Development Kit
CANopen Safety Chip02

QuickStart Manual

Edition May 2009

system house for distributed automation

In this manual are descriptions for copyrighted products which are not explicitly indicated as such. The absence of the trademark (™) and copyright (©) symbols does not infer that a product is not protected. Additionally, registered patents and trademarks are similarly not expressly indicated in this manual

The information in this document has been carefully checked and is believed to be entirely reliable. However, SYS TEC electronic GmbH assumes no responsibility for any inaccuracies. SYS TEC electronic GmbH neither gives any guarantee nor accepts any liability whatsoever for consequential damages resulting from the use of this manual or its associated product. SYS TEC electronic GmbH reserves the right to alter the information contained herein without prior notification and accepts no responsibility for any damages which might result.

Additionally, SYS TEC electronic GmbH offers no guarantee nor accepts any liability for damages arising from the improper usage or improper installation of the hardware or software. SYS TEC electronic GmbH further reserves the right to alter the layout and/or design of the hardware without prior notification and accepts no liability for doing so.

	EUROPE	NORTH AMERICA
Address:	SYS TEC electronic GmbH August-Bebel-Str. 29 D-07973 Greiz GERMANY	PHYTEC America LLC 203 Parfitt Way SW, Suite G100 Bainbridge Island, WA 98110 USA
Ordering Information:	+49 (0) 36 61 / 62 79-0 info@systec-electronic.de	1 (800) 278-9913 info@phytec.com
Technical Support:	+49 (0) 36 61 / 62 79-0 support@systec-electronic.de	1 (800) 278-9913 support@phytec.com
Fax:	+49 (0) 36 61 / 6 32 48	1 (206) 780-9135
Web Site:	http://www.systec-electronic.de	http://www.phytec.com

Edition Mai 2009

Table of Contents

Index of Figures

Index of Tables

1 Introduction

The following documentation describes the CANopen Safety Chip Kit-156. It provides information about the hardware and the function of the example programs and included tools.

There are additional detailed function descriptions of the CANopen Safety Chip CSC02 in the document entitled "Technical Documentation and User Manual CANopen Safety Chip CSC02" (Document Nr.: L-1027_10).

Caution:
The KIT-156 has not been verified by any certification authority and cannot be used for any safety-related procedures in this form. It is intended exclusively for the demonstration of possible applications of the partially certified CANopen Safety Chip CSC02 and development with this chip.

Number Denotation Used:
Hexadecimal numbers are depicted in C notation.

Example:
Hexadecimal value: 0x1234 corresponds to the decimal value: 4460

1.1 Kit Contents

The kit consists of the following components:

- 2x CSC02 reference boards
- USB-CANmodul with PCANview^{TM} for Windows
- CAN cable with terminating resistors
- RS-232 expansion cable
- Tasking demo version for M16C compiler v3.1rl and appropriate patch 5 – registration see 4.1
- Demo version CANopen configurator with safety expansion
- Example software for CSC02 implementation in sensors and actuators (SO-1058)
- CRC builder (SO-1022)
- Renesas Flash Development Toolkit
- Documentation in PDF format on CD in the following directory: ***\Products\CANopen Safety Chip02 KSPS-0208\Manual***
  - "Technical Documentation and User Manual CANopen Safety Chip CSC02" (L-1027_10)
  - "Product Requirements CANopen Safety Chip CSC01" (L-1027_1)
  - Hardware and software documentation for RENESAS M306N4FGTFP microcontroller
  - Reference board circuit diagram

If it is ordered a reference board only, the following is included:

- Example software for CSC02 implementation in sensors and actuators (SO-1058)
- Documentation in PDF format on CD in the following directory: ***\Products\CANopen Safety Chip02 KSPS-0208\Manual***
  - "Technical Documentation and User Manual CANopen Safety Chip CSC02" (L-1027_10)
  - "Product Requirements CANopen Safety Chip CSC01" (L-1027_1)
  - Hardware and software documentation for RENESAS M306N4FGTFP microcontroller
  - Reference board circuit diagram

2 Hardware

2.1 Overview

The hardware includes the following circuit components in addition to the CANopen Safety Chip:

- Screw clamps for connecting the supply voltage
- Slide switch for simulation of an emergency stop button
- 2 LEDs including connections for simulation of an emergency shutdown
- External Watchdog integrated in the actuator example
- Voltage monitoring
- CAN connectivity over DB-9 plug
- First serial interface (RS-232) for programming on DB-9 socket
- Second serial interface (RS-232) on pin header
- Various jumpers for error simulation
- Pin header carrying all of the CANopen Safety Chip's relevant signals
- 4 freely programmable LEDs

The boards included in the kit are identical in terms of hardware. The relevant components are used for the application example in question (depending on whether it is a sensor or an actuator).

The emergency shutdown portion of the system includes the Watchdog, voltage monitor and CANopen Safety Chip itself.

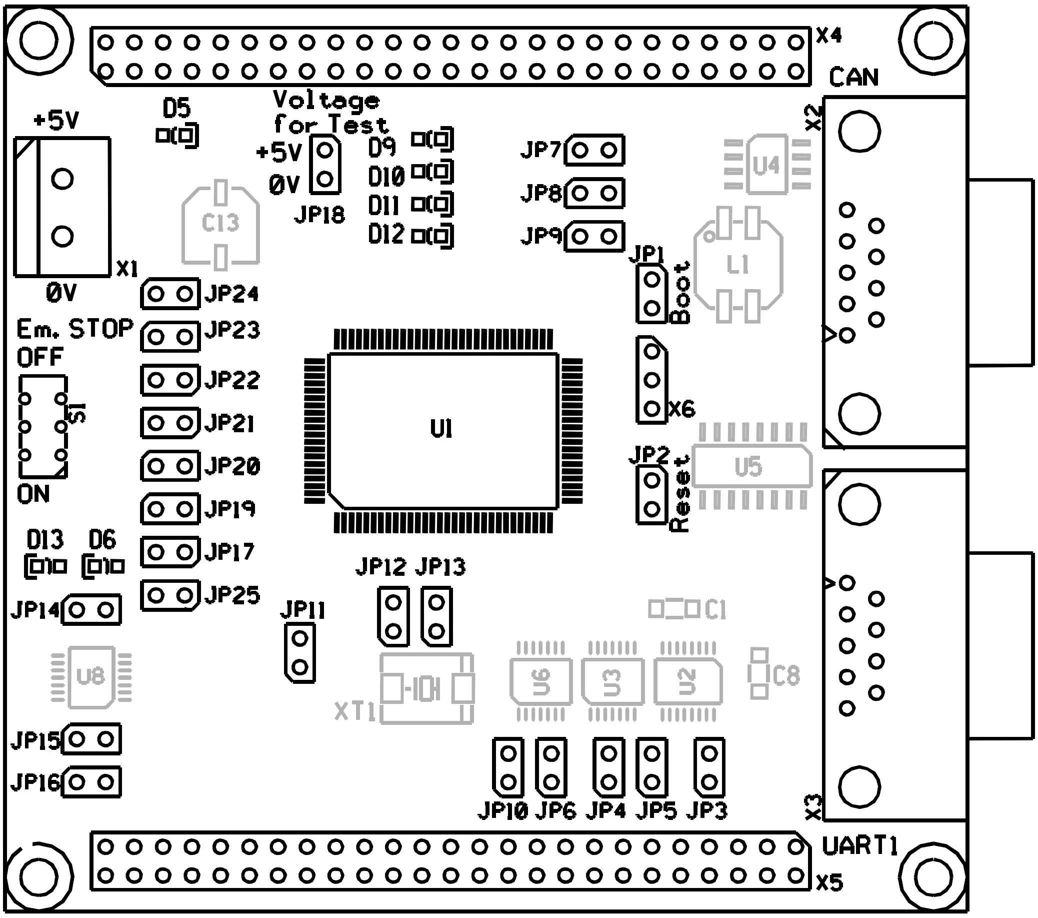


Figure 1: Hardware Overview

2.2 Power Supply

The hardware should be supplied with a regulated voltage of 5V connected to the screw clamp connector X1. The voltage monitoring device MAX6458UKD0A-T is configured so that if the voltage level falls below 4.52 V or exceeds 5.26 V the outputs are disconnected.

Be sure that the polarity is correct *(refer to Figure 1).*

2.3 Simulated Emergency Stop Button

To simulate an emergency stop button there is a dual sliding switch on the hardware. The switch is connected to the CANopen Safety Chip over port pins P2.0 through P2.2 and P3.0 through P3.2.

2.4 Simulated Emergency Shutdown

To simulate a dual-channel emergency shotdown, the hardware has a logical connection to the Watchdog outputs, the voltage monitor as well as port pins P4.0 and P0.0, which affect two LEDs. If the LEDs are off, the hardware is "disconnected". The shutdown occurs as soon as an input signal assumes a low level.

2.5 Jumpers

Number	**Description**
JP1	Activating the programming mode; Default open, BOOT closed
JP2	Resetting the CANopen Safety Chip; Default open, Active closed
JP3 through JP6, JP10	Error simulation of the Watchdog circuitry; Default closed
JP7 through JP9	Error simulation of the CAN connection; Default closed
JP11	Error simulation of the voltage supervision; Default open
JP12, JP13	Error simulation of the clock generation; Default closed
JP15 through JP17, JP25	Error simulation of the disconnect; Default closed
JP18	Test voltage 5V and GND for error simulation; Open
JP19 through JP24	Error simulation of the Emergency Shutoff circuitry; Default closed

Table 1: Jumper Description

2.6 RS-232 Interface

UART1 on the CANopen Safety Chip is used for debugging and programming purposes which extends to the DB-9 socket at X3.

Pin #	Description
2	RS-232 Tx
3	RS-232 Rx
5	GND

Table 2: RS-232 (UART1) Interface at X3, Pin Assignment

UART0 on the CANopen Safety Chip is connected with an RS-232 level to pin header X6 and is available for use in the safety application.

Pin #	Description
1	RS-232 Tx
2	GND
3	RS-232 Rx

Table 3: RS-232 (UART0) Interface at X6, Pin Assignment

2.7 CAN Bus Interface

The CAN interface extends to a DB-9 at plug X2. The pin assignment corresponds to the CiA[1] recommendation DRP[2] 303-1. The CAN interface is not optically isolated.

Pin #	Description
1	CAN_L
2	CAN_GND
3	CAN_H

Table 4: CAN Interface at X2, Pin Assignment

[1] : CiA: CAN in Automation

[2] : DRP303-1: CANopen Cabling and Connector Pin Assignment, Draft Recommendation Proposal

2.8 User Programmable LEDs

The LEDs D9 through D12 are connected on ports 3.4 through 3.7. These can be used in the safety application.

LED D12 is used in the example programs to signal the secure state STOP.

D9 signals the NMT state OPERATIONAL.

2.9 Pin Headers X4 and X5 Pin Assignment

The pin assignment of the header connector at X4 is depicted in the following tables. Px.x means that it is a port pin of the CANopen Safety Chip. The pins are numbered as follows:

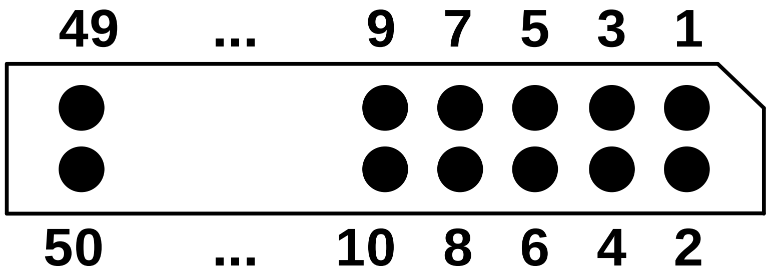


Figure 2: Pin Header X4/X5 Numbering Scheme

Pin #	**Description**
1	VCC (5VDC)
2 ... 9	P0.0 ... P0.7
10, 11	GND
12 ... 19	P1.0 ... P1.7
20, 21	GND
22 ... 29	P2.0 ... P2.7
30, 31	GND
32 ... 40	P3.0 ... P3.7
41, 42	GND
43 ... 50	P4.0 ... P4.7

Table 5: Pin Header X4 Assignment

Pin #	**Description**
1	VCC
2	P8.5 - /NMI - /PowerFail
3	/Reset
4	P8.3 – Actuator Channel 1
5	GND
6 ... 13	P5.0 ... P5.7
14	GND
15 ... 20	P6.0 ... P6.5
21 ... 26	P7.0 ... P7.5
27	P8.0
28	P8.1
29	P8.2
30	P8.4
31	P8.6
32	P8.7
33	GND
34 ... 39	P9.0 ... P9.4, P9.7
40, 41	Vref
42 ... 49	P10.0 ... P10.7
50	GND

Table 6: Pin Header X5 Assignment

2.10 Circuit Diagram

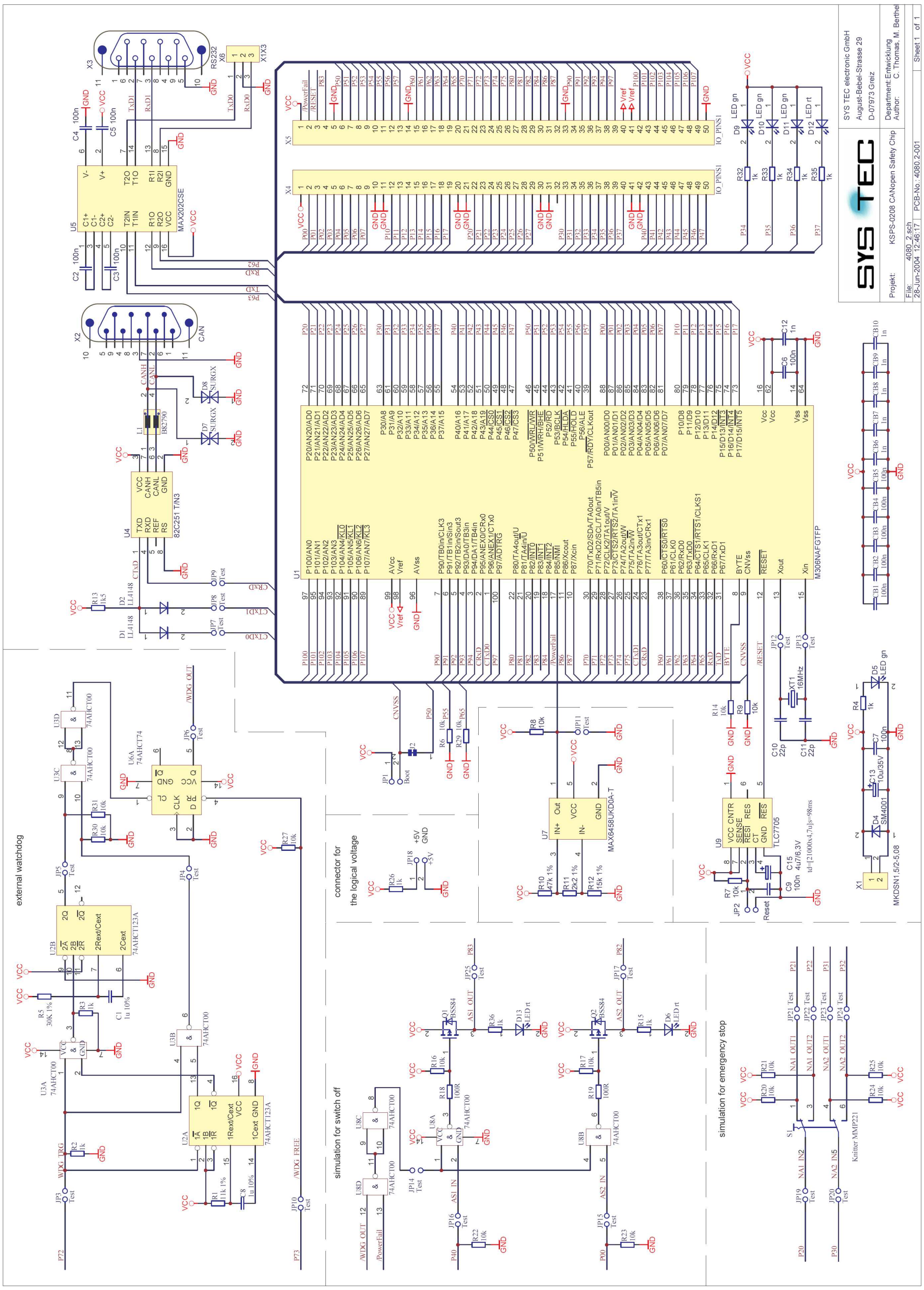


Figure 3: Circuit Diagram

3 Example Programs

3.1 Software Installation

Before installation it is necessary to register the tasking development environment. See 4.1.

To install the software start the file `setup.exe` on the CD in the folder `products\KIT CANopen Safety Chip02 KIT-156`. The example programs are located then in folder `c:\systec\kit-csc` on your hard drive.

The Renesas Flash Development Toolkit has to be installed completely.

3.2 Concept

KIT-156 includes two example programs. One example is for a safety-relevant sensor and the second is an example for a safety-relevant actuator. Each example is based on the other in terms of function and configuration. If the examples are programmed in both boards then they will communicate with eachother.

The example programs with the circuits for the simulated emergency shutoff and the disconnect are only used for demonstarting of the CSC functionality and should be replaced by the user with appropriate safety-related technology and corresponding application software.

File Structure:

- `/callgate`
  - `/include` contains the Callgate internal header file
  - `/source` contains the portion of the Callgate, that has to be included in the safety application
- `/include` contains the common header files for the safety application and the CSC software

- `/userappl_task_v3`
  - `/include` contains the prototypes for application functions that are called by Callgate
  - `/source` contains the example programs for sensors (`usersens.c`) and actuators (`useractr.c`)
  - `/task_prj` contains the Tasking example projects for sensors and actuators and the corresponding Make and Linker files

3.3 Sensors

Configuration:

CAN bit rate:	125kBit/s
CANopen node address:	1

TxSRDO1: ID 0x101/0x102, Length 1, Data xx

Communication Parameter: Index 0x1301

SubIndex 1:	Direction:	1
SubIndex 2:	SCT:	20
SubIndex 3:	SRVT:	10
SubIndex 5:	COB-ID 1:	0x101
SubIndex 6:	COB-ID 2:	0x102

Mapping Parameter: Index 0x1381

SubIndex 0:	2	
SubIndex 1:	Mapping Data:	0x20000108
SubIndex 2:	Mapping Data:	0x21000108
SubIndex 3:	Mapping Data:	0x00000000
...		
SubIndex 16:	Mapping Data:	0x00000000

CRC of this configuration: 0xE7A4

TxSRDO2: disabled

RxSRDO1: disabled

RxSRDO2: disabled

The sensor stresses the Emergency Stop Button with a dynamic signal sent over two channels and reads the response back over two channels.

If the signal combination for "ON" is read, meaning that the Emergency Stop Button has not be pressed, then a 0 is transmitted as the Emergency Stop status in TxSRDO.

If the signal combination for "OFF" is read, meaning that the Emergency Stop Button has been pressed, then a 0xFF is transmitted as the status of the Emergency Stop in TxSRDO.

If an undefined signal combination is read, meaning that the Emergency Stop Button or the connection for the Emergency Stop Button is faulty, then a GFC[1] is transmitted.

The Emergency Stop Button query is debounced. A state machine in the following form has been realized for this purpose.

[1] : Global Failsafe Command, refer to CiA DS304.

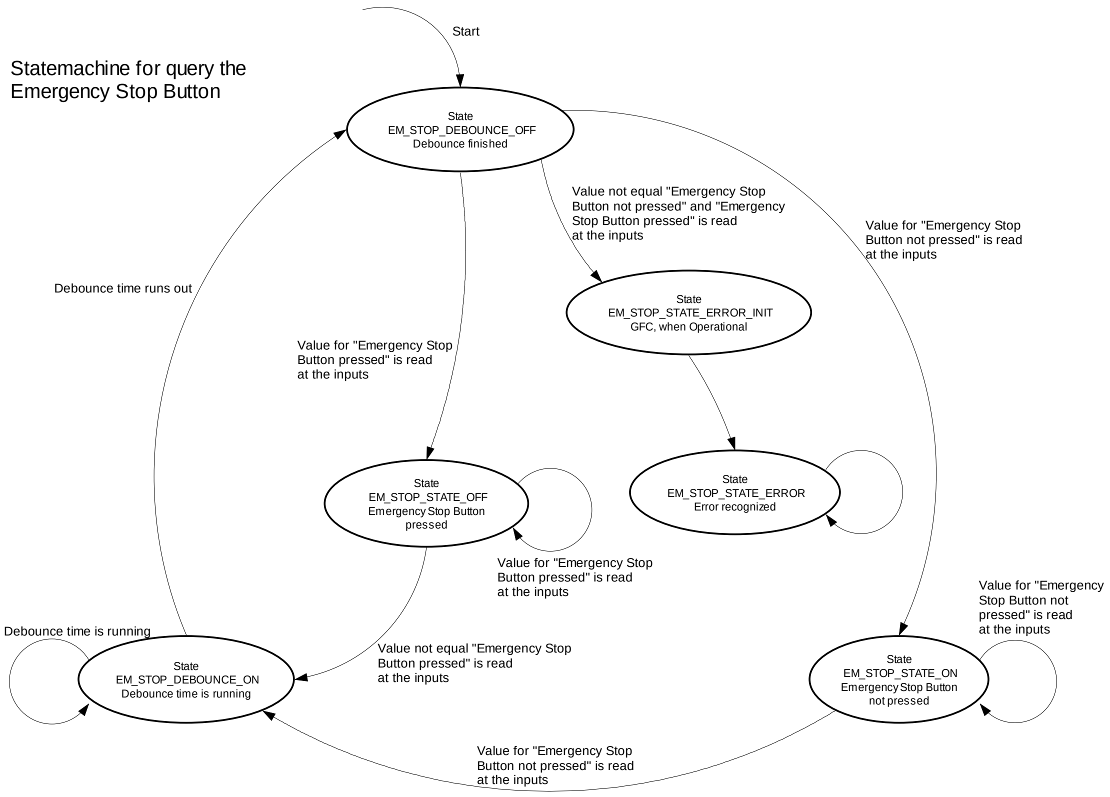

Start
Statemachine for query the Emergency Stop Button
State
EM_STOP_DEBOUNCE_OFF
Debounce finished
Value not equal "Emergency Stop Button not pressed" and "Emergency Stop Button pressed" is read at the inputs
Value for "Emergency Stop Button not pressed" is read at the inputs
Debounce time runs out
State
EM_STOP_STATE_ERROR_INIT
GFC, when Operational
Value for "Emergency Stop Button pressed" is read at the inputs
State
EM_STOP_STATE_OFF
Emergency Stop Button pressed
State
EM_STOP_STATE_ERROR
Error recognized
Value for "Emergency Stop Button pressed" is read at the inputs
Value for "Emergency Stop Button not pressed" is read at the inputs
Debounce time is running
Value not equal "Emergency Stop Button pressed" is read at the inputs
State
EM_STOP_DEBOUNCE_ON
Debounce time is running
State
EM_STOP_STATE_ON
Emergency Stop Button not pressed
Value for "Emergency Stop Button not pressed" is read at the inputs

Figure 4: State Machine for Querying the Emergency Shutoff

The sensor evaluates the SRDO transmission in the following manner:

The status of the SRDO is checked in the SRDO event callback function ***AppSrdoEvent()***. If the status doesn't show the SRDO as having been sent, then the call of the callback function does not match the status. This is safety critical and will result in a shift to STOPP state. A non-matching status means that the callback function was not called from the SRDO module.

In the SRDO error callback function ***AppSrdoError()*** the status of the SRDO is checked as well. If the status doesn't indicate that the SRDO transmission was faulty, then the call of the callback function will not match the status. This is safety critical and will result in a shift to STOPP state. A non-matching status means that the callback function was not called from the SRDO module.

The SRDO number is checked in this manner as well. If it is not equal to 1 (TxSRDO1) it is safety critical, since the SRDOs are shut off in the example. This case will also result in a shift to STOPP state (in the event callback the CSC status in the safety related RAM is set to STOPP; in the error callback the SRDO state is not toggled, whereby the CSC stack will likewise shift to STOPP state).

Furthermore the SRDO state (Variable `m_Srdox.m_bState` in safety related RAM) is monitored in the applications process function. If a status is discovered there that does not match the idle state, it is safety critical and will result in a shift to STOPP state. Registration of an SRDO state that does not match the idle state means that the SRDO callback function was not executed.

3.4 Actuators

Configuration:

CAN Bit Rate:	125 kBit/s
CANopen Node address:	2

TxSRDO1: disabled

TxSRDO2: disabled

RxSRDO1: ID 0x101/0x102, Length 1, Data xx

Communication Parameter: Index 0x1303

SubIndex 1:	Direction:	2
SubIndex 2:	SCT:	30
SubIndex 3:	SRVT:	10
SubIndex 5:	COB-ID 1:	0x101
SubIndex 6:	COB-ID 2:	0x102

Mapping Parameter: Index 0x1383

SubIndex 0:	2	
SubIndex 1:	Mapping Data:	0x20010108
SubIndex 2:	Mapping Data:	0x21010108
SubIndex 3:	Mapping Data:	0x00000000
...		
SubIndex 16:	Mapping Data:	0x00000000

CRC of this configuration: 0x46FA

RxSRDO2: disabled

The actuator evaluates the Emergency Stop Button sent by the sensor in the SRDO, and if a shutdown event is registered there (status 0xFF) then there will be a dual-channel emergency shutdown (received over RxSRDO1).

If the status in SRDO returns to be 0x00, then the shutdown will be deactivated.

Furthermore the actuator serves the external Watchdog. The initialization of the Watchdog occurs in the function ***AppInitialization()***, if the parameter is `bInitType p=APP_INIT_WDG`.

The monitoring, diagonsis and retriggering occurs in the function ***AppWdgProcess()***. The Watchdog is triggered at the end of a safety cycle. It is recognized by monitoring the variable `CsCRAMBlockA g.m wCscState=CSC_STATE_SAFETY_CYCLE_RDY.`

The diagnosis occurs once every hour. This is also carried out in the function ***AppWdgProcess()***. No RAM test or output diagnosis occurs during the Watchdog diagnosis. A Watchdog diagnosis is only possible if the outputs are set to ON, because only then can the reaction of the Watchdog shutoff be recognized and evaluated. The shutoff is then diagnosed if the level falls below the lower trigger threshold or exceeds the upper trigger threshold.

The design of the Watchdog on the reference board has the following characteristics:

Nominal trigger time:	20 ms
Lower limit:	10.6 ±1.4 ms
Upper limit:	28.9 ±4 ms

If the actuator recognizes that the sensor is in the safe state STOPP (SRDO is no longer being received), or that SRDO is faulty, an emergency shutdown will occur as well. In this case a restart will be prevented.

The actuator evaluates the receipt of SRDO in the following manner:

The status of the SRDO is checked in the SRDO event callback function ***AppSrdoEvent()***. If the status doesn't indicate that the SRDO was received successfully, then the call of the callback function does not match the status. This is safety critical and will result in a shift to STOPP state. A non-matching status means that the callback function was not called from the SRDO module.

The status of the SRDO is also checked in the SRDO error callback function ***AppSrdoError()***. If this doesn't indicate that the SRDO receipt was faulty, then the call of the callback function does not match the status. This is safety critical and will result in a shift to STOPP state. A non-matching status means that the callback function was not called from the SRDO module.

The SRDO number is also checked. If it does not equal 3 (RxSRDO1) the result is also safety critical, since in the example these SRDOs are shut off. In this case it will switch to STOPP state as well (in the event callback the SRDO state is not toggled, whereby the CSC stack will also switch to STOPP state).

Furthermore, the SRDO state (Variable `m_Srdox.m_bState` in safety related RAM) will be monitored in the application's process function. If a status is discovered there that does not match the idle state, it is safety critical and will result in a switch to STOPP state. Registration of an SRDO state that does not match the idle state means that the SRDO callback function was not executed.

3.5 QuickStart

First install the included software like it is discribed in *3.1 Software Installation*. Furthermore install the Tasking Development Environment.

The kit contains two pre-programmed devices, a sensor and an actuator.

1. Connect both modules to the CAN bus.
2. Connect the USB-CANmodul to the CAN bus.
3. Display the CAN bus traffic with the program PCANview^{TM} using the USB-CANmodul (bit rate 125 kBit/s).
4. Connect the supply voltage to the modules.
5. A bootup message for each module will be displayed on the CAN bus.
6. Switch both modules to OPERATIONAL state (e.g. send the following CAN message using PCANview™: CAN-ID 0x000, Length (DLC) 2, data: 0x01, 0x00,).
7. Both modules begin sending SRDOs and PDOs.

3.6 Error Diagnosis

Transmission of emergency messages has been implemented in the example program for external error diagnosis.

An emergencey is sent:

1. in the Function ***AppPreSafetyStop()***

The emergency contains the following data:

Emergency Error Code:	0xFF00
Error Register:	0x01

Manufacturer specific Error Field

Byte 0...1:	CSC state
Byte 2:	last CANopen return code

Byte 3:	last diagnosis return code
Byte 4:	application-internal error status

The application-internal error states are defined as follows

`APP_STATE_NO_ERROR`	`0x00` no error
`APP_STATE_UNKNOWN_INT`	`0x01` unknown interrupt ***AppDefaultHandler ()***
`APP_STATE_RAM_ERROR`	`0x02` error during cross comparison of safety related RAM in the safety application
`APP_STATE_SRDO_ERROR`	`0x04` SRDO state error
`APP_STATE_ACTR_ERROR`	`0x08` Error during disconnect (output should be switched to "ON", but it was read back as "OFF")
`APP_STATE_ACTR_ERROR_OFF`	`0x10` Error during disconnect (output should be set to "OFF", however it was read back as "ON")
`APP_STATE_WDT_ERROR_LOW`	`0x20` Error during diagnosis of the Watchdog's lower limit, Watchdog does not shut off
`APP_STATE_WDT_ERROR_UPP`	`0x40` Error during diagnosis of the Watchdog's upper limit, Watchdog does not shut off
`APP_STATE_WDT_ERROR_UPP_TIME`	`0x40` Error during diagnosis of the Watchdog's upper limit, time exceeded

This emergency can only be sent after the CANopen has been initialized. If there was already a shift to STOPP state in the beginning initialization (e.g. because the safety application's CRC sum is not correct), then this error will not be displayed.

2. in the function ***AppSrdoError()***

The emergency contains the following data:

Emergency Error Code:	0xFF01
Error Register:	0x01

Manufacturer specific Error Field

Byte 0:	SRDO number
Byte 1:	error code
Byte 2...4:	not used

3. in the function ***AppPdoEvent()***

If an incorrect PDO length is received the emergency containing the following data will be sent:

Emergency Error Code:	0x8220
Error Register:	0x01

Manufacturer specific Error Field

Byte 0...4:	not used

4 Software

4.1 Tasking Demo

The demo version of the TASKING Tools development environment for M16C v3.1rl is included in the kit.
The appropriate patch 5 must be installed too.

Caution:
Before installation the necessary serial number and the 30-day-licencefile must be requeseted at
info@systec-electronic.com
The subject for the request is „Licence Request Tasking for CSC“.

The TASKING Tools for M16C v3.1rl demo version makes it possible to compile and debug example projects.

When you open an example project it is possible that Tasking will require that you switch the tool chain. This is because the projects were created with the TASKING full version. This will not cause problems and is not an error. The tool chain can be set to accomodate the demo version.

When switching between the individual example projects it is important to make sure that the complete project is re-compiled each time. If this does not occur problems can result when linking.

The code generation does not occur with the settings in the tool chain, instead an external make file is used. The make files are entered in the tool chain using the menu options *"Build" -> "Options" -> "Use external makefile"*. This guarantees that the project settings cannot be altered by mistake.

Since the make files are path dependent, the path may have to be adapted depending on the installation of the TASKING compiler. The following default settings are entered in the make files:

```
#TASKING Tools Demo
PROJDIR = c:\systec\kit-csc\userappl_task_v3\task_prj
PRODDIR = c:\Program Files\TASKING\cm16c v3.1
```

Please modify the path settings according to your configuration.

There are important differences between the Debug project settings and the settings in the release projects (for the release project settings refer to the *"Technical Documentation and User Manual CANopen Safety Chip CSC02"*).

1. Including the file `cstart_csc.src` to the project

The file `cstart_csc.src` is already included in the project. It is used as the starting point for the monitor and must not be modified.

4.2 Debugging Procedures

1. Render the target hardware into Boot mode.
   Insert Jumper JP1 – Boot.
   Generate a hardware reset on jumper JP2

2. Erase the Flash sectors from address 0x0FC000 – 0x0FFFFF and program the monitor file included in the kit

This can be accomplished with the help of the "Flash Development Toolkit" by Renesas. This tool is included in the kit or can be found on the Internet under the following link:

http://www.renesas.com

It is also possible to use other Flash tools. It is important that a partial erasure of the Flash is possible with whichever Flash tools you use.

The use of the "Flash Development Toolkit" is described in the following sections with the help of an example. The FDT is a powerful tool for flash development but here only the feature for easing the flash is used. So it is not necessary to add files to the FDT workspace. Only the CPU type and the connection to the target board has to set up.

The kit includes a ready workspace and batches for easy erasing the right flash sectors.
In this workspace the CPU type is configured an the COM1 for serial connection to the target is selected.
This configuration can be adapted for this workspace like described in 7.

Caution:
The wrong setting can cause the erasure of the permanent firmware rendering the hardware inoperable.

To prevent for this the batches in directory `c:\systec\kit-csc\fdt\kit-156` can be used.

erase_app.bat	erase the flash 0xe0000…0xeffff
erase_startup.bat	erase the flash 0xfc000…0xfffff
prog_monitor.bat	erase the flash 0xfc000…0xfffff and program the monitor csc02mon
prog_startup.bat	erase the flash 0xfc000…0xfffff and program the startup of perment firmware

If the batches are not used, the handling of the FDT is described in Annex 1 step by step. If the batches are used, it is not necessary!

A call of the batch *prog_monitor.bat* erases the needed flash sectors and programs the monitor.

3. Exit Boot mode.

Remove jumper JP1 and generate a hardware reset.

The steps 1 through 3 only have to be carried out once. If the monitor has been programmed already previously, you can begin at step 4.

4. Open the example project.

Open the file `userappl.psp` by selecting *"File"* -> *"Open Project Space"* in the Tasking EDE.

Set one of the two debug projects `usersens_dbg` or `useractr_dbg` as an active project (click on the project using the right mouse button and select *"Set as Current Project"*).

5. Compile/build the entire project.

6. Calculate the CRC sums of the SRDOs and enter them into the example program.

The values are entered in the structure of the default SRDO communication parameters over the constants `SRDO_TX1_CRC, SRDO_TX2_CRC, SRDO_RX1_CRC` and `SRDO_RX2_CRC`.

In the example various CRC sums have been entered based on the bit rate. This is based on the fact that different values for SCT and SRVT are used for the various bit rates. The bit rate itself has no influence on the CRC of the SRDOs.

7. Determine the size of the application's code range and enter it in the example program.

The code size is entered over the variable `wAppcodeBlockSize_g`. The determined length should be given directly in this variable.

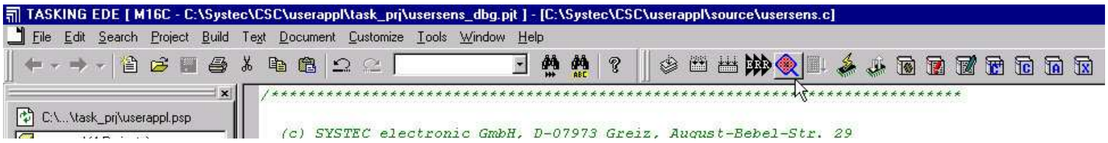
8. Calculate the CRC sum of the application and enter into the example program.

The CRC sum of the application is entered over the variable `wCrcUsrAppl_g`.

The values from points 6 through 8 can easily be calculated with the "CRC Builder" tool *(refer to section 5)*.

9. Start the Debugger.

- Start the *"Cross View Pro"* debugger by clicking on the following icon:


TASKING EDE [M16C - C:\Systec\CSC\userappl\task_prj\usersens_dbg.pjt] - [C:\Systec\CSC\userappl\source\usersens.c]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\task_prj\userappl.psp

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

  (c) SYSTEC electronic GmbH, D-07973 Greiz, August-Bebel-Str. 29
```

Figure 5: Starting the Debugger

10. Debugging the example program.

After loading the project onto the hardware the program will stop at the label `Start` in the startup code.

The program cycle can be interrupted by setting a breakpoint in the file `usersens.c/useractr.c`.

It is important to note that by interrupting the program the time cycle of the permanent CSC main software can no longer be guaranteed. There is also no longer a trigger for the external Watchdog (only used with the actuator).

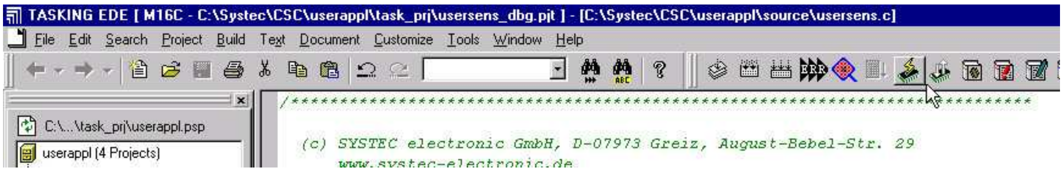
4.3 Release Version

If a release version (the projects without _`dbg`) is to be programmed into the CANopen Safety Chip, proceed as follows.

1. Activate Boot mode.

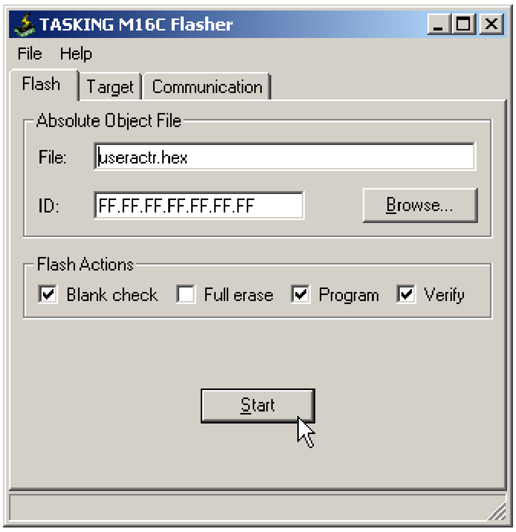
2. Erase Flash blocks beginning at address 0x0E0000 – 0x0EFFFF
   Call batch *erase_app.bat*
3. Erase Flash block beginning at address 0x0FC000 – 0x0FFFFF and program the starup of permanent firmware
   Call batch *prog_startup.bat*
4. Program the file `usersens.hex/useractr.hex` to the Flash with the Tasking Flasher.

Start the Flasher by clicking the button indicated in the image below


TASKING EDE [M16C - C:\Systec\CSC\userappl\task_prj\usersens_dbg.pjt] - [C:\Systec\CSC\userappl\source\usersens.c]
File Edit Search Project Build Text Document Customize Tools Window Help
C:\...\task_prj\userappl.psp
userappl (4 Projects)

```
/*******************************************************************************
  (c) SYSTEC electronic GmbH, D-07973 Greiz, August-Bebel-Str. 29
      www.systec-electronic.de
```

Figure 6: Starting the Tasking Flasher



Absolute Object File

File: useractr.hex

ID: FF.FF.FF.FF.FF.FF.FF

Flash Actions

☑ Blank check ☐ Full erase ☑ Program ☑ Verify

Figure 7: Loading the Monitor

Download by clicking the *"Start"* button.

5. Deactivate Boot mode.

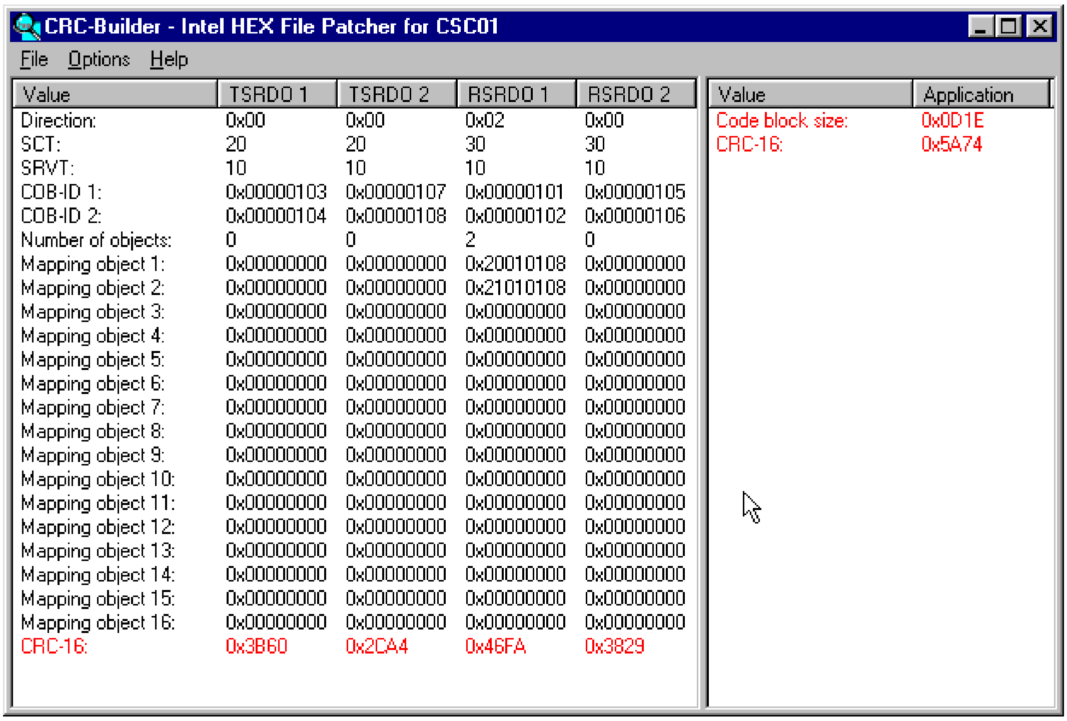
6. Reset the hardware. The program will start.

The calculation of the CRC sums and the code size of the application can be performed easily with the CRC Builder tool. This tool can also enter the calculated value directly into a generated Hex file.

5 CRC Builder

The CRC Builder is a tool used to calculate the CRC sums over the safety application and the SRDO configuration. It enables the corresponding Intel hex-files to be patched. The CRC builder is located on th CD in the folder `products\so-1022`. Please copy the folder to your hard drive, for example: `c:\systec\kit-csc\tools\crcbuild`.

The tool is started by calling `crcptwin.exe`. The following dialog box will open:



Value	TSRDO 1	TSRDO 2	RSRDO 1	RSRDO 2
Direction:	0x00	0x00	0x02	0x00
SCT:	20	20	30	30
SRVT:	10	10	10	10
COB-ID 1:	0x00000103	0x00000107	0x00000101	0x00000105
COB-ID 2:	0x00000104	0x00000108	0x00000102	0x00000106
Number of objects:	0	0	2	0
Mapping object 1:	0x00000000	0x00000000	0x20010108	0x00000000
Mapping object 2:	0x00000000	0x00000000	0x21010108	0x00000000
Mapping object 3:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 4:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 5:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 6:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 7:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 8:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 9:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 10:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 11:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 12:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 13:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 14:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 15:	0x00000000	0x00000000	0x00000000	0x00000000
Mapping object 16:	0x00000000	0x00000000	0x00000000	0x00000000
CRC-16:	0x3B60	0x2CA4	0x46FA	0x3829

Value	Application
Code block size:	0x0D1E
CRC-16:	0x5A74

Figure 8: CRC Builder Window

The corresponding file is selected with *"File"->"Open Intel HEX File"*. It calculates the CRC sum and the size of the safety application (right window: in example CRC=0x5A74, size=0xD1E) and the CRC sums of the SRDO configurations for all 4 SRDOs (left window: in example for RxSRDO 1 0x46FA). The CRC sums can subsequently be entered into the corresponding C file (e.g. usersens.c) and the project can be re-compiled.

The CRC sums can be entered directly in the corresponding Intel HEX file over the menu item *"File"->"Save patched Intel HEX File"*.

6 CANopen Device Monitor CDM

This tool is used for configuration of single Nodes.
For a detailed description of installation and controling use the document L-1056 .

Der CDM works per default with the kit included SYS TEC USB-CANmodul. So connect this device with the PC and the CAN-network.

Exemplary the configuration of an SRDO is here described. The identifier of the SRDO will be changed.

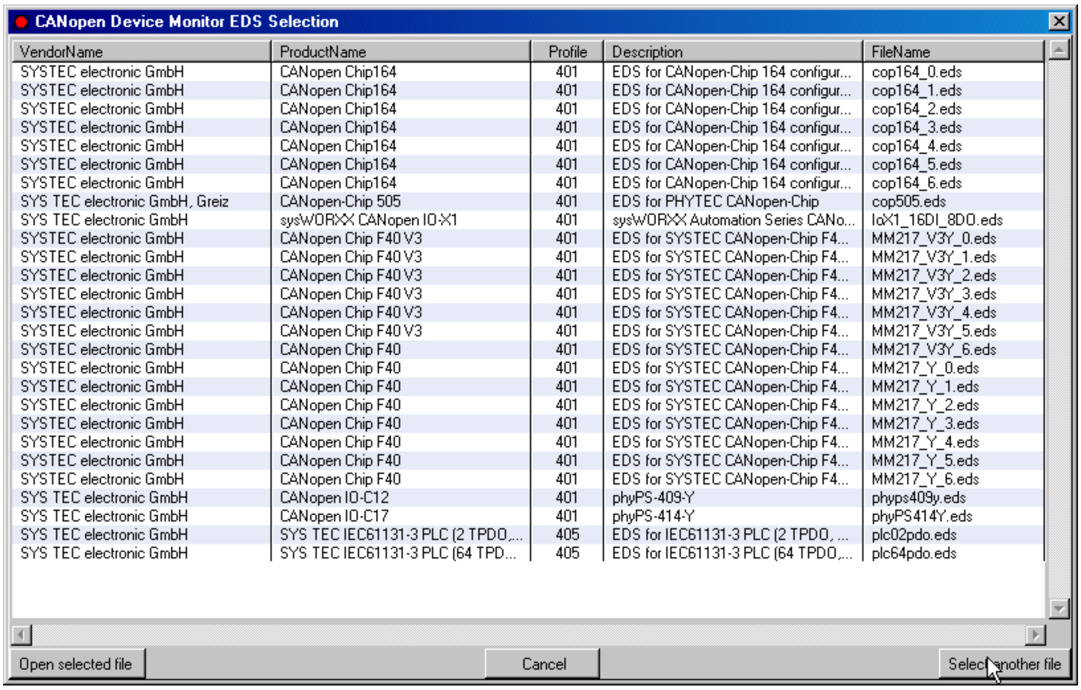
6.1 Activate the Safety-Plugin

The Safety-Plugin is to activate ones. This is done in menu *“Extras”* -> *“Plug-ins”* -> *“Safety”*.
After this the Tab “SRDO” in the right side of the window becomes visible.

6.2 Load EDS-File

To be able to configure the devices it is necessary to load the correspondent EDS-File. Choose in menu *“File”* -> *“Load EDS”*.

In the following window choose *“Select another File”*.



VendorName	ProductName	Profile	Description	FileName
SYSTEC electronic GmbH	CANopen Chip164	401	EDS for CANopen-Chip 164 configur...	cop164_0.eds
SYSTEC electronic GmbH	CANopen Chip164	401	EDS for CANopen-Chip 164 configur...	cop164_1.eds
SYSTEC electronic GmbH	CANopen Chip164	401	EDS for CANopen-Chip 164 configur...	cop164_2.eds
SYSTEC electronic GmbH	CANopen Chip164	401	EDS for CANopen-Chip 164 configur...	cop164_3.eds
SYSTEC electronic GmbH	CANopen Chip164	401	EDS for CANopen-Chip 164 configur...	cop164_4.eds
SYSTEC electronic GmbH	CANopen Chip164	401	EDS for CANopen-Chip 164 configur...	cop164_5.eds
SYSTEC electronic GmbH	CANopen Chip164	401	EDS for CANopen-Chip 164 configur...	cop164_6.eds
SYS TEC electronic GmbH, Greiz	CANopen-Chip 505	401	EDS for PHYTEC CANopen-Chip	cop505.eds
SYS TEC electronic GmbH	sysWORXX CANopen IO-X1	401	sysWORXX Automation Series CANo...	IoX1_16DI_8DO.eds
SYSTEC electronic GmbH	CANopen Chip F40 V3	401	EDS for SYSTEC CANopen-Chip F4...	MM217_V3Y_0.eds
SYSTEC electronic GmbH	CANopen Chip F40 V3	401	EDS for SYSTEC CANopen-Chip F4...	MM217_V3Y_1.eds
SYSTEC electronic GmbH	CANopen Chip F40 V3	401	EDS for SYSTEC CANopen-Chip F4...	MM217_V3Y_2.eds
SYSTEC electronic GmbH	CANopen Chip F40 V3	401	EDS for SYSTEC CANopen-Chip F4...	MM217_V3Y_3.eds
SYSTEC electronic GmbH	CANopen Chip F40 V3	401	EDS for SYSTEC CANopen-Chip F4...	MM217_V3Y_4.eds
SYSTEC electronic GmbH	CANopen Chip F40 V3	401	EDS for SYSTEC CANopen-Chip F4...	MM217_V3Y_5.eds
SYSTEC electronic GmbH	CANopen Chip F40	401	EDS for SYSTEC CANopen-Chip F4...	MM217_V3Y_6.eds
SYSTEC electronic GmbH	CANopen Chip F40	401	EDS for SYSTEC CANopen-Chip F4...	MM217_Y_0.eds
SYSTEC electronic GmbH	CANopen Chip F40	401	EDS for SYSTEC CANopen-Chip F4...	MM217_Y_1.eds
SYSTEC electronic GmbH	CANopen Chip F40	401	EDS for SYSTEC CANopen-Chip F4...	MM217_Y_2.eds
SYSTEC electronic GmbH	CANopen Chip F40	401	EDS for SYSTEC CANopen-Chip F4...	MM217_Y_3.eds
SYSTEC electronic GmbH	CANopen Chip F40	401	EDS for SYSTEC CANopen-Chip F4...	MM217_Y_4.eds
SYSTEC electronic GmbH	CANopen Chip F40	401	EDS for SYSTEC CANopen-Chip F4...	MM217_Y_5.eds
SYSTEC electronic GmbH	CANopen Chip F40	401	EDS for SYSTEC CANopen-Chip F4...	MM217_Y_6.eds
SYS TEC electronic GmbH	CANopen IO-C12	401	phyPS-409-Y	phyps409y.eds
SYS TEC electronic GmbH	CANopen IO-C17	401	phyPS-414-Y	phyPS414Y.eds
SYS TEC electronic GmbH	SYS TEC IEC61131-3 PLC (2 TPDO,...	405	EDS for IEC61131-3 PLC (2 TPDO, ...	plc02pdo.eds
SYS TEC electronic GmbH	SYS TEC IEC61131-3 PLC (64 TPD...	405	EDS for IEC61131-3 PLC (64 TPDO,...	plc64pdo.eds

Figure 9: CDM Load EDS-File

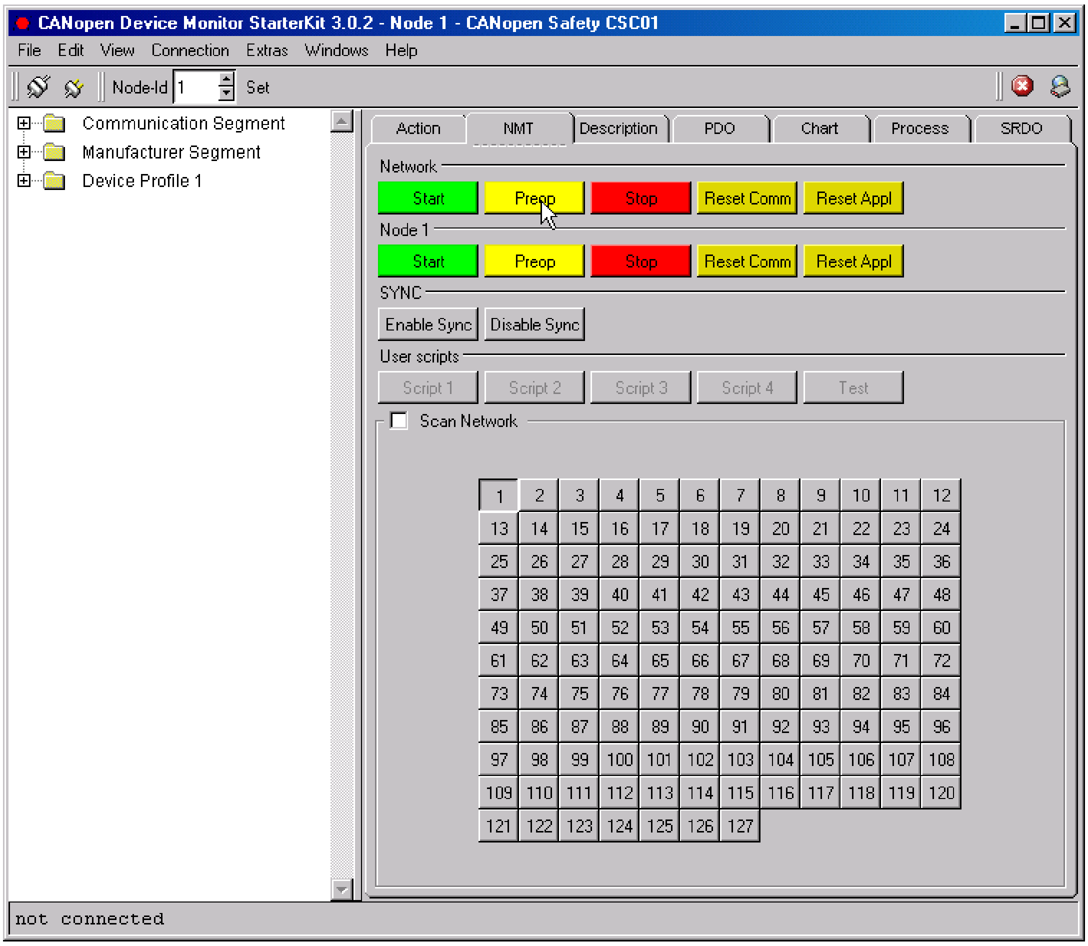
There select the EDS-File for the Safety Chips. It is located in the directory of the demoprogramms under `c:\systec\kit-csc\userappl_task_v3\eds`.
Use this EDS-Files for Actuator and Sensor.

6.3 Connect CDM with network

Connect the CANopen Device Monitor in menu *“Connection”* -> *“Connect”* with the CAN-network.

6.4 Set both nodes in state Preoperational

The configuration of an SRDO is only possible in the state Preoperational.
To do this press the button *“Network Preop”* in Tab *“NMT”*.

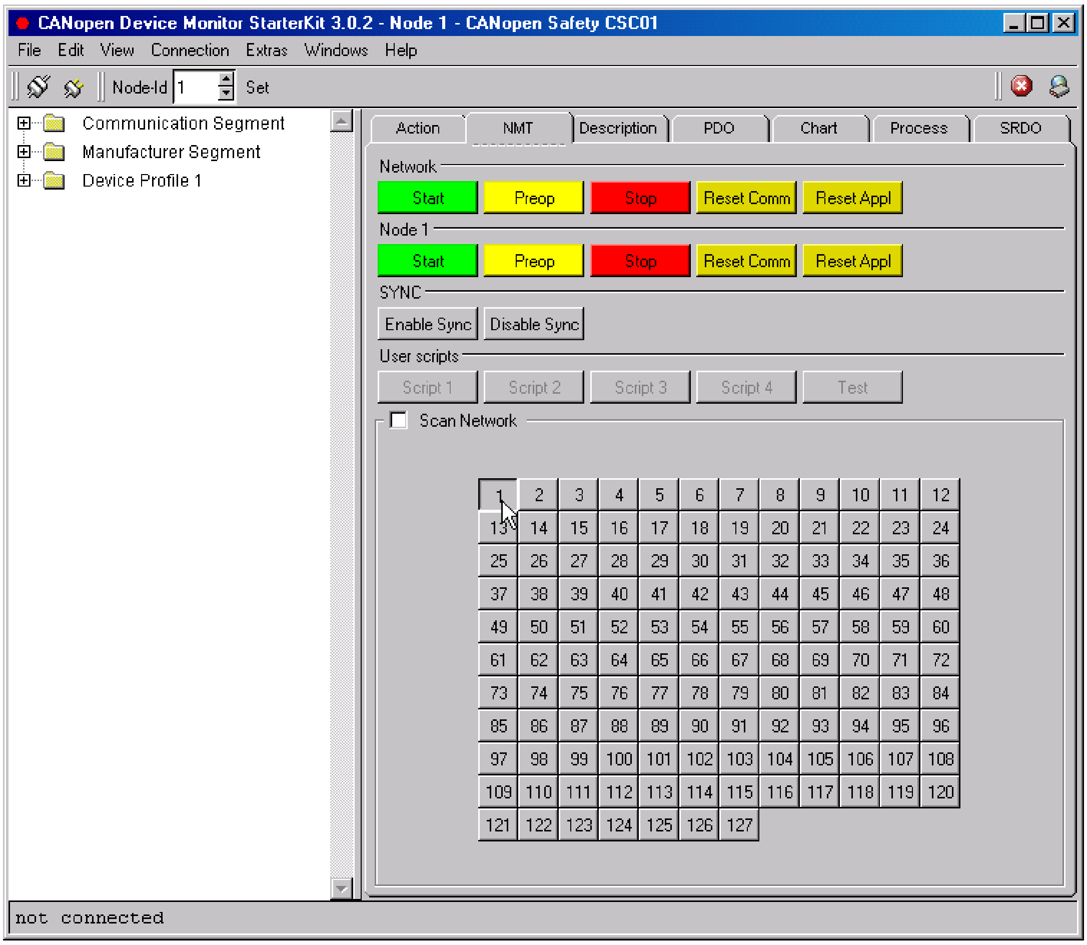


```
not connected
```

Figure 10: CDM set Nodestate Preoperational

6.5 Configure the sensor

Connect the CDM with node number 1 in tab „*NMT*“.



Node-Id 1

□ Scan Network

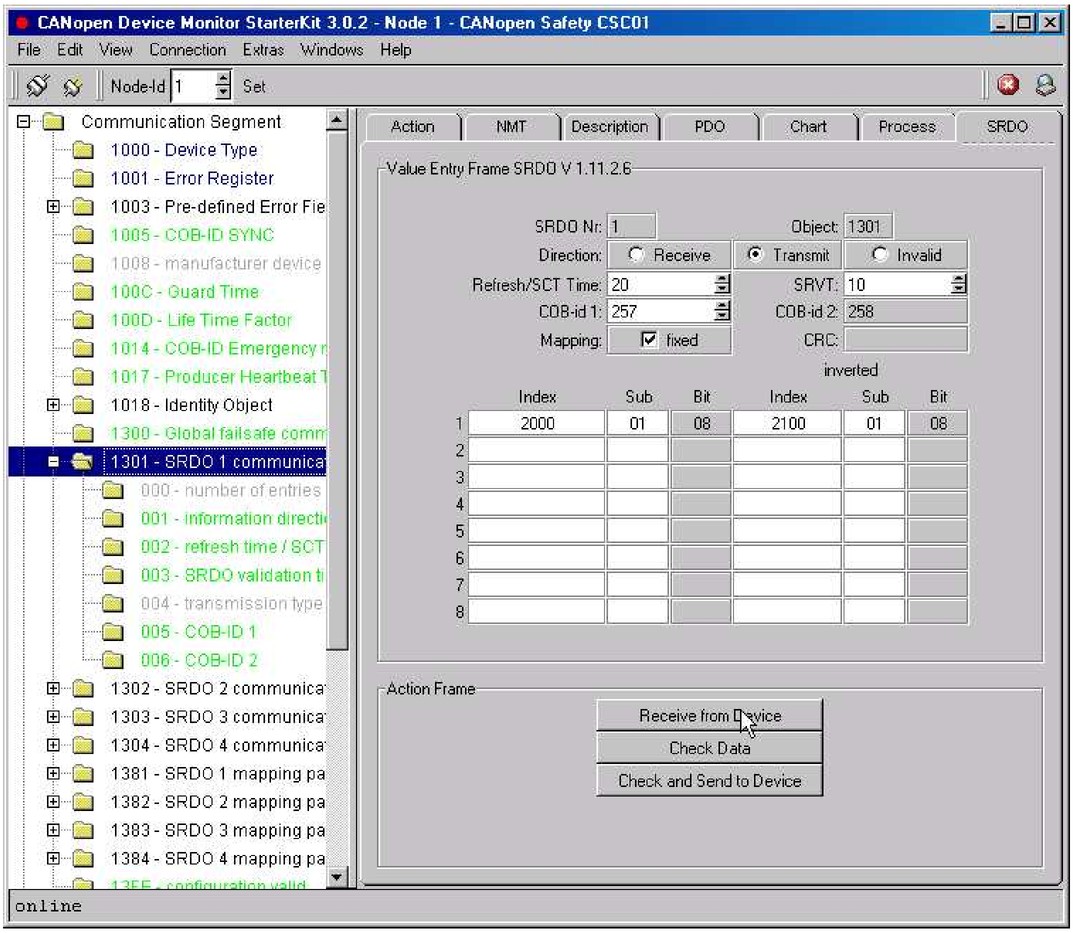
1	2	3	4	5	6	7	8	9	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104	105	106	107	108
109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127					

```
not connected
```

Figure 11: CDM set Nodenumber

Click on Number 1.

Open the Communicationparameter of the first TSRDO and read the data from the sensor by pressing the button *"Receive from Device"*.



Value Entry Frame SRDO V 1.11.2.6

SRDO Nr: 1 Object: 1301

Direction: ○ Receive ● Transmit ○ Invalid

Refresh/SCT Time: 20 SRVT: 10

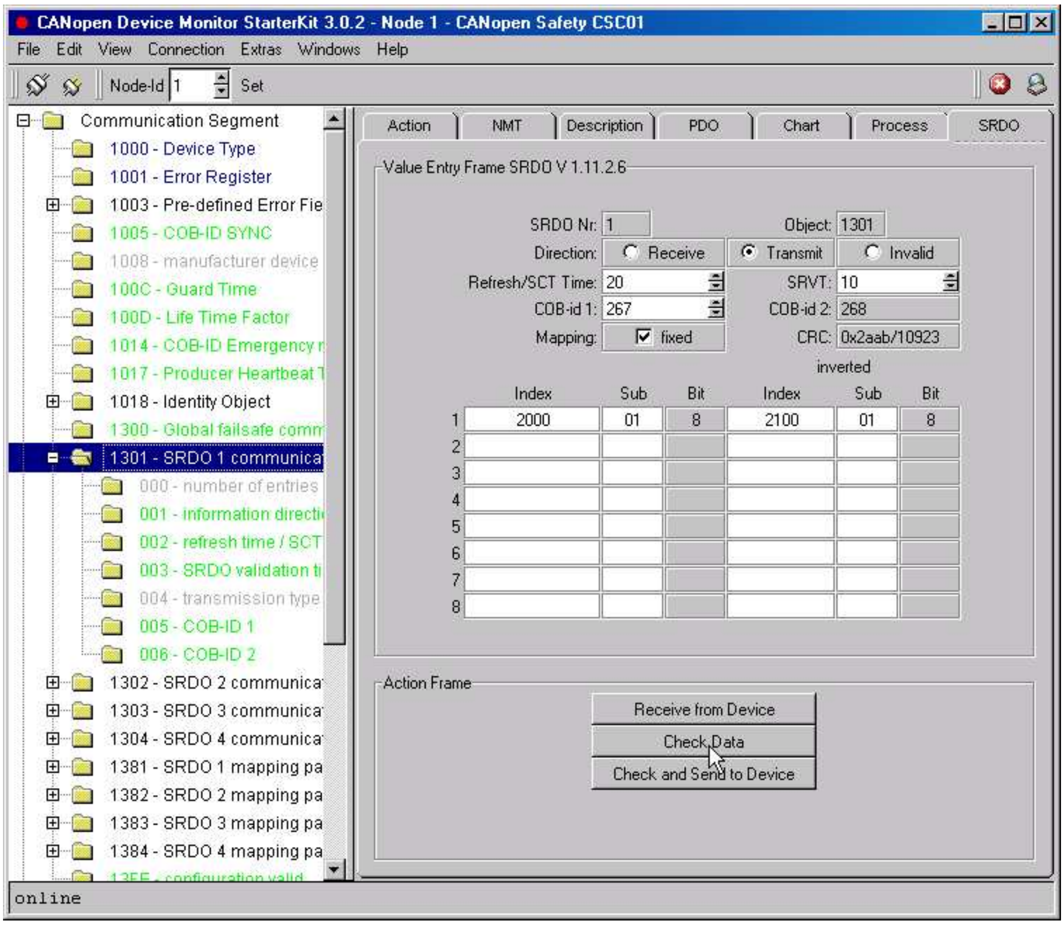
COB-id 1: 257 COB-id 2: 258

Mapping: ☑ fixed CRC:

				inverted		
	Index	Sub	Bit	Index	Sub	Bit
1	2000	01	08	2100	01	08
2						
3						
4						
5						
6						
7						
8						

Figure 12: CDM read SRDO-Configuration from Node

Change the identifier in the field *"COB-ID 1"* such as 267. By pressing the button *"Check Data"* the configuration can be checked and the CRC of the SRDO can be calculated.



Value Entry Frame SRDO V 1.11.2.6

SRDO Nr: 1 Object: 1301
Direction: ○ Receive ● Transmit ○ Invalid
Refresh/SCT Time: 20 SRVT: 10
COB-id 1: 267 COB-id 2: 268
Mapping: ☑ fixed CRC: 0x2aab/10923

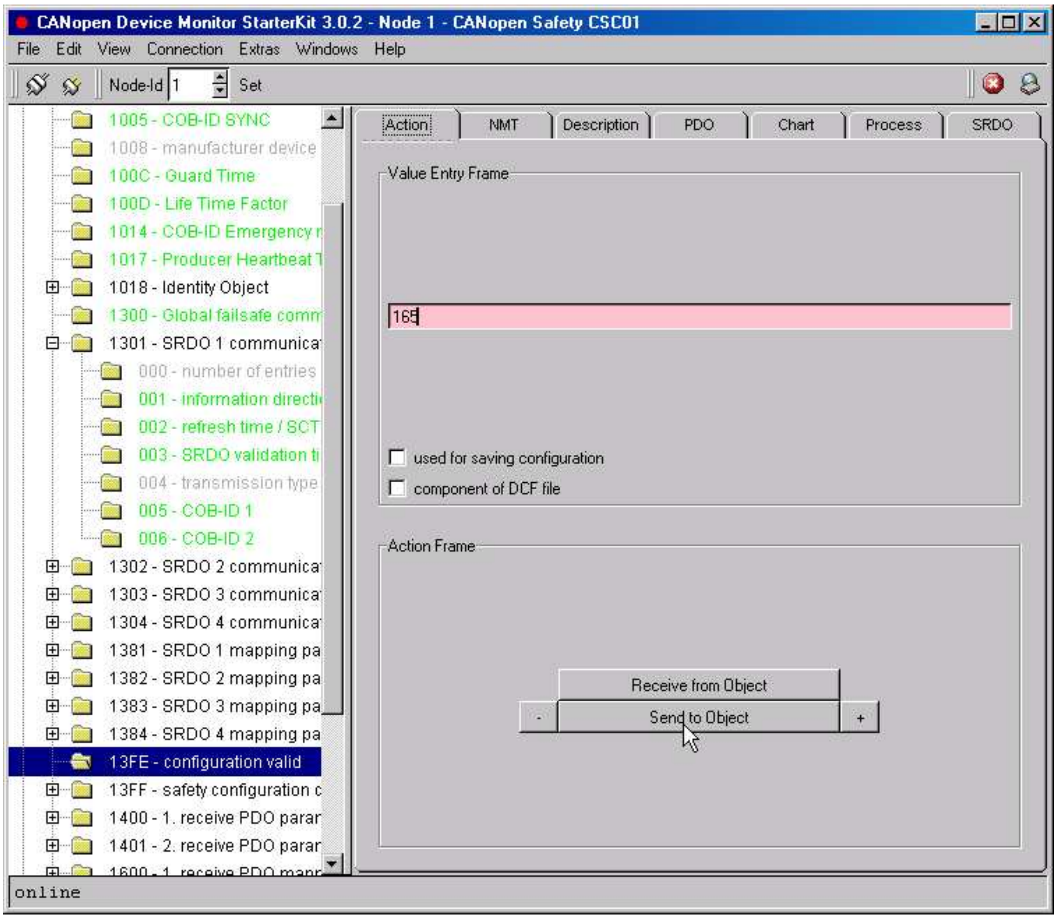
	Index	Sub	Bit	Index	Sub	Bit
				inverted		
1	2000	01	8	2100	01	8
2						
3						
4						
5						
6						
7						
8						

online

Figure 13: CDM change SRDO-Configuration (TSRDO 1)

By pressing the button *"Check and Send to Device"* the configuration will be transmitted to the node.

By write the indexentry 0x13FE “configuration valid” with the value 165 (0xA5) the configuration of the node is set valid.



CANopen Device Monitor StarterKit 3.0.2 - Node 1 - CANopen Safety CSC01

File Edit View Connection Extras Windows Help

Node-Id 1 Set

1005 - COB-ID SYNC
1008 - manufacturer device
100C - Guard Time
100D - Life Time Factor
1014 - COB-ID Emergency
1017 - Producer Heartbeat T
1018 - Identity Object
1300 - Global failsafe comm
1301 - SRDO 1 communica
000 - number of entries
001 - information directi
002 - refresh time / SCT
003 - SRDO validation ti
004 - transmission type
005 - COB-ID 1
006 - COB-ID 2
1302 - SRDO 2 communica
1303 - SRDO 3 communica
1304 - SRDO 4 communica
1381 - SRDO 1 mapping pa
1382 - SRDO 2 mapping pa
1383 - SRDO 3 mapping pa
1384 - SRDO 4 mapping pa
13FE - configuration valid
13FF - safety configuration c
1400 - 1. receive PDO parar
1401 - 2. receive PDO parar
1600 - 1. receive PDO mapp

Action | NMT | Description | PDO | Chart | Process | SRDO

Value Entry Frame

165

□ used for saving configuration
□ component of DCF file

Action Frame

Receive from Object
- Send to Object +

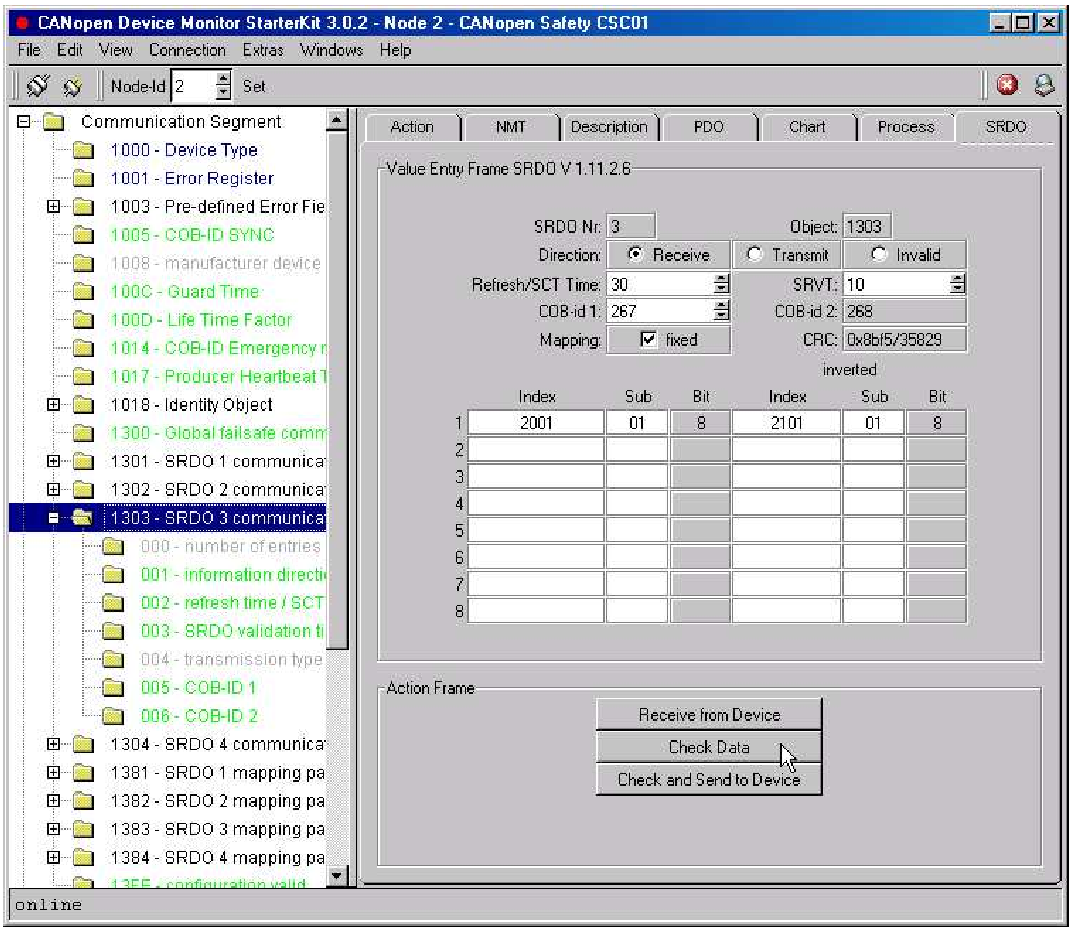
online

Figure 14: CDM set SRDO-Configuration valid

6.6 Configure the actuator

The actuator will be configured in the same manner like the sensor.

- Change the ID to 2.
- Read the first RSRDO from the Node and change the identifier.



SRDO Nr: 3 Object: 1303

Direction: ● Receive ○ Transmit ○ Invalid

Refresh/SCT Time: 30 SRVT: 10

COB-id 1: 267 COB-id 2: 268

Mapping: ☑ fixed CRC: 0x8bf5/35829

	Index	Sub	Bit	Index	Sub	Bit
				inverted		
1	2001	01	8	2101	01	8
2						
3						
4						
5						
6						
7						
8						

Figure 15: CDM change SRDO-Configuration (RSRDO 1)

- Transmit configuration to the node
- Set configuration at Index 0x13FE valid

6.7 Set both nodes in state Operational

Press the button *“Network Start”* in Tab *“NMT”*.

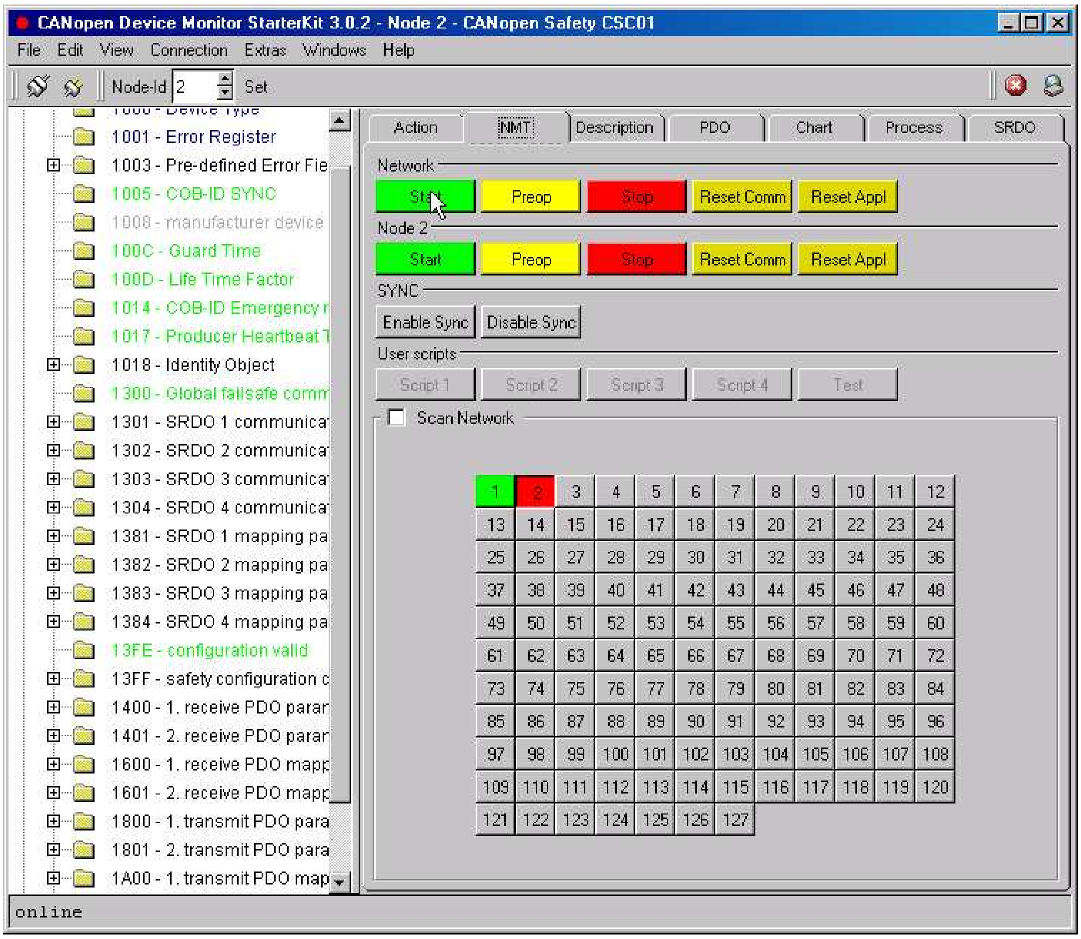
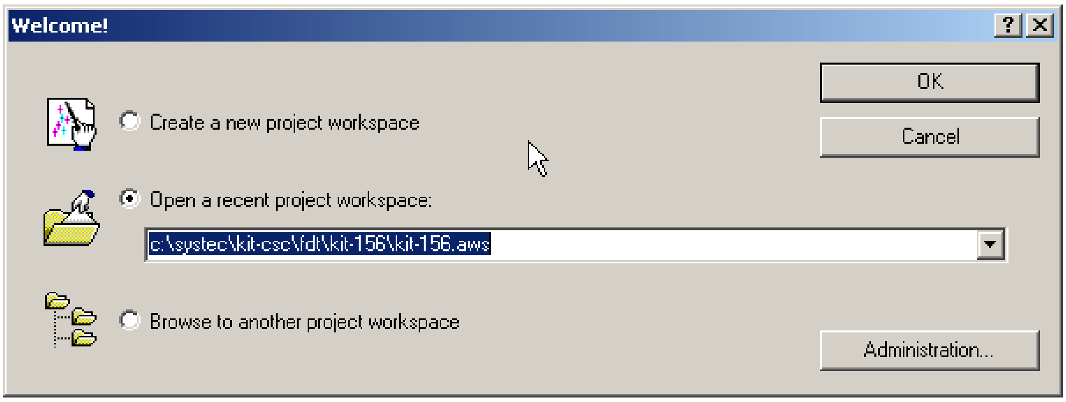


Figure 16: CDM set nodestate Operational

Now the SRDO-communication runs with the identifiers 0x10B and 0x10C.

7 Annex 1 Flash Development Toolkit

Start the FDT and open the FDT workspace KIT-156 located in c:\systec\kit-csc\fdt:



○ Create a new project workspace

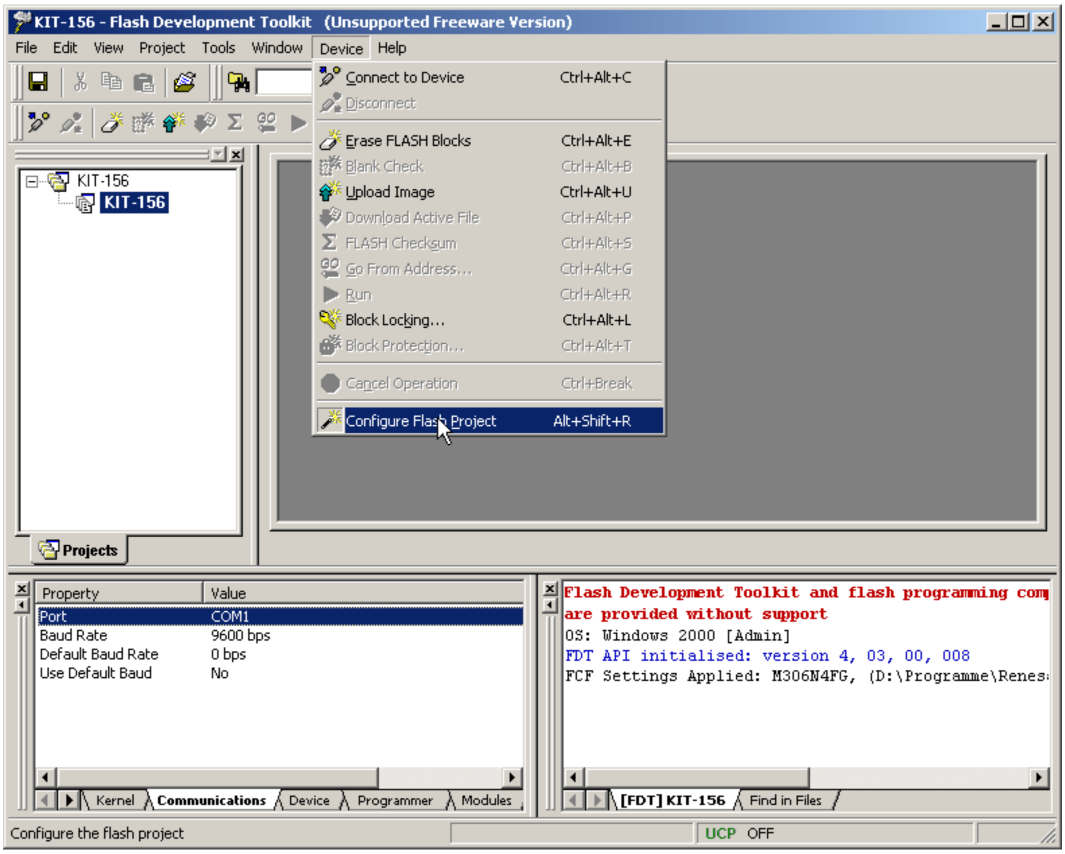
● Open a recent project workspace:

c:\systec\kit-csc\fdt\kit-156\kit-156.aws

○ Browse to another project workspace

Figure 17: Start of FDT workspace

In this workspace the CPU type is configured an the COM1 for serial connection is selected. This can be adapted in menu “Device” and there “Configure Flash Project”.


KIT-156 - Flash Development Toolkit (Unsupported Freeware Version)
File Edit View Project Tools Window Device Help
Connect to Device Ctrl+Alt+C
Disconnect
Erase FLASH Blocks Ctrl+Alt+E
Blank Check Ctrl+Alt+B
Upload Image Ctrl+Alt+U
Download Active File Ctrl+Alt+P
FLASH Checksum Ctrl+Alt+S
Go From Address... Ctrl+Alt+G
Run Ctrl+Alt+R
Block Locking... Ctrl+Alt+L
Block Protection... Ctrl+Alt+T
Cancel Operation Ctrl+Break
Configure Flash Project Alt+Shift+R
KIT-156
KIT-156
Projects
Kernel
Communications
Device
Programmer
Modules
[FDT] KIT-156
Find in Files
Configure the flash project
UCP OFF

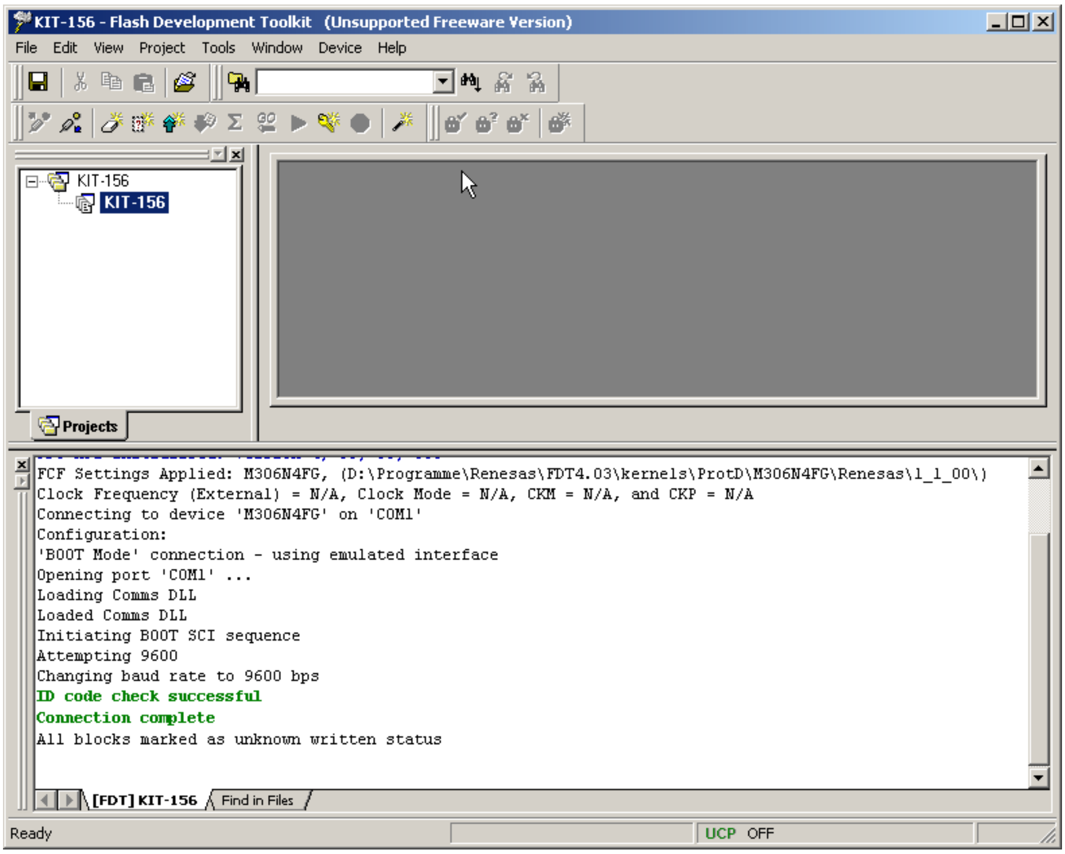
Property	Value
Port	COM1
Baud Rate	9600 bps
Default Baud Rate	0 bps
Use Default Baud	No

```
Flash Development Toolkit and flash programming com
are provided without support
OS: Windows 2000 [Admin]
FDT API initialised: version 4, 03, 00, 008
FCF Settings Applied: M306N4FG, (D:\Programme\Renes
```

Figure 18: Configuration of FDT

Establishing Connection to the Hardware

Click in the menu "Device" on “Connect to Device” to establish a connection to the target hardware. The following window will appear:

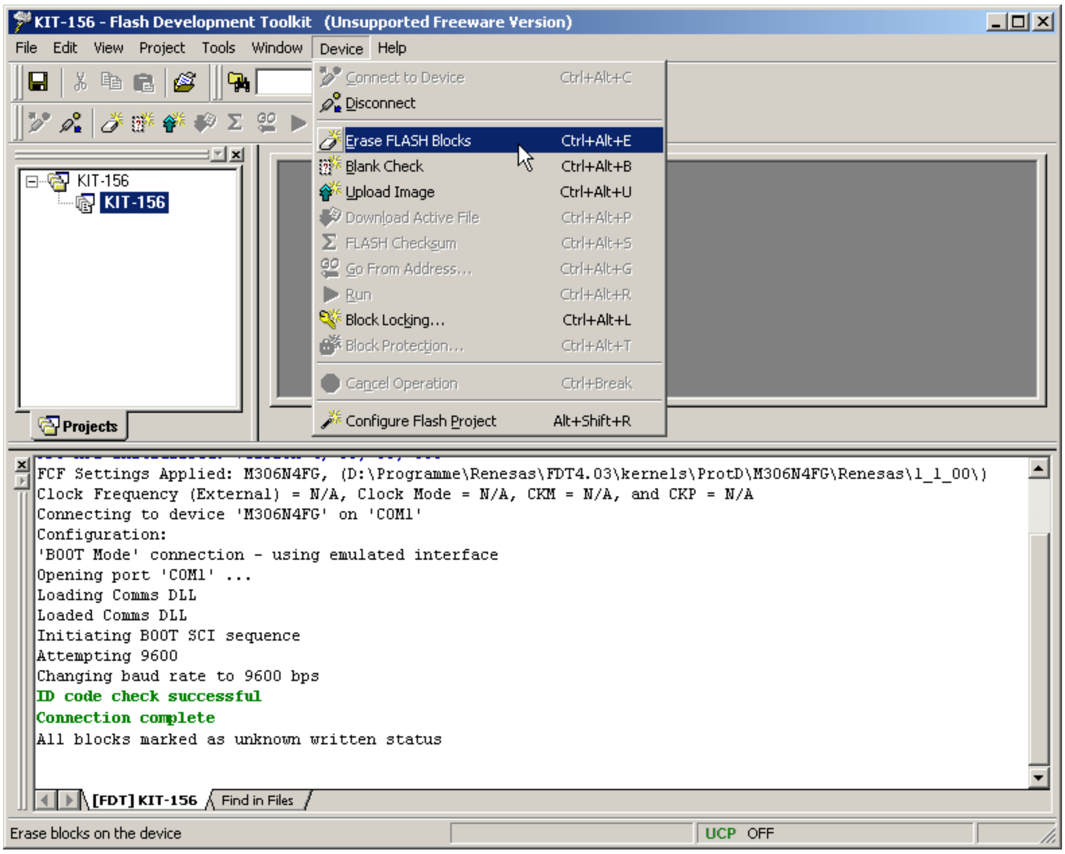

KIT-156 - Flash Development Toolkit (Unsupported Freeware Version)
File Edit View Project Tools Window Device Help
KIT-156
KIT-156
Projects
[FDT] KIT-156
Find in Files
Ready
UCP OFF

```
FCF Settings Applied: M306N4FG, (D:\Programme\Renesas\FDT4.03\kernels\ProtD\M306N4FG\Renesas\1_1_00\)
Clock Frequency (External) = N/A, Clock Mode = N/A, CKM = N/A, and CKP = N/A
Connecting to device 'M306N4FG' on 'COM1'
Configuration:
'BOOT Mode' connection - using emulated interface
Opening port 'COM1' ...
Loading Comms DLL
Loaded Comms DLL
Initiating BOOT SCI sequence
Attempting 9600
Changing baud rate to 9600 bps
ID code check successful
Connection complete
All blocks marked as unknown written status
```

Figure 19: Establishing a Connection to the Hardware

Erasing the Flash Block

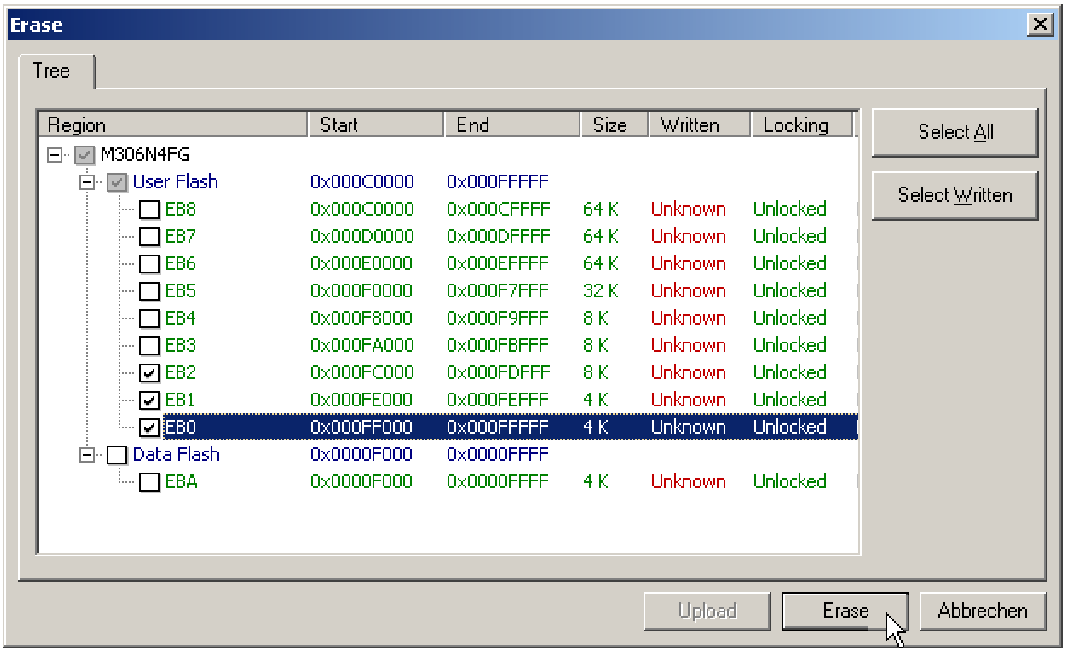
To erase the flash area 0x0FC000 – 0x0FFFFF choose in menu “Device” the entry “Erase Flash Blocks”


KIT-156 - Flash Development Toolkit (Unsupported Freeware Version)
File
Edit
View
Project
Tools
Window
Device
Help
Connect to Device Ctrl+Alt+C
Disconnect
Erase FLASH Blocks Ctrl+Alt+E
Blank Check Ctrl+Alt+B
Upload Image Ctrl+Alt+U
Download Active File Ctrl+Alt+P
FLASH Checksum Ctrl+Alt+S
Go From Address... Ctrl+Alt+G
Run Ctrl+Alt+R
Block Locking... Ctrl+Alt+L
Block Protection... Ctrl+Alt+T
Cancel Operation Ctrl+Break
Configure Flash Project Alt+Shift+R
KIT-156
KIT-156
Projects
[FDT] KIT-156
Find in Files
Erase blocks on the device
UCP OFF

```
FCF Settings Applied: M306N4FG, (D:\Programme\Renesas\FDT4.03\kernels\ProtD\M306N4FG\Renesas\1_1_00\)
Clock Frequency (External) = N/A, Clock Mode = N/A, CKM = N/A, and CKP = N/A
Connecting to device 'M306N4FG' on 'COM1'
Configuration:
'BOOT Mode' connection - using emulated interface
Opening port 'COM1' ...
Loading Comms DLL
Loaded Comms DLL
Initiating BOOT SCI sequence
Attempting 9600
Changing baud rate to 9600 bps
ID code check successful
Connection complete
All blocks marked as unknown written status
```

Figure 20: Erase Flash Blocks Menu

Now select the sectors, what should be erased and click the the “Erase” button.



Region	Start	End	Size	Written	Locking
☑ M306N4FG					
☑ User Flash	0x000C0000	0x000FFFFF			
□ EB8	0x000C0000	0x000CFFFF	64 K	Unknown	Unlocked
□ EB7	0x000D0000	0x000DFFFF	64 K	Unknown	Unlocked
□ EB6	0x000E0000	0x000EFFFF	64 K	Unknown	Unlocked
□ EB5	0x000F0000	0x000F7FFF	32 K	Unknown	Unlocked
□ EB4	0x000F8000	0x000F9FFF	8 K	Unknown	Unlocked
□ EB3	0x000FA000	0x000FBFFF	8 K	Unknown	Unlocked
☑ EB2	0x000FC000	0x000FDFFF	8 K	Unknown	Unlocked
☑ EB1	0x000FE000	0x000FEFFF	4 K	Unknown	Unlocked
☑ EB0	0x000FF000	0x000FFFFF	4 K	Unknown	Unlocked
□ Data Flash	0x0000F000	0x0000FFFF			
□ EBA	0x0000F000	0x0000FFFF	4 K	Unknown	Unlocked

Figure 21: Erasing flash blocks

Document:	**Development Kit CANopen Safety Chip02**
Document number:	**L-1228e_1, May 2009**

How would you improve this manual?

____

____

____

Did you find any mistakes in this manual? page

____

____

____

Submitted by:

Customer number: ______________________________

Name: ______________________________

Company: ______________________________

Address: ______________________________

____

Return to:

SYS TEC electronic GmbH
August-Bebel-Str. 29
D-07973 Greiz, Germany
Fax : +49 (0) 3661 63248