CANopen ChipF40

System Manual

Edition June 2012

System House for Distributed Automation

This side was left blank intentionally.

In this manual are descriptions for copyrighted products which are not explicitly indicated as such. The absence of the trademark (©) symbol 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.

Contact	**Direct**	**Your local distributor**
Address:	SYS TEC electronic GmbH August-Bebel-Str. 29 D-07973 Greiz GERMANY	Please find a list of our distributors under http://www.systec-electronic.com/distributors
Ordering Information:	+49 (3661) 6279-0 info@systec-electronic.com	
Technical Support:	+49 (3661) 6279-0 support@systec-electronic.com	
Fax:	+49 (3661) 62 79 99	
Web Site:	http://www.systec-electronic.com	

9th Edition June 2012

This side was left blank intentionally.

Table of Content

This side was left blank intentionally.

Index of Figures

This side was left blank intentionally.

Index of Tables

1 Preface

This manual describes only the functions of the CANopen ChipF40.

In this manual low active signals are denoted by a "/" in front of the signal name (i.e.: /RD). A "0" indicates a logic-zero or low-level signal, while a "1" represents a logic-one or high-level signal.

Declaration of the Electro Magnetic Conformity for the CANopen ChipF40

The CANopen ChipF40 (henceforth product) was designed for installation in electrical appliances or as dedicated Evaluation Boards (i.e.: for use as a test and prototype platform for hardware/software development) in laboratory environments.

Note:
SYS TEC products lacking protective enclosures are subject to damage by Electro Static Discharge (ESD) and, hence, may only be unpacked, handled or operated in environments in which sufficient precautionary measures have been taken in respect to ESD dangers. It is also necessary that only appropriately trained personnel (such as electricians, technicians and engineers) handle and/or operate these products. Moreover, SYS TEC products should not be operated without protection circuitry if connections to the product's pin header rows are longer than 3 m.

SYS TEC products fulfill the norms of the European Union’s Directive for Electro Magnetic Conformity only in accordance to the descriptions and rules of usage indicated in this manual (particularly in respect to the pin header row connectors, power connector and serial interface to a host-PC).

Implementation of SYS TEC products into target devices, as well as user modifications and extensions of SYS TEC products, is subject to renewed establishment of conformity to, and certification of, Electro Magnetic Directives. Users should ensure conformance following any

modifications to the products as well as implementation of the products into target systems.

The CANopen ChipF40 is one of a series of SYS TEC DIPmodules that can be fitted with different controllers and, hence, offers various functions and configurations. SYS TEC supports all common 8- and 16-bit controllers in two ways:

(1) as the basis for Development Kits in which user-designed hardware can be implemented on a wrap-field around the controller and

(2) as insert-ready, fully functional ECUcore modules which can be directly embedded into the user's peripheral hardware design.

SYS TEC's microcontroller modules allow engineers to shorten development horizons, reduce design costs and speed project concepts from design to market.

2 Introduction to the CANopen ChipF40

The CANopen ChipF40 is available in various firmware-versions:

MM-217-Y	CANopen ChipF40 standard version
MM-217-V3Y	CANopen ChipF40 V3, compatible to CANopen Chip164.
3301002	CANopen ChipF40, 63 Node-ID's adjustable via DIP-switch, default I/O configuration 1, adjustable via CANopen SDO only.

The differences and peculiarities of the both versions are described later in this document.

The CANopen ChipF40 is a tiny yet highly cost-effective Single Board I/O device. In the size of a 40-pin DIP device, the module is designed for use as core component in a customer application design. The CANopen ChipF40 features, besides the implemented standard CANopen firmware, digital inputs and outputs as well as analog input channels and PWM outputs. Using the various on-board configuration options, the module is adaptable to different applications.

All applicable controller signals extend to standard-width (2.54 mm.) pin header rows aligning two edges of the board, making the features and functionality of the Chip readily available to the user. To achieve the compact form factor of a 40-pin DIP device, small package types of the different components are used. The CANopen ChipF40 operates with a single 5 V supply voltage according to the specifications of standard TTL devices.

The firmware implemented in the CANopen ChipF40 has the complete functionality of a CANopen Slave device and has been certified by CiA (CAN in Automation e.V.). The present version of the CANopen-Chip supports 11-Bit identifier (CAN 2.0B passive). The firmware supports the standard Device Profile according to **CiA 401** and the Communication Profile according to **CiA 301 V4.01**. The CANopen ChipF40 provides various on-board configuration options for both the CANopen network parameters and the selection of

input und output characteristics (number and type of I/O's). The serial EEPROM device on the CANopen ChipF40 stores the configuration data of the CANopen Slave during runtime. This provides the advantage that, in the event of a temporary loss of the power supply, configuration data are still valid in the EEPROM.

2.1 Features

- single board CANopen I/O-node in 40-pin DIP dimensions (24x56mm) populated with Fujitsu MB90F352 16-bit microcontroller
- controller signals and ports extending to standard-width (2.54 mm.) pin rows lining the edges of the board
- CAN signals (CANH, CANL) in CAN 2.0B passive mode
- reference voltage pins for the on-chip A/D-converter
- EEPROM for storage of CANopen configuration parameters
- on-board PCA82C251 CAN transceiver, supporting up to 100 nodes on one CAN bus
- CAN signals also available for connection to an external, optically isolated CAN transceiver
- 8-position DIP-switch enables selection of:
  Node-ID (4-bit), baud rate (2-bit) and I/O configuration (2-bit),
  Node-ID (6-bit) and baud rate (2-bit) by version 3301002

3 Hardware Overview

3.1 Pin Layout

Please note that all module connections are not to exceed their expressed maximum voltage or current. Maximum signal input values are indicated in the corresponding controller User's Manual/Data Sheets located on the SYS TEC Products CD. As damage from improper connections varies according to use and application, it is the user's responsibility to take appropriate safety measures to ensure that the module connections are protected from overloading through connected peripherals. All controller signal extend to the pin header connector without any additional protection circuitry. External circuitry in which the CANopen ChipF40 is implemented must follow the guidelines for EMC conformance. As *Figure 1* indicates, all controller signals extend to standard-width (2.54 mm / 0.10 in.) pin rows lining two sides the board (referred to as DIPmodul-connector). This allows the CANopen ChipF40 to be plugged into any target application like a "big chip".

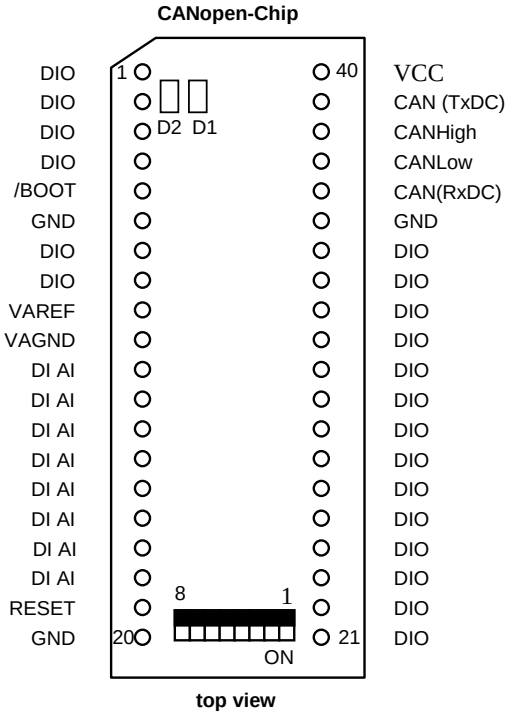


Figure 1: Pin Layout

A more detailed description of pin signals and functions is available in *Table 1*.

3.2 Pin Description Of The Board

Pin Number	Function	I/O	Description
1, 2	P1.2, P1.3	I/O	Port Pin P1.2, P1.3 of the microcontroller
3, 4	P4.4, P4.5	I/O	Port Pin P4.4, P4.5 of the microcontroller
5	/BOOT	I	/BOOT = 0 & RESIN = 1-0-Edge → activate Boot-Mode. Is used for firmware update over serial line.
6	GND	-	Ground 0V
7, 8	P2.0, P2.1	I/O	Port Pin P2.0, P2.1 of the microcontroller
9	VAREF	-	Reference voltage input of the on-chip A/D converter
10	VAGND	-	Analog Ground
11, 12, 13, 14, 15, 16, 17, 18	P6.0, P6.1, P6.2, P6.3, P6.4, P6.5, P6.6, P6.7	I	Port Pin P6.0-P6.7 of the microcontroller
19	RESIN	I	Reset input of the module, 0 - 1 transition triggers the RESET signal
20	GND	-	Ground 0V
21, 22	P5.0, P5.1	I/O	Port Pin P5.0, P5.1 of the microcontroller
23, 24, 25, 26, 27, 28, 29, 30	P3.0, P3.1, P3.2, P3.3, P3.4, P3.5, P3.6, P3.7	I/O	Port Pin P3.1-P3.7 of the microcontroller
31, 32	P5.2, P5.3	I/O	Port Pin P5.2, P5.3 of the microcontroller
33, 34	P2.2, P2.3	I/O	Port Pin P2.2, P2.3 of the microcontroller
35	GND	-	Ground 0 V
36	RxDC	I	Receive line of the on-chip CAN controller
37	CANL	I/O	CANL in-/output of the CAN transceiver
38	CANH	I/O	CANH in-/output of the CAN transceiver
39	TxDC	O	Send line of the on-chip CAN controller
40	VCC	-	Power supply +5 V =

Table 1: Pinout of the DIPmodul-connector

The assignment between applicable I/O lines and their specific function is provided in *Table 7*.

The use of an external CAN driver or an external galvanic isolation of CAN interface (use of signals RxDC and TxDC instaed of CANL and CANH) demands the disassembling of resistors on J1 of the module. A simulaneously use of onboard CAN driver and external CAN driver is not possible.

3.3 Board Configuration

3.3.1 DIP-Switch

An 8-position DIP-switch is located on the topside of the CANopen ChipF40. Four of these switches enable configuration of the Node-ID for CANopen, two set the CAN baud rate CAN bus, and the remaining two switches are used to enable I/O configuration.

Additional configurations are possible within CANopen (*refer to section 3.5*).

Figure 2 shows the DIP-switch pinout and functions. The switch position "OFF" corresponds to logic-zero or low-level signals while "ON" represents high level or logic-one. See *Figure* 1 for location and switch positions.

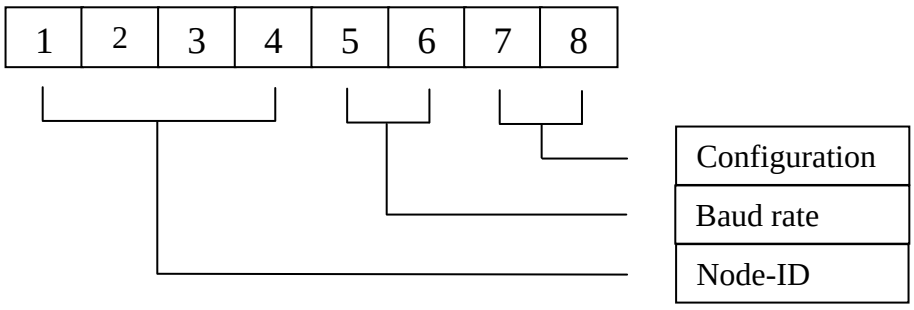


Figure 2: DIP-switch Pinout and Functions

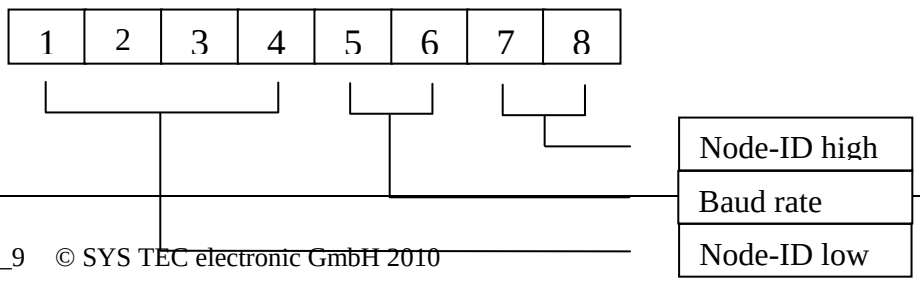


Figure 3: DIP-switch Pinout and Functions, version 3301002 only

3.3.2 CAN Transceiver

The firmware operates with 11-bit identifier (Full CAN 2.0B passive). The selection of the CAN transceiver is made by selection of the corresponding pins of the CANopen ChipF40. Pins 37 and 38 for use of the on-board or pins 36 and 39 for use of an external optically isolated CAN transceiver device.

3.4 Reset

A 1µF capacitor is connected to the microcontroller‘s RESET input. This enables automatic release of a power-on reset. The capacitor is charged via a 50kOhm resistor when power is turned on and holds the RESET input at a low level for a duration of approximately 50 milliseconds

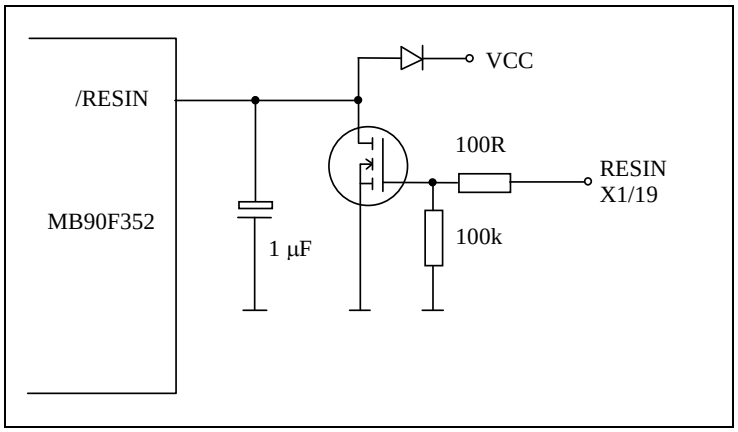


Figure 4: structure of /RESIN Line

3.5 Configuration of Communication Parameters

Node Address Configuration via DIP-switch :
The CANopen ChipF40 is configured for a basic Node-ID of 40hex. An additional, device-specific offset can be added to this base address and configured with the DIP-switch S1 DIP1 to DIP4. The resulting Node-ID is then calculated as follows:

$$\text{Node-ID} = 40H + DIP1*2^0 + DIP2*2^1 + DIP3*2^2 + DIP4*2^3$$

with DIPx = 1 if the corresponding switch is closed (position ON) and DIPx=0 if the switch is open (position OFF).

Example:
If both DIP-switches DIP1 and DIP4 are closed the following Node-ID will be configured:
$\text{Node-ID} = 40H + 1*2^0 + 0*2^1 + 0*2^2 + 1*2^3$
Node-ID = 40H + 1*1 + 1*8
Node-ID = 40H + 1 + 8
Node-ID = 49H

Note:
The configuration 0FH on DIP1...DIP4 is reserved and may not be used in an application. This setting resets the CANopen ChipF40 into its factory default state.

The following table gives an overview of the possible configurations:

DIP1	DIP2	DIP3	DIP4	Node -ID
OFF	OFF	OFF	OFF	0x40 (default)
ON	OFF	OFF	OFF	0x41
OFF	ON	OFF	OFF	0x42
ON	ON	OFF	OFF	0x43
OFF	OFF	ON	OFF	0x44
ON	OFF	ON	OFF	0x45
OFF	ON	ON	OFF	0x46
ON	ON	ON	OFF	0x47
OFF	OFF	OFF	ON	0x48

ON	OFF	OFF	ON	0x49
OFF	ON	OFF	ON	0x4A
ON	ON	OFF	ON	0x4B
OFF	OFF	ON	ON	0x4C
ON	OFF	ON	ON	0x4D
OFF	ON	ON	ON	0x4E
ON	*ON*	*ON*	*ON*	*Reserved*

Table 2: Configuration of Node-ID

Node Address Configuration via DIP-switch, version 3301002:
The CANopen ChipF40, version 33010002, is configured for a basic Node-ID of 40hex. An additional, device-specific offset can be added to this base address and configured with the DIP-switch S1 DIP1 to DIP4 and DIP7 to DIP8. The resulting Node-ID is then calculated as follows:

$$\text{Node-ID} = 40H + DIP1*2^0 + DIP2*2^1 + DIP3*2^2 + DIP4*2^3 + DIP7*2^4 + DIP8*2^5$$

with DIPx = 1 if the corresponding switch is closed (position ON) and DIPx=0 if the switch is open (position OFF). Therefore you could select different Node-ID's from 40H (64dez) to 07EH (126dez).

Note:
The Node-ID 7FH (127dez) is reserved and may not be used in an application. This setting resets the CANopen ChipF40 into its factory default state.

Node- Address Configuration via CANopen
Layer Setting Services (LSS) implemented in the CANopen protocol can be used to freely assign a Node-ID on the CANopen ChipF40. This address can be in the range from 1 to 127 decimal (01h...7Fh). After such LSS configuration the DIP-switch settings for the Node-ID are no longer valid.

Note:
Changing the Node-ID via CANopen LSS also results in resetting the Object Dictionary (OD) with default values.

Bit Rate Configuration via DIP-switch:
Switches DIP5 and DIP6 of DIP-switch can be used to configure one of 4 pre-defined CAN bit rates available on the CANopen ChipF40. The following bit rates are available:

DIP5	DIP6	Bit rates kBit/s
ON	OFF	20
OFF	OFF	125
OFF	ON	500
ON	ON	1000

Table 3: Configuration of CAN Bit Rate

Bit Rate Configuration via CANopen
Layer Setting Services (LSS) implemented in the CANopen protocol can be used to freely assign a desired bit rate on the CANopen ChipF40. After such LSS configuration, the DIP-switch settings for the bit rate are no longer valid.

Index in LSS Timing Table	Bit rate kBit/s
0	1000
1	800
2	500
3	250
4	125
5	100
6	50
7	20
8	10

Table 4: Configuration of CAN Bit Rate over LSS

I/O Configuration via DIP-switch:
The CANopen ChipF40 firmware features four pre-defined I/O configurations that can be configured using switches DIP7 and DIP8 of DIP-switch. The following I/O configurations are available:

DIP7	DIP8	I/O Configuration
OFF	OFF	I/O Configuration 0
ON	OFF	I/O Configuration 1
OFF	ON	I/O Configuration 2
ON	ON	I/O Configuration 3

Table 5: I/O Configuration

The I/O configuration of the CANopen ChipF40, version 3301002, cannot be set via the DIP-switch. The I/O configuration is set to configuration 1 by default. To change the I/O configuration a SDO access to the object dictionary index 2000H is necessary.

I/O Configuration via CANopen
The desired I/O configuration on the CANopen ChipF40 can also be selected via an entry in the Object Dictionary (OD). The manufacturer-specific OD entry 0x2000 is provided for this purpose. The selection is done by writing the applicable I/O configuration number to this object. For example, when writing the value 5 to index 0x2000 the resulting I/O configuration will be configuration number 5. After such OD configuration, the DIP-switch settings for the I/O configuration are no longer valid.

Note:
Changing the Node-ID via CANopen OD entry also results in resetting the Object Dictionary (OD) with default values.

3.6 Default Configuration

At the time of delivery all DIP-switches on the CANopen ChipF40 are open. This results in the following factory default settings:

- Node-ID = 40hex
- bit rate = 125 kBit/s
- configuration = I/O configuration 0

3.7 Pin Assignments for Selected I/O Configurations F40

Configuration	Digital Inputs	Digital Outputs	Analog Inputs	PWM Outputs
0	14	8	2	4
1	8	8	8	4
2	16	8	-	4
3	8	16	-	4
4	16	-	8	4
5	24	-	-	4
6	16	4	4	4

Table 6: Number of I/Os Depending on the Selected Configuration F40

Pin#	Config 0	Config 1	Config 2	Config 3	Config 4	Config 5	Config 6
1	DO 4	DO 0	DO 0	DO 0	DI 12	DI 20	DI 0
2	DO 5	DO 1	DO 1	DO 1	DI 13	DI 21	DI 1
3	DO 6	DO 2	DO 2	DO 2	DI 14	DI 22	DO 0
4	DO 7	DO 3	DO 3	DO 3	DI 15	DI 23	DO 1
5							
6							
7	PWM 0	PWM 0	PWM 0	PWM 0	PWM 0	PWM 0	PWM 0
8	PWM 1	PWM 1	PWM 1	PWM 1	PWM 1	PWM 1	PWM 1
9							
10							
11	AI 0	AI 0	DI 0	DI 0	AI 0	DI 0	AI 0
12	AI 1	AI 1	DI 1	DI 1	AI 1	DI 1	AI 1
13	DI 0	AI 2	DI 2	DI 2	AI 2	DI 2	AI 2
14	DI 1	AI 3	DI 3	DI 3	AI 3	DI 3	AI 3
15	DI 2	AI 4	DI 4	DI 4	AI 4	DI 4	DI 2
16	DI 3	AI 5	DI 5	DI 5	AI 5	DI 5	DI 3
17	DI 4	AI 6	DI 6	DI 6	AI 6	DI 6	DI 4
18	DI 13	AI 7	DI 7	DI 7	AI 7	DI 7	DI 5
19							
20							
21	DI 12	DO 4	DO 4	DO 4	DI 8	DI 16	DI 14
22	DI 11	DO 5	DO 5	DO 5	DI 9	DI 17	DI 15
23	DI 10	DI 0	DI 8	DO 8	DI 0	DI 8	DI 6
24	DI 9	DI 1	DI 9	DO 9	DI 1	DI 9	DI 7
25	DI 8	DI 2	DI 10	DO 10	DI 2	DI 10	DI 8
26	DI 7	DI 3	DI 11	DO 11	DI 3	DI 11	DI 9
27	DO 3	DI 4	DI 12	DO 12	DI 4	DI 12	DI 10
28	DO 2	DI 5	DI 13	DO 13	DI 5	DI 13	DI 11
29	DO 1	DI 6	DI 14	DO 14	DI 6	DI 14	DI 12
30	DO 0	DI 7	DI 15	DO 15	DI 7	DI 15	DI 13
31	DI 6	DO 6	DO 6	DO 6	DI 10	DI 18	DO 2
32	DI 5	DO 7	DO 7	DO 7	DI 11	DI 19	DO 3
33	PWM 2	PWM 2	PWM 2	PWM 2	PWM 2	PWM 2	PWM 2
34	PWM 3	PWM 3	PWM 3	PWM 3	PWM 3	PWM 3	PWM 3
35							
36							
37							
38							
39							
40							

Table 7: Input/Output Configuration and I/O Pin Assignment F40

Description of table entries:

DI - Digital Input
DO - Digital Output
AI - Analog Input
PWM - PWM Output

Note:
Configuration 0 is set up at time of delivery. Whenever changing the I/O configuration user must ensure that the circuitry connected to the applicable I/O pins meets the requirements of the specific I/O signal type. In appropriate signal connection could damage or destroy the CANopen ChipF40. We recommend to disconnect the I/O pins when changing the I/O configuration.

3.8 Pin Assignments for Selected I/O Configurations F40 V3

Configuration	Digital Inputs	Digital Outputs	Analog Inputs
0	18	8	2
1	12	8	8
2	20	8	-
3	12	16	-
4	20	-	8
5	28	-	-
6	20	4	4

Table 8: Number of I/Os Depending on the Selected Configuration F40 V3

Pin#	Config. 0	Config. 1	Config. 2	Config. 3	Config. 4	Config. 5	Config. 6
1	DO 4	DO 0	DO 0	DO 0	DI 12	DI 20	DI 0
2	DO 5	DO 1	DO 1	DO 1	DI 13	DI 21	DI 1
3	DO 6	DO 2	DO 2	DO 2	DI 14	DI 22	DO 0
4	DO 7	DO 3	DO 3	DO 3	DI 15	DI 23	DO 1
5							
6							
7	DI 14	DI 8	DI 16	DI 8	DI 16	DI 24	DI 16
8	DI 15	DI 9	DI 17	DI 9	DI 17	DI 25	DI 17
9							
10							
11	AI 0	AI 0	DI 0	DI 0	AI 0	DI 0	AI 0
12	AI 1	AI 1	DI 1	DI 1	AI 1	DI 1	AI 1
13	DI 0	AI 2	DI 2	DI 2	AI 2	DI 2	AI 2
14	DI 1	AI 3	DI 3	DI 3	AI 3	DI 3	AI 3
15	DI 2	AI 4	DI 4	DI 4	AI 4	DI 4	DI 2
16	DI 3	AI 5	DI 5	DI 5	AI 5	DI 5	DI 3
17	DI 4	AI 6	DI 6	DI 6	AI 6	DI 6	DI 4
18	DI 13	AI 7	DI 7	DI 7	AI 7	DI 7	DI 5
19							
20							
21	DI 12	DO 4	DO 4	DO 4	DI 8	DI 16	DI 14
22	DI 11	DO 5	DO 5	DO 5	DI 9	DI 17	DI 15
23	DI 10	DI 0	DI 8	DO 8	DI 0	DI 8	DI 6
24	DI 9	DI 1	DI 9	DO 9	DI 1	DI 9	DI 7
25	DI 8	DI 2	DI 10	DO 10	DI 2	DI 10	DI 8
26	DI 7	DI 3	DI 11	DO 11	DI 3	DI 11	DI 9
27	DO 3	DI 4	DI 12	DO 12	DI 4	DI 12	DI 10
28	DO 2	DI 5	DI 13	DO 13	DI 5	DI 13	DI 11
29	DO 1	DI 6	DI 14	DO 14	DI 6	DI 14	DI 12
30	DO 0	DI 7	DI 15	DO 15	DI 7	DI 15	DI 13
31	DI 6	DO 6	DO 6	DO 6	DI 10	DI 18	DO 2
32	DI 5	DO 7	DO 7	DO 7	DI 11	DI 19	DO 3
33	DI 16	DI 10	DI 18	DI 10	DI 18	DI 26	DI 18
34	DI 17	DI 11	DI 19	DI 11	DI 19	DI 27	DI 19
35							
36							
37							
38							
39							
40							

Table 9: Input/Output Configuration and I/O Pin Assignment F40 V3

Description of table entries:

DI - *Digital Input*
DO - *Digital Output*
AI - *Analog Input*

Note:
Configuration 0 is set up at time of delivery. Whenever changing the I/O configuration user must ensure that the circuitry connected to the applicable I/O pins meets the requirements of the specific I/O signal type. In appropriate signal connection could damage or destroy the CANopen ChipF40. We recommend to disconnect the I/O pins when changing the I/O configuration.

3.9 Technical Data

Electrical Parameters

Operating Voltage:	5V DC ± 10%
Power consumption :	typ. 65 mA, 24 MHz CPU clock, 25°C; max. 140mA depends on circuitry of I/Os and CAN
Clock generation :	24 MHz CPU clock achieved through multiplication the external 4 MHz crystal frequency

Output voltage of the I/O lines:

low level:	< 0.4 V
high level:	> 4.5 V

Output current of the I/O lines:

low level 1 pin maximum current:	15mA
low level 1 pin average current:	4mA
low level all pins maximum current:	100mA
low level all pins average current:	50mA
high level 1 pin maximum current:	-15mA
high level 1 pin average current:	-4mA
high level all pins maximum current:	-100mA
high level all pins average current:	-50mA

Input voltage of the I/O lines:

low level:	>VSS – 0.3V
high level:	> 0.8V * VCC; <VCC + 0.3V

Analog inputs:

Input voltage:	VAGND .... VREF
Resolution:	10-bit
Input capacity:	10.7 pF
Reference voltage:	VAGND + 2.7V ... VCC

On-board CAN transceiver:

Maximum bit rate:	1000 kBit/s
Number of nodes:	<100

Environmental Conditions

Operational temperature:	-40°C to +85°C
Storage temperature:	-40°C

Mechanical Specifications

Dimensions:	24.0 mm * 58.7 mm., ± 0.3 mm
Weight:	approximately 10.5 grams
Connector Type:	40-pin dual–inline IC socket (2.54 mm pitch), pin diameter 0.47 mm, contact length 3.2 mm

These specifications describe the standard configuration of the CANopen ChipF40 as of the printing of this manual.

4 Setting up the CANopen ChipF40

4.1 Power Supply

The CANopen ChipF40 requires a power supply of +5V DC ±10%. Power can be supplied via pins 6 (GND) and pin 40 (+5V), according to the standard for TTL-level devices. Additional there are pins 20 and 35 for GND connection. It is recommended to use a standard DIL-40 socket in order to integrate the CANopen ChipF40 into a target hardware environment.

4.2 CAN Interface

The CANopen ChipF40 is populated with the Fujitsu MB90F352 microcontroller, featuring Full 2.0B on-chip CAN, and an on-board CAN transceiver (P82C251). The following CAN signals are available at header pins aligning two edges of the board:

CAN_HIGH	Pin 38
CAN_LOW	Pin 37
CAN_GND	Pin 6, 20 or 35

The potentials CAN_GND and GND (general board-level Ground) are identical.

These CAN signals can be routed as follows to a DB-9 socket, according to the CiA 301 Communication Profile standard

CAN_HIGH	Pin 7
CAN_LOW	Pin 2
CAN_GND	Pin 6 and/or Pin 3

The CAN signals can be optically isolated from the CANopen ChipF40 using an external transceiver IC. The maximum transmission rate is 1MBit/s. The maximum CAN bus load should not exceed 50%.

This side was left blank intentionally.

5 QuickStart

This section describes basic start-up of the CANopen ChipF40. It assumes basic knowledge of CANopen networks. It also requires, that the CANopen ChipF40 is properly connected to the CAN bus and power is supplied to the CANopen ChipF40. Please *refer to sections 1* and *1* for basic description of CAN and CANopen.

5.1 Start-Up of the CANopen ChipF40

All values in the Object Dictionary (OD) are pre-configured with default-values. Hence, start-up configuration of the CANopen ChipF40 is not necessary. The CANopen ChipF40 supports the CANopen Minimum Boot-Up. Following reset and internal initialization, the board is in PRE-OPERATIONAL state (*refer to section 8.3 PRE-OPERATIONAL*). Upon receipt of a single message (*Start_Remote_Node*) the board switches to OPERATIONAL state (*refer to section 8.4 OPERATIONAL*).

11-bit CAN Identifier	2 Byte Data	
0	*0x01h*	*Node_ID*

The first data byte contains the Start command, while the second byte contains the node address (*Node_ID*). If you specify the value 00h then all nodes in the network (Broadcast) are started.

5.2 Shut-Down of the CANopen ChipF40

All node activity can be stopped by receipt of the *Enter_Pre_Operational_State* message. This message consists of the following:

11-bit CAN Identifier	2 Byte Status Data	
0	*0x80h*	*Node_ID*

The *Node-ID* identifies the node-addresses. *Node-ID = 00h* addresses all devices on a network (Broadcast).

When in “PRE-OPERATIONAL“ state, SDO-Transfer is still possible. All configuration parameters are then captured and “frozen” as they were in their most recent active state.

Note:
CANopen configuration tools, such as ProCANopen from Vector or CDM and CCM from Port always use SDO-Transfer when accessing a CANopen node. Due to this fact, these tools also work in “PRE-OPERATIONAL“ state.

5.3 CAN Message and Identifier

According to **CiA 301**, a specific CAN identifier is assigned to each CAN message containing process data (Process-Data-Object - PDO). The CAN identifier for input and output data is derived from the node address.

CAN Identifier (hex)	Data Type
180h + node address	1st Tx PDO
280h + node address	2nd Tx PDO
380h + node address	3rd Tx PDO
480h + node address	4th Tx PDO
200h + node address	1st Rx PDO
300h + node address	2nd Rx PDO
400h + node address	3rd Rx PDO (not F40 V3)
500h + node address	4th Rx PDO (not F40 V3)

Table 10: CAN ID for Different PDO Types

5.4 PDO Mapping for I/O’s

The PDO mapping of the available I/O's depends on the selected I/O configuration (refer to *Table 7*).

The CANopen ChipF40 also supports variable PDO mapping. This allows for free mapping of inputs to Tx PDOs and Rx PDOs to output lines. Such free mapping settings can be saved in the on-board EEPROM by writing to index 0x1010.

5.4.1 Default Mapping CANopen ChipF40

In the default mapping, the 4th Tx PDO and the 2nd Rx PDO and the 4th Rx PDO are invalid. This results in the following arrangement of I/Os and PDOs:

Config	1st Tx PDO	2nd Tx PDO	3rd Tx PDO	1st Rx PDO	3rd Rx PDO
0	DI 0...DI 7 DI 8...DI 13	AI 0...AI 1	invalid	DO 0...DO7	duty cycle PWM 0 ... PWM 3
1	DI 0...DI 7	AI 0...AI 3	AI 4...AI 7	DO 0...DO7	duty cycle PWM 0 ... PWM 3
2	DI 0...DI 7 DI 8...DI 15	invalid	invalid	DO 0...DO7	duty cycle PWM 0 ... PWM 3
3	DI 0...DI 7	invalid	invalid	DO 0...DO7 DO 8...DO15	duty cycle PWM 0 ... PWM 3
4	DI 0...DI 7 DI 8...DI 15	AI 0...AI 3	AI 4...AI 7	invalid	duty cycle PWM 0 ... PWM 3
5	DI 0...DI 7 DI 8...DI 15 DI 16...DI 23	invalid	invalid	invalid	duty cycle PWM 0 ... PWM 3
6	DI 0...DI 7 DI 8...DI 15	AI 0...AI 3	invalid	DO 0...DO3	duty cycle PWM 0 ... PWM 3

Table 11: PDO Mapping for I/O‘s F40

5.4.2 Default Mapping CANopen ChipF40 V3

In the default mapping, the 4th Tx PDO and the 2nd Rx PDO are invalid. This results in the following arrangement of I/Os and PDOs:

I/O Config	1st Tx PDO	2nd Tx PDO	3rd Rx PDO	1st Rx PDO
0	DI 0...DI 7 DI 8...DI 13	AI 0...AI 1	invalid	DO 0...DO7
1	DI 0...DI 7	AI 0...AI 3	AI 4...AI 7	DO 0...DO7
2	DI 0...DI 7 DI 8...DI 15	invalid	invalid	DO 0...DO7
3	DI 0...DI 7	invalid	invalid	DO 0...DO7 DO 8...DO15
4	DI 0...DI 7 DI 8...DI 15	AI 0...AI 3	AI 4...AI 7	invalid
5	DI 0...DI 7 DI 8...DI 15 DI 16...DI 23	invalid	invalid	invalid
6	DI 0...DI 7 DI 8...DI 15	AI 0...AI 3	invalid	DO 0...DO3

Table 12: PDO Mapping for I/O‘s F40 V3

5.5 Board Reset

Following each board reset, the CANopen ChipF40 transmits an Boot-up message without data content. Temporary suspension of CANopen ChipF40 activity and subsequent restart can be recognized without Nodeguarding *(refer to section 5.6 Node Guarding)*. The transmitter of this message will be detected by the CAN identifier.

11-bit CAN Identifier	1 Byte Data
700h+ Node_ID	*00h*

5.6 Node-Guarding

Nodeguarding and *Lifeguarding* functions monitor operation of the CANopen network. Distributed peripheral CAN devices are monitored via Nodeguarding, while the Lifeguarding function supervises the guarding Master. To realize Nodeguarding, the Master requests a cyclic status message from the slave nodes. This status request is initiated with a Remote frame message that contains only the status data. The RTR-Bit (Remote Transmit Request Bit) is set for this reason.

11-bit CAN Identifier	1 Byte Data
700h + Node_ID	*Node-Guarding*

Following transmission of the Remote-Frame message, the Slave-nodes responds with a status message consisting of 1 byte of service data.

11-bit CAN Identifier	1 Byte Data
700h + Node_ID	*X*

The data bytes within status message further contain a toggle bit that is supposed to change after each message. Should the status and toggle bits not correspond to the message pattern expected by the Master, or should no response to a message follow, the Master assumes a Slave malfunction. If the Master requests cyclic guard messages, a Slave node can recognize shut-down of the Master. This is the case if the Slave does not receive a message request from the Master within the

pre-configured *Life Time*. The Slave then assumes failure of the Master, sets its inputs into Error state, transmits an Emergency message and switches into *Pre-Operational* state (condition index [67FEH] or [1029H] = 0).

The *Life Time Factor* is configured within the Object [100D] and is multiplied by the *Guard Time* [100C]. This results in the *Life Time* of the "Nodeguarding Protocol". The time base of these cycles is 1 ms. The *Guard Time* specifies how much time must elapse between two *Node-Guarding* messages. The *Life Time Factor* indicates how many times the *Guard Time* can elapse before an error is generated.

Default Values:

Life Time Factor	0
Guard Time	0 ms.
Life Time	0 sec.

Example Values:

Life Time Factor	3
Guard Time	1000 ms.
Life Time	3 sec.

This side was left blank intentionally.

6 Controller Area Network – CAN

6.1 Communication with CANopen

The Controller Area Network (the CAN bus) is a serial data communications bus for real-time applications. CAN was originally developed by the German company Robert Bosch for use in the automotive industry. It is a two-wire bus system that provides a cost-effective communication bus alternative to expensive and cumbersome harness wiring. CAN operates at data rates of up to 1 Mbit per second and has excellent error detection and confinement capabilities. On account of its proven reliability and robustness, CAN is being used in many other automation and industrial applications. CAN is now an international standard and is documented in ISO 11898 (for high-speed applications) and ISO 11519 (for lower-speed applications) documents.

CANopen is a higher-layer network protocol based on the CAN serial bus system, specifically, it is a software-level protocol standard for industrial communication between automated devices. CANopen is authorized by the User and Manufacturers' Group "CAN in Automation e.V." (CiA) and adheres to ISO/OSI standards.

CANopen unleashes the full power of CAN by allowing direct peer to peer data exchange between nodes in an organized, hierarchical manner. The network management functions specified in CANopen simplify project design, implementation and diagnosis by providing standard mechanisms for network start-up and error management.

CANopen supports both cyclic and event-driven communication. This makes it possible to reduce the bus load to a minimum, while still maintaining extremely short reaction times. High communication performance can be achieved at relatively low baud rates, thus reducing EMC problems and minimizing cable costs. CANopen is the ideal networking system for all types of automated machinery. One of the distinguishing features of CANopen is its support for data

exchange at the supervisory control level as well as accommodating the integration of very small sensors and actuators on the same physical network. This avoids the unnecessary expense of gateways linking sensor/actuator bus systems with higher communication networks and makes CANopen particularly attractive to original equipment manufacturers.

CANopen Advantages

- Vendor-independent open-source structure
- Universal standards
- Supports inter-operability of different devices
- High speed real-time capability
- Modular - covers simple to complex devices
- User-friendly - wide variety of support tools available
- Real-Time-capable communication for process data without protocol overhead;
- a modular, configurable structure that can be tailored to the needs of the user and his or her networked application
- Interbus-S, Profibus and MMS oriented-profiles

CANopen Features

- Auto configuration of the network
- Easy access to all device parameters
- Device synchronization
- Cyclic and event-driven data transfer
- Synchronous reading or setting of inputs, outputs or parameters

In addition to its designation as a physical CAN layer standard, CANopen is a “layer-7 protocol” implementation of CAL and is defined by the CANopen Communications Profile in CiA 301. CAL, in turn, is based on an existing and proven protocol originally developed by Philips Medical Systems. CAL is an application-independent application layer that has been specified and is also maintained by the CAN in Automation (CiA) user group.

6.2 CAN Application Layer

The CAN Application Layer (CAL) supports various applications and the integration of CAN hardware from different vendors. A CAL implementation consists of four blocks, each of which can operate as network Master and Slave.

CAN Message Specification (CMS)
CMS defines the communication objects, such as multiplexed variables, Events and Domains.

- Variables: serve data exchange of basic messages
- Events: handle the activity of specifically defined events, such as switches and transmission of asynchronous messages
- Domains: support transmission of data packages larger than the maximum eight bytes of a standard CAN message

CMS further regulates the communication structure between the object targets.

Network Management (NMT)
NMT implements network management functions for NMT-Master and NMT-Slave. These functions support start-up and expansion of a network, as well as Error supervision (Lifeguarding) and prevention of bus overload.

Distributor (DBT)
DBT supports the use of CAN nodes from various vendors through its automatic assigning of message identifiers. The DBT-Master/Slave functions enable administration of a global data basis for communication objects (COBs) of varying priority classes.

Layer Management (LMT)
LMT assigns parameters to the lower layers of data communication, such as timing parameters of CAN nodes or management of a manufacturer code by node name designation.

6.3 CANopen – Open Industrial Communication

The following Special Interest and Working Groups have developed the CAL-based CANopen communication profile:

- SIG Distributed I/O
- SIG Motion Control

and the Working Group (WG)

- WG Higher Layer Protocols

The CiA 301 CANopen standard derived from the results of the ASPIC ESPRIT project. The communication profile describes in detail how data are exchanged over the CAN bus based on the functions provided by CAL. This data can be sorted into two main types:

- Process data
- Service data

Process data is real-time data generated by a networked device. This data is transmitted via a Process Data Object (PDO). The CANopen communication profile determines how a PDO functions within CAL communication objects, as well as which protocol is used for transmission of data. PDOs can be used simultaneously by multiple networked devices, hence enabling broadcast operations.

Service data are used to configure and establish parameters for networked devices. Service data directly communicate to the Object Dictionary of each device and are transmitted using Service Data Objects (SDO).

The CANopen communication profile also determines how these objects are connected and which CAL functions and services can be used. An SDO can only be used between two networked devices, typically a configuration Master and another device that is to be configured. The SDO-Transfer is also capable of confirmation of message receipt.

Each individual networked device provides several PDOs and SDOs. This enables configuration of multi-master networks, in addition to typical single Master / multiple Slave networks.

In addition to data classes, CANopen defines the communication classes that describe:

- Synchronized communication
- Event processing
- Communication initialization

CANopen also defines device profiles that describe the basic functions of networked devices. These device profiles consist of the following two primary components:

- Functional Description
- Operational Description

The ***Functional Description*** of a device is represented by functional blocks and data flows. Descriptive parameters are stored in the Object Dictionary. Each Object Dictionary has a pre-defined structure. Hence, parameters for networked devices of a certain type (for instance I/O modules or drives) are always located in the same place within an Object Dictionary. Parameters can be classified as mandatory, optional and manufacturer-specific.

The ***Operational Description*** of a device is described by state flow diagram (*refer to Figure 5 in Section 7.9*).

Device Profiles are standardized for:

- Generic I/O Modules CiA 401
  digital I/O's
  analog I/O's
- Drives and Motion Control CiA 402

Servo drivers,
Step motors and
Frequency transformers

- Measurement Devices and Closed Loop Controllers CiA 405
- IEC61131-3 Programmable Devices CiA 405
- Encoder CiA 406
- Inclinometer CiA 410

Please refer to the CAN in Automation homepage www.can-cia.org for up-to-date information of available device profiles. All device profiles correspond to the DRIVECOM Profile with CAN-specific modifications to enable multi-master capability.

Software for CANopen Slave functions is based on services for data exchange and network management as defined in CAL standards. In particular only certain parts of CAL have been implemented in CANopen, such as standards for Multiplexed-Domain-Transfer for SDOs and Stored Event-Transfer for transmission of PDOs.

7 CANopen Communication

7.1 CANopen Fundamentals

Open fieldbus systems enable design of distributed network systems by connecting components from multiple vendors while minimizing the effort required for interfacing. To achieve an open networking system, it is necessary to standardize the various layers of communication used.

CANopen uses the international CAN standard, ISO 11898 as the basis for communication. This standard covers the lower two layers of communication specified by the OSI model. Based on this, the CANopen profile family specifies standardized communication mechanisms and device functionality for CAN-based systems. The profile family, which is available and maintained by CAN in Automation e.V. (CiA) consists of the Application layer and communication profile (CiA 301), various frameworks and recommendations (CiA 30x) and various device profiles (CiA 40x).

The network management functions specified in CANopen simplify project design, implementation and diagnosis by providing standard mechanisms for network start-up and error management.

CANopen is the ideal networking system for all types of automated machinery. One of the distinguishing features of CANopen is its support for data exchange at the supervisory control level as well as accommodating the integration of very small sensors and actuators on the same physical network. This avoids the unnecessary expense of gateways linking sensor/actuator bus systems with higher communication networks and makes CANopen particularly attractive to original equipment manufacturers.

7.2 CANopen Device Profiles

CANopen profiles are defined for communication in CiA 301, for I/O Modules in CiA 401, for Drives and Motion Control in CiA 402 and for Encoder in CiA 406. Other profiles are in preparation.

The profiles of a CANopen device are stored in the Object Dictionary (OD) in a defined manner. The Object Dictionary manages the objects using a 16-bit index. This index can be further subdivided with an 8-bit sub-index. All entries are summarized within groups.

For example, the Communication profile is located at index 1000h to 1FFFh.

Certain types of object entries are mandatory; others are optional or manufacturer-specific. The following types of objects are available:

- Domain a variable number of data
- Deftyp a definition entry, such as unsigned16
- Defstruct record type, such as PDO mapping
- Var an individual variable
- Array a multiple data field, whereby each individual data field is a simple variable of the same type
- Record a multiple data field, whereby the data fields are any combination of simple variables

With structured entries, subindex 0 indicates the number of following subindices.

7.3 Communication Profile

The interface between application and CANopen device is clearly defined by a uniform communication profile based on CAL. The CANopen communication protocol defines several methods for transmission and receipt of messages over the CAN bus, including transfer of synchronous and asynchronous messages. Coordinated data exchange across an entire network is possible by means of synchronous message transmission. Synchronous data transfer allows network wide coordinated data exchange. Pre-defined communication objects, i.e. SYNC Objects transmitted on a cyclic time period and Time Stamp objects support synchronous transfers. Asynchronous or event messages may be transmitted at any time and allow a device to immediately notify another device without having to wait for the next synchronous data transfer cycle.

7.4 Service Data Objects

Network management controls communication and device profiles of all networked devices. For this type of access service data objects (SDO) are used. In CANopen devices, all parameters and variables that are accessible via CAN are clearly arranged in the Object Dictionary.

All objects in the Object Dictionary can be read and/or written via SDOs. SDO represent a peer-to-peer communication between networked nodes. This access occurs according to the Multiplexed Domain protocol, whereby the index and subindex of the addressed objects are used as a multiplexor. This protocol is based on handshaking.

Individual parameters are addressed using a 16-bit index and an 8-bit subindex addressing mechanism. In this mode data packages may be larger than 8 bytes using multiple CAN messages. Messages smaller than 5 bytes can be transferred with a transmission acknowledgement. The owner of the Object Dictionary is the server of the domain. Read and write accesses via SDOs are supervised by the CANopen server and are checked for validity.

A variety of access restrictions must be taken into account, such as; *Read only, Write only* and *No PDO mapping.* Error messages provide detailed information on any access conflicts. Service Data Objects (SDOs) are normally used for device configuration such as setting device parameters. They are also used to define the type and format of information communicated using the Process Data Objects.

7.5 Process Data Objects

A Process Data Object (PDO) is a CAN message whose data contents, identifier, inhibit time, transmission type and CMS priority are configurable via entries in the Object Dictionary. PDO format and data content of the message may be fixed or dynamically configured using SDO data transfers. PDOs do not contain any explicit protocol overhead, hence enabling very fast and flexible exchange of data between applications running on each node. Hence, PDO transfers are typically used for high speed, high priority data exchange. Data size in a PDO message is limited to 8 bytes or less. PDOs can be transmitted directly from any device on the network simultaneously to any number of other devices. Data exchange across a CANopen network does not require a bus Master. This multicast capability is one of the unique features of CAN and is fully exploited in CANopen.

PDO entries start at index 1400h for receipt objects and at 1800h for transmission objects. CANopen permits cyclic and event-controlled communication. The type of transfer indicates the manner of the reaction to the SYNC message; while the inhibit time is the minimum time that must elapse between two transmissions of the PDO. PDOs reduce the bus load to a minimum, achieve a high information flow-rate and can be accessed via remote frames.

A simple CANopen device usually supports four PDOs. These are initialized with preset identifiers. Additional PDOs can be designated, yet to avoid message collision they may be set invalid (deactivated). This deactivation is configured by setting the MSB (bit 31) in the identifier of the PDO.

The message identifier can be found in the Object Dictionary under the entry for communication parameter in subindex 1. Bit 30 indicates if remote request for this PDO is enabled (bit 30 = 0) or not. Bit 29 configures the CAN frame format, bit 29 = 0 indicates 11-bit identifier.

Bit	31	30	29	28 – 11	10 - 0
11-bit-ID	0/1	0/1	0	000000000000000000	11-bit Identifier
29-bit-ID	0/1	0/1	1	29-bit Identifier	

Table 13: COB-Identifier (Communication Target Object Identifier)

The transmission types in subindex 2 can be configured within a range of 0 to 255. The values 0 to 240 define that the transfer of the PDO is in relation to the SYNC message. The value 0 indicates that current input values are only transmitted upon arrival of a SYNC message and if the requested input value has changed. Values between 1 and 240 indicate that the PDO is transmitted upon arrival of a corresponding number of SYNC messages. The values 241 to 251 are reserved. The values 252 and 253 are intended only for remote objects. For value 252, data is updated but not transmitted upon receipt of the SYNC message. The value 253 updates data upon receipt of the remote request. Values 254 and 255 are used for asynchronous PDOs. The release of these asynchronous PDOs is manufacturer or Device Profile-specific.

The inhibit time is stored in multiples of 100 µs as unsigned16 values at subindex 3.

At subindex 4, the priority group for this particular PDO is defined. The priority group is only effective in case DBT services (communication object identifier distribution services) are executed. Depending on the supported subindices, subindex 0 must be set to the applicable value.
PDO settings must correspond to the I/O profile rules:

- the first transmit and receipt PDO is used for exchange of ***digital*** data;

- the second transmit and receipt PDO is used for exchange of ***analog*** data.

If a CANopen device does not support digital inputs or outputs, it is recommended that the first transmit and receipt PDO remains unused. If a CANopen device does not support analog signals, it is recommended that the second transmit and receipt PDO remains unused.

7.6 PDO-Mapping

A unique mapping entry exists for each communication parameter entry of a PDO. This mapping entry is located in the Object Dictionary 200h above the corresponding communication parameter entry for this PDO. This mapping table corresponds to PDO data contents. The requirement for PDO mapping is the presence of variables in the Object Dictionary that are capable of mapping. For example, digital outputs at index 6200h and digital inputs at index 6000h can be mapped. These values can also be set and read out via SDO. However, in order to use the benefits of the CAN bus, the variables of a CANopen device are put in PDOs.

The mapping of variables is organized as follows:
All mapping entries are 4 bytes in size. The number of objects to be mapped is written to subindex 0. Each following subindex contains a reference to the index and subindex of variables and their length stored in “Bit“. For example:

```
                    60000108h
• reference to index 6000
• subindex              01
• length                  08 bit
```

In this example the value of the digital input is indicated by the first byte of a transmit PDO. For most CANopen devices, mapping occurs with a granularity of eight (8). This means that a maximum of eight entries per byte is possible for a mapping table.
In special cases mapping of bit objects can be supported. It is also advisable to sometimes exclude areas from mapping. For example, a

CANopen device might evaluate only the fifth byte of a PDO. In this case, 2 unsigned16 dummy objects are inserted in the mapping identity, if supported by the CANopen device. A mapping table can be used to appropriately configure communication parameters to encode a PDO for transmission or to decode a received PDO.

7.7 Error Handling

Each node in the network is able to signal error states as far as they are detected by the hardware and software. Error Handling is enabled by Emergency Objects. Internal fatal error states are encoded in error codes and sent only once to the other nodes. If other errors occur, the node remains in error state and transmits a new Emergency Object. If the error is recovered, the node then transmits an error message with the code *No error*. The Emergency message consists of 8 bytes, whereby the first and second bytes contain additional information that is found in the device profiles. The third byte contains the contents of the error register; while the remaining five bytes contain manufacturer-specific information. The Emergency Error code is stored in object [1003h], the *Pre-Defined Error Field*. This creates an error log that chronologically sorts errors. The oldest error is situated at the highest subindex.

Byte	0	1	2	3	4	5	6	7
Content	Emergency Error Code		Error Register, Object [1001]	Manufacturer Specific Error Field				

Table 14: Emergency-Message Contents

7.8 Network Services

In addition to services enabling configuration and data exchange, various CAN network services also support monitoring of networked devices. NMT (network management) services require a node in the network that assumes the functions of the NMT-Masters. The NMT-Master services include initialization of NMT-Slaves, distribution of the identifiers, the node monitoring and network booting.

7.8.1 Life-Guarding

Optional node monitoring is achieved by "*Life-Guarding*". The NMT-Master periodically transmits a Lifeguard message to the Slave. The Slave responds to the Lifeguard message with a return message indicating its present status and a bit that toggles between two messages. Should the Slave not respond or indicate an unexpected status, the NMT-Master application is informed by means of a status message. Moreover, the Slave can detect failure of the Master. *Life-Guarding* is started with the transmission of the initial message from the Master.

7.8.2 Heartbeat

Similar to Lifeguarding, the Heartbeat function is an additional network supervisory service. But unlike the Lifeguarding, the Heartbeat does not require a NMT-Master. Only CANopen Slaves are able to function as Heartbeat Producer and Consumer because they provide an Object Dictionary in order to store the Heartbeat times.

7.8.3 Heartbeat Producer

The Heartbeat Producer cyclically sends a Heartbeat message. The configured Producer Heartbeat time (16-bit – value in ms), located at index 1017h, will be used as an interval time. If this interval time expires, a message with the following contents will be sent:

Byte	0	1...7
Content	Producer State	reserved

Table 15: Heartbeat Message Structure

The COB-ID that is used is 0700h + the node address.
The Heartbeat Producer gives its status, which can be any of the following values, in the first byte of the message:

00h BOOTUP
04h STOPPED
05h OPERATIONAL
7Fh PRE-OPERATIONAL

The Heartbeat Producer is deactivated when the producer Heartbeat time is set to 0.

7.8.4 Heartbeat Consumer

The Heartbeat Consumer analyzes Heartbeat messages sent from the producer. In order to monitor the Producer, the Consumer requires every producers' node address, as well as the consumer Heartbeat time.

For every monitored Producer, there is a corresponding sub-entry that has the following contents:

	MSB		**LSB**
Bit	31-24	23-16	15-0
Value	00h	Node-ID	Consumer Heartbeat Time

Table 16: Structure of a Consumer Heartbeat Time Entry

The Consumer is activated when a Heartbeat message has been received and a corresponding entry is configured in the OD. If one of the activated Heartbeat times expires during an active Heartbeat consumer without receipt of a corresponding Heartbeat message, then the consumer for this producer is deactivated.

The Heartbeat consumer is completely deactivated when the consumer Heartbeat time is given a value of 0.

7.9 Network Boot-Up

The NMT-Master is responsible for booting of the network. The boot procedure takes place over several steps. According to the type of networked CANopen device, the identifier defaults to pre-defined values (for minimum CANopen devices) or is configured via DBT services. The pre-defined configuration for the identifier values include Emergency Objects, PDOs and SDOs. These are calculated according to node addresses, which can be located between 1 and 128 and are added to a base identifier that determines the function of an object.

Bit	10			7	6						0
COB-Identifier											
	Function Code				Device-ID						

Table 17: Calculation of the COB-Identifier from the Node Addresses

This base identifier is determined as follows:

Object	Resulting COB-ID [hex]	Resulting COB-ID [decimal]	Communication Parameter at Index
EMERGENCY	80h + Device-ID	129 – 255	
PDO1 (tx)	180h + Device-ID	385 – 511	1800h
PDO1 (rx)	200h + Device-ID	513 – 639	1400h
PDO2 (tx)	280h + Device-ID	641 – 767	1801h
PDO2 (rx)	300h + Device-ID	769 – 895	1401h
PDO3 (tx)	380h + Device-ID	896 – 1022	1802h
PDO3 (rx)	400h + Device-ID	1025 – 1050	1402h
PDO4 (tx)	480h + Device-ID	1152 – 1278	1803h
PDO4 (rx)	500h + Device-ID	1281 – 1406	1403h
SDO (tx)	580h + Device-ID	1409 – 1535	
SDO (rx)	600h + Device-ID	1537 – 1663	
Nodeguarding	700h + Device-ID	1793 – 1919	(100Eh)

Table 18: Base Identifier

Configuration data can be loaded on Slave devices via the pre-defined SDO. PDOs can be transmitted after nodes are set from *Pre_Operational* to *Operational* state by the NMT service *Start_Remote_Node*. Minimum CANopen devices also support the *Stop_Remote_Node, Enter_Pre-Operational_State, Reset_Node, Reset_Communication* services. As indicated in Figure 3, networked nodes automatically enter *Pre_Operational* following boot-up and

initialization. The *Reset_Node* service completely resets target nodes. *Reset_Communication* resets communication parameters.

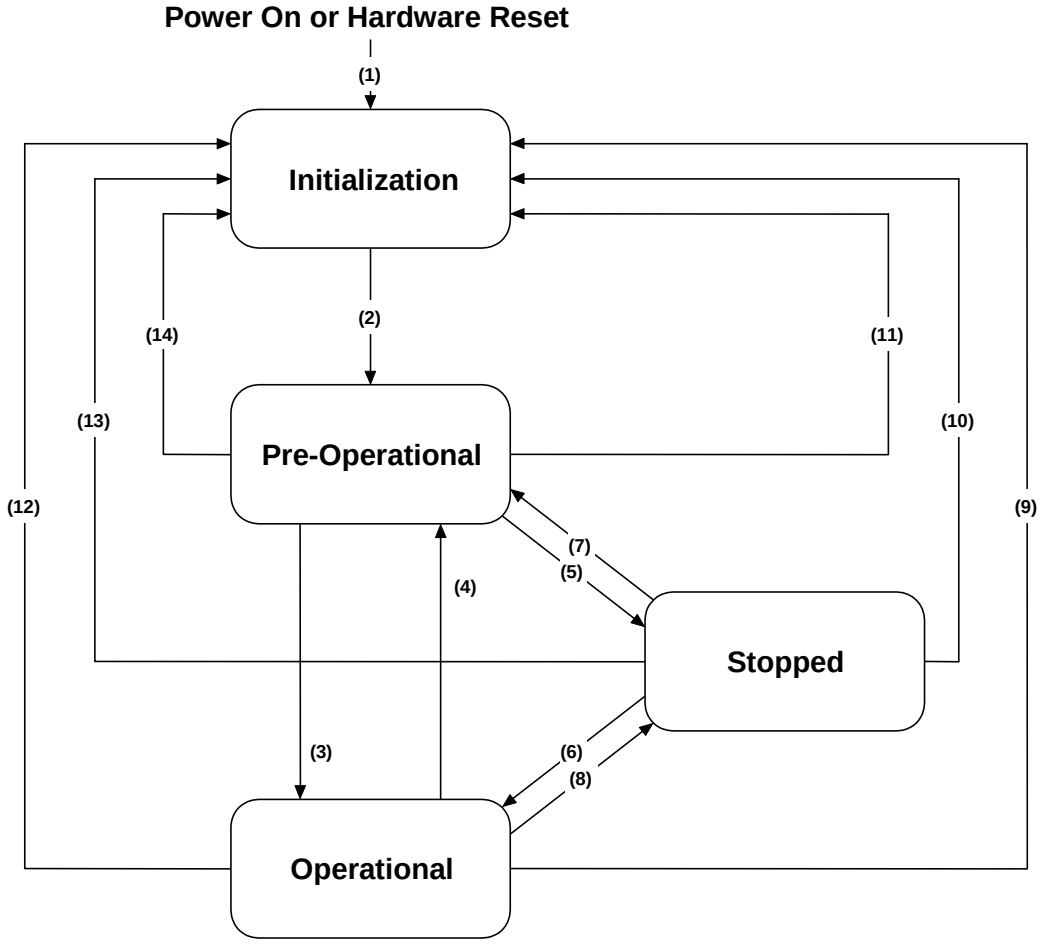


Figure 5: State Diagram of a CANopen Device

State transition	Action required
(1)	following "Power On", automatically switches into "Initialization" state
(2)	"Initialization" finished, automatically switches into "Pre-Operational" state
(3),(6)	NMT service "Start_Remote_Node"
(4),(7)	NMT service "Enter_Pre-Operational_State"
(5),(8)	NMT service "Stop_Remote_Node"
(9),(10),(11)	NMT service "Reset_Node"
(12),(13),(14)	NMT service "Reset_Communication"

Table 19: Description of State Flow Diagram Symbols

For networked devices operating in a network with or without DBT capabilities, it is necessary to reserve the identifier for “minimum devices“ in the database of the DBT-Master.

Extended Boot-up is based on CAL specifications. The device states *Pre-Operational* and *Initializing* have been implemented in addition.

7.10 Object Dictionary Entries

Beside the parameters for the PDOs, a number of additional entries in the Object Dictionary belong to the data that specify a CANopen device. The communication profile contains such information as:

- the device type at index [1000];
- the error register at index [1001];
- the Pre-Defined Error Field at [1003];
- the identifier of the SYNC message at [1005];
- the device name at [1008],
- the hardware and software version of the manufacturer at [1009] and [100A];
- the node address at [100B];
- the parameter Guard-Time at [100C] and
- the parameter Life-Time-Factor at [100D].

In the device type, information about the implemented device profile and the capabilities of the device is encoded. The error register gives information about internal errors of the device; the pre-defined error field provides an error log. In case the Guard-Time and Life-Time-Factor are unequal to Zero, the multiplied values result in the Life Time of the CANopen device for the node monitoring protocol.

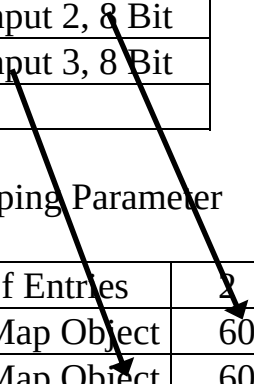
7.11 PDO Mapping Example

All network variables can be transferred by PDOs, which can transmit a maximum of 8 bytes of information. The allocation of variables to PDOs is defined by mapping tables. These variables are addressable via the Object Dictionary. Reading and writing of entries to the Object Dictionary occurs by means of Service Data Objects (SDO), which are used to configure the network by means of a special configuration tool.

This process is illustrated below in *Table 20*. Inputs 2 and 3 of device A are to be transferred to the outputs 1 and 3 of device B. Both devices support complete mapping.

Device A:

1000H	Device Type
.....	
6000H,1	Input 1, 8 Bit
6000H,2	Input 2, 8 Bit
6000H,3	Input 3, 8 Bit
....	



Transmit PDO Mapping Parameter

1A00H,0	# of Entries	2
1A00H,1	1.Map Object	60000208H
1A00H,2	2.Map Object	60000308H

Transmit PDO Communication Parameter:

1800H,0	# of Entries	2
1800H,1	COB-ID	501
1800H,2	Transm. Type	255
....		

Resulting PDO:

COB-ID	DATA	
501	Output 1	Output 3

Table 20: PDO Mapping Example

Transmit and receive PDOs utilize the same CAN identifier 501. Thus device B automatically receives the PDO transmitted by device A. The recipient, device B, interprets the data in accordance with its mapping scheme; it passes the first byte at output 1 and the second byte at output 3. These correspond to inputs 2 and 3, respectively of the transmitting device A.

7.12 Input/Output Assignment to Object Dictionary Entries

The CANopen ChipF40 allows an easy configuration for a specific CANopen application. The fixed number of inputs and outputs on the CANopen ChipF40 makes easy configuration of Process Data Objects (PDOs) possible. Both digital and analog inputs, as well as the digital outputs, are configured in accordance with CiA standards. Configuration of Object Dictionary Input/Output entries for the CANopen ChipF40 and CANopen ChipF40 V3**,** according to data type, is shown in *Table 21*and *Table 22*.

Data Type	Index / Subindex	Size
Digital Input		
DI0 ... DI7	6000h / 1	BYTE
DI8 ... DI15	6000h / 2	BYTE
DI16 ... DI23	6000h / 3	BYTE
Digital Outputs		
DO0 ... DO4	6200h / 1	BYTE
DO8 ... DO15	6200h / 2	BYTE
Analog Inputs		
AI0	6401h / 1	WORD
AI1	6401h / 2	WORD
AI2	6401h / 3	WORD
AI3	6401h / 4	WORD
AI4	6401h / 5	WORD
AI5	6401h / 6	WORD
AI6	6401h / 7	WORD
AI7	6401h / 8	WORD
PWM Outputs		
PWM0	6500H / 1; 6510H / 1	WORD
PWM1	6500H / 2; 6510H / 2	WORD
PWM2	6500H / 3; 6510H / 3	WORD
PWM3	6500H / 4; 6510H / 4	WORD

Table 21: Object Dictionary Input/Output Entries F40

Data Type	Index / Subindex	Size
Digital Input		
DI0 ... DI7	6000h / 1	BYTE
DI8 ... DI15	6000h / 2	BYTE
DI16 ... DI23	6000h / 3	BYTE
DI24 ... DI27	6000h / 3	BYTE
Digital Outputs		
DO0 ... DO4	6200h / 1	BYTE
DO8 ... DO15	6200h / 2	BYTE
Analog Inputs		
AI0	6401h / 1	WORD
AI1	6401h / 2	WORD
AI2	6401h / 3	WORD
AI3	6401h / 4	WORD
AI4	6401h / 5	WORD
AI5	6401h / 6	WORD
AI6	6401h / 7	WORD
AI7	6401h / 8	WORD

Table 22: Object Dictionary Input/Output Entries F40 V3

Note:
After boot-up of the CANopen ChipF40, objects can be accessed via SDOs. If the node is in state *Operational***,** objects can be accessed via PDOs. The default mapping parameters applies for Object Dictionary Input/Output entries. Any modification of mapping parameters can be done via SDO with the help of a network configuration tool.

8 CANopen ChipF40 Operation

8.1 CANopen State Transitions

The structure of messages that changes the state of a CANopen node is as follows:

11-bit CAN Identifier	2 Byte Data	
0	*cs*	*NODE_ID*

Node_ID Node address; Node_ID = 0 to address all devices (Broadcast)
cs Command

Table 23 summarizes all NMT-Master messages used for status control:

Command (cs)	Description	Function	State after Execution
1 (01h)	Start_Remote_ Node	Starts the CANopen device and PDO transmission, activates outputs	OPERATIONAL
2 (02h)	Stop_Remote_ Node	Stops PDO transmission, renders outputs in error state	STOPPED or PREPARED
128 (80h)	Enter_Pre_ Operational_ State	Stops PDO transmission, SDO remains active	PRE-OPERATIONAL
129 (81h)	Reset_Node	Executes a system Reset; Initial Start-up, resets all settings to default values	PRE-OPERATIONAL
130 (82h)	Reset_ Communication	Resets all communication parameters to default values	PRE-OPERATIONAL

Table 23: NMT-Master Messages for Status Control

8.2 Power On

After “Power-On”, the CANopen ChipF40 executes required initialization routines and switches into *Pre_Operational* state.

8.3 PRE-OPERATIONAL

Process Data Objects (PDOs) are not active in *Pre_Operational* state. The default identifier for Service Data Objects (SDOs) is available and all necessary network configurations can be executed via SDO. At the end of the configuration process, the CANopen device can be rendered into *Operational* state. This can be done by the network Master or by the user with the help of a network configuration tool.

8.4 OPERATIONAL

All Process Data Objects (PDOs) can be exchanged in *Operational* state. Access via SDO is also possible.

8.5 STOPPED

Network communication is suspended in state *STOPPED*. This does not affect the Node-Guarding and the “Heartbeat“, if this was enabled before. This state can be used to render the application into a “Safety State“. In *STOPPED* state PDO, SDO, SYNC and Emergency communication are **NOT** functioning. Leaving this state is only possible with a NMT message.

8.6 Restart Following Reset / Power-On

Each Reset of the CANopen ChipF40 transmits an Emergency message without data contents. Temporary operational failure of the CANopen ChipF40 and subsequent power-up of the device are detected without Node Guarding (*refer to Section 5.6*), as the sending device can be determined by the message identifier.

The CANopen ChipF40 distinguishes between *"Load"*_Start and *"Save"*_ Start. *"Load"*_Start is necessary:

- for initial operation of the CANopen ChipF40 after its delivery
- if the device parameters (Object Dictionary entries in RAM) should be overwritten by default values

With *"Load"*_Start, all default CANopen ChipF40 Object Dictionary entries are copied to RAM after Reset/Power On (manufacturer default values).

The string "save" must be written to object [1010] at subindex 1 in order to carry out the *"Save"*_ Start routine. With *"Save"*_ Start all Object Dictionary entries are copied from EEPROM to RAM after a Reset/Power-On using the saved user-specific values. If the bus Master or the user, by means of a network configuration tool, modifies Object Dictionary entries, then the modifications are only active as of the next RESTART if *"Save"* is written to object [1010] in subindex 1. This means that only the stored values are valid after the Reset/Power-On of the CANopen ChipF40. These values are stored then in the EEPROM and, in the event of power-down, are not lost. A *"Save"*_ Start can take up to 1.5 seconds, because the entire OD is read from the EEPROM and written into RAM.

All device parameters can be stored in the EEPROM using object [1010] in subindex 1. In order to prevent unintended storage of parameters in the EEPROM device, a special *"Save"* signature must be written to subindex 1. This 32-bit signature (in hex format) appears as follows:

MSB			LSB
'e'	'v'	'a'	's'
65h	76h	61h	73h

All device parameters can be reset to manufacturer default values according to CiA 301 or CiA 401 standards via the object [1011] in subindex 1. In order to prevent an unintended reset following a store instruction with the *"Save"* signature, the *"Load"* signature must be written to subindex 1. This 32-bit signature (in hex format) appears as

follows:

MSB			LSB
‘d’	‘a’	‘o’	‘l’
64h	61h	6fh	6ch

In order to set the default values, a Reset/Power-On must be subsequently executed.

8.7 NMT-Boot-Configuration

The CANopen ChipF40 can be configured, that it works as a NMT-Boot-Master for all CANopen-Nodes in the network. The configurations is made in the Object Dictionary at Index [2001H]. Subindex 1 is the entry for NMT-Boot-Enable:

- 0 (default), Node is no Boot-Master
- 1 Node is Boot-Master.

Subindex 2 is the entry for NMT-Start-Time: It is the delay-time [ms] after this the node sends the NMT-Boot-Message (default 500ms, max. 65s).
The entries for NMT-Boot-Configuration are saved into EEPROM when they are written. They are not used at save and recover the OD *(see also section 8.6*). So the entry has to written directly with the recommended values to configure the CANopen ChipF40 as Boot-Master or not.
The configuration will be active at the next reset of the CANopen ChipF40.

Example for SDO messages for write NMT-Boot-Enable und NMT-Start-Time:

Action	**CAN ID for Node 0x40**	**DLC**	**SDO Cmd**	**Index**		**Sub-index**	**Value**			
set NMT-Boot-Enable	640	8	2F	01	20	01	01	00	00	00
clear NMT-Boot-Enable	640	8	2F	01	20	01	00	00	00	00
set NMT-Start-Time to 0x3E8	640	8	2B	01	20	02	E8	03	00	00

Table 24: SDOs zum Setzen der NMT-Boot-Konfiguration

The used CAN-Identifier is 0x600 + Node-ID. The values are according to CANopen-Standard LSB first.

8.8 Analog Input Operation

8.8.1 Handling Analog Values

This section provides general information on data storage of analog values in a CANopen frame.

The CANopen Standard CiA 401 defines that all analog values till 15 bit have to be stored as 16-bit value aligned left with a sign bit. On the CANopen ChipF40 all A/D-conversion values are stored with 10-bit data. Consequently, for each analog channel, two data bytes must be transmitted.

These data bytes are stored and transmitted on the CAN bus as shown in *Table 25*.

Byte 2								Byte 1							
Sign	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
+/-	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	0	0	0	0	0

Table 25: Storage of Analog Values

On the CAN bus, first byte 1 and then byte 2, is transmitted.

8.8.2 Formula for Calculating the Analog Input Value

The formula listed below is used to calculate a voltage value of an analog input from the A/D-conversion result:

$$AIn_{/V} = \frac{result\ A/D\ conversion_{/hex} \bullet voltage\ range_{/V}}{2^{\ resolution\ ADC}}$$

The following example will explain the use of this formula in more detail:

over CAN transmitted A/D-value	= 012A0H (4768 dec)
voltage range	= 5V (standard supply on pins VAREF and VAGND)
logical resolution ADC	= 15 Bit
analog input value (AI)	= 0,727V

For the lowest quantization of the A/D-value the real resolution of the ADC has to be used.

A/D-value	= 01H
voltage range	= 5V (standard supply on pins VAREF and VAGND)
real resolution ADC	= 10 Bit
lowest resolution	= 4,88 mV/Digit

8.8.3 Selecting the Interrupt Trigger

This object entry determines which event can release an interrupt. For this purpose the object [6421] "Interrupt_Trigger_Selection" is available. If the "Global_Interrupt_Enable" [6423] is activated, the release of an interrupt transmits the TX-PDO for analog inputs. A specific subindex is available for each analog input channel. This allows precise configuration of the interrupt event for each channel.

The following values are available:

Bit Number	Interrupt Trigger
0	Upper limiting value exceeded
1	Lower limiting value exceeded
2	Input value fluctuates more than *DELTA [6426]*
3	*Not supported!*
4	*Not supported!*
5 to 7	Reserved

Table 26: Interrupt Trigger Bits

Example:
6421,1 = 04h means: the first analog input must fluctuate by more than *DELTA* in order to send the PDO.

Note:
The default values for all analog inputs are set to 04h.

8.8.4 Interrupt Source

This object entry stores which analog input caused the interrupt. The object [6422] "Analog_Input_Interrupt_Source" is available for this purpose. Every single bit refers to the corresponding analog input channel. These bits will be reset automatically if the entry has been read by a SDO or the object entry was transmitted with a PDO.

The following convention is used:
"1" : Channel caused an interrupt,
"0" : Channel caused no interrupt.

Example:
6422,1 = 01h means: analog input channel 0 caused an interrupt.

8.8.5 Interrupt Enable

All interrupts can be enabled or disabled using the object entry [6423] "Analog_Input_Global_Interrupt_Enable". The default value is "0", indicating interrupt execution is disabled. To enable the interrupt execution, the value "1" must be written to the object entry (*also refer to section 8.8.3*).

8.8.6 Interrupt Upper and Lower Limit

An interrupt is released, if the analog input value is higher or lower than the specified limiting value in the applicable subindex. The upper limit is specified in object [6424], the lower limit in [6425]. To release an interrupt, the OD entry [6423] must be set to "1".

Each analog input value will be transmitted as long the trigger condition is given. This assumes that no other trigger condition, such as the Delta Function, is enabled. The limit values must be specified as 32-bit value aligned left.

For this purpose, the objects:
- [6424] "Analog_Input_Interrupt_Upper_Limit_Integer" and
- [6425] "Analog_Input_Interrupt_Lower_Limit_Integer"

are available.

Note:
The default value in both entries for all analog inputs is "0".

Example:
6423 = 1h, 6421,1 = 05h and **6424,1 = 2000h**:
The analog input #1 releases an interrupt if the value exceeds the limit of 2000h, and then the value fluctuates by more than specified in the Delta function (see following section).

8.8.7 Delta Function

The delta function allows configuration of the extent to which an analog input value can fluctuate since the most recent transmission. Only if the fluctuation on the analog input exceeds the value specified in the delta function transmission of the corresponding PDO on the CAN bus is initiated. This configuration can be done using the object [6426] *Analog_Input_Interrupt_Delta*. Entries specify the number of digits in the conversion result that are allowed to fluctuate. The default value for all four analog inputs is 5. This means that the A/D-conversion result may change by up to 5 digits before a PDO is transmitted. The value must be specified as aligned left and assumes 10-bit resolution.

Note:
The default value for delta for all analog inputs is "A0H".

8.8.8 Example for Trigger Conditions

In the following figure the combination of objects 6424H to 6426H is shown.

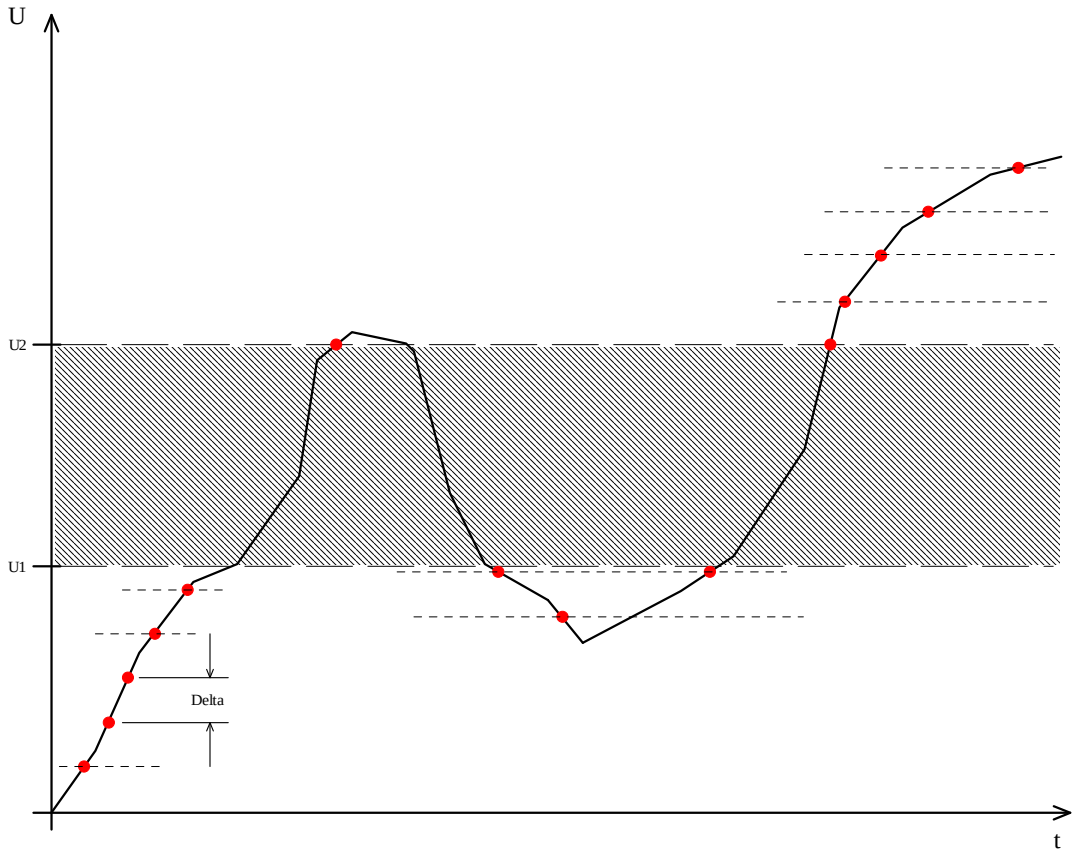


Figure 6: Example Trigger Conditions

The value *U1* was entered at index [6424H] - Lower Limit and *U2* at index [6425H] - Upper Limit in OD. Furthermore the value *Delta* was entered at index [6426H] - Delta. The shown voltage trace is located at an analogue input. At the moments, what are marked with λ , a corresponding PDO will be transmitted from CANopen ChipF40. Is the analogue input vale between U1 and U2 (the hatched area), no PDO will be transmitted.

8.9 Functionality of PWM Outputs

(not available for CANopen ChipF40 V3)
The CANopen ChipF40 can generate PWM-signals. For every Output it exists one OD-entry for period (index [6510H]) and one for duty cycle (Index [6500H]).
Both parameter have the format unsigned 16.

The duty cycle is declared in percent. That means, the value 0H – 0% and 0FFFFH – 100% are corresponding.

The default value for all PWM-Outputs is 0.

The period is declared as a multiple of 1µs. The value of 1000 corresponds with a frequency of 1kHz.

The default value for all PWM-Outputs is 1000.

The lowest adjustable value is 43 (CAN-bus baud rate different to 10kBit), or 86 (CAN-bus baud rate equal 10 kBit).

8.10 Emergency Message

In the event of an error, the status of the CANopen Chip164 is transmitted via a high-priority Emergency Message. These messages consist of 8 data bytes and contain error information. The Emergency Message is transferred as soon as one of the specified errors occurs. A specific Error Message is only transmitted once, even if the recent error is not resolved for a longer period of time. If all error causes are eliminated, then an Error Message with contents “0“ (error eliminated) is transmitted. The structure of the 8-byte Emergency Message is depicted below:

BYTE 0	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7
Error Code		Error-Register [1001]	Manufacturer-specific Error Code				

Table 27: Emergency Message

8.10.1 Error Code

The Error Code (byte field 0+1, LSB, MSB) indicates whether an error is present or whether the error has already been eliminated (no error). The following error codes are valid:

- 0000H: no error
- 1000H: global error
- 5000H Hardware Error (Hardware reset was detected)
- 6100H Software Error (Software reset was detected)
- 6101H Software Error (Watchdog reset was detected)
- 6102H Software Error (Stack overrun of microcontroller)
- 8110H CAN-Message lost (busload too high)
- 8120H Device is in Error passive mode
- 8130H: Lifeguard or Heartbeat Error
  In case of a Heartbeat Consumer error, the Node-ID of the failing node is transmitted in the manufacturer-specific error code field.
- 8140H Device has detected CAN-BUSOFF
- 8210H The PDO was not processed due a length error.

8.10.2 Error Register

The Error Register (byte field 2) can contain the following values:

- 81h: Occurrence of a manufacturer-specific error
- 11h: CAN communication error
- 01h: Occurrence of a common error
- 00h: Error has been eliminated - error reset

8.11 Display State at Run and Error LED

The current state of the CANopen ChipF40 is displayed at the both state-LEDs D1 und D2. The functionality of both LED's is defined in standard **CiA 303-3 V1.0** .

8.11.1 Run LED

The green Run LED (D2) displays the NMT State of the device. The *Table 28* shows the different states and their meaning.

RUN LED	state	meaning
On	OPERATIONAL	CANopen ChipF40 is in state OPERATIONAL
Blinking relation 50:50	PREOPERATIONAL	CANopen ChipF40 is in state PREOPERATIONAL
Single Flash	STOPPED	CANopen ChipF40 is in state STOPPED
Triple Flash synchronous with the Error LED	Stack overrun	The software caused a stack overrun of the microcontrollers.
Blinking by turns with Error LED	access with LSS	A LSS service is running.
Synchronous fast Blinking with Error LED	Reset to default values	At DIP-Switch is the reset to default values set.

Table 28: Run LED States

8.11.2 Error LED

The red Error LED (D1) displays the current error state of the CANopen ChipF40. The *Table 29* shows the different states and their meaning.

ERROR LED	State	Description
Off	no error	no error on device detected
Single Flash	Warning Limit reached	The Warning Limit in CAN Controller was reached (too much Error Frames on CAN Bus).
Blinking by turns with Run LED	access with LSS	A LSS service is running.
Double Flash	Error Control Event	An error at Lifeguarding, Nodeguarding or Heartbeat was detected.
Triple Flash synchronous with the Run LED	Stack overrun	The software caused a stack overrun of the microcontrollers.
On	Bus Off	The CAN controller is in state "Bus Off".
Synchronous fast Blinking with Run LED	Reset to default values	At DIP-Switch is the reset to default values set.

Table 29: Error Led States

This side was left blank intentionally.

9 Operation in the Event of Errors

9.1 State of the CANopen ChipF40 in the Event of Errors

The object dictionary entry "Error Behavior" at index [1029] for F40 and index [67FE] for CANopen ChipF40 V3 can be used to define which state the CANopen Chip164 should transfer to in case of an error.

The following entries are possible:

0: change state to PRE-OPERATIONAL
1: do not change state
2: change state to STOPPED

These settings over all possible error sources described in *sections 8.10.1*. The entries Output Error (subindex 2) and Input Error (subindex 3) are not supported.
The error code 6102H sets the outputs into error state. This error is critical and it can leave only by reset.

9.2 Output Handling in the Event of Errors

The user can determine how each output is supposed to behave in the event of an error. The outputs are only changed, when at index “Error Behavior” a state change was activated. All errors that are not lead to a state change do not change the outputs.

9.2.1 Digital Outputs

On digital outputs, error handling can be pre-defined via the objects [6206] (“*Error_Mode_Output_8-Bit*“) and [6207] (“*Error_Value_Output_8-Bit”*). These entries can be configured by means of a network configuration tool. In the default configuration, the outputs do not change their states in the event of an error.

A value of “1“ at the bit position for an applicable output in the object [6206] results in writing the bit value (“0“ or “1“) located in the object [6207] to the corresponding output.

Example for digital outputs:

Index	Subindex	DO 3	DO 2	DO 1	DO 0	Description
6206	1	0	0	1	1	Error Mode Output 8-bit
6207	1	X	X	0	1	Error Value Output 8-b

Table 30: Example for Error Handling digital outputs

In the event of an error, the digital output DO0 is set to 1 while DO1 is set to 0. The status of the outputs OUT2 and OUT3 remain unchanged.

9.2.2 PWM Outputs

On PWM outputs, error handling can be pre-defined via the objects [6543] (“*PWM_Output_Error_Mode*“) and [6544] (*”PWM_Output_Error_Value”*). These entries can be configured by means of a network configuration tool. In the default configuration, the outputs do not change their states in the event of an error.

A value of “1“ in the object [6543] results in writing the value located in the object [6544] to the corresponding output.

Example for PWM outputs:

Index	Subindex	PWM0 Case 1	PWM0 Case 2	Description
6543 H	1	0	1	PWM Output Error Mode
6544 H	1 = 0x4000	no change	0x4000 , 25%	PWM Output Error Value

Table 31: Example for Error Handling PWM outputs

In the event of an error, in case 1 the PWM output 0 is not changing. In Case 2 the duty cycle changes to 25%.

9.3 Changing from Error State to Normal Operation

In the event of an error, the outputs retain their active values until overwritten (by means of PDO/SDO) by new output values. This requires that the error, such as “Bus Off” or “Life-Guarding” error, is eliminated and the CANopen Chip164 be switched into *Operational* state by a Master *“Start_Remote_Node”* message.

This side was left blank intentionally.

10 Object Dictionary CANopen ChipF40

Index [hex]	Object	Name	Data type	in PDO mappable
1000	Var	Device Type	Unsigned32	-
1001	Var	Error Register	Unsigned8	-
1003	Array	Error Message	Unsigned32	-
1005	Var	Identifier SYNC-message	Unsigned32	-
1007	Var	SYNC window length	Unsigned32	-
1008	Var	Device name	String	-
1009	Var	Hardware Version	String	-
100A	Var	Software Version	String	-
100C	Var	Guard Time	Unsigned16	-
100D	Var	Life Time Factor	Unsigned8	-
1010	Array	User-Parameter save	Unsigned32	-
1011	Array	Default-Parameter reload	Unsigned32	-
1014	Var	Identifier Emergency	Unsigned32	-
1016	Array	Consumer Heartbeat Time	Unsigned32	-
1017	Var	Producer Heartbeat Time	Unsigned16	-
1018	Record	Identity Object	Identity	-
1029	Array	Error Behavior	Unsigned8	-
1200	Record	1st Server SDO Parameter	SDO Parameter	-
1201	Record	2nd Server SDO Parameter	SDO Parameter	-
1400	Record	RxPDO1 Communication parameter	PDOComPar	-
1401	Record	RxPDO2 Communication parameter	PDOComPar	-
1402	Record	RxPDO3 Communication parameter	PDOComPar	-
1403	Record	RxPDO4 Communication parameter	PDOComPar	-
1600	Record	RxPDO1 Mapping parameter	PDOMapping	-
1601	Record	RxPDO2 Mapping parameter	PDOMapping	-
1602	Record	RxPDO3 Mapping parameter	PDOMapping	-
1603	Record	RxPDO4 Mapping parameter	PDOMapping	-
1800	Record	TxPDO1 Communication parameter	PDOComPar	-
1801	Record	TxPDO2 Communication parameter	PDOComPar	-
1802	Record	TxPDO3 Communication parameter	PDOComPar	-
1803	Record	TxPDO4 Communication parameter	PDOComPar	-
1A00	Record	TxPD01 Mapping parameter	PDOMapping	-

1A01	Record	TxPD02 Mapping parameter	PDOMapping	-
1A02	Record	TxPD03 Mapping parameter	PDOMapping	-
1A03	Record	TxPD04 Mapping parameter	PDOMapping	-
2000	Var	I/O Configuration	Unsigned8	-
2001	Var	NMT-Boot-Configuration	Unsigned8	-
6000	Array	PDO Digital Input	Unsigned8	x
6200	Array	PDO Digital Output	Unsigned8	x
6206	Array	Error Mode Digital Output	Unsigned8	-
6207	Array	Error State Digital Output	Unsigned8	-
6401	Record	PDO Analog Input	Integer16	x
6421	Array	Interrupt Trigger Selection	Unsigned8	-
6422	Array	Interrupt Source	Unsigned32	x
6423	Var	Global Interrupt Enable	Boolean	-
6424	Array	Interrupt upper Limit	Integer32	-
6425	Array	Interrupt lower Limit	Integer32	-
6426	Record	Input Interrupt Delta	Unsigned32	-
6500	Array	PWM Pulse	Unsigned16	x
6510	Array	PWM Period	Unsigned16	x
6543	Array	PWM Output Error Mode	Unsigned8	-
6544	Array	PWM Output Error Value	Unsigned16	-

Table 32: Object Dictionary of the CANopen ChipF40

11 Object Dictionary CANopen ChipF40 V3

Index [hex]	Object	Name	Data type	in PDO mappable
1000	Var	Device Type	Unsigned32	-
1001	Var	Error Register	Unsigned8	-
1003	Array	Error Message	Unsigned32	-
1005	Var	Identifier SYNC-message	Unsigned32	-
1007	Var	SYNC window length	Unsigned32	-
1008	Var	Device name	String	-
1009	Var	Hardware Version	String	-
100A	Var	Software Version	String	-
100C	Var	Guard Time	Unsigned16	-
100D	Var	Life Time Factor	Unsigned8	-
1010	Array	User-Parameter save	Unsigned32	-
1011	Array	Default-Parameter reload	Unsigned32	-
1014	Var	Identifier Emergency	Unsigned32	-
1016	Array	Consumer Heartbeat Time	Unsigned32	-
1017	Var	Producer Heartbeat Time	Unsigned16	-
1018	Record	Identity Object	Identity	-
1200	Record	1st Server SDO Parameter	SDO Parameter	-
1201	Record	2nd Server SDO Parameter	SDO Parameter	-
1400	Record	RxPDO1 Communication parameter	PDOComPar	-
1401	Record	RxPDO2 Communication parameter	PDOComPar	-
1600	Record	RxPDO1 Mapping parameter	PDOMapping	-
1601	Record	RxPDO2 Mapping parameter	PDOMapping	-
1800	Record	TxPDO1 Communication parameter	PDOComPar	-
1801	Record	TxPDO2 Communication parameter	PDOComPar	-
1802	Record	TxPDO3 Communication parameter	PDOComPar	-
1803	Record	TxPDO4 Communication parameter	PDOComPar	-
1A00	Record	TxPD01 Mapping parameter	PDOMapping	-
1A01	Record	TxPD02 Mapping parameter	PDOMapping	-
1A02	Record	TxPD03 Mapping parameter	PDOMapping	-
1A03	Record	TxPD04 Mapping parameter	PDOMapping	-
2000	Var	I/O Configuration	Unsigned8	-
2001	Var	NMT-Boot-Configuration	Unsigned8	-
6000	Array	PDO Digital Input	Unsigned8	x
6200	Array	PDO Digital Output	Unsigned8	x
6206	Array	Error Mode Digital Output	Unsigned8	-

6207	Array	Error State Digital Output	Unsigned8	-
6401	Record	PDO Analog Input	Integer16	x
6421	Array	Interrupt Trigger Selection	Unsigned8	x
6422	Array	Interrupt Source	Unsigned32	x
6423	Var	Global Interrupt Enable	Boolean	-
6424	Array	Interrupt upper Limit	Integer32	x
6425	Array	Interrupt lower Limit	Integer32	x
6426	Record	Input Interrupt Delta	Unsigned32	x
67FE	Array	Error Behavior	Unsigned8	-

Table 33: Object Dictionary of the CANopen ChipF40 V3

12 Revision History of this Document

Date	Manual Version	Changes
22/11/2004	L-1062e_1	Initial translation based on L-1062d_1
22/04/2005	L-1062e_2	Error code for stack overrun and display on the LEDs included
10/02/2006	L-1062e_3	QA Revision, reformatting
12/03/2007	L-1062e_4	description for Error Led changed
23/01/2008	L-1062e_5	description for CANopen Chip F40 version 3301002
26/06/2009	L-1062e_6	max. value for PWM frequency at 10kBit CAN changed
03/03/2010	L-1062e_7	values for output current of digital I/O pins attached

This side was left blank intentionally.

Index

This side was left blank intentionally.

Document: CANopen ChipF40
Document number: L-1062e_9, Edition June 2012

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 : +49-3661-6279-99

Published by

SYS TEC ELECTRONIC

Ordering No. L-1062e_9
Printed in Germany