SYS TEC
ELECTRONIC

System Manual
PLCcore-F407

User Manual
Version 8

Edition September 2018

Document No.: L-1515e_08

Status/Changes

Status: released

Date/Version	Section	Changes	Editor
2014/01/28 Version 1	All	Creation	T. Volckmann
2014/02/20	Figures	Updated figures to the latest core and baseboard revision	T. Volckmann
2014/02/24	Table 12	Add column for pull up/down mapping	T. Volckmann
	Section 7.5.2	Add frequency for PWM output Add frequency for Counter input	
2014/04/10	Table 36	Add reference for using NVDATA function blocks	T. Volckmann
2014/07/23 Version 2	Section 7.7.2	Add note regarding CAN1 interface	T. Volckmann
2014/08/04	Section 3	Amplitude information for PWM output	T. Volckmann
2014/09/19	Appendix A	Add mapping for SIO PORT parameters	T. Volckmann
2014/09/22	Section 7.3.2	Add how to interpret current measurement	T. Volckmann
	Section 9	Add jumper configuration	
2015/02/10 Version 3	Section 7.3.2	Corrected current consumption	T. Volckmann
2015/06/08 Version 4	Figure 16	Corrected description of Figure 16	T. Volckmann
	Section 7.8.1	Reworked parameter description for SIO Function Blocks	
	Table 7	Add IO Timings	
2015/10/05 Version 5	Section 11.2.8	Add command description for GET_RTC	T. Volckmann
	Section 11.2.9	Add command description for SET_RTC	
2016/03/15 Version 6	Section 7.5.2	Adjust description for pulse outputs Add description for AB encoding	T. Volckmann
	Section 7.9	Add new section 7.9	
	Section 11.2.7	Add new section 11.2.7	
	Appendix	Extend description for PTO_PWM function block Add function blocks for Modbus and file system	
	Section 9.4	Add new section 9.4	
2018/07/31 Version 7	Section 6	Correct wrong pinout of ETH_RXM, ETH_RXP, ETH_TXM and ETH_TXP	T. Volckmann

2018/09/19 Version 8	Section 5.2	Characteristics of Analog Inputs added	A. Süß

This manual includes descriptions for copyrighted products that 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 guarantees 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 does not accept responsibility for any damages which might result.

Additionally, SYS TEC electronic GmbH neither guarantees nor assumes 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.

© Copyright 2018   SYS TEC electronic GmbH, D-08468 Heinsdorfergrund.
All rights – including those of translation, reprint, broadcast, photomechanical or similar reproduction and storage or processing in computer systems, in whole or in part – are reserved. No reproduction may occur without the express written consent from SYS TEC electronic GmbH.

Inform yourselves:

Contact	Direct	Your local distributor
Address:	SYS TEC electronic GmbH Am Windrad 2 D-08468 Heinsdorfergrund GERMANY	
Ordering Information:	+49 (0) 37 65 / 38 600-0 info@systec-electronic.com	Please find a list of our distributors under: http://www.systec-electronic.com/distributors
Technical Support:	+49 (0) 37 65 / 38 600-0 support@systec-electronic.com	
Fax:	+49 (0) 37 65 / 38 600-4100	
Web Site:	http://www.systec-electronic.com	

8th Edition September 2018

Table of Contents

1 Introduction

Thank you that you have decided for the SYS TEC PLCcore-F407. This product provides to you an innovative and high-capacity PLC-kernel. Due to its high performance on a small manufactured size and due to its low power consumption, it is well-suitable as communication and control processor for embedded applications.

Please take some time to read through this manual carefully. It contains important information about the commissioning, configuration and programming of the PLCcore-F407. It will assist you in getting familiar with the functional range and usage of the PLCcore-F407. This document is complemented by other manuals, e.g. for the *OpenPCS* IEC 61131 programming system and the CANopen extension for IEC 61131-3. Table 1 in section 2 shows a listing of relevant manuals for the PLCcore-F407. Please also refer to those complementary documents.

For more information, optional products, updates et cetera, we recommend you to visit our website: http://www.systec-electronic.com. The content of this website is updated periodically and provides to you downloads of the latest software releases and manual versions.

Declaration of Electro Magnetic Conformity for PLCcore-F407 (EMC law)

The PLCcore-F407 has been designed to be used as vendor part for the integration into devices (further industrial processing) or as Development Board for laboratory development (hard- and software development).

After the integration into a device or when changes/extensions are made to this product, the conformity to EMC-law again must be assessed and certified. Only thereafter products may be launched onto the market.

The CE-conformity is only valid for the application area described in this document and only under compliance with the following commissioning instructions! The PLCcore-F407 is ESD-sensitive and may only be unpacked, used and operated by trained personal at ESD-conform work stations.

The PLCcore-F407 is a module for the application in automation technology. It features IEC 61131-3 programmability, uses standard CAN-bus and Ethernet network interfaces and a standardized network protocol. Consequently, development times are short and hardware costs are reasonable. PLC-functionality is created on-board through a CANopen network layer. Hence, it is not necessary for the user to create firmware.

2 Overview / Where to find what?

The PLCcore-F407 is based on SYS TEC ECUcore-F407 hardware and is extended by PLC-specific functionality (PLD software, PLC firmware). There are different hardware manuals for all hardware components such as the ECUcore-F407 and the PLCcore-F407 (the hardware of both modules is identical), development boards and reference circuitry. Software-sided, the PLCcore-F407 is programmed with IEC 61131-3-conform *OpenPCS* programming environment. There are additional manuals for *OpenPCS* that describe the handling of programming tools and SYS TEC-specific extensions. Those are part of the software package *"OpenPCS"*. Table 1 lists up all relevant manuals for the PLCcore-F407.

Table 1: Overview of relevant manuals for the PLCcore-F407

Information about…	In which manual?
Basic information about the PLCcore-F407 (configuration, administration, process image, connection assignment, firmware update, hardware description et cetera)	In this manual
Basics about the *OpenPCS* IEC 61131 programming system	Brief instructions for the programming system (Entry "*OpenPCS Documentation*" in the *OpenPCS* program group of the start menu) (Manual no.: L-1005)
Complete description of the *OpenPCS* IEC 61131 programming system, basics about the PLC programming according to IEC 61131-3	Online help about the *OpenPCS* programming system
Command overview and description of standard function blocks according to IEC 61131-3	Online help about the *OpenPCS* programming system
SYS TEC extension for IEC 61131-3: - String functions - UDP function blocks - SIO function blocks - FB for RTC, Counter, PWM/PTO	User Manual *"SYS TEC-specific extensions for OpenPCS / IEC 61131-3"* (Manual no.: L-1054)
CANopen extension for IEC 61131-3 (Network variables, CANopen function blocks)	User Manual *"CANopen extension for IEC 61131-3"* (Manual no.: L-1008)
Textbook about PLC programming according to IEC 61131-3	IEC 61131-3: Programming Industrial Automation Systems John/Tiegelkamp Springer-Verlag ISBN: 3-540-67752-6 (a short version is available as PDF on the *OpenPCS* installation CD)

Section 4 of this manual explains the **commissioning of the PLCcore-F407** based on the Development Kit for the PLCcore-F407 and their **electrical characteristics**.

Section 6 describes the **connection assignment** of the PLCcore-F407.

Section 7 explains details about the **application of the PLCcore-F407**, e.g. the **setup of the process image**, the **meaning of control elements** and it provides basic information about programming the module. Moreover, information is given about the usage of CAN interfaces in connection with **CANopen**.

Section 7.9 describes the **firmware update** process of the PLCcore-F407.

Section 9 explains the baseboard configuration of the PLCcore-F407 Development Kit.

Section 10 shows how to **install and configure the *OpenPCS*** environment for the PLCcore-F407.

Section 11 explains the **configuration command shell** and the **available commands**. This section not only includes the command names itself, but also their parameters and shows some example configurations.

3 Product Description

The PLCcore-F407 as another innovative product extends the SYS TEC electronic GmbH product range within the field of control applications. In the form of an insert-ready core module, it provides to the user a complete and compact PLC. Due to CAN and Ethernet interfaces, the PLCcore-F407 is best suitable to perform decentralized control tasks.

Figure 1: Top view of the PLCcore-F407

These are some significant features of the PLCcore-F407:

- High-performance CPU kernel (ARM 32-bit Cortex-M4, 168 MHz CPU Clock, 210 DMIPS)
- 192 KByte SDRAM Memory, 1 MByte FLASH Memory
- 2x52 pin card edge (Mini PCie connector footprint)
- 1x 10/100 Mbps Ethernet LAN interface (with on–board PHY)
- 2x CAN 2.0B interface, usable as CANopen Manager (CiA 302-conform)
- 3x synchronous/asynchronous serial ports (USART)
- 24 digital inputs (2 of these are usable as high-speed counter with 1 direction and 1 input each)
- 22 digital outputs (2 of these are usable as PWM/PTO output with pulse/dir)
- 8 analog inputs
- 2 analog outputs
- On-board peripherals:
  - RTC
  - Temperature sensor
- Programmable in IEC 61131-3
- Function block libraries for communication (CANopen, Ethernet and UART/USART)
- Function block libraries for hardware components (RTC, Counter, PWM/PTO)
- Support of typical PLC control elements (e.g. Run-LED, Error-LED)
- Small dimension (67.5x52.5 mm)

There are different types of firmware available for the PLCcore-F407. They differ regarding the protocol used for the communication between Programming PC and PLCcore-F407:

Order number: 3390094:   PLCcore-F407/Z4 (CANopen)
communication with Programming PC via CANopen Protocol
(Interface CAN0)

Order number: 3390095:   PLCcore-F407/Z5 (Ethernet)
communication with Programming PC via UDP Protocol
(Interface ETH0)

Making PLC available as an insert-ready core module with small dimensions reduces effort and costs significantly for the development of user-specific controls. The PLCcore-F407 is also very well suitable as intelligent network node for decentralized processing of process signals (CANopen and UDP). Additionally, it can be used as basic component for special assemblies or as PLC in hard-to-access areas.

The on-board firmware of the PLCcore-F407 contains the entire PLC runtime environment including CANopen connection with CANopen master functionality. Thus, the module is able to perform control tasks such as linking in- and outputs or converting rule algorithms. Data and occurrences can be exchanged with other nodes (e.g. superior main controller, I/O slaves and so forth) via CANopen network, Ethernet (UDP protocol) and serial interfaces (UART). Moreover, the number of in- and outputs either is locally extendable or decentralized via CANopen devices. For this purpose, the CANopen-Chip is suitable. It has also been designed as insert-ready core module for the appliance in user-specific applications.

The PLCcore-F407 provides 24 digital inputs (DI0...DI23, 3.3V level), 22 digital outputs (DO0...DO21, 3.3V level), 2 high-speed counter input (with 1 direction and 1 input each) and 2 PWM output (3.3V amplitude). This default I/O configuration can be adapted for specific application requirements by SYS TEC (please contact support@systec-electronic.com if you are interested in this option). Saving the PLC program in the on-board Flash-Disk of the module allows an automatic restart in case of power breakdown.

Programming the PLCcore-F407 takes place according to IEC 61131-3 using the *OpenPCS* programming system of the company infoteam Software GmbH (http://www.infoteam.de). This programming system has been extended and adjusted for the PLCcore-F407 by the company SYS TEC electronic GmbH. Hence, it is possible to program the PLCcore-F407 graphically in KOP/FUB, AS and CFC or textually in AWL or ST. Downloading the PLC program onto the module takes place via Ethernet or CANopen – depending on the firmware that is used. Addressing in- and outputs and creating a process image follows the SYS TEC scheme for compact control units. Like all other SYS TEC controls, the PLCcore-F407 supports backward documentation of the PLC program as well as the debug functionality including watching and setting variables, single cycles, breakpoints and single steps.

4 Development Kit PLCcore-F407

4.1 Overview

The Development Kit PLCcore-F407 is a high-capacity, complete package at a particularly favorable price. Based on a compact PLC, it enables the user to perform decentralized, network-compatible automation projects. Moreover, it facilitates the user to get to know the advantages of graphical and textual PLC programming according to IEC 61131-3 – compared to conventional programming languages.
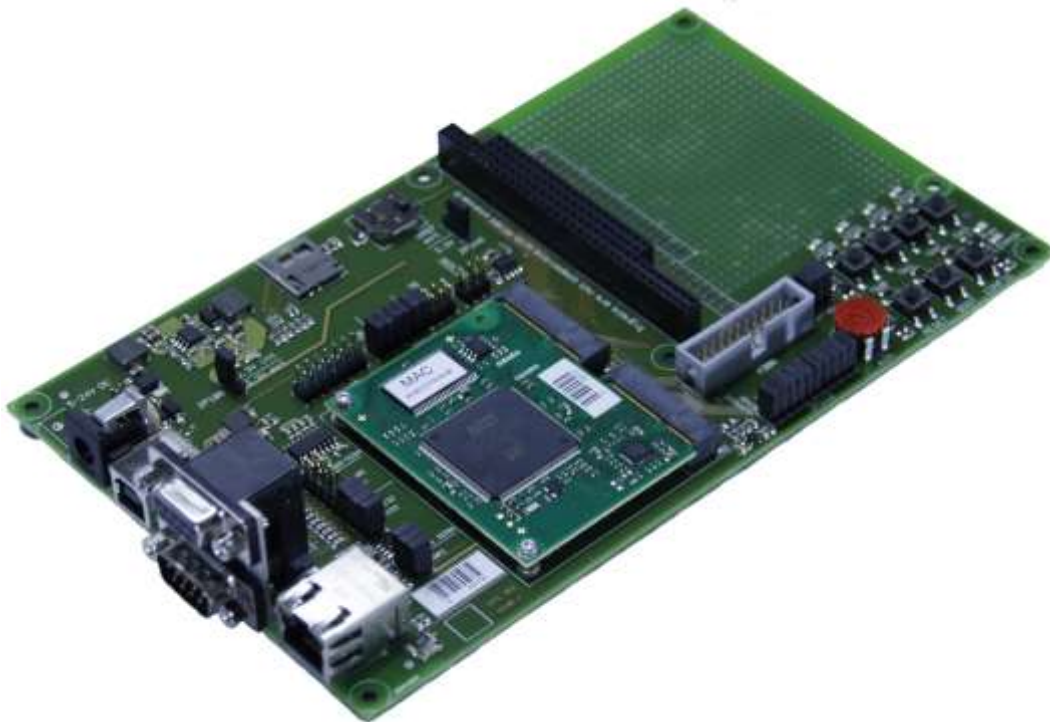
Figure 2: Development Kit PLCcore-F407

The Development Kit PLCcore-F407 ensures quick and problem-free commissioning of the PLCcore-F407. Therefore, it combines all hard- and software components that are necessary to create own applications: the core module PLCcore-F407, the corresponding Development Board containing I/O periphery and numerous interfaces, the *OpenPCS* IEC 61131 programming system as well as further accessory. Thus, the Development Kit forms the ideal platform for developing user-specific applications based on the PLCcore-F407. It allows for a cost-efficient introduction into the world of decentralized automation technology. All components included in the Kit enable in- and output extensions of the PLCcore-F407 through CANopen-I/O-assemblies. Thus, the Development Kit may also be used for projects that require PLC with network connection.

The Development Kit PLCcore-F407 contains the following hardware components:

- PLCcore-F407
- Development Board for the PLCcore-F407
- USB cable
- CD with programming software, examples, documentation and other tools

The Development Board included in the Kit facilitates quick commissioning of the PLCcore-F407 and simplifies the design of prototypes for user-specific applications based on this module. Among other equipment, the Development Board comprises different power supply possibilities, Ethernet interface,

CAN interface, 4 push buttons and 4 LED as control elements for digital in- and outputs and it comprises a potentiometer for the analog input. Signals that are available from plug connectors of the PLCcore-F407 are linked to pin header connectors and enable easy connection of own peripheral circuitry. Hence, the Development Board forms an ideal experimentation and testing platform for the PLCcore-F407.

The *OpenPCS* IEC 61131 programming system included in the Kit serves as software development platform and as debug environment for the PLCcore-F407. Thus, the module can either be programmed graphically in KOP/FUB, AS and CFC or textually in AWL or ST. Downloading the PLC program onto the module takes place via Ethernet or CANopen – depending on the firmware that is used. High-capacity debug functionality such as watching and setting variables, single cycles, breakpoints and single steps simplify the development and commissioning of user software for this module.

4.2 Electric commissioning of the Development Kit PLCcore-F407

The Development Kit PLCcore-F407 can be simply powered via USB (P200). An external power adapter is only required for running the Development Kit PLCcore-F407 stand alone, without connection to a Host PC. An USB cable is already included in the Kit delivery. For commissioning the Kit, it is essential to use at least USB (P200, for power supply and configuration via virtual serial interface, see section 11) and – depending on the firmware version – either ETH0 (X200, for order number 3390095) or CAN0 (P201A, for order number 3390094). Table 2 provides an overview over the connections of the Development Kit PLCcore-F407.

Table 2: Connections of the Development Kit PLCcore-F407

Connection	Labeling on the Development Board	Remark
USB (Power supply and COM0/RS232)	P200	USB is used for: - Power supply and - Serial connection to run configuration shell (e.g. setting CAN and IP configuration, see section 11)
Alternative Power Supply	X101 or X102	An external power supply (5..24V) can be used optionally for running the Development Kit PLCcore-F407 stand alone, without connection to a Host PC.
ETH0 (Ethernet)	X200	This interface serves as communication interface with the Programming PC and is necessary for the program download (PLCcore-F407/Z5, order number 3390095), besides can be used freely for the user program.
CAN0 (CAN)	P201A	This interface serves as communication interface with the Programming PC and is necessary for the program download (PLCcore-F407/Z4, order number 3390094), besides can be used freely for the user program.
COM1 (RS485)	P201B	This interface can be used freely for general operation of the user program.
COM2 (RS232)	P201B	This interface can be used freely for general operation of the user program.

Figure 3 shows the positioning of the most important connections of the Development Board for the PLCcore-F407.
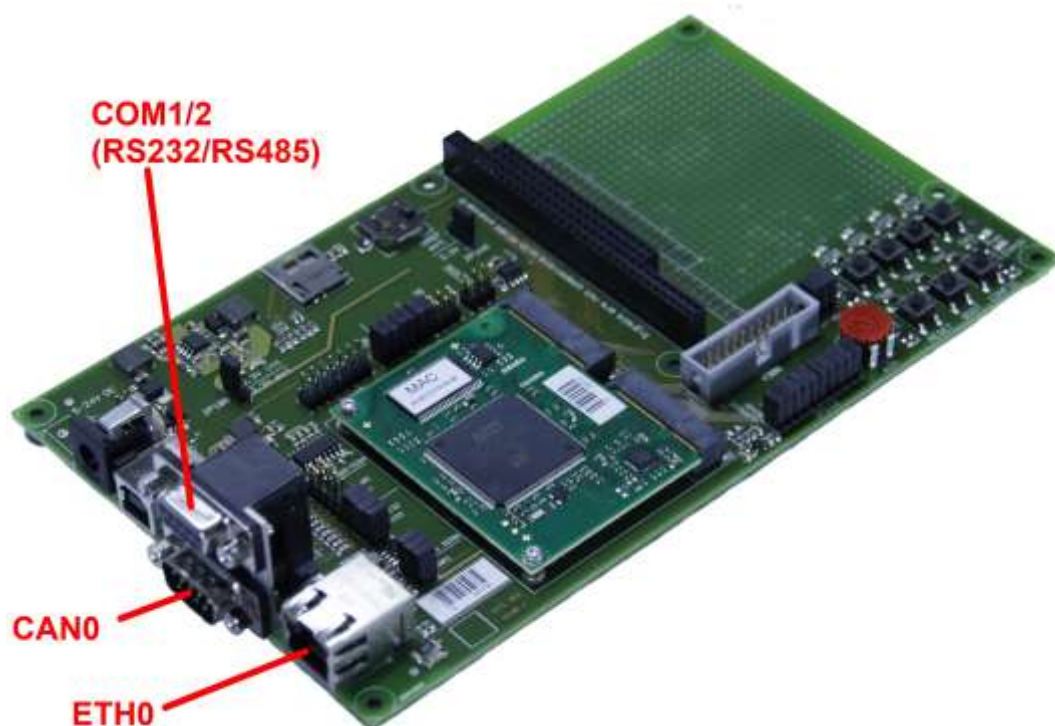
Figure 3: Positioning of most important connections on the Development Board for the PLCcore-F407

Advice
Upon commissioning, cables for Ethernet (ETH0 / X200) and RS232 (COM0 / P201B) must be connected prior to activating the power supply (USB / P200 or external power supply via X101 / X102).

4.3 Control elements of the Development Kit PLCcore-F407

The Development Kit PLCcore-F407 allows for easy commissioning of the PLCcore-F407. It has available various control elements to configure the module and to simulate in- and outputs for the usage of the PLCcore-F407 as PLC kernel. In Table 3 control elements of the Development Board are listed and their meaning is described.

Table 3: Control elements of the Development Board for the PLCcore-F407

Control element	Name	Meaning	
Pushbutton 0	SW0	Digital Input DI0	(Process Image: %IX0.0)
Pushbutton 1	SW1	Digital Input DI1	(Process Image: %IX0.1)
Pushbutton 2	SW2	Digital Input DI2	(Process Image: %IX0.2)
Pushbutton 3	SW3	Digital Input DI3	(Process Image: %IX0.3)
LED 0	LED0	Digital Output DO0	(Process Image: %QX0.0)
LED 1	LED1	Digital Output DO1	(Process Image: %QX0.1)
LED 2	LED2	Digital Output DO2	(Process Image: %QX0.2)
LED 3	LED3	Digital Output DO3	(Process Image: %QX0.3)
Poti (ADC)	AIN	Analog Input AI0	(Process Image: %IW8.0)
Run-LED	RUN	Display of activity state of the PLC (see section 7.6.1)	
Error-LED	ERROR	Display of error state of the PLC (see section 7.6.2)	

4.4 Optional accessory

4.4.1 USB-RS232 Adapter Cable

The SYS TEC USB-RS232 Adapter Cable (order number 3234000) provides a RS232 interface via an USB-Port of the PC. Together with a terminal program, it enables the configuration of the PLCcore-F407 from PCs, e.g. laptop computers which do not have RS232 interfaces any more.

Figure 4: SYS TEC USB-RS232 Adapter Cable

5 Electrical characteristic of the PLCcore-F407

5.1 General operating conditions

The following table shows the general operating conditions together with their respective operating range of the PLCcore-F407.

Table 4: General operating conditions

Symbol	Parameter	Min	Typ	Max	Unit
3V3	Standard operating voltage 3V3	3.135	3.3	3.465	V
I$_{3V3}$	Total current consumption 3V3-Domain	-	250	450	mA
VREF	Reference voltage (3V3-VREF < 1.2V!)	2.3	-	3V3	V
I$_{VREF}$	VREF DC current consumption	-	300	500	µA
VBAT	Backup operating voltage (internal, external RTC)	1.8	-	3.6	V
IVBAT	Backup domain supply current	-	-	TBD	µA
T$_A$	Operating Temperature Range	-40	-	85	°C

5.2 I/O characteristic

The tables below show the most important characteristics of analog and digital in- and outputs.

Table 5: Analog input characteristic

Symbol	Parameter	Min	Typ	Max	Unit
V$_{AIN}$	Conversion voltage range	0	-	VREF	V
Resolution			12		bit
R$_{ADC}$	Sampling switch resistance			6	kΩ
R$_{AIN}$	External input impedance			50	kΩ
C$_{ADC}$	Internal sample and hold capacitor		4		pF
t$_S$	Sampling time		2.67		µs
t$_{Cycle}$	Cycle time to convert the 8 AI channels		30.5		µs
ET	Total unadjusted error		±3	±5	LSB
EO	Offset error		±2	±3	LSB
I$_{INJ+}$	Positive injection current (1)			5	mA
I$_{INJ-}$	Negative injection current (1)			0	mA

(1) ADC accuracy versus negative injection current:
Injecting a negative current on any analog input pins should be avoided as this significantly reduces the accuracy of the conversion being performed on another analog input. It is recommended to add a Schottky diode (pin to ground) to analog pins which may potentially inject negative currents.

Table 6: Analog output characteristic

Symbol	Parameter	Min	Typ	Max	Unit
AO min	Lower AO voltage with buffer OFF	-	0.5	-	mV
AO min	Higher AO voltage with buffer OFF	-	-	VREF-1LSB	V
R_O	Impedance output with buffer OFF	-	-	15	kΩ
C_{LOAD}	Capacitive load	-	-	50	pF

Table 7: Digital input characteristic

Symbol	Parameter	Min	Typ	Max	Unit
V_{IL}	Input low level voltage	GND-0.3	0	0.3 x 3V3	V
V_{IH}	FT I/O input high level voltage	0.7 x 3V3	3.3	5.2	V
V_{IH}	TC I/O input high level voltage	0.7 x 3V3	3.3	3.6	V
V_{hys}	IO FT Schmitt trigger voltage mV hysteresis	0.05 x 3V3			V
I_{lkg}	I/O FT input leakage current	-	-	3	µA
C_{IO}	I/O pin capacitance	-		5	pF
$t_{f(IO)out}$	Output high to low level fall time and output low to thigh level rise time for C_L=10 pF			2.5	ns

Table 8: Digital output characteristic

Symbol	Parameter	Min	Typ	Max	Unit
V_{OL}	Output low level voltage for an I/O pin when 8 pins are sunk at same time (I_{IO}=8mA)	-	-	0.4	V
V_{OH}	Output high level voltage for an I/O pin when 8 pins are sourced at same time	2.4	-	-	V
I_{IO}	Output current sunk/source by any I/O pin	-	-	25	mA
$I_{IOmaxsource}$	The maximum sum of I_{IO} sourced by the device	-	-	150	mA
$I_{IOmaxsource}$	The maximum sum of I_{IO} sunk by the device	-	-	150	mA

5.3 Ethernet characteristics

In the table below you will find important information about the Ethernet characteristics of the PLCcore-F407.

Table 9: Ethernet receive or transmit signal characteristic

Symbol	Parameter	Min	Typ	Max	Unit
V_{IHETH}	Input high level voltage	2	-	-	V
VILETH	Input low level voltage	-	-	0.8	V
V_{OHETH}	Output high level voltage	2.40	-		V
V_{OLETH}	Output low level voltage	-	-	0.4	V
I_{INETH}	Input Current	-	-10	10	µA
|Ioz|	Output Tri-State Leakage	-	-	10	µA

Table 10: Ethernet LED output characteristic

| Symbol	Parameter	Min	Typ	Max	Unit
I_{LED}	LED output drive current	-	8	-	mA

6 Pinout of the PLCcore-F407

The PLCcore-F407 provides board I/O functionalities through simple connector footprint using two pin card edge connectors (X301 and X303, see Figure 6). Appropriate pin header connectors corresponding to the PLCcore-F407 are available from company "Tyco" and "JAE":

Tyco name: 56 pol Mini PCIexpress Connector
Tyco order number: 1717831

JAE name: 56 pol Mini PCIexpress Connector
JAE order number: MM60-52B1-E1-R650

It is also recommended to use spacers to fix the module onto the baseboard. They are also available from company "JAE":

JAE name: Spacer
JAE order number: NT4R1600
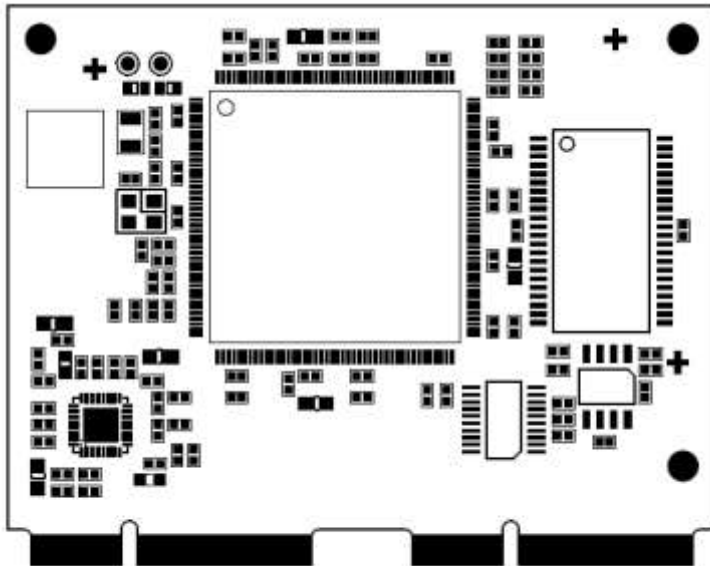
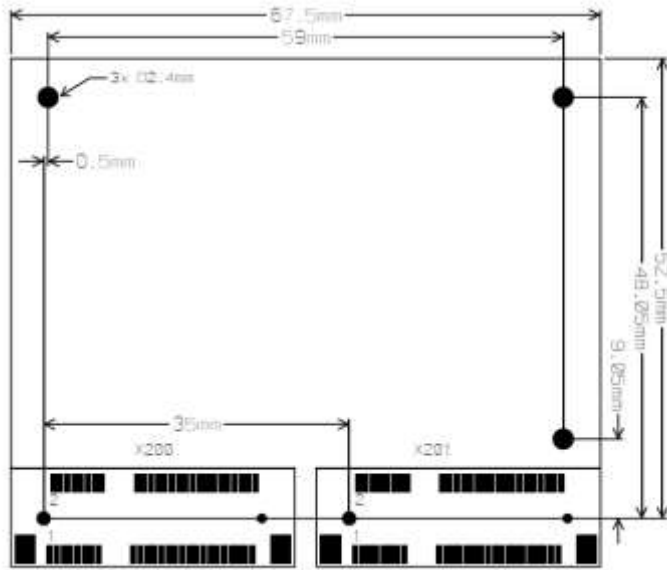

Figure 5: Footprint of the PLCcore-F407

Figure 6: Pinout of the PLCcore-F407 - top view

Figure 6 exemplifies the position of the pin card edge connectors (X200 and X201) on the PLCcore-F407. The figure below shows the mapping between the connectors of the ECUcore-F407 and the Development Kit PLCcore-F407.
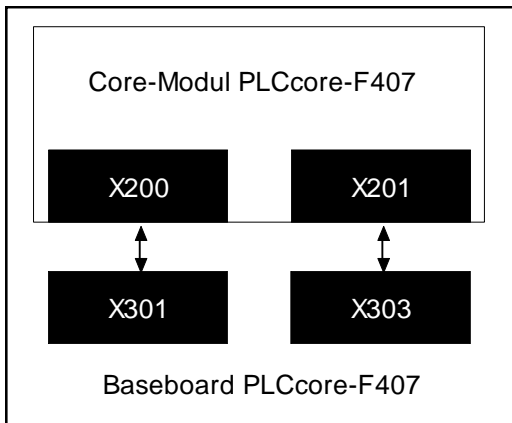

Figure 7: Connector Mapping between PLCcoreF407 and ECUcore-F407

A complete connection assignment of this module is listed up in Table 11.

Table 11: Connections of the PLCcore-F407, completely, sorted by connection pin

Signal	Pin	Pin	Signal	Signal	Pin	Pin	Signal
+3V3_Core	X301/1	X301/2	SIO1	+3_3V_Core	X303/1	X303/2	CNT0
GND	X301/3	X301/4	SIO1	GND-	X303/3	X303/4	CNT0_DIR
ETH_RXM	X301/5	X301/6	/RUN-LED	JTMS	X303/5	X303/6	CNT1
ETH_RXP	X301/7	X301/8	/ERROR-LED	JTCK	X303/7	X303/8	CND1_DIR
GND	X301/9	X301/10	ETH_LINK/ACT	DO16	X303/9	X303/10	DI18
ETH_TXM	X301/11	X301/12	ETH_SPEED	DO17	X303/11	X303/12	DI20

Signal	Pin	Pin	Signal		Signal	Pin	Pin	Signal
ETH_TXP	X301/13	X301/14	VREF		DI16	X303/13	X303/14	VBAT
GND	X301/15	X301/16	GND		DO18	X303/15	X303/16	GND
AO0	X301/17	X301/18	SIO0_TX		DO19	X303/17	X303/18	DI8
AO1	X301/19	X301/20	SIO0_RX		DI17	X303/19	X303/20	DI9
GND	X301/21	X301/22	SIO1		GND	X303/21	X303/22	DI10
AIN0	X301/23	X301/24	CAN1		SIO2	X303/23	X303/24	DI11
AIN1	X301/25	X301/26	CAN1		SIO2	X303/25	X303/26	GND
GND	X301/27	X301/28	PWM0		SIO2	X303/27	X303/28	DI12
DI0	X301/29	X301/30	PWM1		SIO2	X303/29	X303/30	DI13
DI1	X301/31	X301/32	GND		GND	X303/31	X303/32	DI14
DI2	X301/33	X301/34	DO0		CAN2	X303/33	X303/34	DI15
DI3	X301/35	X301/36	DO1		CAN2	X303/35	X303/36	DO8
GND	X301/37	X301/38	DO2		GND	X303/37	X303/38	DO9
DI4	X301/39	X301/40	DO3		AIN2	X303/39	X303/40	DO10
DI5	X301/41	X301/42	GND		AIN3	X303/41	X303/42	DO11
DI6	X301/43	X301/44	DO4		AIN4	X303/43	X303/44	GND
DI7	X301/45	X301/46	DO5		AIN5	X303/45	X303/46	DO12
/CONFIG	X301/47	X301/48	DO6		AIN6	X303/47	X303/48	DO13
BOOT0	X301/49	X301/50	DO7		AIN7	X303/49	X303/50	DO14
/RESET	X301/51	X301/52	/RST_OUT		GND3	X303/51	X303/52	DO15

Table 12 is a subset of Table 11 and only includes all in- and outputs of the PLCcore-F407 sorted by their function.

Table 12: Connections of the PLCcore-F407, only I/O, sorted by function

Connector	I/O structure	PLC Function 1	PLC Function 2 A=alternative, S=simultaneous	Pull Up / Pull Down
X301/29	FT/DI	DI0 (Pushbutton SW0)		Pull Down
X301/31	FT/DI	DI1 (Pushbutton SW1)		Pull Down
X301/33	FT/DI	DI2 (Pushbutton SW2)		Pull Down
X301/35	FT/DI	DI3 (Pushbutton SW3)		Pull Down
X301/39	FT/DI	DI4		Pull Down
X301/41	FT/DI	DI5		Pull Down
X301/43	FT/DI	DI6		Pull Down
X301/45	FT/DI	DI7		Pull Down
X303/18	FT/DI	DI8		Pull Down
X303/20	FT/DI	DI9		Pull Down
X303/22	FT/DI	DI10		Pull Down
X303/24	FT/DI	DI11		Pull Down
X303/28	FT/DI	DI12		Pull Down
X303/30	FT/DI	DI13		Pull Down
X303/32	FT/DI	DI14		Pull Down
X303/34	FT/DI	DI15		Pull Down

X303/13	FT/DI	DI16		Pull Down
X303/19	FT/DI	DI17	A: SD-Card	Pull Down
X303/10	FT/DI	DI18	A: SD-Card	Pull Down
X303/12	FT/DI	DI19	A: SD-Card	Pull Down
X303/2	FT/DI	DI20	S: CNT0	Pull Down
X303/4	FT/DI	DI21	S: CNT0_DIR	Pull Down
X303/6	FT/DI	DI22	S: CNT1	Pull Down
X303/8	FT/DI	DI23	S: CNT1_DIR	Pull Down
X301/34	FT/DO	DO0 (User-LED0)		No Pull
X301/36	FT/DO	DO1 (User-LED1)		No Pull
X301/38	FT/DO	DO2 (User-LED2)		No Pull
X301/40	FT/DO	DO3 (User-LED3)		No Pull
X301/44	FT/DO	DO4		No Pull
X301/46	FT/DO	DO5		No Pull
X301/48	FT/DO	DO6		No Pull
X301/50	FT/DO	DO7		No Pull
X303/36	FT/DO	DO8		No Pull
X303/38	FT/DO	DO9		No Pull
X303/40	FT/DO	DO10		No Pull
X303/42	FT/DO	DO11		No Pull
X303/46	FT/DO	DO12		No Pull
X303/48	FT/DO	DO13		No Pull
X303/50	FT/DO	DO14		No Pull
X303/52	FT/DO	DO15		No Pull
X303/9	FT/DO	DO16		No Pull
X303/11	FT/DO	DO17		No Pull
X303/15	FT/DO	DO18		No Pull
X303/17	FT/DO	DO19		No Pull
X301/28	FT/DO	DO20	A: PWM0	No Pull
X301/30	FT/DO	DO21	A: PWM1	No Pull
X301/23	FT/AI	AIN0		No Pull
X301/25	FT/AI	AIN1		No Pull
X303/39	FT/AI	AIN2		No Pull
X303/41	FT/AI	AIN3		No Pull
X303/43	FT/AI	AIN4		No Pull
X303/45	FT/AI	AIN5		No Pull
X303/47	FT/AI	AIN6		No Pull
X303/49	FT/AI	AIN7		No Pull
X301/17	TC/AO	AO0		No Pull
X301/19	TC/AO	AO1		No Pull

7 PLC Functionality of the PLCcore-F407

7.1 System start of the PLCcore-F407

By default, the PLCcore-F407 loads all necessary firmware components upon Power-on or Reset and starts running the PLC program afterwards. Hence, the PLCcore-F407 is suitable for the usage in autarchic control systems. In case of power breakdown, such systems resume the execution of the PLC program independently and without user intervention. Figure 8 illustrates the system start in detail:
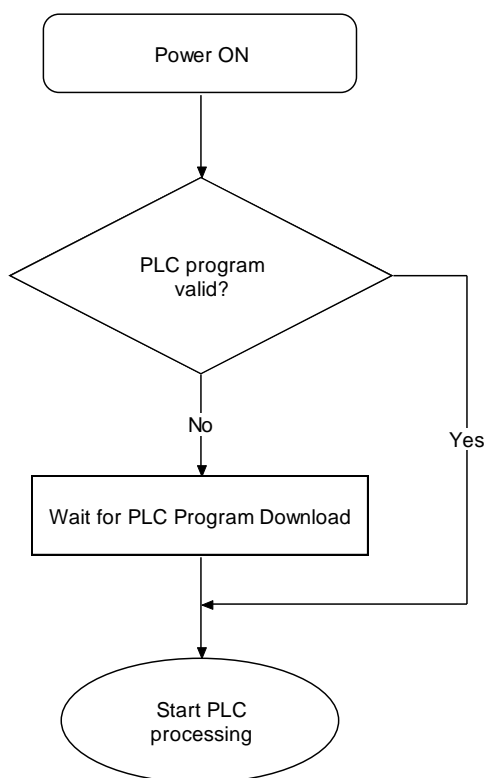

Figure 8: System start of the PLCcore-F407

If the firmware has found a valid control program in the non-volatile memory, this program is restarted and processed. Otherwise the reactivated program is ready for execution and the PLCcore-F407 is in stop mode.

7.2 Programming the PLCcore-F407

The PLCcore-F407 is programmed with IEC 61131-3-conform *OpenPCS* programming environment. There exist additional manuals about *OpenPCS* that describe the handling of this programming tool. Those are part of the software package *"OpenPCS"*. All manuals relevant for the PLCcore-F407 are listed in Table 1.

The PLCcore-F407 firmware is based on standard firmware for SYS TEC`s compact control units. Consequently, it shows identical properties like other SYS TEC control systems. This affects

especially the process image setup (see section 7.3) as well as the functionality of control elements (Run-LED, Error-LED).

Depending on the firmware version used, PLCcore-F407 firmware provides numerous function blocks to the user to access communication interfaces. Table 13 specifies the availability of FB communication classes (SIO, CAN, UDP) for different PLCcore-F407 firmware versions.

Table 13: Support of FB communication classes for different types of the PLCcore

Type of Interface	PLCcore-F407/Z3 Art. no: 3390093	PLCcore-F407/Z4 Art. no: 3390094	PLCcore-F407/Z5 Art. no: 3390095	Remark
CAN	-	X	X	FB description see manual L-1008
UDP	-	X	X	FB description see manual L-1054
SIO	X	X	X	FB description see manual L-1054

Table 36 in Appendix A contains a complete listing of firmware functions and function blocks that are supported by the PLCcore-F407.

Detailed information about using the CAN interfaces in connection with CANopen is provided in section 7.7.

7.3 Process image of the PLCcore-F407

7.3.1 Local In- and Outputs

Compared to other SYS TEC compact control systems, the PLCcore-F407 obtains a process image with identical addresses. All in- and outputs listed in Table 14 are supported by the PLCcore-F407.

Table 14: Assignment of in- and outputs to the process image of the PLCcore-F407

I/O of the PLCcore-F407	Address and Data type in the Process Image	
DI0 ... DI7	**%IB0.0** %IX0.0 … %IX0.7	as Byte with DI0 … DI7 as single Bit for each input
DI8 ... DI15	**%IB1.0** %IX1.0 … %IX1.7	as Byte with DI8 … DI15 as single Bit for each input
DI16 … DI23	**%IB2.0** %IX2.0 … %IX2.7	as Byte with DI16 … DI23 as single Bit for each input
AIN0	**%IW8.0**	15Bit + sign (0 … +32767)
AIN1	**%IW10.0**	15Bit + sign (0 … +32767)
AIN2	**%IW12.0**	15Bit + sign (0 … +32767)
AIN3	**%IW14.0**	15Bit + sign (0 … +32767)
AIN4	**%IW16.0**	15Bit + sign (0 … +32767)
AIN5	**%IW18.0**	15Bit + sign (0 … +32767)
AIN6	**%IW20.0**	15Bit + sign (0 … +32767)

I/O of the PLCcore-F407	Address and Data type in the Process Image	
AIN7	**%IW22.0**	15Bit + sign (0 … +32767)
C0	**%ID40.0**	31Bit + sign (-2^{31} - 2^{31} -1) Counter: counter input: DI2.4, direction: DI2.5, see section 7.5.1 AB Encoder: channel A: DI2.4, channel B: DI0.4, see section 7.5.1
C1	**%ID44.0**	31Bit + sign (-2^{31} - 2^{31} -1) Counter: counter input: DI2.6, direction: DI2.7, see section 7.5.1 AB Encoder: channel A: DI2.6, channel B: DI0.5, see section 7.5.1
On-board Temperature Sensor	**%ID72.0**	31Bit + sign as 1/10000 °C
DO0 ... DO7	**%QB0.0** %QX0.0 … %QX0.7	as Byte with DO0 … DO7 as single Bit for each output
DO8 … DO15	**%QB1.0** %QX1.0 … %QX1.7	as Byte with DO8 … DO15 as single Bit for each output
DO16 … DO21	**%QB2.0** %QX2.0 … %QX2.5	as Byte with DO16 … DO21 as single Bit for each output
AO0	**%QW8.0**	15Bit + sign (0 … +32767)
AO1	**%QW10.0**	15Bit + sign (0 … +32767)
P0	**%QX0.0** Impulse output: DO0, see section 7.5.2	(default value for inactive generator)
P1	**%QX0.1** Impulse output: DO1, see section 7.5.2	(default value for inactive generator)

In- and outputs of the PLCcore-F407 are not negated in the process image. Hence, the H-level at one input leads to value "1" at the corresponding address in the process image. Contrariwise, value "1" in the process image leads to an H-level at the appropriate output.

7.3.2 Calculate current measurement

The baseboard of the Development Kit PLCcore-F407 features a current measurement circuit. In the process image, the value is located at AIN1. This value can be interpreted as follows:

(0.702127659574 / 32767) • Value of AIN1 = Current Consumption
Current Measurement • 0.1k = Current Consumption in A

AIN1 typically has a value around 6800. Hence, the typical power consumption is 0.145 A / 145 mA. Please note that the value of AIN1 also depends on the current load of the module. For instance, enabling LED's increases the consumption.

7.3.3 In- and outputs of user-specific baseboards

The connection lines leading towards the outside provides to the user most effective degrees of freedom for designing the in-/output circuit of the PLCcore-F407. Therewith, all in- and outputs of the PLCcore-F407 can be flexibly adjusted to respective requirements. This implicates that the process image of PLCcore-F407 is significantly conditioned by the particular, user-specific in-/output circuit. Please contact our support employee if you are interested in this option:

support@systec-electronic.com

7.4 Communication interfaces

7.4.1 Serial interfaces

The PLCcore-F407 features 3 serial interfaces (COM0…COM2).

COM0: This interface is used for communication between PC and PLC to allow for:
- Configuration of module settings (see section 11)
- Firmware updates (see section 7.9)
- PLC program download and debugging (Art. no: 3390093 only)

On the Development Kit PLCcore-F407 the interface SIO0 is tunneled via USB on connector P200.

COM1: This interface can be used freely for general operation of the user program.

COM2: This interface can be used freely for general operation of the user program.

7.4.2 CAN interfaces

The PLCcore-F407 features 2 CAN interfaces (CAN0 and CAN1). CAN interfaces allow for data exchange with other devices via network variables and they are accessible from a PLC program via function blocks of type *"CAN_Xxx"* (see section 7.7 and *"User Manual CANopen Extension for IEC 61131-3"*, Manual no.: L-1008).

Section 7.7 provides detailed information about the usage of the CAN interface in connection with CANopen.

7.4.3 Ethernet interfaces

The PLCcore-F407 features 1 Ethernet interface (ETH0). It serves as service interface to administer the PLCcore-F407 and enables data exchange with other devices. The interface is accessible from a PLC program via function blocks of type *"LAN_Xxx"* (see manual *"SYS TEC-specific Extensions for OpenPCS / IEC 61131-3"*, Manual no.: L-1054).

7.5 Specific peripheral interfaces

7.5.1 Counter inputs

The PLCcore-F407 features 2 fast counter inputs (C0 and C1). Prior to its usage, all counter inputs must be parameterized via function block *"CNT_FUD"* (see manual *"SYS TEC-specific Extensions for OpenPCS / IEC 61131 3"*, Manual no.: L 1054). Afterwards, in a PLC program the current counter value is accessible via process image (see Table 14 in section 7.3.1) or via function block *"CNT_FUD"*. Table 15 lists the allocation between counter channels and inputs.

Please note that the counting direction is determined once during initialization. So, if hardware direction control is used, the corresponding pin is checked only during initialization. The direction cannot be changed during runtime, except by calling *"CNT_FUD"* again with the desired direction or hardware detection.

Table 15: Allocation between counter channels and inputs

Counter channel	Counter input	Optional direction input	Counter value in process image
C0[1]	C0 (DI20) %IX2.4	DI21 %IX2.5	%ID40.0
C1	C1 (DI22) %IX2.6	DI23 %IX2.7	%ID44.0

The theoretical maximum frequency for counter inputs is 24 MHz. Practically the frequency depends on the number of interrupts done before the counter interrupt is invoked.

Counter C0 and C1 are also supporting AB encoding in X4 mode. For using the AB encoding mode, it is necessary to use the counter input for channel A and either DI4 or DI5 (depending on the used counter input) for channel B. Please see Table 16 for more information regarding the channel mapping. Using AB encoding will automatically disable the corresponding digital input. Reading them will return 0 for the corresponding digital input.

Table 16: Allocation between counter channels and AB inputs

AB Encoder	Channel A	Channel B	Counter value in process image
C0[1]	C0 (DI20) %IX2.4	DI4 %IX0.4	%ID40.0
C1	C1 (DI22) %IX2.6	DI5 %IX0.5	%ID44.0

[1] Hardware direction cannot be used while SD card mode is enabled. See section 7.9 more information.

7.5.2 Pulse outputs

To release PWM signal sequences, the PLCcore-F407 features 2 pulse outputs (P0 and P1). Prior to its usage, all pulse outputs must be parameterized using function block *"PTO_PWM"* (see manual *"SYS TEC-specific Extensions for OpenPCS / IEC 61131 3"*, Manual no.: L 1054).

After the impulse generator is started, it takes over the control of respective outputs. If the impulse generator is deactivated, the respective output adopts the corresponding value that is filed in the process image for this output (see Table 14 in section 7.3.1). Table 17 lists the allocations between impulse channels and outputs.

Table 17: Allocation between impulse channels and outputs

Impulse channel	Impulse output
P0	P0 (DO20) %QX2.4
P1	P1 (DO21) %QX2.5

Table 18 shows the characteristics of the pulse width modulation and pulse train outputs. Please note that using PTO to generate more than 255 pulses requires reloading the counting register of the corresponding timer. Hence the higher the frequency of PTO output the more inaccurate it becomes because of the required reloading. Table 18 shows the recommended frequencies for PTO and the minimum/maximum output for PWM.

Table 18: Characteristics of PWM/PTO output

Function	Resolution	Max. Cycle Time	Min. Cycle Time
Pulse Width Modulation (PWM)	32 Bit	65535 ms (15 mHz)	2 µs (500 kHz)
Pulse Train Output (PTO)	16 Bit	65 ms (15384 mHz)	13 µs (77 kHz)

7.6 Control and display elements

7.6.1 Run-LED (green)

The module connection *"/Run-LED"* (see Table 11) is designed for connecting a Run-LED. This Run-LED provides information about the activity state of the control system. The activity state is shown through different modes:

Table 19: Display status of the Run-LED

LED Mode	PLC Activity State
Off	The PLC is in state *"Stop"*: • the PLC does not have a valid program, • the PLC has received a stop command from the *OpenPCS* programming environment or • the execution of the program has been canceled due to an internal error
Quick flashing in relation 1:8 to pulse	The PLC is on standby but is not yet executing: • The PLC has received a start command from the *OpenPCS* programming environment but the local Run/Stop switch is still positioned to *"Stop"*
Slow flashing in relation 1:1 to pulse	The PLC is in state *"Run"* and executes the PLC program.
Quick flashing in relation 1:1 to pulse	The PLC is in mode *"Reset"*.

7.6.2 Error-LED (red)

Module connection *"/Error-LED"* (see Table 11) is designed for connecting an Error-LED. This Error-LED provides information about the error state of the control system. The error state is represented through different modes:

Table 20: Display status of the Error-LED

LED Mode	PLC Error State
Off	No error has occurred; the PLC is in normal state.
Permanent light	A severe error has occurred: • The PLC was started using an invalid configuration (e.g. CAN node address 0x00) and had to be stopped or • A severe error occurred during the execution of the program and caused the PLC to independently stop its state *"Run"* (division by zero, invalid Array access, ...), see below
Slow flashing in relation 1:1 to pulse	A network error occurred during communication to the programming system; the execution of a running program is continued. This error state will be reset independently by the PLC as soon as further communication to the programming system is successful.
Quick flashing in relation 1:1 to pulse	The PLC is in mode *"Reset"*.
Quick flashing in relation 1:8 to pulse	The PLC is on standby, but is not yet running: • The PLC has received a start command from the *OpenPCS* programming environment but the local Run/Stop switch is positioned to *"Stop"*

In case of severe system errors such as division by zero of invalid Array access, the control system passes itself from state *"Run"* into state *"Stop"*. This is recognizable by the permanent light of the Error-LED (red). In this case, the error cause is saved by the PLC and is transferred to the computer and shown upon next power-on.

7.7 Using CANopen for CAN interfaces

The PLCcore-F407 features 2 CAN interfaces (CAN0 and CAN1), usable as CANopen Manager (conform to CiA Draft Standard 302).

The CAN interface allow for data exchange with other devices via network variables and is usable from a PLC program via function blocks of type *"CAN_Xxx"*. More details are included in *"User Manual CANopen Extension for IEC 61131-3"*, Manual no.: L-1008.

The CANopen services **PDO** (*P*rocess *D*ata *O*bjects) and **SDO** (*S*ervice *D*ata *O*bjects) are two separate mechanisms for data exchange between single field bus devices. Process data sent from a node (**PDO**) are available as broadcast to interested receivers. PDOs are limited to 1 CAN telegram and therewith to 8 Byte user data maximum because PDOs are executed as non-receipt broadcast messages. On the contrary, **SDO** transfers are based on logical point-to-point connections ("Peer to Peer") between two nodes and allow the receipted exchange of data packages that may be larger than 8 Bytes. Those data packages are transferred internally via an appropriate amount of CAN telegrams. Both services are applicable for interface CAN0 as well as for CAN1 of the PLCcore-F407.

SDO communication basically takes place via function blocks of type *"CAN_SDO_Xxx"* (see *"User Manual CANopen Extension for IEC 61131-3"*, Manual no.: L-1008). Function blocks are also available for PDOs (*"CAN_PDO_Xxx"*). Those should only be used for particular cases in order to also activate non-CANopen-conform devices. For the application of PDO function blocks, the CANopen configuration must be known in detail. The reason for this is that the PDO function blocks only use 8 Bytes as input/output parameter, but the assignment of those Bytes to process data is subject to the user.

Instead of PDO function blocks, network variables should mainly be used for PDO-based data exchange. Network variables represent the easiest way of data exchange with other CANopen nodes. Accessing network variables within a PLC program takes place in the same way as accessing internal, local variables of the PLC. Hence, for PLC programmers it is not of importance if e.g. an input variable is allocated to a local input of the control or if it represents the input of a decentralized extension module. The application of network variables is based on the integration of DCF files that are generated by an appropriate CANopen configurator. On the one hand, DCF files describe communication parameters of any device (CAN Identifier, etc.) and on the other hand, they allocate network variables to the Bytes of a CAN telegram (mapping). The application of network variables only requires basic knowledge about CANopen.

In a CANopen network, exchanging PDOs only takes place in status *"OPERATIONAL"*. If the PLCcore-F407 is not in this status, it does not process PDOs (neither for send-site nor for receive-site) and consequently, it does not update the content of network variables. The CANopen Manager is in charge of setting the operational status *"OPERATIONAL"*, *"PRE-OPERATIONAL"* etc. (mostly also called "CANopen Master"). In typical CANopen networks, a programmable node in the form of a PLC is used as CANopen-Manager. The PLCcore-F407 is optionally able to take over tasks of the CANopen Manager.

As CANopen Manager, the PLCcore-F407 is able to parameterize the CANopen I/O devices ("CANopen-Slaves") that are connected to the CAN bus. Therefore, upon system start via SDO it transfers DCF files generated by the CANopen configurator to the respective nodes.

7.7.1 CAN interface CAN0

Interface CAN0 features a dynamic object dictionary. This implicates that after activating the PLC, the interface does not provide communication objects for data exchange with other devices. After downloading a PLC program (or its reload from the non-volatile storage after power-on), the required communication objects are dynamically generated according to the DCF file which is integrated in the PLC project. Thus, CAN interface CAN0 is extremely flexible and also applicable for larger amount of data.

For the PLC program, all network variables are declared as *"VAR_EXTERNAL"* according to IEC61131-3. Hence, they are marked as „outside of the control", e.g.:

```
VAR_EXTERNAL
    NetVar1 : BYTE ;
    NetVar2 : UINT ;
END_VAR
```

A detailed procedure about the integration of DCF files into the PLC project and about the declaration of network variables is provided in manual *"User Manual CANopen Extension for IEC 61131-3"* (Manual no.: L-1008).

When using CAN interface CAN0 it must be paid attention that the generation of required objects takes place upon each system start. This is due to the dynamic object directory. "Design instructions" are included in the DCF file that is integrated in the PLC project. **Hence, changes to the configuration can only be made by modifying the DCF file.** This implies that after the network configuration is changed (modification of DCF file), the PLC project must again be translated and loaded onto the PLCcore-F407.

7.7.2 CAN interface CAN1

On the contrary to interface CAN0, interface CAN1 is provided as static object dictionary. This means that the amount of network variables (communication objects) and the amount of PDOs available are both strongly specified. During runtime, the configuration of PDOs is modifiable. This implies that communication parameters used (CAN Identifier, etc.) and the allocation of network variables to each

Byte of a CAN telegram (mapping), can be set and modified by the user. Thus, only the amount of objects (amount of network variables and PDOs) is strongly specified in the static object dictionary. Consequently, application and characteristics of objects can be modified during runtime. For this reason, at interface CAN1 the PLCcore-F407 acts as a CANopen I/O device.

Note
CAN1 cannot be used while CAN0 is disabled. Hence, to use CAN1 it is necessary to enable CAN0.

All network variables of the PLC program are available through the marker section of the process image. Therefore, 252 Bytes are usable as input variables and also 252 Bytes as output variables. To enable any data exchange with other CANopen I/O devices, the section of static network variables is mapped to different data types in the object dictionary (BYTE, SINT, WORD, INT, DWORD, DINT). Variables of the different data types are located within the same memory area which means that all variables represent the same physical storage location. Hence, a WORD variable interferes with 2 BYTE variables, a DWORD variable with 2 WORD or 4 BYTE variables. Figure 9 exemplifies the positioning of network variables for CAN1 within the marker section.

Figure 9: Positioning of network variables for CAN1 within the marker section

Table 21 shows the representation of network variables through appropriate inputs in the object dictionary of interface CAN1.

Table 21: Representation of network variables for CAN1 by entries in the object dictionary

OD section	OD variable / EDS input	Data type CANopen	Data type IEC 61131-3
Inputs (inputs for the PLCcore-F407)			
Index 2000H Sub 1 … 252	CAN1InByte0 … CAN1InByte251	Unsigned8	BYTE, USINT
Index 2001H Sub 1 … 252	CAN1InSInt0 … CAN1InSInt251	Integer8	SINT
Index 2010H Sub 1 … 126	CAN1InWord0 … CAN1InWord125	Unsigned16	WORD, UINT
Index 2011H Sub 1 … 126	CAN1InInt0 … CAN1InInt125	Integer16	INT
Index 2020H Sub 1 … 63	CAN1InDword0 … CAN1InDword62	Unsigned32	DWORD, UDINT
Index 2021H Sub 1 … 63	CAN1InDInt0 … CAN1InDInt62	Integer32	DINT
Outputs (outputs for the PLCcore-F407)			
Index 2030H Sub 1 … 252	CAN1OutByte0 … CAN1OutByte251	Unsigned8	BYTE, USINT
Index 2031H Sub 1 … 252	CAN1OutSInt0 … CAN1OutSInt251	Integer8	SINT
Index 2040H Sub 1 … 126	CAN1OutWord0 … CAN1OutWord125	Unsigned16	WORD, UINT
Index 2041H Sub 1 … 126	CAN1OutInt0 … CAN1OutInt125	Integer16	INT
Index 2050H Sub 1 … 63	CAN1OutDword0 … CAN1OutDword62	Unsigned32	DWORD, UDINT
Index 2051H Sub 1 … 63	CAN1OutDInt0 … CAN1OutDInt62	Integer32	DINT

The object dictionary of interface CAN1 in total has available 16 TPDO and 16 RPDO. The first 4 TPDO and RPDO are preconfigured and activated according to the Predefined Connection Set. The first 32 Byte of input and output variables are mapped to those PDOs. Table 22 in detail lists all preconfigured PDOs for interface CAN1.

Table 22: Preconfigured PDOs for interface CAN1

PDO	CAN-ID	Data
1. RPDO	0x200 + NodeID	%MB0.0 … %MB7.0
2. RPDO	0x300 + NodeID	%MB8.0 … %MB15.0
3. RPDO	0x400 + NodeID	%MB16.0 … %MB23.0
4. RPDO	0x500 + NodeID	%MB24.0 … %MB31.0
1. TPDO	0x180 + NodeID	%MB256.0 … %MB263.0
2. TPDO	0x280 + NodeID	%MB264.0 … %MB271.0
3. TPDO	0x380 + NodeID	%MB272.0 … %MB279.0
4. TPDO	0x480 + NodeID	%MB280.0 … %MB287.0

Due to limitation to 16 TPDO and 16 RPDO, only 256 Bytes (2 * 16PDO * 8Byte/PDO) of total 504 Bytes for network variables in the marker section (2 252Bytes) can be transferred via PDO. Irrespective of that it is possible to access all variables via SDO.

The configuration (mapping, CAN Identifier etc.) of interface CAN1 typically takes place via an external Configuration Manager that parameterizes the object dictionary on the basis of a DCF file created by the CANopen configurator. By using default object inputs 1010H und 1011H, the PLCcore-F407 supports the persistent storage and reload of a backed configuration.

Alternatively, the configuration (mapping, CAN Identifier etc.) of the static object dictionary for interface CAN1 can take place from the PLC program by using SDO function blocks. Therefore, inputs *NETNUMBER* and *DEVICE* must be used as follows:

```
NETNUMBER  := 1;          (* Interface CAN1 *)
DEVICE     := 0;          (* local Node     *)
```

7.8 Controller specific PLC Function Blocks

7.8.1 The Function Blocks SIO_*

This group of function blocks controls the serial interfaces *SIO0, SIO1 and SIO2*. The function blocks are described in the manual "L-1054 SYS TEC-specific Extensions for OpenPCS / IEC61131-3".

This section gives additional information about the function block operands PORT and PROTOCOL depending on the interface.

<u>Definition of the Operands:</u>

PORT: Number of serial interface to be used
                0 = SIO0 (COM0, RS232)
                1 = SIO1 (COM1, RS422 / RS485)
                2 = SIO2 (COM2, RS232)

PROTOCOL: The PROTOCOL operand defines whether RS232, XON/XOFF, RS422,
                RS485, RTS/CTS is used.
                0 = RS232 (available for SIO0, SIO2)
                1 = XON/XOFF (available for SIO0, SIO1)
                2 = Hardware Handshake via RTS/CTS (available for SIO2)
                3 = RS485 (available for SIO1)
                4 = RS422 (available for SIO1)

7.9 Using Micro SD-Card

The PLCcore-F407 supports Micro SD-Cards since firmware version 5.08.02.00. The support for Micro SD-Cards must be enabled using the configuration shell before they can be accessed (see section 11.2.7 for how to enable Micro SD-Card support). Additionally, the jumpers of the corresponding development board must be set (see section 9.4).

Micro SD-Cards are not hot pluggable for PLCcore-F407. After enabling the Micro SD-Card support, the Micro SD-Card has to be plugged in before the PLCcore-F407 is turned on. The PLCcore-F407 checks whether a Micro SD-Card is plugged in or not and mounts the Card if found.

To release the Micro SD-Card, the PLCcore-F407 has to be turned off. It is not recommended to remove the Card during runtime.

When enabling Micro SD-Card support, some pins cannot be used because of overlapping functionality (see Table 23). The corresponding interface or functionality returns an error or 0 for digital inputs.

Table 23: Reserved SD-Card pins

Signal	Pin	Pin	Signal
DI18	X303/10	X303/4	CNT0_DIR
DI20	X303/12	X303/17	DO19
SIO1	X301/2	X303/19	DI17
SIO1	X301/4		

8 Updating the PLCcore-F407 Firmware

All necessary firmware components to run the PLCcore-F407 are already installed on the module upon delivery. Hence, firmware updates should only be required in exceptional cases, e.g. to input new software that includes new functionality.

The PLCcore-F407 Firmware includes the PLC Runtime System. The Firmware can only be generated and modified by the producer; it is not identical with the PLC user program which is created by the PLC user. The PLC user program is directly transferred from the *OpenPCS* programming environment onto the module. No additional software is needed.

An update of the PLC Firmware is made via SIO0 of the PLCcore-F407. On the PLCcore-F407 Development Kit the interface SIO0 is tunneled via USB on connector P200. Therefore, connect the Development Kit to the PC using the USB cable delivered with the Kit.

On the PC side the virtual COM port driver delivered with the Kit *("CP210xVCPInstaller.exe")* has to be installed. This will add an additional virtual serial interface to the Host PC. This allows for serial download the new firmware to the PLCcore-F407.
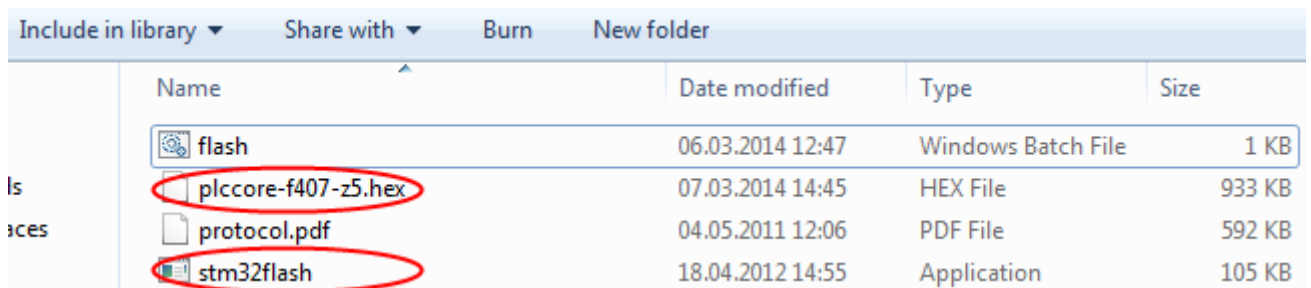
To set the PLCcore-F407 in update mode, the following sequence is necessary:

1. Both signals *"/RESET"* and *"BOOT0"* (see Table 11) must be set to active
2. Signal *"/RESET"* has to become inactive while *"BOOT0"* must still be set active
3. Signal *"BOOT0"* has to become inactive too, so that both signals *"/RESET"* and *"BOOT0"* are in there normal operation state again

On the Development Kit PLCcore-F407 signal *"/RESET"* is connected pushbutton *"RESET"* and *"BOOT0"* is connected to pushbutton *"BOOT"*. To set the PLCcore-F407 in update mode, the following sequence is necessary:

1. Press pushbutton *"RESET"* and hold it down
2. Press pushbutton *"BOOT"* while *"RESET"* is still down
3. Release pushbutton *"RESET"* while *"BOOT"* is still down
4. Release pushbutton *"BOOT"*

To update the firmware you have to extract the firmware package to a directory of your choice. The package contains at least the update tool and a *hex* file, containing the new firmware.

Figure 10: Content of firmware package

If the firmware package contains a file called *flash.bat* you can update the firmware by executing this file. Otherwise you need to open a command shell and navigate to the directory containing the extracted files. In this case the update process is started by entering the following line in your command shell:

stm32flash.exe -e 60 -v -b 115200 -w "<hex_file_name>" COM<num>

For example, the following command line programs the firmware hex file "*plccore-f407-z5.hex*" via *"COM3"* (COM3 at Host PC) to the PLCcore-F407:

stm32flash.exe -e 60 -v -b 115200 -w "plccore-f407-z5.hex" COM3

Table 24: Parameter STM32 firmware update tool

Parameter	Description
-e 60	Erases 60 pages before writing the flash
-v	Verifies the written flash
-b 115200	Use baud rate 115200
-w "plccore-f407-z5.hex"	Write plccore-f407-z5.hex file to flash
COM3	COM3 is used for file transfer

It must be considered that the file name for the firmware may differ between firmware versions. Hence you have to modify the command above regarding the file name and version number of the *hex* file.

After calling *flash.bat* (or executing the command above) you have to wait several minutes till the update is finished.

Figure 11: Firmware update process

When the update finished you need to press the reset switch to restart the PLC which applies the firmware update. Run- and user led are toggling as described in section 7.6.

9 Baseboard Configuration

The following chapter describes possible jumper configurations. These configurations are referring to the supplied baseboard of the PLCcore-F407.

9.1 General Jumpers

The general configuration of the PLCcore-F407 is done using the JP112 connector. This connector is used to enable or disable the push buttons, LED's and the analog input. Figure 12 shows the position of JP112. Additionally, this figure is also showing the position of the push buttons, the LED's and the analog input, represented by the potentiometer.

Figure 12: Positon of JP202 and JP212 for serial output

JP112 consists of 10 jumpers, mapped as follows:

Table 25: General jumper mapping

Jumper name on baseboard	Description
LED0 … LED3	JP101 … JP104
SW0 … SW3	JP105 … JP108
AIN (Potentiometer)	JP110
Current Sense	JP111

All jumpers are set by default. So, all push buttons, LED's and analog inputs are enabled. In order to disable a specific feature (e.g. push button SW2), requires removing the corresponding jumper (JP107 for SW2).

9.2 Serial Output

The serial interfaces of the PLCcore-F407 are controlled by JP212. COM1 and COM2 are additionally controlled by JP202. The positions of the corresponding jumpers are marked in the Figure 13.
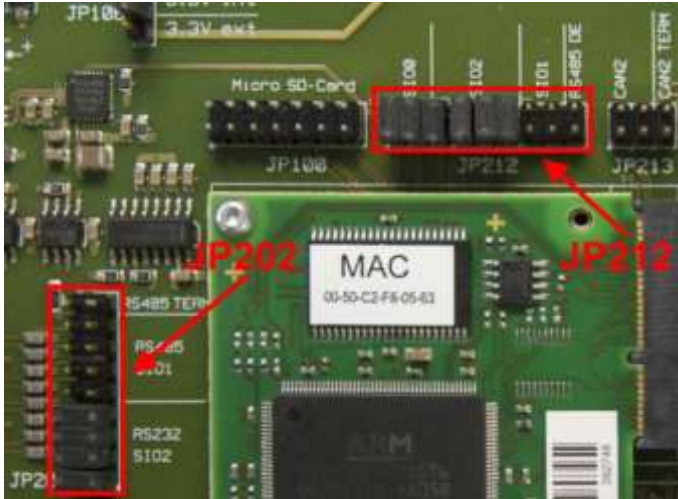
Figure 13: Position of JP202 and JP212 for serial output

SIO0 refers to the UART/USB-Bridge (COM0) which is the default interface for the configuration shell and can be used for user output during runtime. The interface is enabled when the first 2 jumpers from the left of JP212 are set. Removing these jumpers is disabling this serial interface. Figure 14 shows the corresponding jumper configuration where the SIO0 interface is enabled.

Figure 14: Jumper configuration to enable SIO0 (USB/UART)

The remaining serial interfaces SIO1 (COM1) and SIO2 (COM2) are also configured by JP212. Both serial outputs are using the same (female) SUB-D 9 connector. Hence, only SIO1 or SIO2 should be enabled simultaneously. Do not enable both interfaces at the same time on the reference baseboard.

Enabling SIO1 or SIO2 requires setting the corresponding jumpers on the baseboard for JP212 and JP202. The position of these jumpers is shown in Figure 13 which is also showing the jumper configuration where SIO2 is enabled.

Contrarily enabling SIO1 requires setting the corresponding jumpers on JP212 and JP202. Please note that SIO1 supports RS485 and RS422. Depending on the bus configuration, it might require to terminate the bus using the "RS485 TERM" jumper on JP202 or not. So, if the bus is already terminated by another device, it is not necessary to set the "RS485 TERM" jumper. Otherwise, setting this jumper is required. Figure 15 shows the jumper position to enable the RS485 interface and the pin configuration.
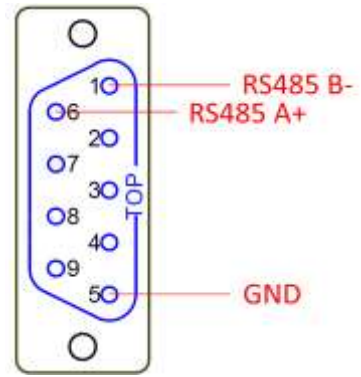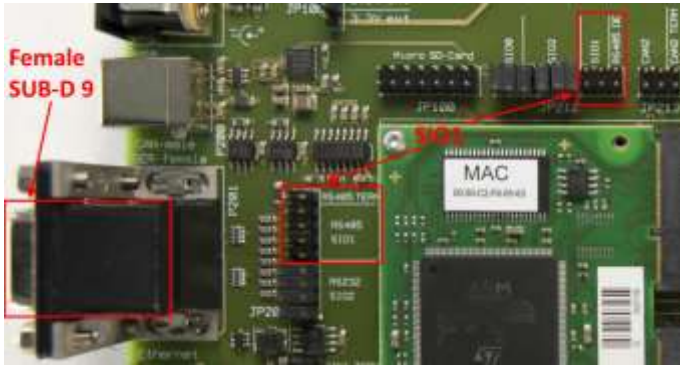
Figure 15: Jumper and pin configuration for RS485

Enabling RS422 support requires removing the "RS485 DE" jumper of the JP212. See Figure 16 for the jumper positions and the pin configuration.
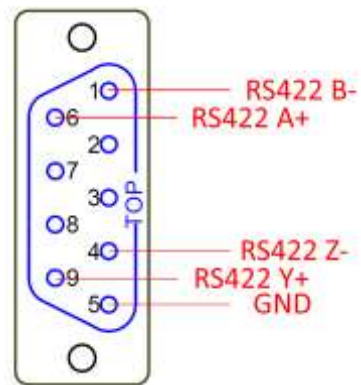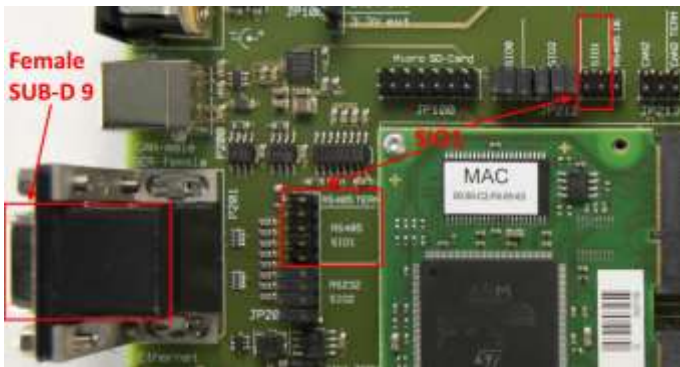

Figure 16: Jumper and pin configuration for RS422

The RS232 is enabled by setting the marked jumpers shown in Figure 17. All serial interfaces can be used in the PLC program by using their corresponding number to call the SIO function blocks. Please found more information about the function blocks in section 7.8.1.
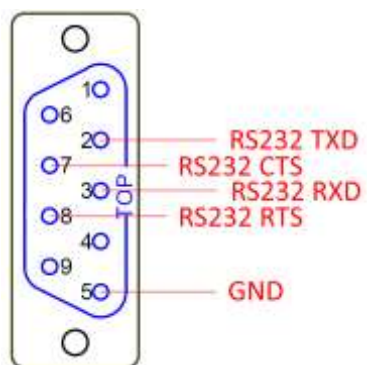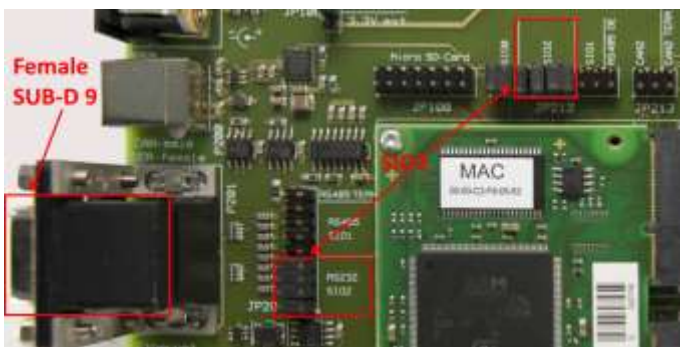
Figure 17: Jumper and pin configuration for RS232

9.3 Control Area Network

The CAN interfaces of the PLCcore-F407 can be controlled with JP214 (CAN1) and JP213 (CAN2). The position of JP213 and JP214 on the baseboard are marked in Figure 18.
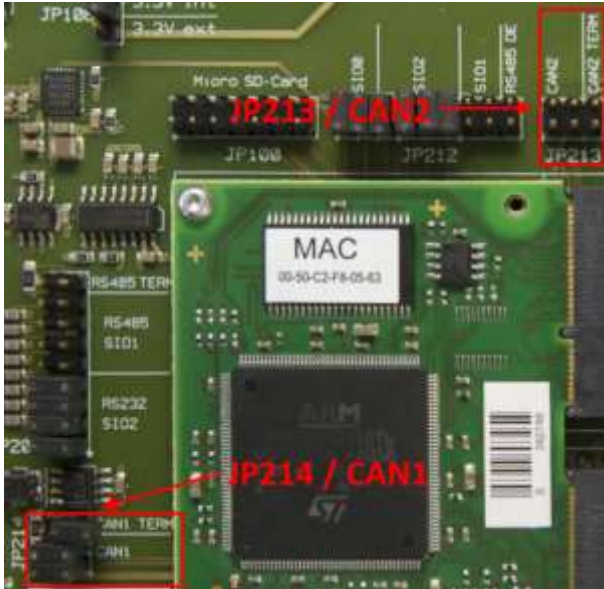

Figure 18: Jumper position of JP213 and JP214

Figure 18 shows the default CAN configuration. So, CAN1 is enabled, while CAN2 is disabled. The termination jumpers are only necessary if the CAN bus is not terminated by another device or resistor.

9.4 Micro SD-Card

The Micro SD-Card interface of the PLCcore-F407 is controlled by JP100. To enable the Micro SD-Card, all jumpers have to be set. The position of JP100 is shown in Figure 19.
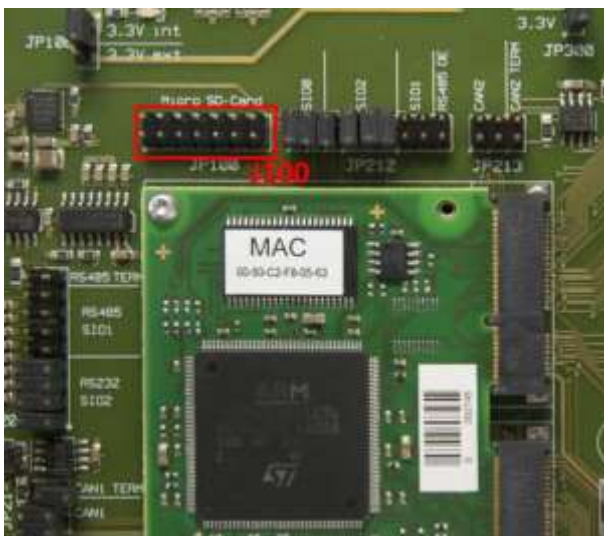

Figure 19: Jumper position of JP100

Micro SD-Card support is not supported by all Firmware version of the PLCcore-F407. Furthermore, the Micro SD-Card support is disabled by default and must be enabled as described in section 7.9.

10 OpenPCS Programming System

10.1 Installation Driver for USB-CANmodul

For using the PLCcore-F407/Z4 (order number: 3390094) the USB-CANmodul is used for data communication with the PLC. To install the USB-CANmodul driver, following steps are necessary:

1. Running driver setup (SO-387.exe)
2. Connect USB-CANmodul to PC

10.2 Installation OpenPCS Programming System

Step 1:

- Start the OpenPCS setup program (e.g. *"PS665e.exe")* and follow the self-explanatory instructions until the following dialog window appears.
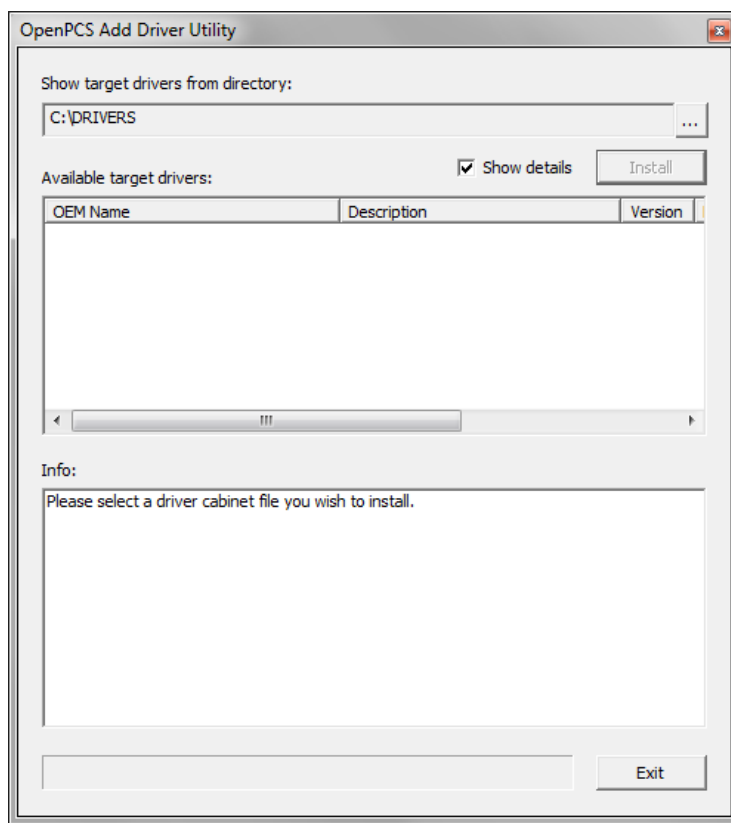

Figure 20: OpenPCS Installation Setup

Close the dialog in shown in Figure 20 by clicking on button *"Exit"* (driver installation will be done with the *"SYSTEC OpenPCS Extension"*, see Step 2).

Step 2:

- Start the *"SYSTEC OpenPCS Extension"* setup program (e.g. *"SYSTEC-OpenPCS-Extension_665e_302.exe".* Follow the instructions of the install program.

- In the *"Select Components"* dialog select *"Standard Installation for SYS TEC Control Units"* and then click on button *"Next"*.
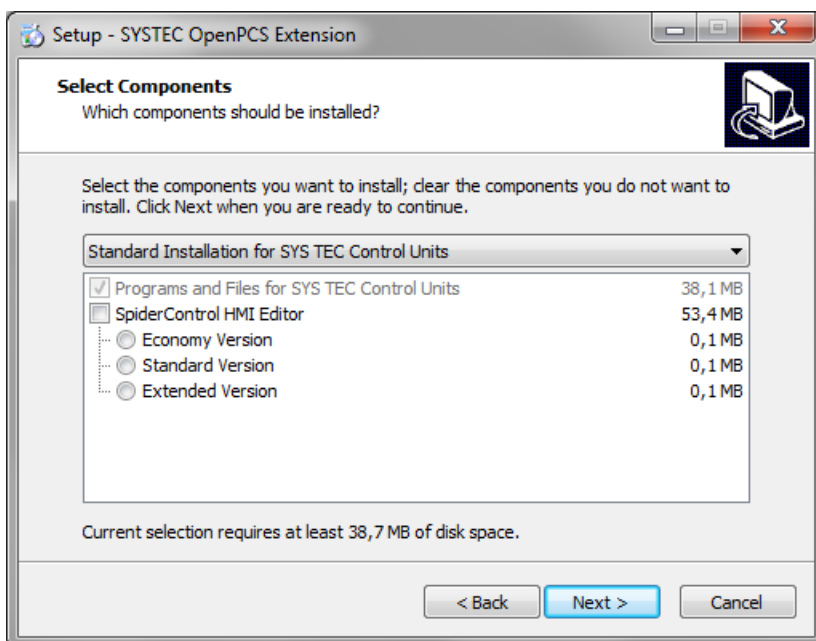

Figure 21: SYS TEC OpenPCS Extension select components dialog

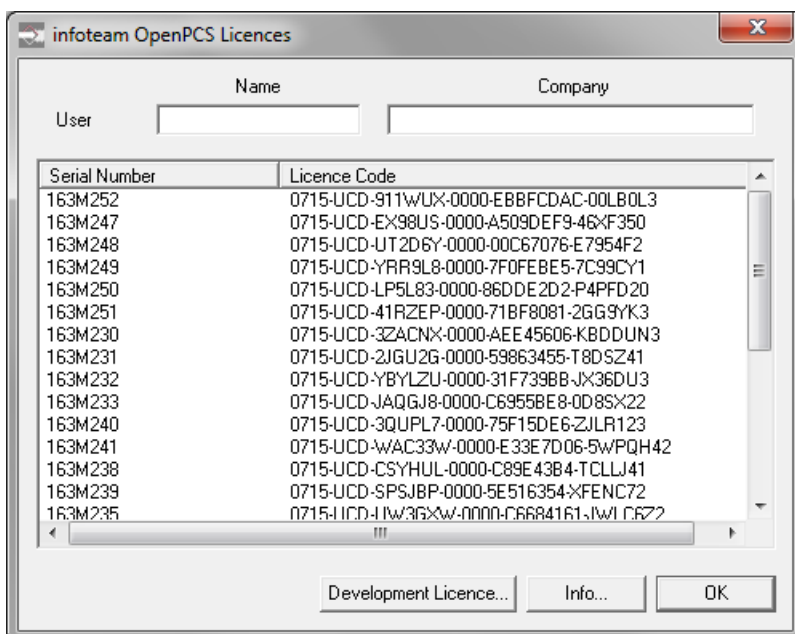- Insert your name and company into the *"infoteam OpenPCS Licenses"* window and click *"OK".*

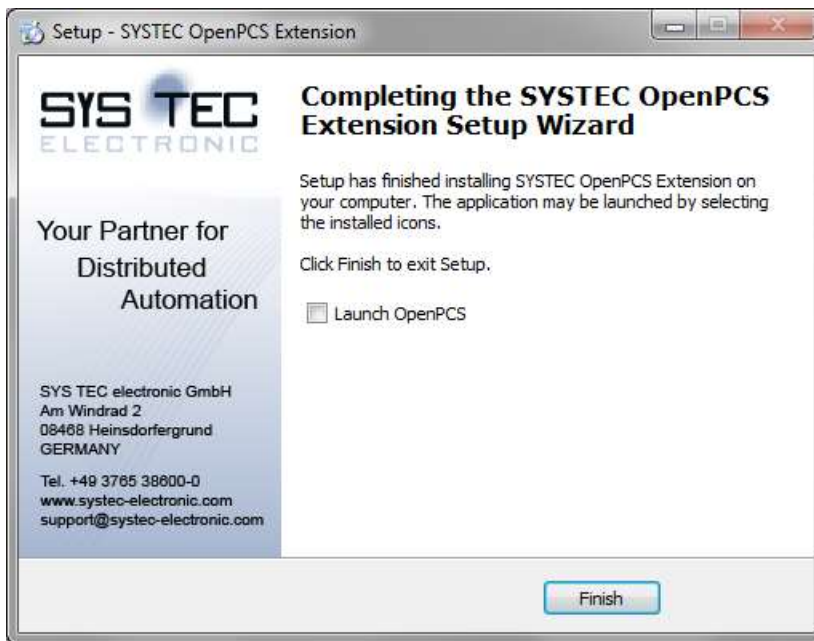*Figure 22: SYS TEC OpenPCS Extension Setup*

- Finish the installation.

Figure 23: SYS TEC OpenPCS Extension Setup

10.3 Define Network Connection

Step 1:

- Select *"PLC"* ➢ *"Connections…"* from the OpenPCS menu.

- In the dialog window *"Connection Setup"*, click at the button *"New"*. This will in turn open the following window.
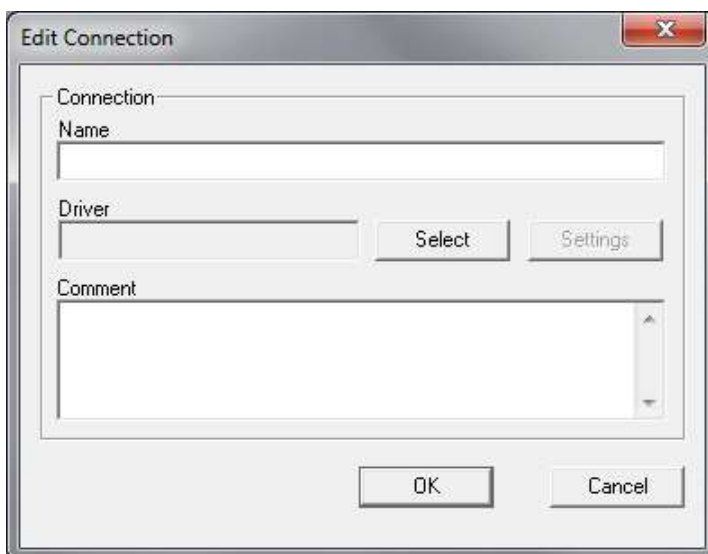

Figure 24: Target Connection – Edit Connection

Step 2:

- Choose the button "Select".

- Select the driver *"SYSTEC Standard Driver"*. This is the first choice for many kinds of connections.
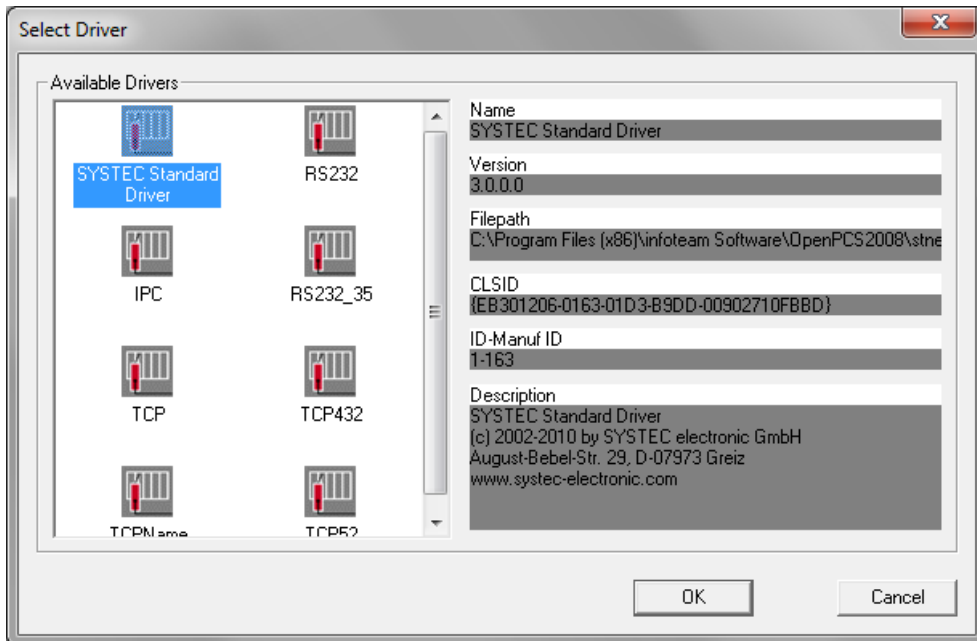

Figure 25: Target Connection – Select Driver

- Confirm the selection by clicking "OK".

Step 3:

- Enter a meaningful name for the connection settings, e.g. *"UDP_192_168_10_180"*.

- Now, the *"Edit Connection"* dialog looks like this:
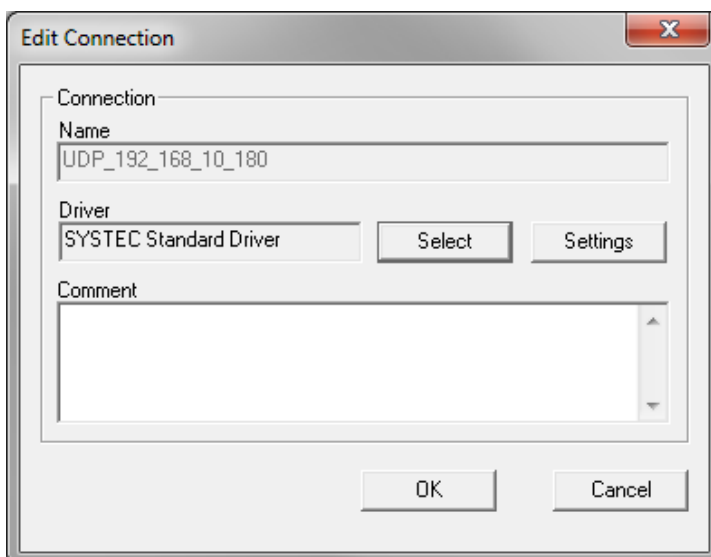
Figure 26: Target Connection – Edit Connection (filled)

- Finish the configuration of the target connection by clicking *"OK"* in the *"Edit Connection"* dialog and *"Close"* in the *"Connection Setup"* window

10.4 Assign Network Connection to Resource

Step1:

- Select "PLC" ➢ "Resource Properties…" from the OpenPCS main menu.

- Choose "SYSTEC - PLCcore-F407/Z4 (339000094/Z4)" for the Hardware Module setting and the Network Connection "UDP_192_168_10_180", which has been created in the previous section.

Figure 27: Resource Properties

- Confirm the settings by clicking *"OK"*.

Step2:

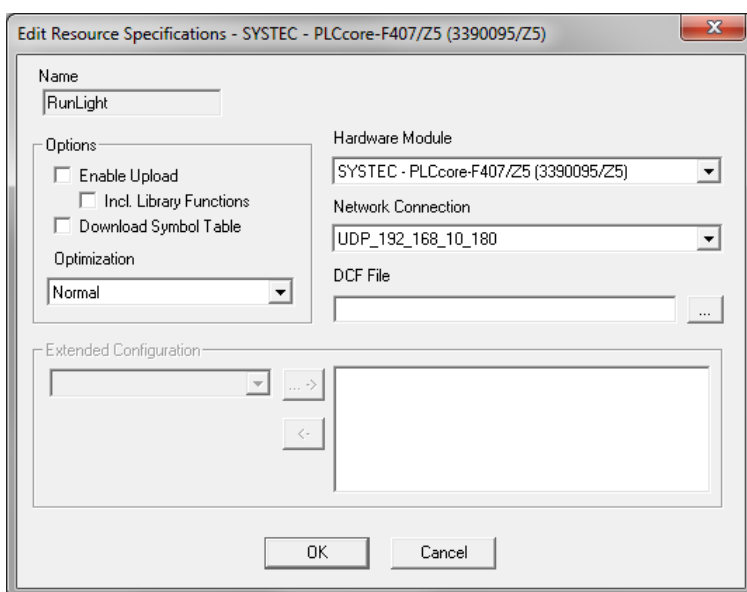- Use either the entry "PLC" ➤ "Rebuild Active Resource" from the main menu or click at the appropriate icon as shown in the screen snapshot.
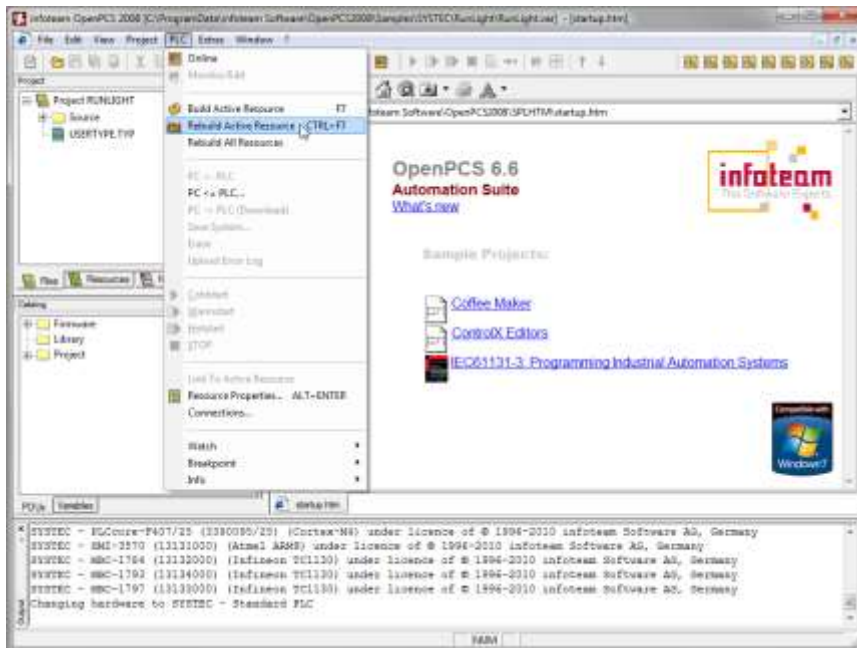

Figure 28: OpenPCS - Rebuild Active Resource

11 Configuration Command Shell

11.1 Entering the Configuration Command Shell

The Configuration Command Shell (herein after called Configuration Shell) of the *Controller* provides a set of commands to read and write device configurations such as TCP/IP address settings and CAN interface configurations.

Figure 29: Built-in Configuration Command Shell

The Configuration Shell uses the serial interface SIO0 of the PLCcore-F407. On the Development Kit PLCcore-F407 the interface SIO0 is tunneled via USB on connector P200. To access the Configuration Shell, connect the Development Kit to the PC using the USB cable delivered with the Kit.

On the PC side the virtual COM port driver delivered with the Kit *("CP210xVCPInstaller.exe")* has to be installed. This will add an additional virtual serial interface to the Host PC. This allows for access to the Command Shell using a Terminal program. Suitable as Terminal program would be *"HyperTerminal"* which is included in the Windows delivery or *"TeraTerm"* which is available as Open Source and meets higher demands (downloadable from: http://ttssh2.sourceforge.jp).

The Terminal program must be configured as follows (see Figure 30):

- 115200 Baud
- 8 Data bit
- 1 Stop bit
- no parity
- no flow control
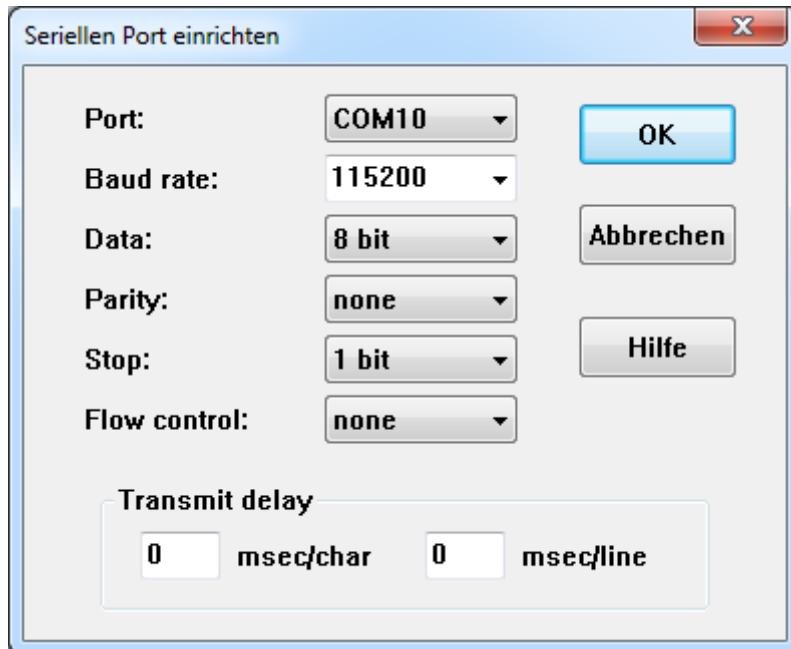
Figure 30: Terminal configuration using the example of "TeraTerm"

To start the Configuration Shell of the PLCcore-F407, the following sequence is necessary:

4. Both signals *"/RESET"* and *"/CONFIG"* (see Table 11) must be set to active
5. Signal *"/RESET"* has to become inactive while *"/CONFIG"* must still be set active
6. Signal *"/CONFIG"* has to become inactive too, so that both signals *"/RESET"* and *"/CONFIG"* are in there normal operation state again

On the Development Kit PLCcore-F407 signal *"/RESET"* is connected pushbutton *"RESET"* and *"/CONFIG"* is connected to pushbutton *"CONFIG"*. To start the Configuration Shell on a PLCcore-F407 mounted to the Development Board, the following sequence is necessary:

5. Press pushbutton *"RESET"* and hold it down
6. Press pushbutton *"CONFIG"* while *"RESET"* is still down
7. Release pushbutton *"RESET"* while *"CONFIG"* is still down
8. Release pushbutton *"CONFIG"*

After entering the Command Shell, the status message should come up on the terminal window as shown in Figure 29. All inputs are case insensitive. The *HELP* command gives you an overview about all commands available. To get help for a specific command just enter *HELP* followed by the command name or type the command followed by an *"?"*. Press *CTRL+R* to repeat the last command. The command *"EXIT"* terminates the Configuration Shell. The configuration is stored in non-volatile memory and the PLC starts with current settings.

Warning
Please note that the following commands cannot be undone: - *DEL_CFG_FILE* - *DEL_PLC_PROG* - *DEL_NVDATA* - *DEL_FILE* - *FORMAT_FS* Please take care about respective commands!

11.2 Command Description

The following section describes the commands available for the Command Configuration Shell on the PLCcore-F407.

11.2.1 GET_DEV_CONFIG

The command *GET_DEV_CONFIG* shows the current device configuration. This command has no parameters.

Example

```
> GET_DEV_CONFIG
Device Information:
Type:               SYSTEC PLCcore-F407/Z5 (3390025/Z5)
DeviceID:           1012005
Serial Number:      262746
License Key:        WXYZ-WXYZ-WXYZ-WXYZ-WXYZ-WXYZ-WXYZ-WXYZ
Product Code:       131301
Hardware:
  CPU Board:        4345.01
  IO Board:         4346.01
I/O Driver:         1.00
Firmware:           5.08.01.04
Build Date/Time:    Sep  10 2015 / 16:03:14
Virtual Machine:    7.03 - 2 (#42)
OEM-ID:             163

Device Configuration:
Config File:        0:/PC-F407.cfg
PLC Backup Dir:     plcdata
Memory available:   512 kByte
ProcessImage:
  Extended I/Os:    disabled
RuntimeSettings:
  Watchdog:         DiagModeOn
  SnglSegDwl Mode:  disabled
  CRC Mode:         disabled

ETH0 MAC Addr:      00-50-C2-F8-04-56
ETH0 IpAddr:        192.168.10.248
ETH0 SubnetMask:    255.255.255.0
ETH0 Gateway:       192.168.10.1
ETH0 PortNum:       8888
```

```
CAN0 Enable:        on
CAN0 NodeID:        32 (0x20)
CAN0 Baudrate:      1000
CAN0 MasterMode:    off
CAN0 ErrorLevel:    AllNetErrors

CAN1 Enable:        off
CAN1 NodeID:        48 (0x30)
CAN1 Baudrate:      1000
CAN1 MasterMode:    off
CAN1 ErrorLevel:    AllNetErrors
Ok

>
```

11.2.2 SET_IP_CONFIG

The command *SET_IP_CONFIG* is used to set the communication parameters for Ethernet interface.

Command

```
SET_IP_CONFIG <IfNum IpAddr [NetMask [Gateway [PortNum]]]>
```

Table 26: Parameter for command SET_IP_CONFIG

Parameters	Attribute	Values	Description
IfNum	Mandatory	0 -> ETH0 1 -> ETH1	Interface number for Ethernet interface
IpAddr	Mandatory	i.e. 192.168.10.130	IP address
NetMask	Optional	i.e. 255.255.255.0	Network mask
Gateway	Optional	i.e. 192.168.10.200	IP address of default gateway
PortNum	Optional	i.e. 8888	Port number

Example

```
SET_IP_CONFIG  0  192.168.10.130  255.255.255.0

OK

>
```

11.2.3 SET_CAN_CONFIG

The command *SET_CAN_CONFIG* is used to set the communication parameters of the two CAN communication interfaces CAN0 und CAN1.

Command

```
SET_CAN_CONFIG <IfNum Enable [NodeID [Bitrate [MstMode ]]]>
```

Table 27: Parameter for command SET_CAN_CONFIG

Parameters	Attribute	Values	Description
IfNum	Mandatory	0 -> CAN0 1 -> CAN1	Interface number for CAN interface
Enable	Mandatory	on/off	Enables or disables the CAN interface
NodeID	Optional	1…0x7F	Node-ID (i.e. 0x21)
Bitrate	Optional	20, 50, 100, 125,250, 500, 1000	CAN-bus baud rate (i.e. 500)
MstMode	Optional	on/off	Master mode (on/off)

Example

```
SET_CAN_CONFIG  0  on  32  1000  off

OK

>
```

11.2.4 SET_CAN_ERRLEV

The command *SET_CAN_ERRLEV* sets the error level for saving CAN errors in the Error Log of the PLC. This Error Log can be uploaded if the PLCcore-F407 is online connected to the OpenPCS Programming System.

Command

```
SET_CAN_ERRLEV <IfNum ErrorLevel>
```

Table 28: Parameter for command SET_CAN_ERRLEV

Parameters	Attribute	Values	Description
IfNum	Mandatory	0 -> CAN0 1 -> CAN1	Interface number for CAN interface
ErrorLevel	Mandatory	no	No CAN errors are logged in the Error Log
		heavy	Only heavy occurring errors (i.e. BusOff) are logged in the Error Log
		all	All occurring CAN errors are logged in the Error Log

Example

```
> SET_CAN_ERRLEV  1  all
Ok

>
```

11.2.5 SET_WDG_MODE

The command *SET_WDG_MODE* is used to define the behavior of the on-board watchdog of the PLCcore-F407. -

<u>Command</u>

```
SET_WDG_MODE <WdgMode>
```

Table 29: Parameter for watchdog configuration

Parameter	Attribute	Values	Description
WdgMode	Mandatory	on	Default-setting, Watchdog guarding is activated. The device is reset on Watchdog-event.
		monitor	Watchdog-events are recognized by the firmware without resetting the device. However, an error message (message box) is sent to the programming system when a Watchdog-event has occurred.
		off	Watchdog-guarding is inactive; the device does not react to any Watchdog-events.

<u>Example</u>

```
> SET_WDG_MODE  off
Ok
>
```

11.2.6 SET_SSDWL_MODE

The command *SET_SSDWL_MODE* sets the mode for the transfer of a PLC program archive from the OpenPCS Programming System to the PLCcore-F407. The command is described only for completeness. The use of the command is very rarely needed.

Background:
A PLC program archive contains several segments of data. To reduce the transfer time the segments are packed together to a container and the device sends only one ACK/NAK for one container. If the PC will not receive the ACK/NAK within a defined timeout, the connection will break with an error message. The time between the transferred container and sending of the ACK/NAK depends on the machine speed of the device. In rare cases it can happen that is needed too long time for the processing of a container. In this case, the command *SET_SSDL_MODE* can be used to switch from container to a single segmented transfer mode.

<u>Command</u>

```
SET_SSDWL_MODE <SSDwlMode>
```

Table 30: Parameter for command SET_SSDWL_MODE

Parameter	Attribute	Values	Description
SSDwlMode	Mandatory	on	A PLC program archive will be transferred with single segments (single segment download).
		off	A PLC program archive will be transferred in a container. This needs the shortest transfer time (Default setting).

Example

```
> SET_SSDWL_MODE  off
OK
```

11.2.7 SET_SDCARD_MODE

The command SET_SDCARD_MODE is used to enable or disable the Micro SD-Card support and available since firmware version 5.08.02.00.

Command

```
SET_SDCARD_MODE <SdMode>
```

Table 31: Parameter for command SET_SDCARD_MODE

Parameter	Attribute	Values	Description
SdMode	Mandatory	on	Enable Micro SD-Card
		off	Disable Micro SD-Card

Example

```
> SET_SDCARD_MODE on
OK

>
```

11.2.8 GET_RTC

Shows the system time and date from the on-board RTC unit. This command has no parameters.

Example:

```
> GET_RTC
Valid: yes
Time  12:30:15
Date: 2015/10/05
```

11.2.9 SET_RTC

Command

```
SET_RTC <hh:mm:ss [yyyy/mm/dd]>
```

SET_RTC enables you to set the system time and date for the on-board RTC. The RTC on PLCcore-F407 is battery backed. The time format is based on 24h. Please refer to the SYS TEC PLC extension Manual (L-1054) for more information on how accessing the RTC from within the PLC program.

Table 32: Parameter for command SET_RTC

Parameter	Attribute	Description	
hh:mm:ss	Mandatory	hh: mm: ss:	hour (0…24) minutes (0…59) seconds (0…59)
yyyy/mm/dd	Optional	yyyy: mm: dd:	year month day

Example

```
> SET_RTC 12:30:59 2015/10/05
OK
```

11.2.10 SET_CRC_MODE

The command *SET_CRC_MODE* enables or disables the CRC check for saving/loading the PLC resource.

Command

```
    SET_CRC_MODE <CrcMode>
```

Table 33: Parameter for command SET_CRC_MODE

Parameter	Attribute	Values	Description
CrcMode	Mandatory	on	Enables the CRC mode
		off	Disables the CRC mode

Example

```
> SET_CRC_MODE  off
OK
>
```

11.2.11 GET_LAST_ERROR

The command *GET_LAST_ERROR* prints detailed information about the last system error occurred. This command has no parameters.

Example

```
> GET_LAST_ERROR
ErrorCode = 0
ErrorText = Controller working normal
Ok
>
```

11.2.12 PRINT_CFG_FILE

The command PRINT_CFG_FILE shows the content of the configuration file if the file exists.

<u>Example</u>

```
> PRINT_CFG_FILE
[ETH0]
IpAddr=192.168.10.180
SubnetMask=255.255.255.0
Gateway=192.168.10.1
PortNum=8888

[CAN0]
Enable=1
NodeID=0x20
Baudrate=1000
MasterMode=0
ErrLevel=2

[CAN1]
Enable=0
NodeID=0x30
Baudrate=1000
MasterMode=0
ErrLevel=2

[PlcSettings]
EnableExtIo=0
WatchdogMode=0
SnglSegDwlMode=0
CrcCheckMode=0
Ok
>
```

11.2.13 DEL_CFG_FILE

The command *DEL_CFG_FILE* deletes the configuration file. Once done, this cannot be undone. This command has no parameters.

<u>Example</u>

```
> DEL_CFG_FILE
Do you really want to delete the device configuration file (y/n)? y
Ok
>
```

11.2.14 DEL_PLC_PROG

The command *DEL_PLC_PROG* deletes the saved PLC program in the PLC program backup archive. Once done, this cannot be undone. This command has no parameters.

<u>Example</u>

```
> DEL_PLC_PROG
Do you really want to delete the PLC program archive (y/n)? y
Ok
>
```

```
> PRINT_CFG_FILE
```

11.2.15 DEL_NVDATA

The command *DEL_NVDATA* deletes all data (variables) stored in non-volatile memory area. Once done, this cannot be undone. This command has no parameters.

Example

```
> DEL_NVDATA
Do you really want to delete the NVDATA archive (y/n)? y
Ok
>
```

11.2.16 DIR

The command *DIR* allows you to see all available files saved in the Flash Files System. This command has no parameters.

Example

```
> DIR

File system directory hierarchy...

+ 0:/
 |
 + -- plcdata
 |     |
 |     + -- PLCARCHV.BIN      2320 Bytes
 |     |
 |     + -- PLCPDATA.BIN         0 Bytes
 +
 |
 + -- PC-F407.cfg            317 Bytes
>
```

11.2.17 DUMP_FILE

The command *DUMP_FILE* prints out the contents of the given file in hexadecimal or in ASCII text form.

Command

```
    DUMP_FILE <FileName [-a]>
```

Table 34: Parameter for command DUMP_FILE

Parameter	Attribute	Description
FileName	Mandatory	Path to the file that should be dumped.
-a	Optional	Print file as ASCII output instead of hex dump

Example

```
> DUMP_FILE  PC-F407.cfg
00000000:    0A 5B 45 54 48 30 5D 0A 49 70 41 64 64 72 3D 31    .[ETH0].IpAddr=1
00000010:    39 32 2E 31 36 38 2E 31 30 2E 31 38 30 0D 0A 53    92.168.10.180..S
00000020:    75 62 6E 65 74 4D 61 73 6B 3D 32 35 35 2E 32 35    ubnetMask=255.25
00000030:    35 2E 32 35 35 2E 30 0D 0A 47 61 74 65 77 61 79    5.255.0..Gateway
00000040:    3D 31 39 32 2E 31 36 38 2E 31 30 2E 31 0D 0A 50    =192.168.10.1..P
00000050:    6F 72 74 4E 75 6D 3D 38 38 38 38 0D 0A 0A 5B 43    ortNum=8888...[C
00000060:    41 4E 30 5D 0A 45 6E 61 62 6C 65 3D 31 0D 0A 4E    AN0].Enable=1..N
00000070:    6F 64 65 49 44 3D 30 78 32 30 0D 0A 42 61 75 64    odeID=0x20..Baud
00000080:    72 61 74 65 3D 31 30 30 30 0D 0A 4D 61 73 74 65    rate=1000..Maste
00000090:    72 4D 6F 64 65 3D 30 0D 0A 45 72 72 4C 65 76 65    rMode=0..ErrLeve
000000A0:    6C 3D 32 0D 0A 0A 5B 43 41 4E 31 5D 0A 45 6E 61    l=2...[CAN1].Ena
000000B0:    62 6C 65 3D 30 0D 0A 4E 6F 64 65 49 44 3D 30 78    ble=0..NodeID=0x
000000C0:    33 30 0D 0A 42 61 75 64 72 61 74 65 3D 31 30 30    30..Baudrate=100
000000D0:    30 0D 0A 4D 61 73 74 65 72 4D 6F 64 65 3D 30 0D    0..MasterMode=0.
000000E0:    0A 45 72 72 4C 65 76 65 6C 3D 32 0D 0A 0A 5B 50    .ErrLevel=2...[P
000000F0:    6C 63 53 65 74 74 69 6E 67 73 5D 0A 45 6E 61 62    lcSettings].Enab
00000100:    6C 65 45 78 74 49 6F 3D 30 0D 0A 57 61 74 63 68    leExtIo=0..Watch
00000110:    64 6F 67 4D 6F 64 65 3D 30 0D 0A 53 6E 67 6C 53    dogMode=0..SnglS
00000120:    65 67 44 77 6C 4D 6F 64 65 3D 30 0D 0A 43 72 63    egDwlMode=0..Crc
00000130:    43 68 65 63 6B 4D 6F 64 65 3D 30 0D 0A             CheckMode=0..
Ok
>
```

11.2.18 DEL_FILE

The command *DEL_FILE* deletes a specific file from the Flash File System. Once done, this cannot be undone.

Command

```
DEL_FILE <FileName>
```

Table 35: Parameter for command DEL_FILE

Parameter	Attribute	Description
FileName	Mandatory	Path to the file that should be deleted.

Example

```
> DEL_FILE  PC-F407.cfg
Ok
>
```

11.2.19 FORMAT_FS

The command *FORMAT_FS* formats the Flash File System located in the external Flash memory. Once done, this cannot be undone. This command has no parameters.

Note
During this operation all data in the Flash File System will be deleted. There is no way to backup or restore this data, so please be **carefully**!

<u>Example</u>

```
> FORMAT_FS

CAUTION: This will delete all data stored in the FLASH File System!
Do you really want to continue (y/n)? y

Formatting the FALSH File System need some seconds, please wait…

OK
>
```

11.2.20 EXIT

The command *EXIT* terminates and leaves the Configuration Shell, save the configuration and restart the PLC.

11.2.21 HELP

The command *HELP* shows all available commands as well as a short command description (*see* Figure 29).

Appendix A: Firmware function scope of PLCcore-F407

Table 36 lists all firmware functions and function blocks available on the PLCcore-F407.

Sign explanation:

FB	Function block
FUN	Function
Online Help	*OpenPCS* online help
L-1054	Manual *"SYS TEC-specific extensions for OpenPCS / IEC 61131-3"*, Manual no.: L-1054)
PARAM:={0,1,2}	values 0, 1 and 2 are valid for the given parameter

Table 36: Firmware functions and function blocks of PLCcore-F407

Name	Type	Reference	Remark
PLC standard Functions and Function Blocks			
SR	FB	Online Help	
RS	FB	Online Help	
R_TRIG	FB	Online Help	
F_TRIG	FB	Online Help	
CTU	FB	Online Help	
CTD	FB	Online Help	
CTUD	FB	Online Help	
TP	FB	Online Help	
TON	FB	Online Help	
TOF	FB	Online Help	
Functions and Function Blocks for string manipulation			
LEN	FUN	L-1054	
LEFT	FUN	L-1054	
RIGHT	FUN	L-1054	
MID	FUN	L-1054	
CONCAT	FUN	L-1054	
INSERT	FUN	L-1054	
DELETE	FUN	L-1054	
REPLACE	FUN	L-1054	
FIND	FUN	L-1054	
GETSTRINFO	FB	L-1054	
CHR	FUN	L-1054	
ASC	FUN	L-1054	
STR	FUN	L-1054	
VAL	FUN	L-1054	
Functions and Function Blocks for OpenPCS specific task controlling			
ETRC	FB	L-1054	
PTRC	FB	L-1054	
GETVARDATA	FB	Online Help	
GETVARFLATADDRESS	FB	Online Help	
GETTASKINFO	FB	Online Help	

Name	Type	Reference	Remark
Functions and Function Blocks for handling of non-volatile data			
NVDATA_BIT	FB	L-1054	DEVICE:={0}, see [1]
NVDATA_INT	FB	L-1054	DEVICE:={0}, see [1]
NVDATA_STR	FB	L-1054	DEVICE:={0}, see [1]
NVDATA_BIN	FB	L-1054	DEVICE:={0}, see [1]
Functions and Function Blocks for handling of time			
GetTime	FUN	Online Help	
GetTimeCS	FUN	Online Help	
DT_CLOCK	FB	L-1054	
DT_ABS_TO_REL	FB	L-1054	
DT_REL_TO_ABS	FB	L-1054	
Functions and Function Blocks for counter inputs and pulse outputs			
CNT_FUD	FB	L-1054	CHANNEL:={0,1}
PTO_PWM	FB	L-1054	CHANNEL:={0,1} [2]
PTO_TAB	FB	L-1054	CHANNEL:={0,1} [2]
Functions and Function Blocks for Serial interfaces			
SIO_INIT	FB	L-1054	PORT:={0,1,2} [3]
SIO_STATE	FB	L-1054	PORT:={0,1,2} [3]
SIO_READ_CHR	FB	L-1054	PORT:={0,1,2} [3]
SIO_WRITE_CHR	FB	L-1054	PORT:={0,1,2} [3]
SIO_READ_STR	FB	L-1054	PORT:={0,1,2} [3]
SIO_WRITE_STR	FB	L-1054	PORT:={0,1,2} [3]
SIO_READ_BIN	FB	L-1054	PORT:={0,1,2} [3]
SIO_WRITE_BIN	FB	L-1054	PORT:={0,1,2} [3]
Functions and Function Blocks for CAN interfaces / CANopen			
CAN_GET_LOCALNODE_ID	FB	L-1008	NETNUMBER:={0,1}
CAN_CANOPEN_KERNEL_STATE	FB	L-1008	NETNUMBER:={0,1}
CAN_REGISTER_COBID	FB	L-1008	NETNUMBER:={0,1}
CAN_PDO_READ8	FB	L-1008	NETNUMBER:={0,1}
CAN_PDO_WRITE8	FB	L-1008	NETNUMBER:={0,1}
CAN_SDO_READ8	FB	L-1008	NETNUMBER:={0,1}
CAN_SDO_WRITE8	FB	L-1008	NETNUMBER:={0,1}
CAN_SDO_READ_STR	FB	L-1008	NETNUMBER:={0,1}
CAN_SDO_WRITE_STR	FB	L-1008	NETNUMBER:={0,1}
CAN_SDO_READ_BIN	FB	L-1008	NETNUMBER:={0,1}
CAN_SDO_WRITE_BIN	FB	L-1008	NETNUMBER:={0,1}
CAN_GET_STATE	FB	L-1008	NETNUMBER:={0,1}
CAN_NMT	FB	L-1008	NETNUMBER:={0,1}
CAN_RECV_EMCY_DEV	FB	L-1008	NETNUMBER:={0,1}
CAN_RECV_EMCY	FB	L-1008	NETNUMBER:={0,1}
CAN_WRITE_EMCY	FB	L-1008	NETNUMBER:={0,1}
CAN_RECV_BOOTUP_DEV	FB	L-1008	NETNUMBER:={0,1}
CAN_RECV_BOOTUP	FB	L-1008	NETNUMBER:={0,1}
CAN_ENABLE_CYCLIC_SYNC	FB	L-1008	NETNUMBER:={0,1}
CAN_SEND_SYNC	FB	L-1008	NETNUMBER:={0,1}

Name	Type	Reference	Remark
CANL2_INIT	FB	L-1008	NETNUMBER:={0,1}
CANL2_SHUTDOWN	FB	L-1008	NETNUMBER:={0,1}
CANL2_RESET	FB	L-1008	NETNUMBER:={0,1}
CANL2_GET_STATUS	FB	L-1008	NETNUMBER:={0,1}
CANL2_DEFINE_CANID	FB	L-1008	NETNUMBER:={0,1}
CANL2_DEFINE_CANID_RANGE	FB	L-1008	NETNUMBER:={0,1}
CANL2_UNDEFINE_CANID	FB	L-1008	NETNUMBER:={0,1}
CANL2_UNDEFINE_CANID_RANGE	FB	L-1008	NETNUMBER:={0,1}
CANL2_MESSAGE_READ8	FB	L-1008	NETNUMBER:={0,1}
CANL2_MESSAGE_READ_BIN	FB	L-1008	NETNUMBER:={0,1}
CANL2_MESSAGE_WRITE8	FB	L-1008	NETNUMBER:={0,1}
CANL2_MESSAGE_WRITE_BIN	FB	L-1008	NETNUMBER:={0,1}
CANL2_MESSAGE_UPDATE8	FB	L-1008	NETNUMBER:={0,1}
CANL2_MESSAGE_UPDATE_BIN	FB	L-1008	NETNUMBER:={0,1}
Functions and Function Blocks for Ethernet interfaces / UDP			
LAN_GET_HOST_CONFIG	FB	L-1054	NETNUMBER:={0}
LAN_ASCII_TO_INET	FB	L-1054	NETNUMBER:={0}
LAN_INET_TO_ASCII	FB	L-1054	NETNUMBER:={0}
LAN_GET_HOST_BY_NAME	FB	L-1054	NETNUMBER:={0}
LAN_GET_HOST_BY_ADDR	FB	L-1054	NETNUMBER:={0}
LAN_UDP_CREATE_SOCKET	FB	L-1054	NETNUMBER:={0}
LAN_UDP_CLOSE_SOCKET	FB	L-1054	NETNUMBER:={0}
LAN_UDP_RECVFROM_STR	FB	L-1054	NETNUMBER:={0}
LAN_UDP_SENDTO_STR	FB	L-1054	NETNUMBER:={0}
LAN_UDP_RECVFROM_BIN	FB	L-1054	NETNUMBER:={0}
LAN_UDP_SENDTO_BIN	FB	L-1054	NETNUMBER:={0}
Functions and Function Blocks for File System			
FILE_OPEN	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_CLOSE	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_READ	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_READ_LINE	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_WRITE	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_SEEK	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_SYNC	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_STAT	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_CHMOD	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_TOUCH	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_DELETE	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_RENAME	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_COPY	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_SPLIT_PATH	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_DIR_OPEN	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_DIR_CLOSE	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_DIR_READ	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_GET_DIR	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_SET_DIR	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_MKDIR	FB	L-1828	Since Firmware Version 5.08.02.00
FILE_RMDIR	FB	L-1828	Since Firmware Version 5.08.02.00
Functions and Function Blocks for Modbus			
MODBUS_OPEN_INSTANCE	FB	L-1829	Since Firmware Version 5.08.02.00

Name	Type	Reference	Remark
MODBUS_CLOSE_INSTANCE	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_REGISTER_VAR_LIST	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_READ_REGS	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_WRITE_SINGLE_REG	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_WRITE_MULTI_REGS	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_READ_WRITE_REGS	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_READ_INPUT_REGS	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_READ_DISCR_INPUTS	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_READ_COILS	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_WRITE_SINGLE_COIL	FB	L-1829	Since Firmware Version 5.08.02.00
MODBUS_RAW_PDU_REQUEST	FB	L-1829	Since Firmware Version 5.08.02.00

[1]   These information are stored outside the file system to protect them against power outages which may cause a corrupted file system. In such case however the data are still getting corrupted if the power loss occurred while erasing one page or writing data to the flash. Writing to NVData requires at least one erase and write operation per 4k flash page which usually requires about 90 ms (max. 350 ms). Writing to multiple pages requires correspondingly more time. It is recommended to use 4k aligned data blocks to prevent writing multiple pages.

[2]   The pulse train output uses a 16 bit timer with 8 bit repetition counter. That implies a maximum cycle time of about 65 ms and 256 time-accurate edges. Using more pulses result in inaccurate pulse timings. These delays depend on the usage of the PLCcore-F407 which enables the controller to invoke the timer reset earlier or later, the used time base and the period length.
Furthermore the PLCcore-F407 does not support all base cycles described in L-1054. Supported TB_IDX values are 1 (base cycle = 1 ms) and 2 (base cycle = 1 us).

[3]   As chapter 7.4.1 states, the PLCcore-F407 supports 3 different serial interfaces. Each of them can be used by passing the corresponding interface number to the function block as *PORT* parameter. The following list shows relation between the interface and the *PORT* parameter.

Index

Document:	System Manual PLCcore-F407
Document number:	L-1515e_08, September 2018

How would you improve this manual?

Did you detect any mistakes in this manual? page

Submitted by:

Customer number:

Name:

Company:

Address:

Please return your suggestions to:	SYS TEC electronic GmbH Am Windrad 2 D - 08468 Heinsdorfergrund GERMANY Fax: +49 (3765) 38600-4100 Email: info@systec-electronic.com