System Manual
IoT-Chip

User Manual
Version 1

Edition June 2017

Document No.: L-1966e_1

SYS TEC electronic GmbH Am Windrad 2 D-08468 Heinsdorfergrund
Telefon: +49 (3765) 38600-0 Telefax: +49 (3765) 38600-4100
Web: http://www.systec-electronic.com Mail: info@systec-electronic.com

Status/Changes

Status: released

Date/Version	Section	Changes	Editor
2017/06/02 Version 1	All	Creation	T. Junghänel

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 2017 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	Please find a list of our distributors under: http://www.systec-electronic.com/distributors
Ordering Information:	+49 (0) 37 65 / 38 600-0 info@systec-electronic.com	
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	

1st Edition June 2017

List of Abbreviations

AI	Analog Input
AO	Analog Output
CAN	Controller Area Network (according to ISO 11898-1:2003 and ISO 11898-2:2003)
CPU	Central Processing Unit
ETH	Ethernet
FB	Function Block
GND	Ground Reference potential
GPIO	General Purpose Input Output
I/O	Input/Output
I^2C	Inter-integrated circuit
IoT	Internet of Things
M2M	Machine-to-Machine
MAC	Media Access Controller (e.g. Ethernet controller)
MQTT	Message Queue Telemetry Transport
PCB	Printed Circuit Board
PDO	Process Data Object
PLC	Programmable Logical Controller
POT	Potentiometer
PWM	Pulse Width Modulation
RTC	Real Time Clock
RX	Receive
SIO	Serial Input Output
SDO	Service Data Object
SPI	Serial Peripheral Interface
tbd	to be defined
TX	Transmit
UART	Universal Asynchronous Receiver Transmitter

Table of Contents

List of Tables

List of Figures

1 Introduction

Thank you that you have decided for the SYS TEC IoT-Chip. Our IoT Chip SE is a plug-in System on Module for cloud connection which is freely programmable by users. Unlike other IoT Chip devices, the SYS TEC electronic IoT Chip SE combines sensor2cloud requirements with data preprocessing features ranging from time-critical signals to efficient control tasks. To that end, the IoT Chip SE comes with appropriate function libraries pre-installed, making it ready for use straight away. Moreover our IoT Chip SE is prepared for the Cloud Services, such as IBM IoT Platform (IBM Watson).

Please take some time to read through this manual carefully. It contains important information about the commissioning, configuration and programming of the IoT-Chip. It will assist you in getting familiar with the functional range and usage of the product.

This device contains open source software protected by respective third party proprietary and copy rights. Please refer to Appendix A for details.

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 the IoT-Chip (EMC law)

The IoT-Chip 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 IoT-Chip is ESD-sensitive and may only be unpacked, used and operated by trained personal at ESD-conform work stations.

The IoT-Chip 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 Product Description

2.1 Orderable Parts

Table 1: Orderable parts

Part number	Product name	Features
3390100	IoT-Chip DIL40	PLC Program Memory: 128kByte Connector: DIL40 RJ45 for Ethernet
3390101	IoT-Chip LGA	PLC Program Memory: 128kByte Connector: LGA
KIT-175	Development Kit for IoT-Chip	IoT-Chip DIL40 Baseboard for IoT-Chip

2.2 Technical Data

The IoT Chip SE is easy to integrate for use in Internet of Things/ Industry 4.0 applications and the connecting devices/machines to the cloud as these fields require. Like every of our devices the IoT Chip SE is prepared for the direct connection to Cloud. Due to the partnership with IBM we make your devices ready for IoT.
The benefits of the SYS TEC electronic IoT Chip SE lie in its use, security and cloud connection. On the chip are libraries and protocols such as CANopen, MQTT or Modbus available, which are ready to use immediately. The templates supplied in the source code are available to users as a starting point for their own customization. I²C and SPI make it possible to directly connect actuators and sensors. This allows the entire measurement, control and regulation to be handled on the IoT chip.
As a result, the chip works independently of the cloud – whereas with other IoT chip devices, these processes typically take place in the cloud. There is also no additional gateway required for cloud connection.

Our IoT Chip SE delivers M2M communication at the highest level, as well as being efficient and available at a moderate cost.

Figure 1: IoT-Chip SE (DIL40) by SYS TEC electronic GmbH

These are some significant features of the IoT-Chip:

- High-performance CPU kernel (ARM® 32-bit Cortex®-M7, 200 MHz CPU Clock, 428 DMIPS)
- PLC Program memory: 128kByte
- DIL-40 socket or LGA
- 1x 10/100 Mbps Ethernet LAN interface (with on–board PHY) and RJ45 connector
- 1x CAN 2.0B interface, usable as CANopen Manager (CiA 302-conform)
- 3x synchronous/asynchronous serial ports (USART)
- Up to 15 digital inputs
- Up to 13 digital outputs
- 4 analog inputs
- 2 analog outputs
- On-board peripherals:
  - RTC
  - Temperature sensor
- Programmable in IEC 61131-3
- Function block libraries for communication (CANopen, Ethernet, I2C, SPI, MQTT, Modbus, 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 (24mm x 70mm)
- Operating temperature: 0°C…70°C (optional: -45°C…85°C)

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 IoT-Chip 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 IoT-Chip 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 IoT-Chip provides up to 15 digital inputs (DI0...DI14, 3.3V level), up to 13 digital outputs (DO0...DO12, 3.3V level), 1 high-speed counter input 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 IoT-Chip 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 IoT-Chip by the company SYS TEC electronic GmbH. Hence, it is possible to program the IoT-Chip graphically in KOP/FUB, AS and CFC or textually in AWL or ST. Downloading the PLC program onto the module takes place via Ethernet. 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 IoT-Chip supports backward documentation of the PLC program as well as the debug functionality including watching and setting variables, single cycles, breakpoints and single steps.

2.3 Block Diagram

The IoT-Chip signals are combined in signal groups. A signal group is selected using the Configuration filed value in configuration section of PLC Process image. Some signal groups serve multiple functions multiplexed by Configuration field value.

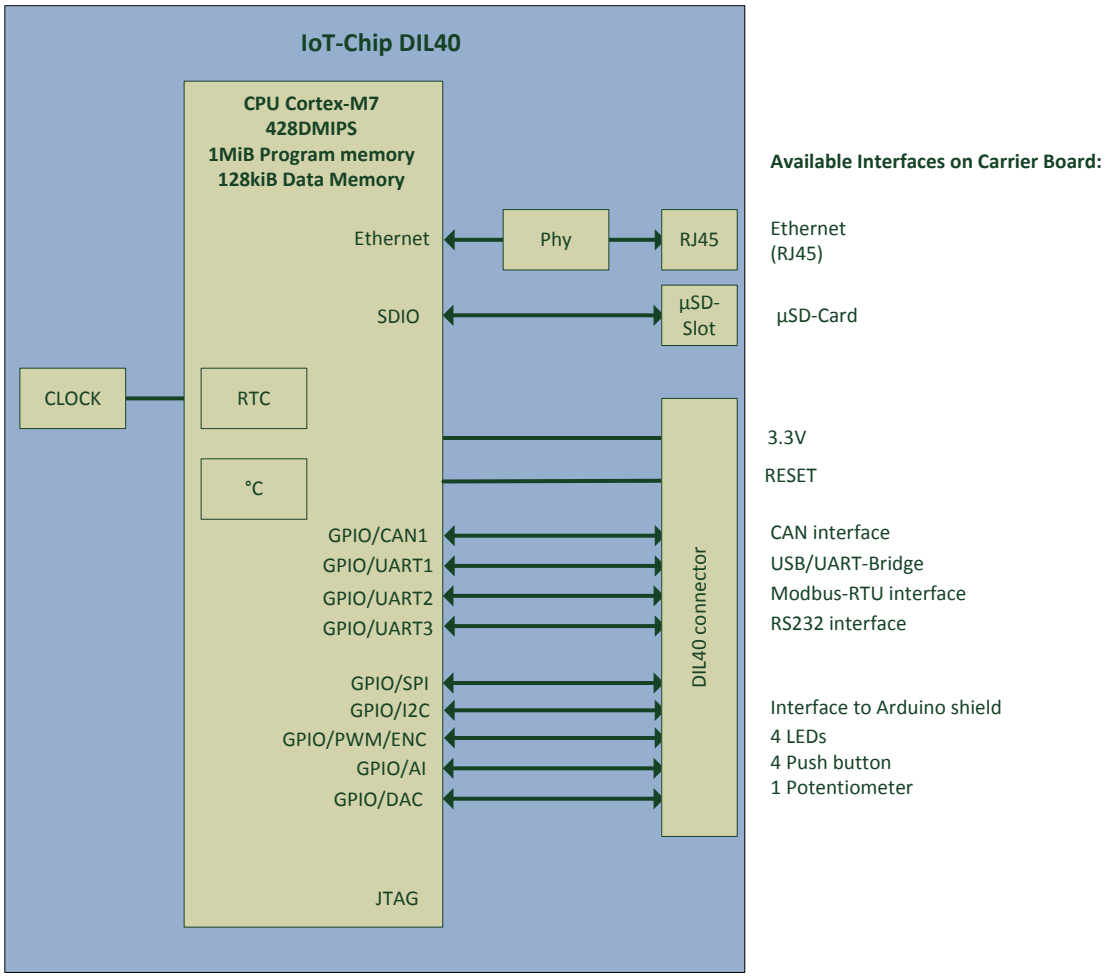

Figure 2: Block diagram IoT-Chip DIL40

3 Development Kit IoT-Chip DIL40

3.1 Overview

The Development Kit IoT-Chip 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. Additionally, it allows the user to get knowledge about the Internet of Things by offering a big variety of input and output interfaces.

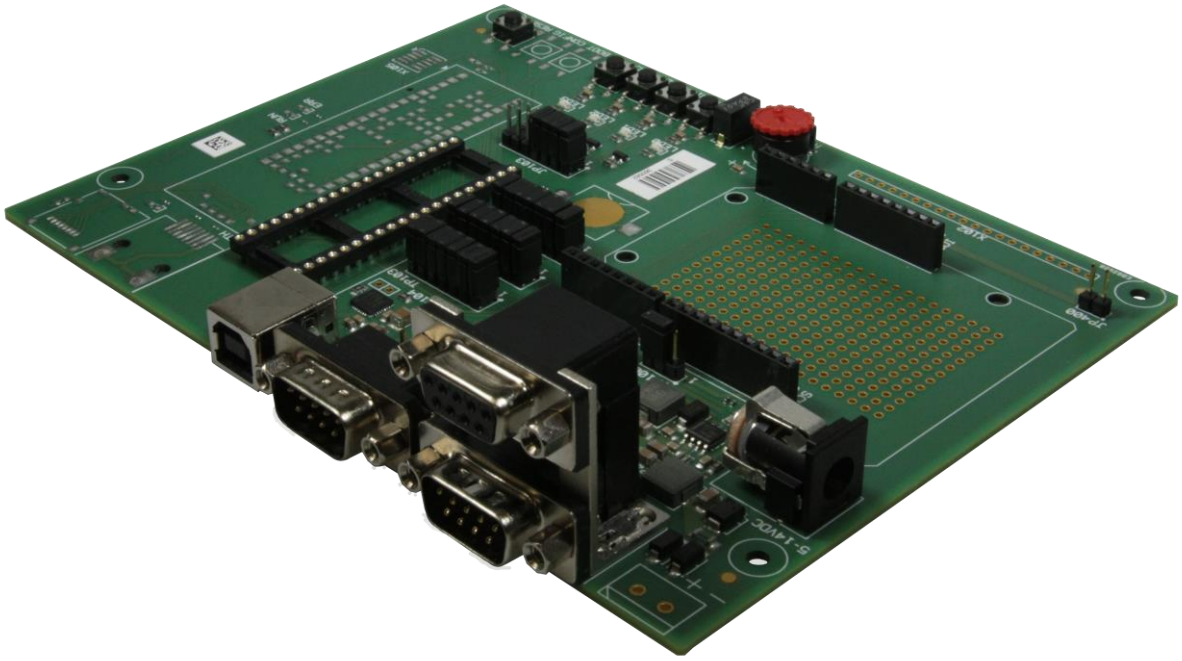

Figure 3: Development Kit IoT-Chip

The Development Kit IoT-Chip DIL40 ensures quick and problem-free commissioning of the IoT-Chip. Therefore, it combines all hard- and software components that are necessary to create own applications: the core module IoT-Chip DIL40, 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 IoT-Chip. It allows a cost-efficient introduction into the world of sensor2cloud communication and data preprocessing. All components included in the Kit enable in- and output extensions of IoT-Chip through CANopen-I/O-assemblies. Thus, the Development Kit may also be used for projects that require PLC with network connection.

The Development Kit IoT-Chip DIL40 contains the following hardware components:

- IoT-Chip (DIL40 connector)
- Development Board for the IoT-Chip

The Development Board included in the Kit facilitates quick commissioning of the IoT-Chip 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, one 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 pins of the IoT-Chip are linked to the baseboard and enable easy connection of own peripheral circuitry. Hence, the Development Board forms an ideal experimentation and testing platform for the IoT-Chip. The operating temperature should lie between 0°C and 70°C.

The *OpenPCS* IEC 61131 programming system included in the Kit serves as software development platform and as debug environment for the IoT-Chip. 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. 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.

3.2 Electric commissioning of the Development Kit IoT-Chip

The Development Kit IoT-Chip can be simply powered via USB. An external power adapter is only required for running the Development Kit 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 (for power supply and configuration via virtual serial interface, see section 10) and ETH. Table 2 provides an overview over the connections of the Development Kit IoT-Chip.

Table 2: Connections of the Development Kit IoT-Chip

Connection	Remark
USB (Power supply and COM0/RS232)	USB is used for: - Power supply and - Serial connection to run configuration shell (e.g. setting CAN and IP configuration, see section 10)
Alternative Power Supply	An external power supply (5...14V) can be used optionally for running the Development Kit stand alone, without connection to a Host PC.
ETH0 (Ethernet)	This interface serves as communication interface with the Programming PC and is necessary for the program, besides can be used freely for the user program.
CAN0	This interface can be used freely for the user program.
COM1 (Modbus)	This interface can be used freely for general operation of the user program.
COM2 (RS232)	This interface can be used freely for general operation of the user program.

Figure 4 shows the positioning of the most important connections of the Development Board for the IoT-Chip besides the Ethernet interface (ETH0). While the ETH0-interface of the Development Kit IoT-Chip DIL40 is placed on the IoT-Chip, in LGA-version it could be located on the baseboard next to the IoT-Chip.

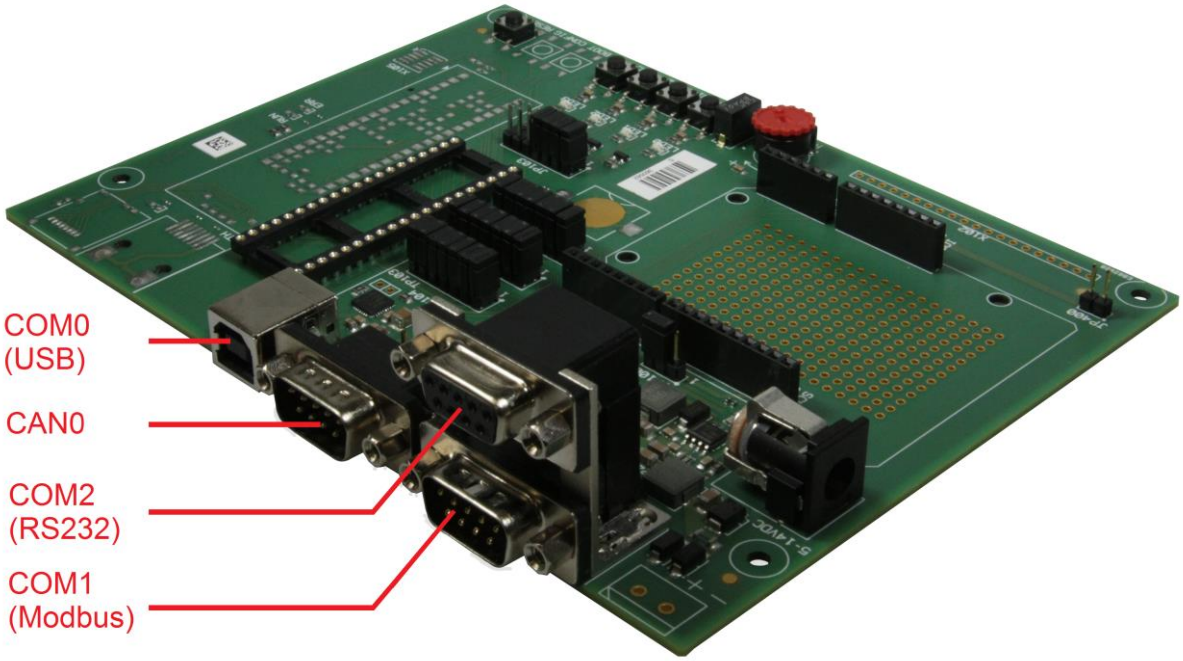

Figure 4: Most important connections on the Development Board of the IoT-Chip

3.3 Control elements of the Development Kit IoT-Chip DIL40

The Development Board allows the easy commissioning of the IoT-Chip. It has various control elements available to configure the module and to simulate inputs and outputs to the IoT-Chip. 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 IoT-Chip

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)
Pot (ADC)	R105	Analog Input AI0	(Process Image:	%IW8.0)
Run-LED	RUN	Display of activity state of the PLC (see section 6.6.1)		
Error-LED	ERROR	Display of error state of the PLC (see section 6.6.2)		

3.4 Optional accessory

3.4.1 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 IoT-Chip from PCs, e.g. laptop computers which do not have RS232 interfaces any more.

Figure 5: SYS TEC USB-RS232 Adapter Cable

3.4.2 CAN-Cable

The Development Board of the IoT-Chip features one CAN-interface. To connect the board to an existing CAN-network, a CAN-cable is needed. This cable is not part of the order. Of course SYS TEC electronic offers the production of customized cables on inquiry.

3.4.3 RS485-Cable

To establish a connection via Modbus, the Development Board of the IoT-Chip uses the COM1-interface. The cable should be attached to the female SUB-D9 connector on the baseboard. As this cable is not part of the order, contact SYS TEC electronic in case of need.

4 Electrical characteristic of the IoT-Chip

4.1 General operating conditions

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

Table 4: General operating conditions

Symbol	Parameter	Min	Typ	Max	Unit
3V3	Standard operating voltage 3V3	3,135	3,300	3,465	V
I_{3V3}	Total current consumption 3V3-Domain	-	250,000	450,000	mA
VREF	Reference voltage (3V3-VREF < 1.2V!)	1,800	-	3V3	V
I_{VREF}	VREF DC current consumption	-	300,000	500,000	µA
VBAT	Backup operating voltage (internal, external RTC)	1,800	-	3,600	V
I_{VBAT}	Backup domain supply current	-	-	TBD	µA
T_A	Operating Temperature Range	-40,000	-	85,000	°C

4.2 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
R_{AIN}	External input impedance	-	-	50	kΩ
C_{ADC}	Internal sample and hold capacitor	-	-	4	pF

Table 6: Analog output characteristic

Symbol	Parameter	Min	Typ	Max	Unit
DAC_OUT min	Lower DAC_OUT voltage with buffer OFF	-	0,5	-	mV
DAC_OUT min	Higher DAC_OUT voltage with buffer OFF	-	-	VREF-1LSB	V
R_O	Impedance output with buffer OFF	-	-	15	kΩ
C_{LOAD}	Capacitive load	-	-	50,000	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	3,300	5,200	V
V_{IH}	TC I/O input high level voltage	0,7 x 3V3	3,300	3,600	V
V_{hys}	IO FT Schmitt trigger voltage mV hysteresis	0,05 x 3V3			V
I_{lkg}	I/O FT input leakage current	-	-	3,000	µA
C_{IO}	I/O pin capacitance	-		5,000	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	2,40	-	-	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	-	-	150,000	mA
$I_{IOmaxsource}$	The maximum sum of I_{IO} sunk by the device	-	-	150,000	mA

4.3 Ethernet characteristics

In the table below you will find important information about the Ethernet characteristics of the IoT-Chip.

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

5 Pinout of the IoT-Chip

The IoT-Chip provides board I/O functionalities through simple connector footprint. Figure 6 shows the footprints of the Development Board and the IoT-Chip including the pin numbers.

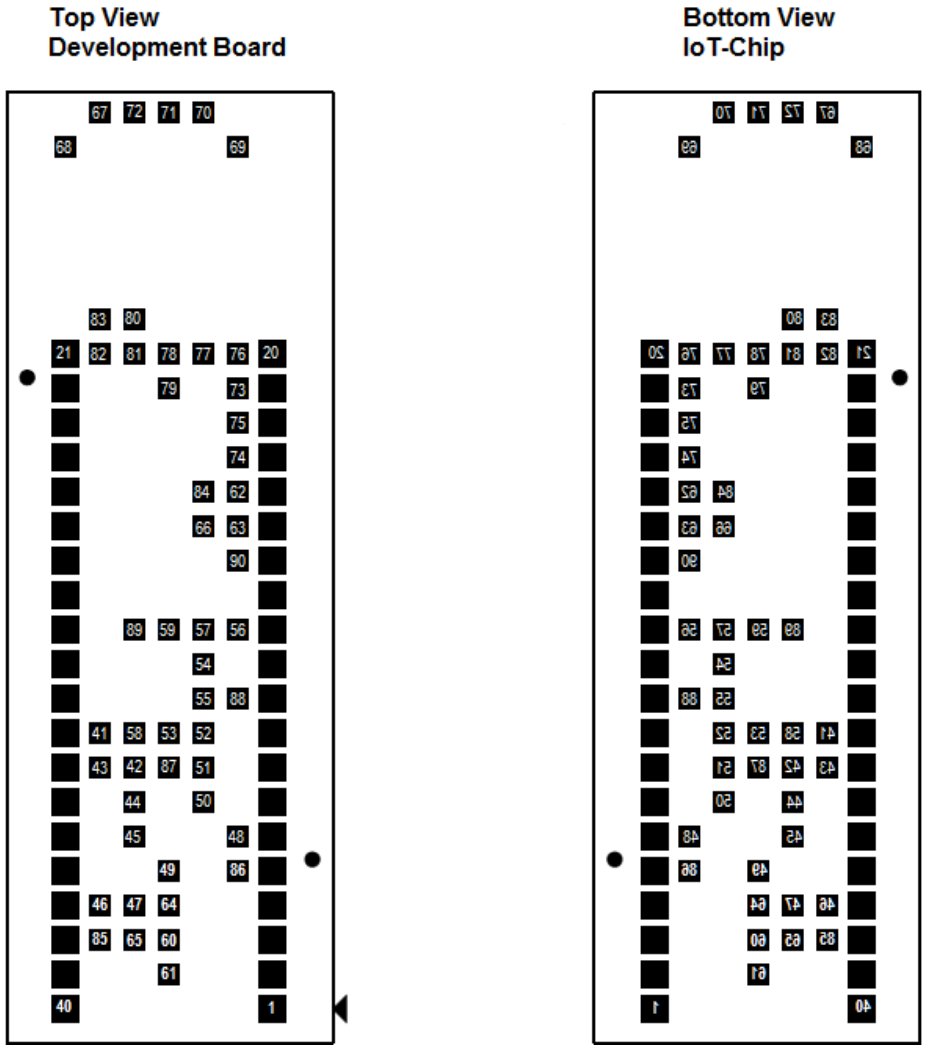

Figure 6: Footprints Development Board and IoT-Chip

Figure 7 exemplifies the dimensions of the Development Board and the IoT-Chip. With the help of Figure 6 and Figure 7 customers are able to develop own PCB´s in which the IoT-Chip could be integrated easily.

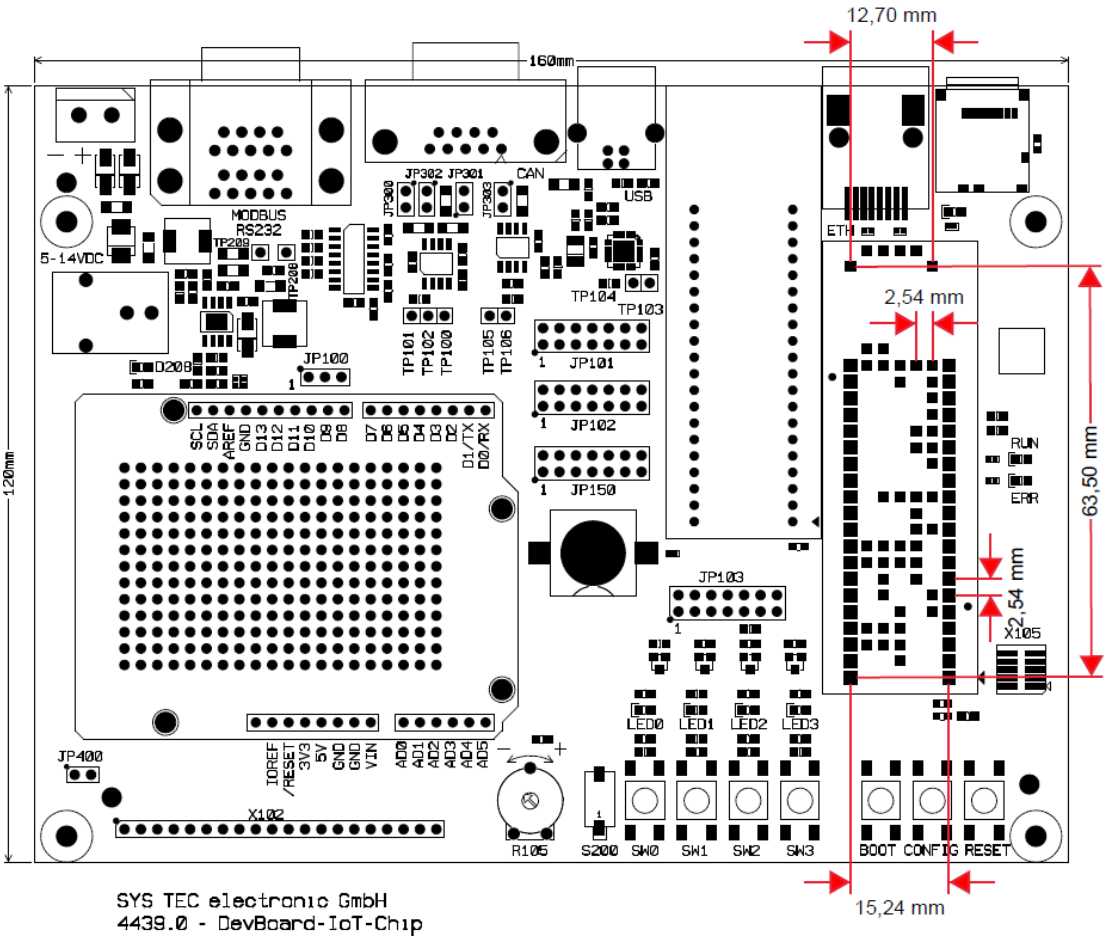

Figure 7: Dimensions of Development Board and IoT-Chip - top view

Table 11 lists all pins which are accessible by the customer and could be used by user-written PLC software.

Table 11: Pin Assignment

Pin #	Pin Label	PLC Function 1	PLC Function 2	Remark
1	3V3			
2	GND			
3	GPIO1	UART0_RXD		
4	GPIO2	UART0_TXD		
5	GPIO3	DO0 [User-LED0]	PWM0	
6	/RESET			
7	GPIO4	CAN0_TXD		
8	GPIO5	CAN0_RXD		
9	GPIO6	UART1_RXD		
10	GPIO7	UART1_TXD		
11	GPIO8	UART1_DE		
12	GPIO9	DO4	SPI0_/CS	To enable SPI, use command described in section 10.2.8.
13	GPIO10	DI5	SPI0_MISO	
14	GPIO11	DI6	SPI0_MOSI	
15	GPIO12	DI7	SPI0_CLK	
16	GPIO13	DI8	I2C0_SDA	To enable I2C, use command described in section 10.2.7.
17	GPIO14	DI9	I2C0_SCL	
18	GND			
19	GPIO15	DI0 [Push-button SW0]		

Pin #	Pin Label	PLC Function 1	PLC Function 2	Remark
20	GPIO16	DI1 [Push-button SW1]		
21	GND			
22	GPIO17	DI2 [Push-button SW2]		
23	GPIO18	DI4	ENC0 (CH1)	
24	GPIO19	DI5	ENC0 (CH2)	
25	GPIO20	DO1 [User-LED1]	PWM1	
26	GPIO21	DO2 [User-LED2]		
27	VREF			
28	GND			
29	GPIO22	AI0 [Poti]		
30	GPIO23	AI1		
31	GPIO24	AO0		
32	GPIO25	AO1		
33	GPIO26	AI2		
34	GPIO27	AI3		
35	GPIO28	DO3 [User-LED3]		
36	GPIO29	DI3 [Push-button SW3]		
37	GPIO30	UART2_CTS		
38	GPIO31	UART2_RTS		
39	GPIO32	UART2_TXD		
40	GPIO33	UART2_RXD		
41	GPIO34			Accessible via pins 2-20 on patch field X102 (only LGA-Version).
42	GPIO35			
43	GPIO36			
44	GPIO37			
45	GPIO38			
46	GPIO39			
47	GPIO40			
48	GPIO41			
49	GPIO42			
50	GPIO43			
51	GPIO44			
52	GPIO45			
53	GPIO46			
54	GPIO47			
55	GPIO48			
56	GPIO49			
57	GPIO50			
58	GPIO51			
59	GPIO52			

Pins 41-59 are accessible via patch field X102 (pins 2-20) only when using the LGA-version of the Development Kit. This feature is not integrated now and will be available in future.

To switch between PLC Functions use the Configuration Command Shell described in section 10. If the PLC Function 2 is enabled, corresponding pins are used for the new function and are no longer available as digital inputs or outputs.
Commands for enabling PLC Function 2 are listed in section 10.2.

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

Table 12: Connections of the IoT-Chip, only I/O, sorted by function

Pin #	I/O structure	PLC Function 1	PLC Function 2 A=alternative, S=simultaneous	Resistor Type internal	Resistor Type external
19	DI	DI0 [Push-button SW0]		Pull Down	
20	DI	DI1 [Push-button SW1]		Pull Down	
22	DI	DI2 [Push-button SW2]		Pull Down	
36	DI	DI3 [Push-button SW3]		Pull Down	
23	DI	DI4	A: ENC0 (CH1)	Pull Down	
24	DI	DI5	A: ENC0 (CH2)	Pull Down	
14	DI	DI6	A: SPI0_MISO	Pull Down	
15	DI	DI7	A: SPI0_CLK	Pull Down	
16	DI	DI8	A: I2C0_SDA	No Pull	Pull Up (1k Ω)
17	DI	DI9	A: I2C0_SCL	No Pull	Pull Up (1k Ω)
5	DO	DO0 [User-LED0]	A: PWM	No Pull	
25	DO	DO1 [User-LED1]	A: PWM	No Pull	
26	DO	DO2 [User-LED2]		No Pull	
35	DO	DO3 [User-LED3]		No Pull	
12	DO	DO4	A: SPI0_/CS	No Pull	
13	DO	DO5	A: SPI0_MOSI	Pull Down	
29	AI	AIN0 [POT]		No Pull	
30	AI	AIN1		No Pull	
33	AI	AIN2		No Pull	
34	AI	AIN3		No Pull	
32	AO	AO0		No Pull	
32	AO	AO1		No Pull	

6 PLC Functionality of the IoT-Chip

6.1 System start of the IoT-Chip

By default, the IoT-Chip loads all necessary firmware components upon Power-on or Reset and starts running the PLC program afterwards. Hence, the IoT-Chip 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:

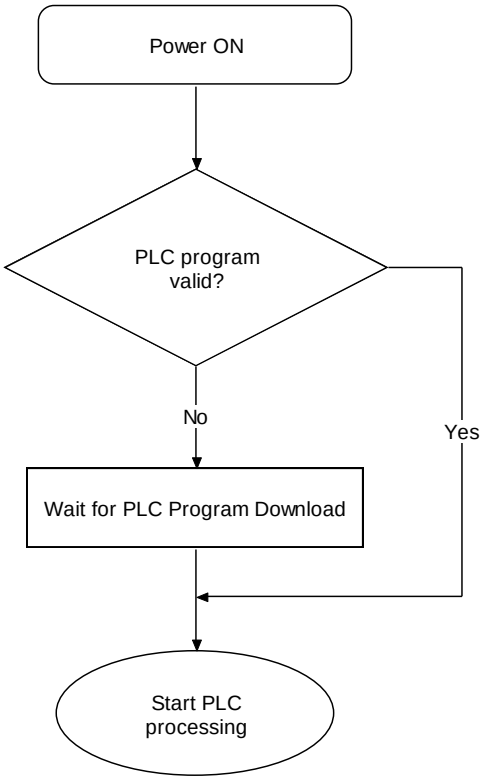

Figure 8: System start of the IoT-Chip

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 IoT-Chip is in stop mode.

6.2 Programming the IoT-Chip

The IoT-Chip 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"*.

The IoT-Chip 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 6.3) as well as the functionality of control elements (Run-LED, Error-LED).

The IoT-Chip firmware provides numerous function blocks to the user to access communication interfaces. Table 13 specifies the availability of FB communication classes (SIO, CAN, UDP).

Table 13: Support of FB communication classes for the IoT-Chip

Type of Interface	IoT-Chip Art. no: 3390100/3390101	Remark
CAN	X	FB description see manual L-1008
UDP	X	FB description see manual L-1054
SIO	X	FB description see manual L-1054

Table 35 in Appendix B contains a complete listing of firmware functions and function blocks that are supported by the IoT-Chip.

Detailed information about using the CAN interfaces in connection with CANopen is provided in section 6.7.

6.3 Process image of the IoT-Chip

6.3.1 Local In- and Outputs

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

Table 14: Assignment of in- and outputs to the process image of the IoT-Chip

I/O of the IoT-Chip	Address and Data type in the Process Image
DI0 ... DI5	**%IB0.0** as Byte with DI0 … DI5 %IX0.0 … %IX0.5 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)
C0	**%ID40.0** 31Bit + sign (-2^{31} - 2^{31} -1) Counter: counter input: DI2.4, direction: DI2.5, see section 6.5.1 AB Encoder: channel A: DI2.4, channel B: DI0.4, see section 6.5.1
On-board Temperature Sensor	**%ID72.0** 31Bit + sign as 1/10000 °C
DO0 ... DO4	**%QB0.0** as Byte with DO0 … DO4 %QX0.0 … %QX0.4 as single Bit for each output
AO0	**%QW8.0** 15Bit + sign (0 … +32767)
AO1	**%QW10.0** 15Bit + sign (0 … +32767)
P0	**%QX0.0** (default value for inactive generator) Impulse output: DO0, see section 6.5.2
P1	**%QX0.1** (default value for inactive generator) Impulse output: DO1, see section 6.5.2

In- and outputs of the IoT-Chip 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.

6.3.2 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 IoT-Chip. Therewith, all in- and outputs of the IoT-Chip can be flexibly adjusted to respective requirements. This implicates that the process image of the IoT-Chip 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

6.4 Communication interfaces

6.4.1 Serial interfaces

The IoT-Chip 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 10)
- Firmware updates (see section 7)
- PLC program download and debugging

On the Development Kit IoT-Chip the interface SIO0 is tunneled via USB on connector USB.

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.

6.4.2 CAN interfaces

The IoT-Chip features one CAN interface (CAN0). The CAN interface allows data exchange with other devices via network variables and it is accessible from a PLC program via function blocks of type *"CAN_Xxx"* (see section 6.7 and *"User Manual CANopen Extension for IEC 61131-3"*, Manual no.: L-1008).

Section 6.7 provides detailed information about the usage of the CAN interface in connection with CANopen.

6.4.3 Ethernet interfaces

The IoT-Chip features one Ethernet interface (ETH0). It serves as service interface to administer the IoT-Chip 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).

6.5 Specific peripheral interfaces

6.5.1 Counter inputs

The IoT-Chip features one fast counter input (C0). Prior to its usage, the counter input 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 6.3.1) or via function block *"CNT_FUD"*. Table 15 lists the allocation between counter channel 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	C0 (DI20) %IX2.4	DI21 %IX2.5	%ID40.0

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

Counter C0 also supports AB encoding in X4 mode. For using the AB encoding mode, it is necessary to use the counter input for channel A and DI4 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	C0 (DI20) %IX2.4	DI4 %IX0.4	%ID40.0

6.5.2 Pulse outputs

To release PWM signal sequences, the IoT-Chip 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 6.3.1). 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 outputs.

Table 18: Characteristics of PWM/PTO output

Function	Resolution	Max. Cycle Time	Min. Cycle Time
Pulse Width Modulation (PWM)	16 Bit	65535 ms (15mHz)	2 µs (500 kHz)

6.6 Control and display elements

There are three different LED mounted on the IoT-Chip (DIL40-version) or on the Development Board (LGA-version): Run-LED, Error-LED and the Link-LED of the Ethernet connection. The different states of the Run-LED and the Error-LED are explained in the following section.

6.6.1 Run-LED (green)

The 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"*.

6.6.2 Error-LED (red)

The 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.

6.7 Using CANopen for CAN interfaces

The IoT-Chip features one CAN interface (CAN0), 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 of the IoT-Chip.
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 IoT-Chip 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 IoT-Chip is optionally able to take over tasks of the CANopen Manager.

As CANopen Manager, the IoT-Chip 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.

6.7.1 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 IoT-Chip.

6.8 Controller specific PLC Function Blocks

6.8.1 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.

Definition of the Operands:

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

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

7 Updating the IoT-Chip Firmware

Updating the firmware of the IoT-Chip is not possible at the moment.
For further information please contact the technical support of SYS TEC electronic:

Telephone: +49 (0) 3765 / 38 600-0
E-Mail: support@systec-electronic.com

8 Baseboard Configuration

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

8.1 Voltage Reference

Connector JP 100 defines the value of the voltage reference. If pin 2 is connected to pin 1, the board uses the voltage which is connected to the pin labelled “AREF”. If pin 2 is connected to pin 3, the voltage reference is set to +3.3V.

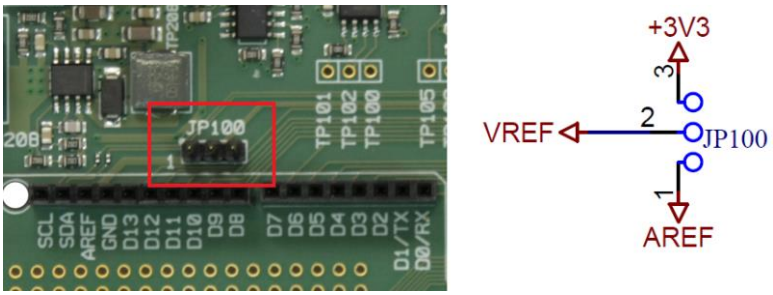

Figure 9: Position and Function of Connector JP100

8.2 GPIO on Connectors JP101 and JP102

These two connectors are responsible for connecting several GPIO pins of the IoT-Chip to the labelled pin header on the baseboard (see Table 22). See section 5 for further information of alternative functions of GPIO pins.

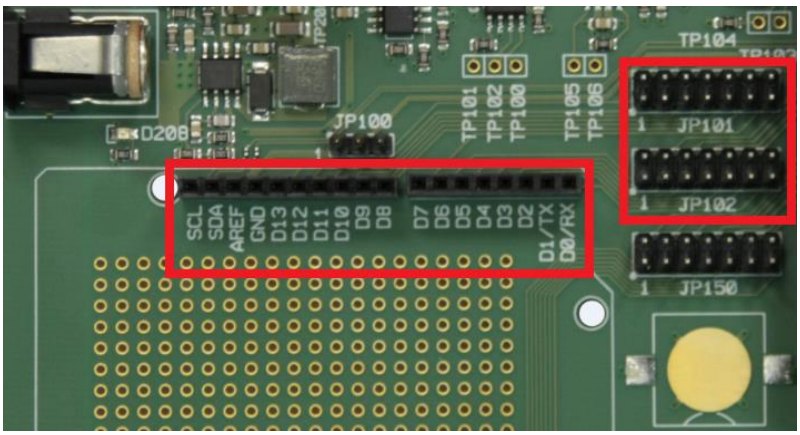

Figure 10: Connectors JP101 and JP102 and corresponding pin header

Table 22: Mapping of Connectors JP101 and JP102

Jumper on baseboard	Function on IoT-Chip	Label on Baseboard
JP101(1)	DI7/SPI0_CLK	D13
JP101(2)	DI6/SPI0_MISO	D12
JP101(3)	DO5/SPI0_MOSI	D11
JP101(4)	DO4/SPI0_CS	D10
JP101(5)	DO0/PWM	D9
JP101(6)	UART2_RTS	D8
JP101(7)	UART2_CTS	D7
JP102(1)	DO3	D6
JP102(2)	DO1/PWM	D5
JP102(3)	DI5/ENC0(CH1)	D4
JP102(4)	DI4/ENC0(CH2)	D3
JP102(5)	DI2	D2
JP102(6)	UART2_TXD	D1/TX
JP102(7)	UART2_RX	D0/RX

8.3 GPIO and electric potential on connector JP103

The first four jumpers on connector JP103 are used to enable or disable the four push buttons. While pins 9 and 10 offer access to GND, pins 11 and 12 are on an electrical potential of +3.3V.

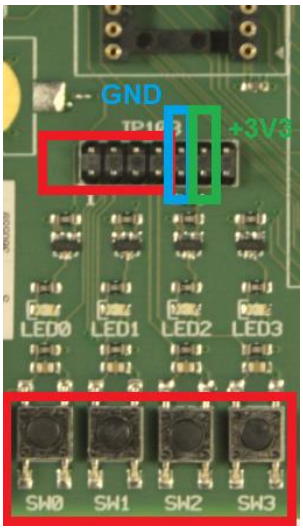

Figure 11: Function of connector JP103

8.4 GPIO on Connector JP150

The mapping of the analog inputs and outputs are defined by the jumpers on connector JP150. Further information are listed in Table 23.

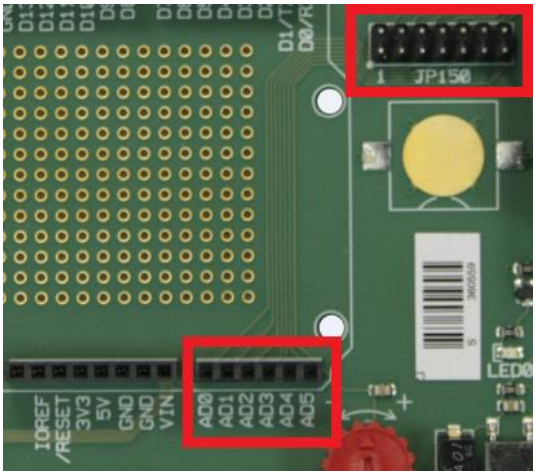

Figure 12: Connector JP150 and corresponding pin header

Table 23: Mapping of Connector JP150

Jumper on baseboard	Function on IoT-Chip	Label on Baseboard
JP150(1)	AI0	AD0
JP150(2)	AI1	AD1
JP150(3)	AI2	AD2
JP150(4)	AI3	AD3
JP150(5)	AO0	AD4
JP150(6)	AO1	AD5
JP150(7)	Connects potentiometer to AI0	

8.5 Serial Output

The IoT-Chip offers three different serial outputs. 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. On the Development Board it is tunneled via USB (see Figure 13).

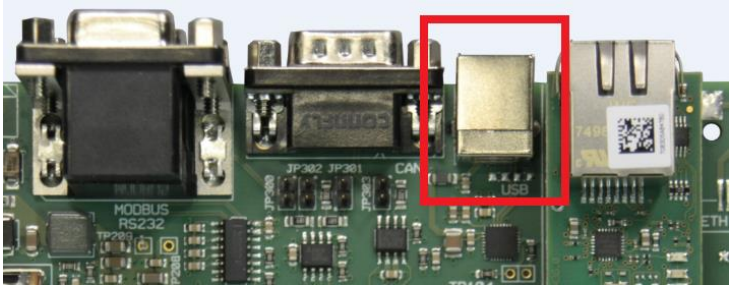

Figure 13: Position of the USB-Interface

SIO1(COM1) refers to the Modbus-RTU (RS485) and uses the male SUB-D 9 connector.
Jumpers 300-302 are responsible for terminating the Modbus. For a successful termination all three jumpers must be set. The positions of the jumpers are marked in Figure 14.

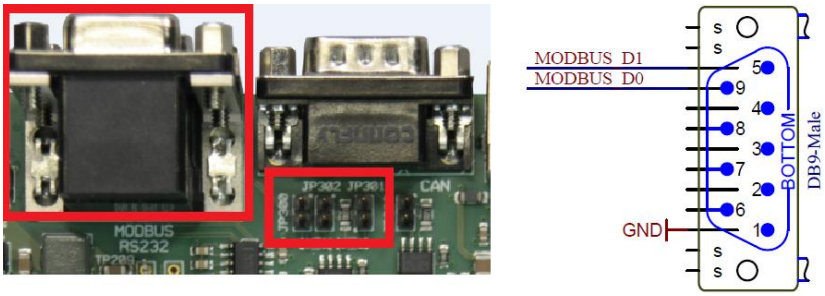

Figure 14: Interface, jumper and pin configuration for Modbus

The serial interface SIO2(COM2) uses the female SUB-D 9 connector positioned over the Modbus-Interface by RS232. The pin configuration for RS232 is shown in Figure 15.

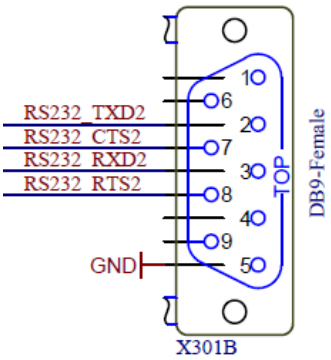

Figure 15: Pin configuration for RS232

8.6 Controller Area Network

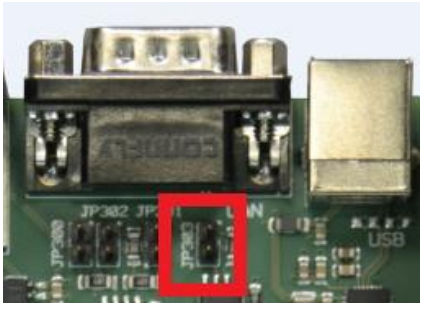

Figure 16: Position of JP303

The jumper JP303 is the termination jumper of the CAN interface and is only necessary if the CAN bus is not terminated by another device or resistor. If the jumper is set, the CAN bus is terminated by a resistor of 120 Ω.

9 OpenPCS Programming System

9.1 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.

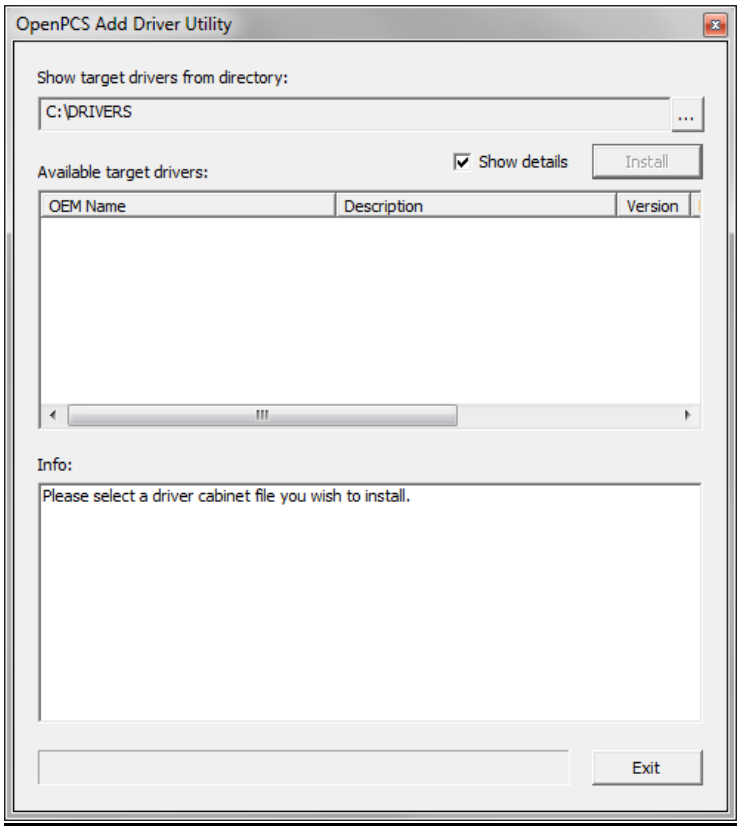

Figure 17: OpenPCS Installation Setup

Close the dialog in shown in Figure 17 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_702e_100.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"*.

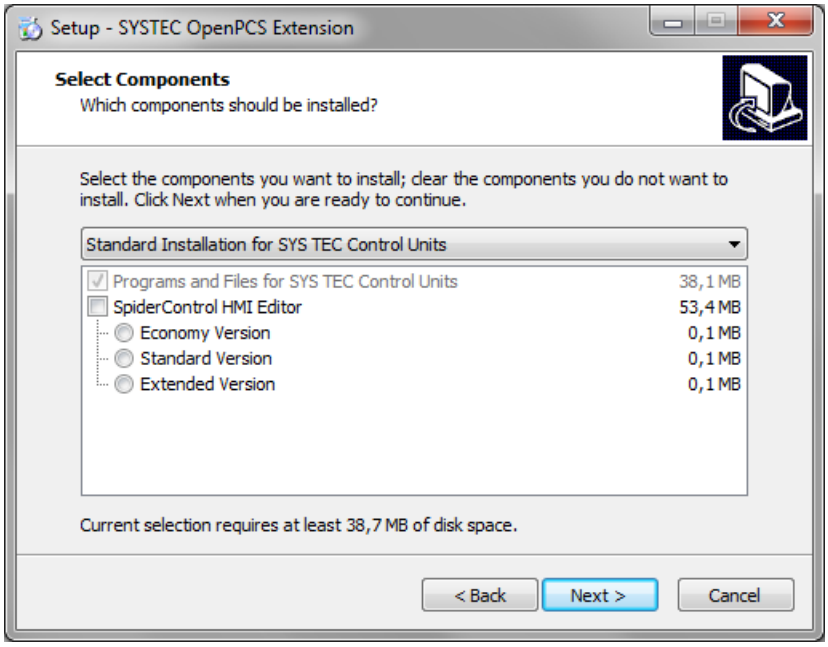

Figure 18: SYS TEC OpenPCS Extension select components dialog

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

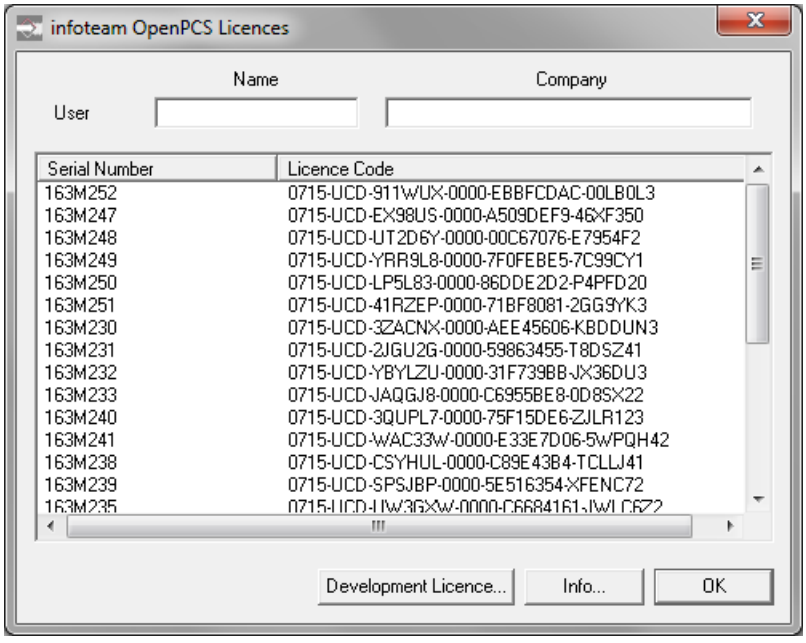

Figure 19: SYS TEC OpenPCS Extension Setup

- Finish the installation.

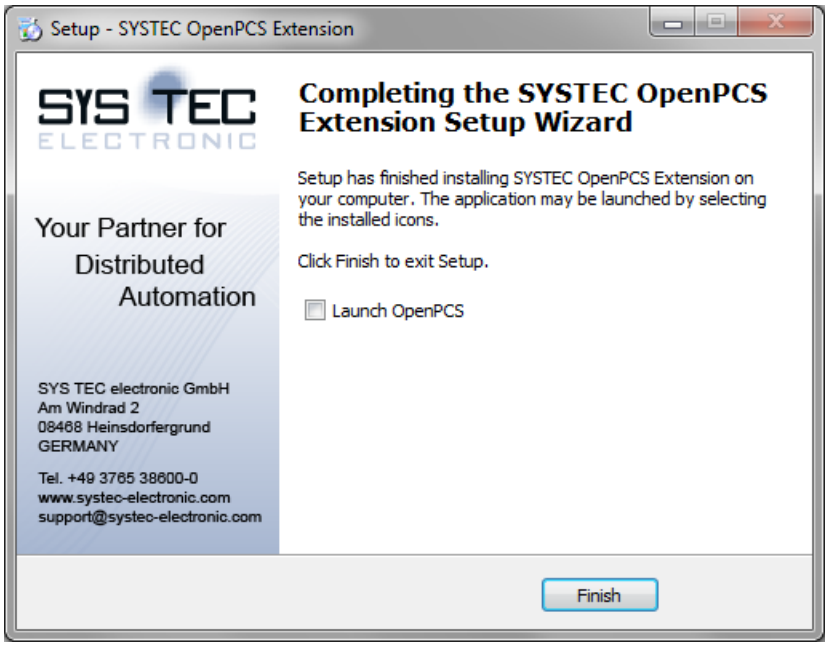

Figure 20: SYS TEC OpenPCS Extension Setup

9.2 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.

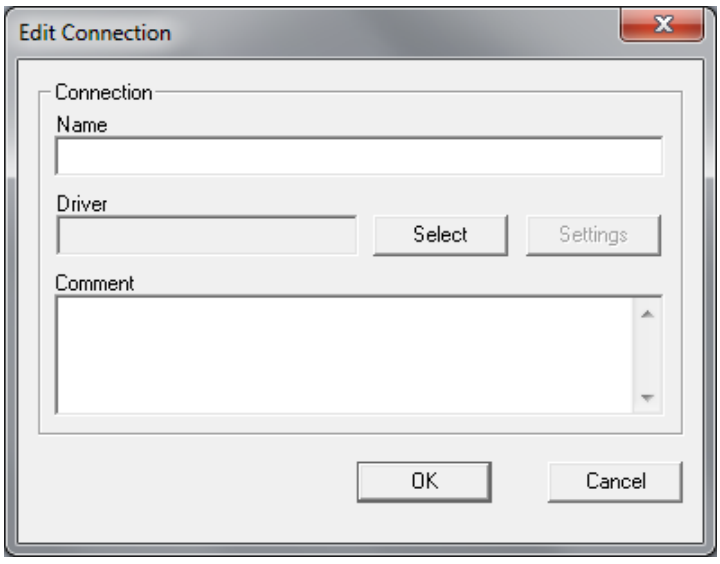

Figure 21: Target Connection – Edit Connection

Step 2:

- Choose the button “Select”.
- Select the driver *“SYSTEC Standard Driver”*. This is the first choice for many kinds of connections.

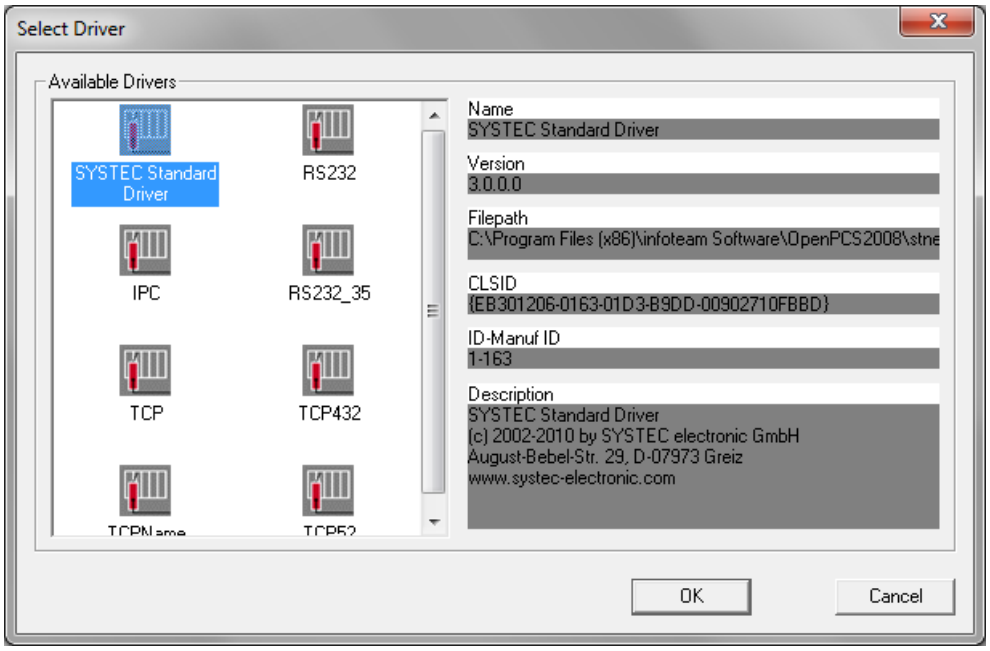

Figure 22: Target Connection – Select Driver

- 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:

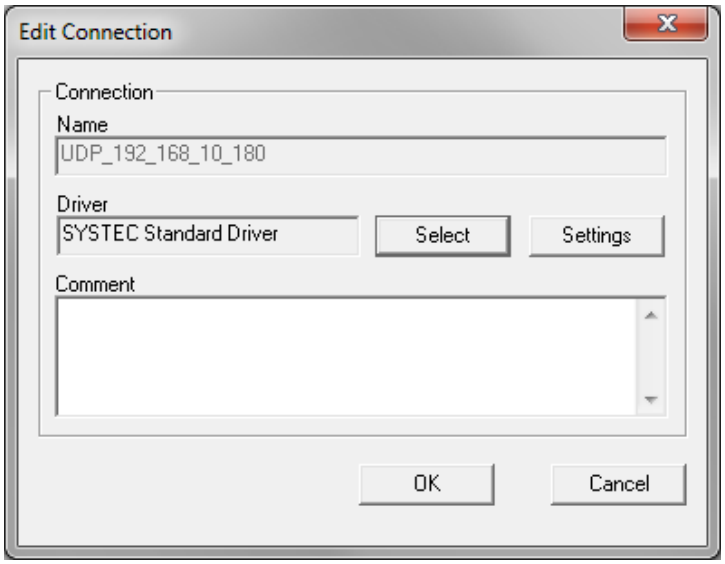

Figure 23: Target Connection – Edit Connection (filled)

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

9.3 Assign Network Connection to Resource

Step1:

- Select "PLC" ➢ "Resource Properties…" from the OpenPCS main menu.
- Choose "SYSTEC – IotChip-F767/Z5 (3390100/Z5)" for the Hardware Module setting and the Network Connection "UDP_192_168_10_180", which has been created in the previous section.

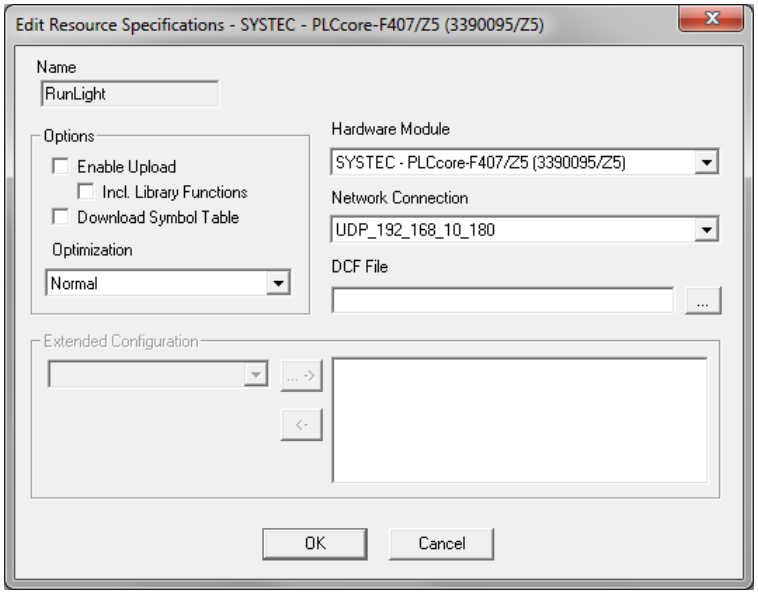

Figure 24: Resource Properties

- 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.

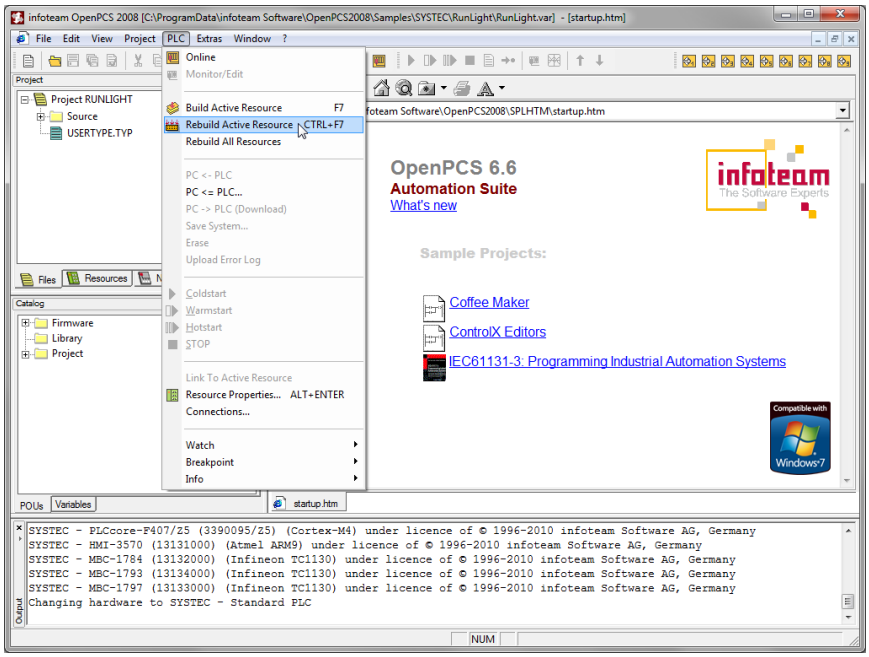

Figure 25: OpenPCS - Rebuild Active Resource

10 Configuration Command Shell

10.1 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.

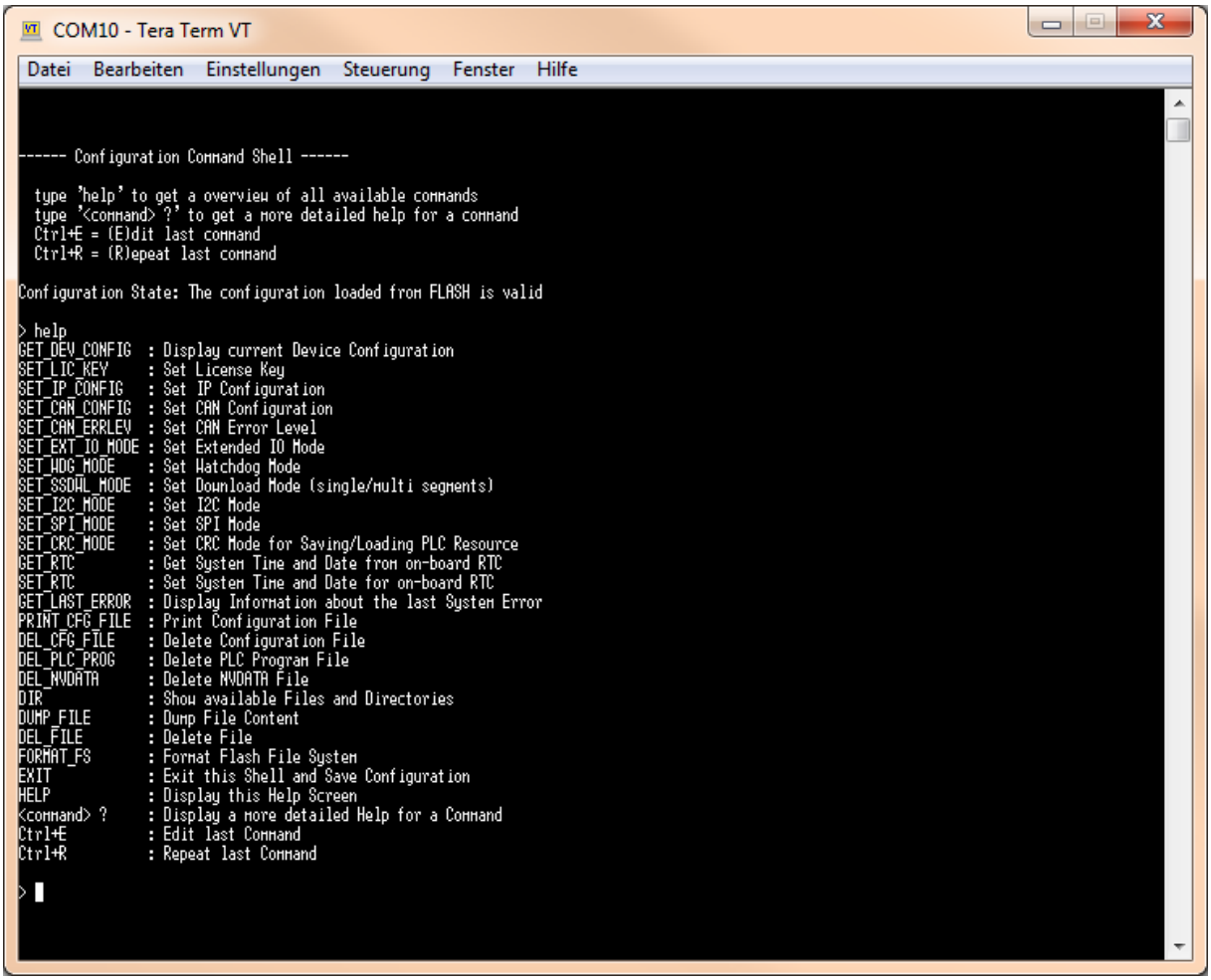

Figure 26: Built-in Configuration Command Shell

The Configuration Shell uses the serial interface SIO0 of the IoT-Chip. On the Development Kit the interface SIO0 is tunneled via USB on connector USB. 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 *"Tera Term"* 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 27):

- 115200 Baud
- 8 Data bit
- 1 Stop bit
- no parity
- no flow control

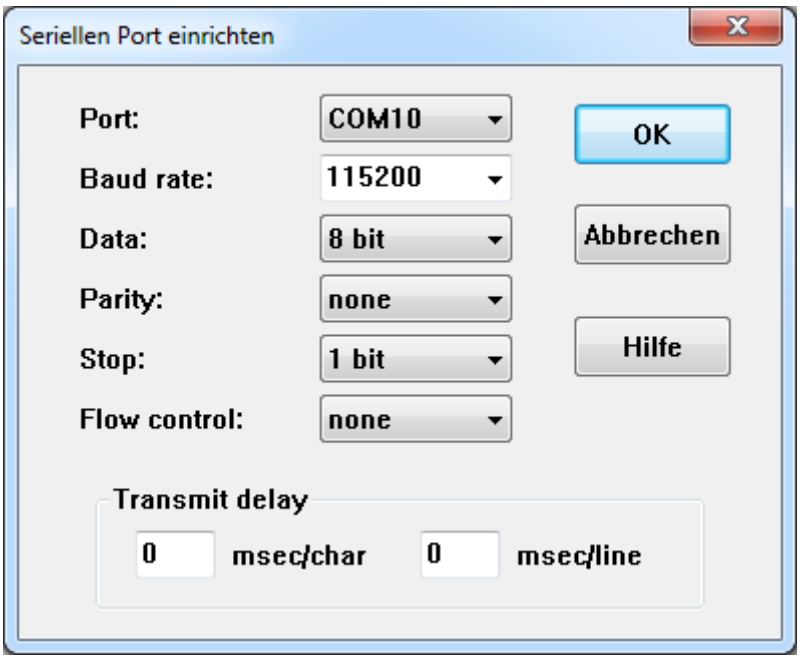

Figure 27: Terminal configuration using the example of "TeraTerm"

To start the Configuration Shell of the IoT-Chip move the dip switch 2 in the upper position and reboot the chip.

After entering the Command Shell, the status message should come up on the terminal window as shown in Figure 26. 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!

10.2 Command Description

The following section describes the commands available for the Command Configuration Shell on the IoT-Chip.

10.2.1 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 IotChip-F767/Z5 (3390100/Z5)
DeviceID:            1013005
Serial Number:       17070007
License Key:         RIKE-AJC8-05V3-62B0-JXJB-82EK-JLDT-8KEK
Product Code:        3390100
Hardware:
  CPU Board:         4436.00
  IO Board:          4439.00
I/O Driver:          1.00
Firmware:            5.10.01.00
Build Date/Time:     May 12 2017 / 11:32:43
Virtual Machine:     7.03 - 2 (#42)
OEM-ID:              163

Device Configuration:
Config File:         0:/IOTC-F767.cfg
PLC Backup Dir:      plcdata
Memory available:    128 kByte
ProcessImage:
  Extended I/Os:     disabled
RuntimeSettings:
  Watchdog:          Off
  I2C:               enabled
  SPI:               enabled
  SnglSegDwl Mode:   enabled
  CRC Mode:          enabled




ETH0 MAC Addr:       70-B3-D5-AB-47-AF
ETH0 IpAddr:         192.168.10.222
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
```

```
Ok
>
```

10.2.2 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 24: Parameter for command SET_IP_CONFIG

Parameters	Attribute	Values	Description
IfNum	Mandatory	0 -> ETH0	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
>
```

10.2.3 SET_CAN_CONFIG

The command *SET_CAN_CONFIG* is used to set the communication parameters of the CAN communication interface CAN0.

Command

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

Table 25: Parameter for command SET_CAN_CONFIG

Parameters	Attribute	Values	Description
IfNum	Mandatory	0 -> CAN0	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
>
```

10.2.4 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 IoT-Chip is online connected to the OpenPCS Programming System.

Command

```
SET_CAN_ERRLEV <IfNum ErrorLevel>
```

Table 26: Parameter for command SET_CAN_ERRLEV

Parameters	Attribute	Values	Description
IfNum	Mandatory	0 -> CAN0	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  0  all
Ok
>
```

10.2.5 SET_WDG_MODE

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

Command

```
SET_WDG_MODE <WdgMode>
```

Table 27: 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.

Example

```
> SET_WDG_MODE  off
Ok
>
```

10.2.6 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 IoT-Chip. 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.

Command

```
SET_SSDWL_MODE <SSDwlMode>
```

Table 28: 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
```

10.2.7 SET_I2C_MODE

The command SET_I2C_MODE is used to enable or disable the I²C communication of the IoT-Chip.

Command

```
SET_I2C_MODE <I2cMode>
```

Table 29: Parameter for command SET_SDCARD_MODE

Parameter	Attribute	Values	Description
I2cMode	Mandatory	on	Enable I²C
		off	Disable I²C

Example

```
> SET_I2C_MODE on
OK
>
```

10.2.8 SET_SPI_MODE

The command SET_SPI_MODE is used to enable or disable the SPI communication of the IoT-Chip.

Command

```
SET_ SPI_MODE <SpiMode>
```

Table 30: Parameter for command SET_SDCARD_MODE

Parameter	Attribute	Values	Description
SpiMode	Mandatory	on	Enable SPI
		off	Disable SPI

Example

```
> SET_ SPI_MODE on
OK
>
```

10.2.9 GET_RTC

This command 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:  2017/05/05
```

10.2.10 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 the IoT-Chip 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 31: 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 2017/05/05
OK
```

10.2.11 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 32: 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
>
```

10.2.12 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
>
```

10.2.13 PRINT_CFG_FILE

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

Example

```
> PRINT_CFG_FILE

[ETH0]
IpAddr=192.168.10.222
SubnetMask=255.255.255.0
Gateway=192.168.10.1
PortNum=8888

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

[PlcSettings]
EnableExtIo=0
WatchdogMode=2
EnableI2c=1
EnableSpi=1
SnglSegDwlMode=1
CrcCheckMode=1

Ok
>
```

10.2.14 DEL_CFG_FILE

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

Example

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

10.2.15 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.

Example

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

10.2.16 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
>
```

10.2.17 DIR

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

Example

```
> DIR

Files and Directories of Flash File System:

 + 0:/
 |
 + -- plcdata
 |      |
 |      + -- PLCARCHV.BIN     13699 Bytes
 +
 |
 + -- IOTC-F~1.CFG             271 Bytes

Ok

>
```

10.2.18 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 33: 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 IOTC-F~1.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 32 32 32 0D 0A 53  92.168.10.222..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 50 6C 63 53 65 74 74 69 6E  l=2...[PlcSettin
000000B0:   67 73 5D 0A 45 6E 61 62 6C 65 45 78 74 49 6F 3D  gs].EnableExtIo=
000000C0:   30 0D 0A 57 61 74 63 68 64 6F 67 4D 6F 64 65 3D  0..WatchdogMode=
000000D0:   32 0D 0A 45 6E 61 62 6C 65 49 32 63 3D 31 0D 0A  2..EnableI2c=1..
000000E0:   45 6E 61 62 6C 65 53 70 69 3D 31 0D 0A 53 6E 67  EnableSpi=1..Sng
000000F0:   6C 53 65 67 44 77 6C 4D 6F 64 65 3D 31 0D 0A 43  lSegDwlMode=1..C
00000100:   72 63 43 68 65 63 6B 4D 6F 64 65 3D 31 0D 0A     rcCheckMode=1..

Ok
>
```

10.2.19 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 34: Parameter for command DEL_FILE

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

Example

```
> DEL_FILE  IOTC-F~1.CFG
Ok
>
```

10.2.20 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**!

Example

```
> 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 FLASH File System need some seconds, please wait…

OK
>
```

10.2.21 EXIT

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

10.2.22 HELP

The command *HELP* shows all available commands as well as a short command description (*see* Figure 26).

Appendix A: Third Party Software Components

This device contains open source software protected by respective third party proprietary and copy rights.

Neither SYS TEC electronic GmbH or above third party providers are responsible for any ill-intend or inappropriate use of the provided software. IN NO EVENT SHALL SYS TEC electronic GmbH BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

[1] lwIP

lwIP (lightweight IP) is an lightweight open source TCP/IP stack designed for embedded systems.

Project URL: http://savannah.nongnu.org/projects/lwip/

Copyright and license:

```
/*
 * Copyright (c) 2001, 2002 Swedish Institute of Computer Science.
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * 1. Redistributions of source code must retain the above copyright notice,
 *    this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. The name of the author may not be used to endorse or promote products
 *    derived from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
 * OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
 * IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
 * INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
 * NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
 * DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
 * THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 * This file is part of the lwIP TCP/IP stack.
 *
 * Author: Adam Dunkels <adam@sics.se>
 *
 */
```

[2] Paho MQTT Embedded/C

The Eclipse Paho MQTT package is a client library for MQTT embedded devices.

Project URL: https://github.com/eclipse/paho.mqtt.embedded-c

Copyright and license:


```
Eclipse Distribution License - v 1.0

Copyright (c) 2007, Eclipse Foundation, Inc. and its licensors.

All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this
list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this
list of conditions and the following disclaimer in the documentation and/or
other materials provided with the distribution.

Neither the name of the Eclipse Foundation, Inc. nor the names of its
contributors may be used to endorse or promote products derived from this
software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

Appendix B: Firmware function scope of IoT-Chip

Table 35 lists all firmware functions and function blocks available on IoT-Chip.

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 35: Firmware functions and function blocks of IoT-Chip

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} (Default), see (1)
NVDATA_INT	FB	L-1054	DEVICE:={0} (Default), see (1)
NVDATA_STR	FB	L-1054	DEVICE:={0} (Default), see (1)
NVDATA_BIN	FB	L-1054	DEVICE:={0} (Default), 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}
PTO_TAB	FB	L-1054	CHANNEL:={0,1}
Functions and Function Blocks for Serial interfaces			
SIO_INIT	FB	L-1054	PORT:={0,1,2}(2)
SIO_STATE	FB	L-1054	PORT:={0,1,2}(2)
SIO_READ_CHR	FB	L-1054	PORT:={0,1,2}(2)
SIO_WRITE_CHR	FB	L-1054	PORT:={0,1,2}(2)
SIO_READ_STR	FB	L-1054	PORT:={0,1,2}(2)
SIO_WRITE_STR	FB	L-1054	PORT:={0,1,2}(2)
SIO_READ_BIN	FB	L-1054	PORT:={0,1,2}(2)
SIO_WRITE_BIN	FB	L-1054	PORT:={0,1,2}(2)
Functions and Function Blocks for CAN interfaces / CANopen			
CAN_GET_LOCALNODE_ID	FB	L-1008	NETNUMBER:={0} (Default)
CAN_CANOPEN_KERNEL_STATE	FB	L-1008	NETNUMBER:={0} (Default)
CAN_REGISTER_COBID	FB	L-1008	NETNUMBER:={0} (Default)
CAN_PDO_READ8	FB	L-1008	NETNUMBER:={0} (Default)
CAN_PDO_WRITE8	FB	L-1008	NETNUMBER:={0} (Default)
CAN_SDO_READ8	FB	L-1008	NETNUMBER:={0} (Default)
CAN_SDO_WRITE8	FB	L-1008	NETNUMBER:={0} (Default)
CAN_SDO_READ_STR	FB	L-1008	NETNUMBER:={0} (Default)
CAN_SDO_WRITE_STR	FB	L-1008	NETNUMBER:={0} (Default)
CAN_SDO_READ_BIN	FB	L-1008	NETNUMBER:={0} (Default)
CAN_SDO_WRITE_BIN	FB	L-1008	NETNUMBER:={0} (Default)
CAN_GET_STATE	FB	L-1008	NETNUMBER:={0} (Default)
CAN_NMT	FB	L-1008	NETNUMBER:={0} (Default)
CAN_RECV_EMCY_DEV	FB	L-1008	NETNUMBER:={0} (Default)
CAN_RECV_EMCY	FB	L-1008	NETNUMBER:={0} (Default)
CAN_WRITE_EMCY	FB	L-1008	NETNUMBER:={0} (Default)
CAN_RECV_BOOTUP_DEV	FB	L-1008	NETNUMBER:={0} (Default)
CAN_RECV_BOOTUP	FB	L-1008	NETNUMBER:={0} (Default)
CAN_ENABLE_CYCLIC_SYNC	FB	L-1008	NETNUMBER:={0} (Default)
CAN_SEND_SYNC	FB	L-1008	NETNUMBER:={0} (Default)

Name	Type	Reference	Remark
CANL2_INIT	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_SHUTDOWN	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_RESET	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_GET_STATUS	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_DEFINE_CANID	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_DEFINE_CANID_RANGE	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_UNDEFINE_CANID	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_UNDEFINE_CANID_RANGE	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_MESSAGE_READ8	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_MESSAGE_READ_BIN	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_MESSAGE_WRITE8	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_MESSAGE_WRITE_BIN	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_MESSAGE_UPDATE8	FB	L-1008	NETNUMBER:={0} (Default)
CANL2_MESSAGE_UPDATE_BIN	FB	L-1008	NETNUMBER:={0} (Default)
Functions and Function Blocks for Ethernet interfaces / UDP			
LAN_GET_HOST_CONFIG	FB	L-1054	NETNUMBER:={0} (Default)
LAN_ASCII_TO_INET	FB	L-1054	NETNUMBER:={0} (Default)
LAN_INET_TO_ASCII	FB	L-1054	NETNUMBER:={0} (Default)
LAN_GET_HOST_BY_NAME	FB	L-1054	NETNUMBER:={0} (Default)
LAN_GET_HOST_BY_ADDR	FB	L-1054	NETNUMBER:={0} (Default)
LAN_UDP_CREATE_SOCKET	FB	L-1054	NETNUMBER:={0} (Default)
LAN_UDP_CLOSE_SOCKET	FB	L-1054	NETNUMBER:={0} (Default)
LAN_UDP_RECVFROM_STR	FB	L-1054	NETNUMBER:={0} (Default)
LAN_UDP_SENDTO_STR	FB	L-1054	NETNUMBER:={0} (Default)
LAN_UDP_RECVFROM_BIN	FB	L-1054	NETNUMBER:={0} (Default)
LAN_UDP_SENDTO_BIN	FB	L-1054	NETNUMBER:={0} (Default)
Functions and Function Blocks for Modbus			
MODBUS_OPEN_INSTANCE	FB	L-1829	
MODBUS_CLOSE_INSTANCE	FB	L-1829	
MODBUS_REGISTER_VAR_LIST	FB	L-1829	
MODBUS_READ_REGS	FB	L-1829	
MODBUS_WRITE_SINGLE_REG	FB	L-1829	
MODBUS_WRITE_MULTI_REGS	FB	L-1829	
MODBUS_READ_WRITE_REGS	FB	L-1829	
MODBUS_READ_INPUT_REGS	FB	L-1829	
MODBUS_READ_DISCR_INPUTS	FB	L-1829	
MODBUS_READ_COILS	FB	L-1829	
MODBUS_WRITE_SINGLE_COIL	FB	L-1829	
MODBUS_RAW_PDU_REQUEST	FB	L-1829	
Functions and Function Blocks for I2C			
tbd			
Functions and Function Blocks for SPI			
tbd			
Functions and Function Blocks for MQTT			
tbd			

(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) As chapter 6.4.1 states, the IoT-Chip 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 IoT-Chip
Document number:	L-1966e_1, June 2017

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