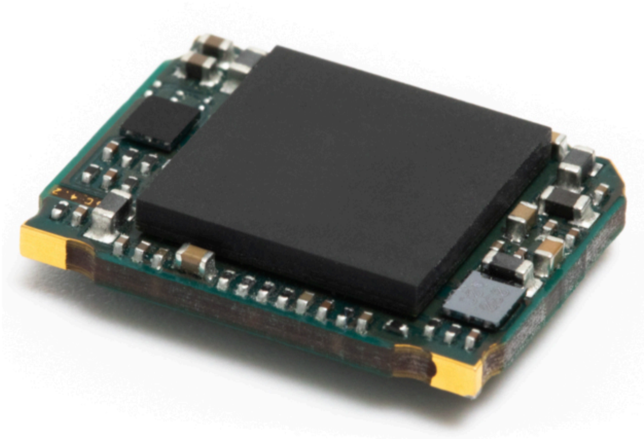
periCORE

Single Pair Ethernet communication module



Datasheet

Seamless IoT Connectivity

Abstract

This technical datasheet describes the *periCORE* multi-interface IIoT module, optimized to connect sensors and actuators to the Local Area Network via Single Pair Ethernet. Its small form factor allows it to be integrated into sensor-housings in order to tie them directly to the IT world. Further, it offers a variety of interfaces on the sensor or actuator side for most analogue front end ICs.

Doc-No.: PRN.100.375 rev: 2

July 22, 2022

Document Information

Title	periCORE
Subtitle	Single Pair Ethernet communication module
Type	Datasheet
Status	Release
Version	2
Date	July 22, 2022
Disclosure Restriction	

Contents

1 Overview

periCORE is an Ethernet communication module, which is designed to be integrated into sensor and actuator devices. It will provide networking capability to these devices both in hardware and in software, so that it can be easily integrated. It allows turning formerly passive sensors and actuators into intelligent devices, which can preprocess data and can operate event based, while all the network stuff, including state-of-the-art security and firmware management, is pre-implemented and ready to use. Further, it allows rebranding and customization for your devices with minimal development efforts. Everyone can create a customized Firmware for the periCORE with the supplementary development kit.

Targeted Applications

- Industrial sensors
- Industrial control
- IoT / IIoT
- Remote sensor access
- Building automation

Key Features

- Fully qualified Industrial IoT module
- Firmware development framework
- Provided TCP/IPv6 stack
- Event-based minimal operating system
- arm Cortex®-R4 250MHz processor core
- 32-MBit flash memory for persistent storage
- Up to 3x 100BASE-T1 Single Pair Ethernet Phys (IEEE 802.3bw compatible)
- Integrated Ethernet switching core
- Compact form factor
- Operated with 24V
- Integrated 3V3 power supply

Interfaces

- 2 x 100BASE-T1 Phy (IEEE 802.3bw)
- 1 x Combined 100BASE-T1/TX Phy
- 1 x MAC to arm processor core (Figure 1)
- 1 x UART
- 1 x I2C (400 kHz)
- 2 x GPIO

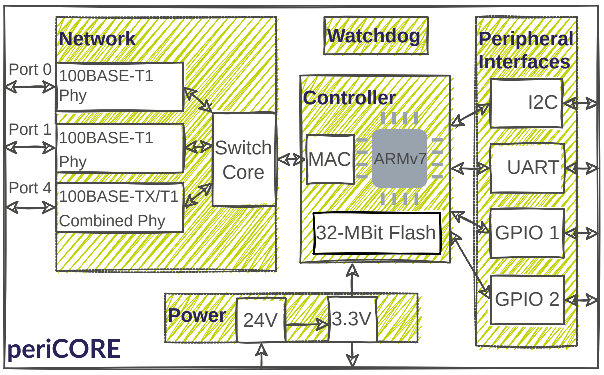


Figure 1: periCOREs hardware blocks.

Operational Parameters

- Operating voltage: 24 VDC
- Power supply: 3.3 VDC (up to 100mA)
- Temperature range: -40°C to +85°C
- Power consumption: 0.6 W

Package

Dimensions: 16.7 x 13 x 3.8 mm (Figure 2)

Mounting: Solder pads, 73 LGA-Pads, Pattern 13 x 10, Pitch 1.27 mm

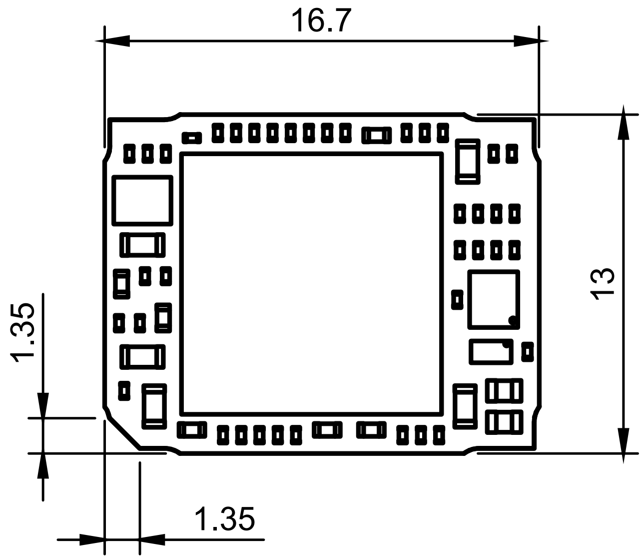


Figure 2: periCOREs dimensions in mm.

Compliance

- RoHS
- WEEE

Security

- NIST compliant TLS implementation
- Role Based Access Control (RBAC)
- Certificate based client authentication
- AES encryption algorithm
- X.509 certificates and PKIX path validation
- Elliptic Curve Cryptography (ECC)

Software Library *libperiCORE*

- Rapid firmware development with *periCORE Development Kit* (see Figure 3)
- mDNS/LLMNR for name resolving
- DNS-SD for automated service discovery
- TCP/UDP endpoints
- TLS-based secure communication endpoints
- RESTful API
- Secure MQTT-client for publishing sensor values or subscribing to actuator commands
- HTTPs server including Web based UI
- Product lifecycle features
- C++20 standard conform

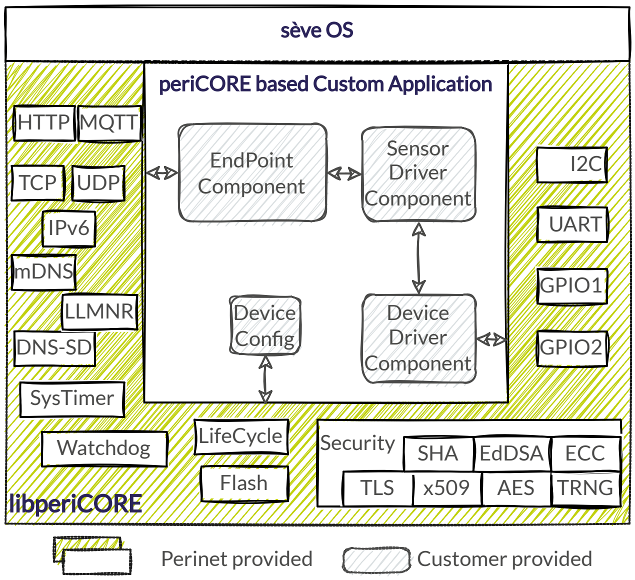


Figure 3: The software architecture with Custom Application template, provided by Perinet.

2 Module Architecture

The *periCORE* communcation module offers a variety of external interfaces to accommodate for the needs of an IIoT Industry 4.0 environment. Those interfaces can be split into three essential blocks that are connected to the internal controller. An overview of those blocks is shown in Figure 4. In addition, the Micro Controller Unit (MCU) is visualized as hardware block Controller.

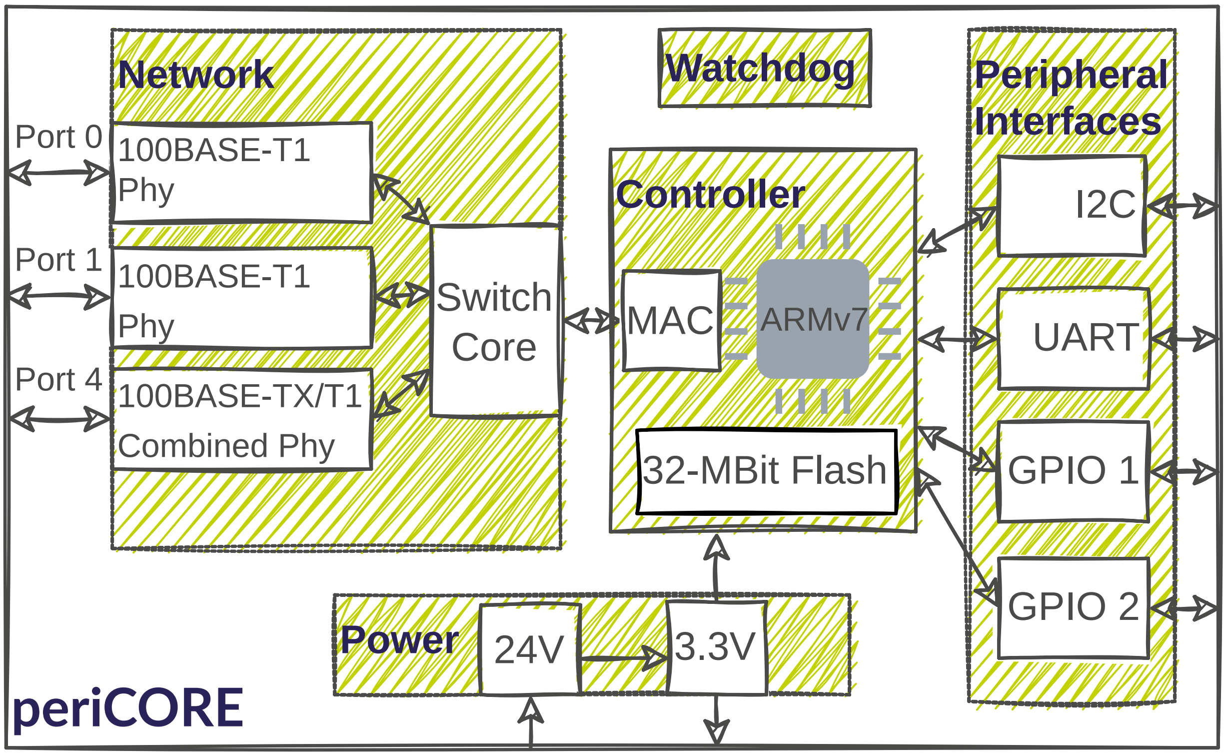


Figure 4: Included hardware blocks of the *periCORE* communcation module .

Network 3 ports for Ethernet communication, which are compatible to 100BASE-T1, are available for SPE communication. Port 4 can either be configured to 100BASE-T1 or 100BASE-TX. For further details, see Section 2.1.

Peripheral Interfaces To retrieve sensor data or control actuators, respectively, the *periCORE* communcation module provides I2C, 2-wire UART and two GPIO ports as peripheral interfaces. They allow the integration of OEMs sensor and/or actuator products and are described in more details in Section 2.2.

Power To power peripheral circuits, a 3.3V output is provided alongside the 24V input, as is shown in the bottom power side. The power block is described in Section 2.3

Controller The MCU executes a platform specific firmware which is dedicated to the sensor/actuator application. It can be programmed to fetch platform specific sensor/actuator data and handles the configuration of the Network interfaces(see Section 2.4).

Watchdog Implements an independent circuit to reset the *periCORE* communcation module in an error condition (see Section 2.5).

2.1 Network Interfaces

Port	Communication Standard	Default
Port 0	100BASE-T1(master,slave)	100BASE-T1(slave)
Port 1	100BASE-T1(master,slave)	100BASE-T1(master)
Port 4	100BASE-T1(master,slave); 100BASE-TX	100BASE-TX

Table 1: Network Interfaces of the *periCORE* communcation module

2.1.1 100BASE-T1 Ports 0, 1 and 4

The periCORE module provides two 100BASE-T1 compliant Ethernet PHYs and a hybrid 100BASE-T1/100BASE-TX PHY.

A 100BASE-T1 PHY can operate either in master or slave mode [15]. The physical layer defines a link between a 100BASE-T1 PHY in master and a 100BASE-T1 PHY in slave mode. A link between two 100BASE-T1 PHYs in master mode nor two 100BASE-T1 PHYs in slave mode cannot be established. Each port is statically configured to master or slave mode during bootup (see: Table 1).

An automatic negotiation for master or slave mode of the 100BASE-T1 PHY is not defined in the physical layer specification (IEEE 802.3bw standard [15]) and is also not implemented within the *periCORE* communcation module module.

Each port is connected with a 100MBit full duplex MAC to the switch core.

Note: The Port 4 of the *periCORE* communcation module provides either 100BASE-T1 or 100BASE-TX functionality. It needs to be configured accordingly.

A typical 100BASE-T1 application circuit where Port 0 is used (although it also applies for other ports) is shown below, in Figure 5. In this example, a hybrid cabling, where the data and power are transferred through the separate wire pairs, was used.

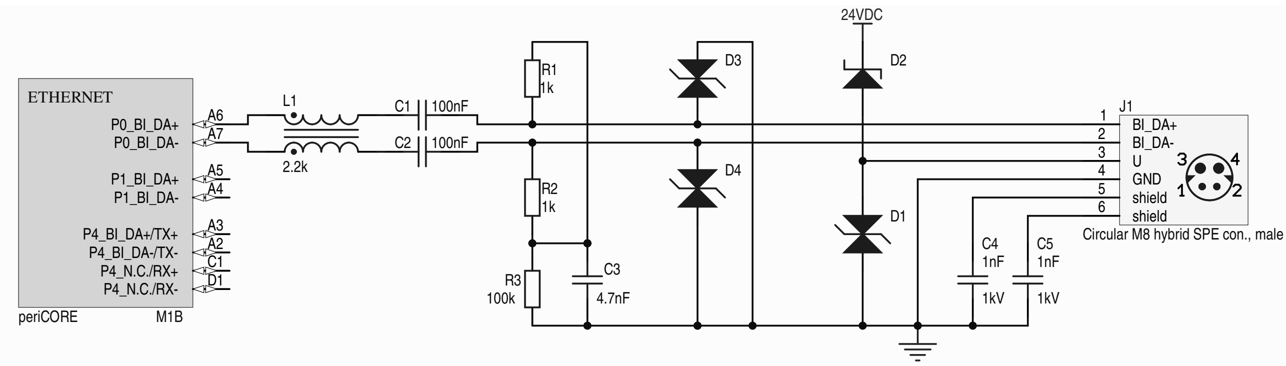


Figure 5: 100BASE-T1 typical application circuit

The circuit consists of:

- C1, C2 - DC blocking capacitors with minimum voltage rating of 50V and 10 % tolerance
- L1 - common mode choke. Recommended parts are: Murata DLW31SH222SQ2#, Bourns SRF3216-222Y and Wurth Elektronik 744232222
- R1, R2 - common mode RF currents termination with 1% tolerance and minimal power rating of 0.4 W
- R3 - discharge resistor. Power rating > 0.1 W
- C3 - HF bypass capacitor with voltage rating >= 100 V and at least 10% tolerance
- C4, C5 - Shield isolation capacitors with 1kV voltage rating
- D1 - ESD protection for power supply lines
- D2 - Reverse polarity protection
- D3, D4 - ESD protection for the data lines with capacitance < 3.5 pF
- J1 - M8 Hybrid connector

Placement and layout guidelines:

- D1, D3 and D4 should be placed as close as possible to the connector.
- MDI differential pairs should be routed as coplanar waveguides with characteristic differential impedance of 100 $\Omega \pm 10$ %. The traces should be kept symmetrical. The distance between the traces should be close to their width.
- Maximize the spacing between the MDI pair and other traces or power planes. A good practice is at least 5H edge-to-edge, where H is dielectric height.

2.1.2 100BASE-T1/TX Port 4

By default, the Port 4 is configured to 100BASE-TX and includes an IEEE 802.3-, IEEE 802.3u- and IEEE 802.3x-compliant (see [12] and [13]) media access controller (MAC). The auto-negotiation mechanism (enabled by default) is defined in the IEEE 802.3u and IEEE 802.3ab specifications (see [11]).

When in 100BASE-TX mode, the MAC automatically selects the appropriate speed (CSMA/CD or full-duplex) based on the PHY auto-negotiation result.

2.1.3 Serial LED Interface

The periCORE module provides link information about the status of the link on Port 0, 1 and 4 via its Serial LED Interface. This interface uses two wires (pins: LED_CLK and LED_DATA). A serial stream of bits is put on pin LED_DATA out, synchronously with clock signal present on pin LED_CLK. For information about the timing see Section 5.2. The stream consists of 24 bits. Bits are active-low and for every port there are 3 bits indicating:

- 100MBit/ACT - this bit is active (low) if 100Mbit link is established. It is toggled whenever an activity on the link occurs.

- ACT - this bit indicates the link activity, and it is toggled whenever an activity occurs.
- LNK - this bit indicates the presence of the link. If the link is established, this bit is low.

The structure of the stream is given in the table below.

Bit	Port	Description
0 - 4	—	Reserved
5	Port 4	100MBit/ACT
6	Port 4	ACT
7	Port 4	LNK
8 - 16	—	Reserved
17	Port 1	100MBit/ACT
18	Port 1	ACT
19	Port 1	LNK
20	—	Reserved
21	Port 0	100MBit/ACT
22	Port 0	ACT
23	Port 0	LNK

Table 2: Link status bit stream structure

In order to utilize the data, an external circuit with shift registers is needed. The circuit is shown below in Figure 6.

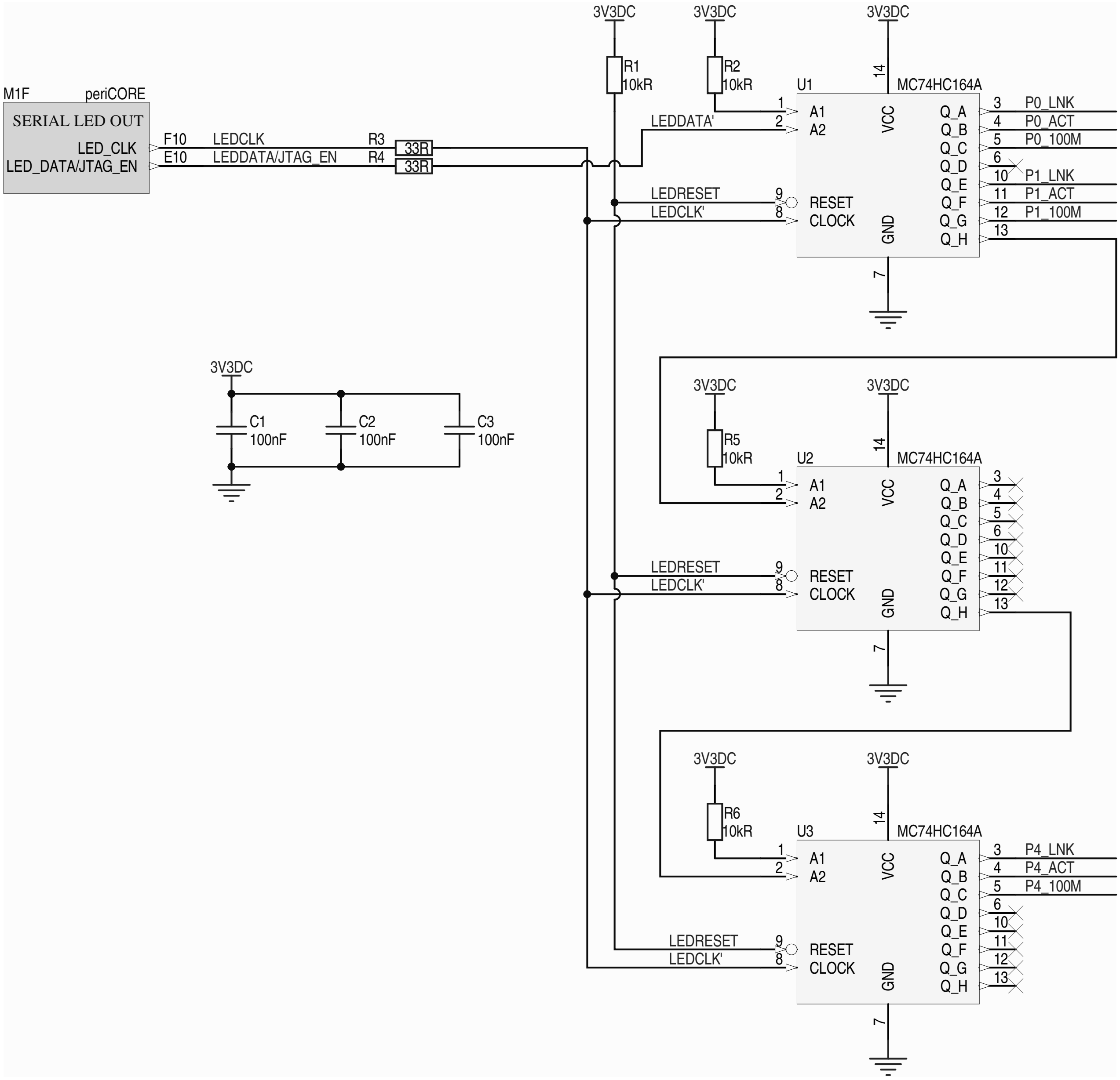


Figure 6: Serial LED interface typical application circuit

The circuit consists of 3 8-bit shift registers. The shift registers are connected in series so that they can accept the whole stream of 24 bits. Based on the application requirements, the link status bits can be used to implement various LED indicators and blinking patterns. The schematic in Figure 7 shows an example of a green (D1) and a yellow (D2) LEDs with the following behaviour:

- D1 is on if the link is established, otherwise it is off. D2 is off.
- If there is any activity on the network, both diodes are blinking out of phase - when D1 is on, D2 is off and vice versa.

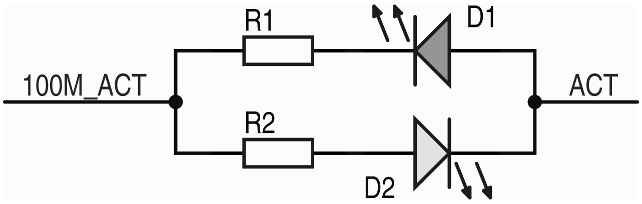


Figure 7: Link Status LED implementation example

2.2 Peripheral Interfaces

2.2.1 I2C

The Inter-Integrated Circuit (I^2C) module is a serial bus interface which consists of two wires: SDA (Serial Data Line) and SCL (Serial Clock Line). I^2C is useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, A/D & D/A Converters, various sensors, etc. For further information refer to the I^2C-bus specification [16]. periCORE supports I2C interface with the following features:

- Fast-mode with 400 kbit/s fixed bit rate
- Master mode in single master bus architecture
- 7-Bit device addresses

An example implementation of 0-10 V sensor interface using an M12 4-pin connector, periCORE and an external ADC, is shown in below. The analog signal, available on pin 4 of the M12 connector is first attenuated and filtered, through R_1, R_2 and C_1. The ADC converts attenuated analog signal into digital samples and the periCORE module reads digital samples from the ADC via I^2C and makes them available over the network. The power to the sensor is supplied via pins 1 and 3 of the M12 connector. D_1 and D_2 protects the circuitry from ESD, and they should always be placed as close as possible to the connector pins.

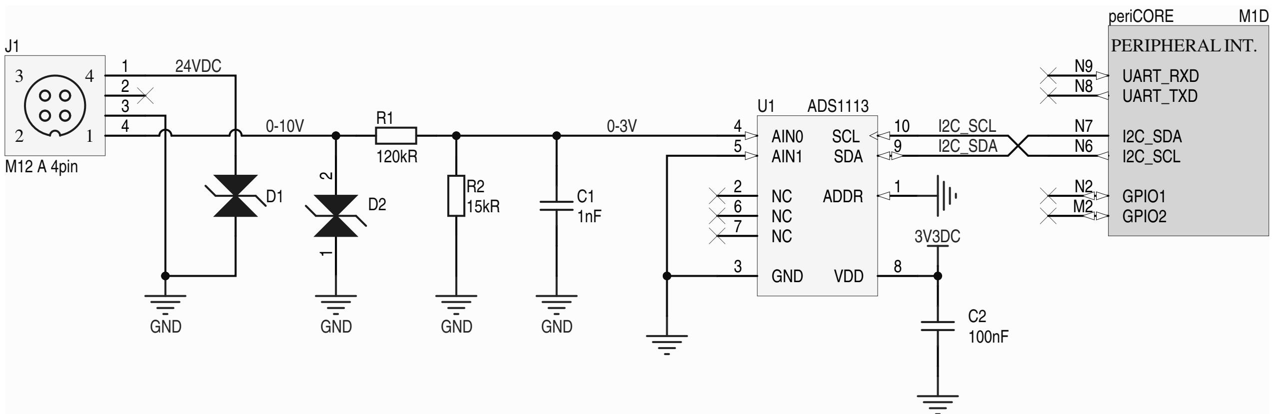


Figure 8: 0-10 V implementation example with I^2C

2.2.2 GPIO

The periCORE supports up to two general-purpose I/O (GPIO) pins.

GPIO logic can be configured as either a digital output or a digital input. Bidirectional functionality can be accomplished by toggling between the input and output configuration in software.

2.2.3 UART

UART (universal asynchronous receiver-transmitter) is hardware peripheral which is used for asynchronous serial communication, where data bits are sent one by one over a single wire. The communication starts with a start bit, followed by the data bits (LSB first), optional parity bit and at least one stop bit. A typical communication frame is shown below:

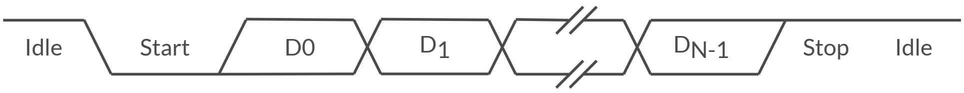


Figure 9: UART frame structure

periCORE module supports single UART. UART contains two I/O pins: TXD and RXD. The data is transmitted on TXD pin, LSB first. RXD is an input where the data is received. periCORE doesn't support hardware flow control, and it needs to be handled in software using XON/XOFF[1].

The default UART configuration on periCORE is:

- Baud rate: 115200
- 1 start bit
- 1 stop bit
- Parity none

2.3 Power Block

2.3.1 Power Input

The *periCORE* communcation module module features a step-down DC-DC converter for alimentation of all basic functions on the module. More information about the input voltage range, current consumption as well as additional electrical parameters could be found in Section 4.3.

2.3.2 Power Output

The periCORE module provides the possibility to take advantage of the internal DC-DC converter as well, to supply external peripheral circuitry blocks with 3.3VDC. 3.3VDC is available at the `3V3_OUT` pin. There is an internal over-current and short circuit protection, although in case of a short-circuit, 3.3V drops and the periCORE is not functional until the cause of the short-circuit is removed and 3.3V is present again. More information about the maximum allowable output current as well as other electrical parameters of the output 3.3V could be found in Section 4.3.

[1] Software flow control is application specific and not part of the *libperiCORE* deliverable.

2.4 Controller Block

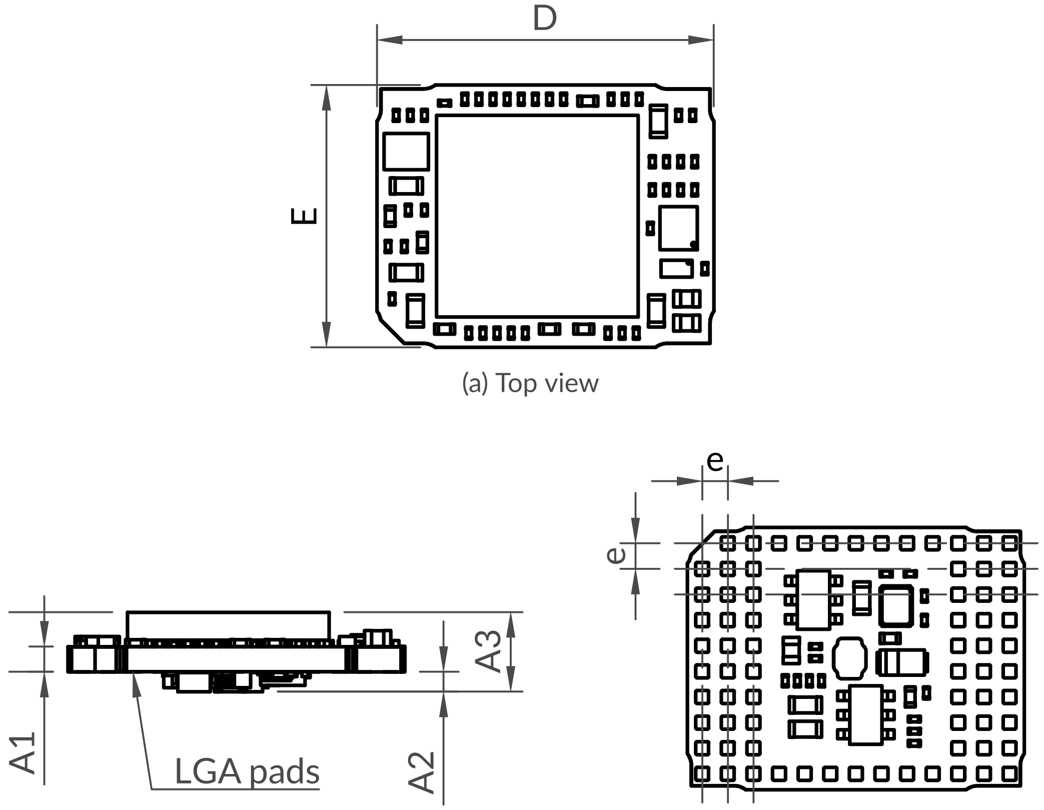
periCORE firmware is executed inside the ARMv7 MCU core. A 32-MBit Flash memory is used as a permanent storage for the periCORE firmware and additional auxiliary data. The MCU clock frequency is 250 MHz and the available RAM is 512 kb.

2.5 Watchdog Block

periCORE module is equipped with a watdchdog timer in order to ensure the recovery of the MCU from a fault situation. If the MCU doesn't serve the watchdog timer within 5 seconds, the watchdog will perform a reset of the MCU. The 5s detection window is not possible to change.

3 Mechanical Specification

3.1 Module Dimensions



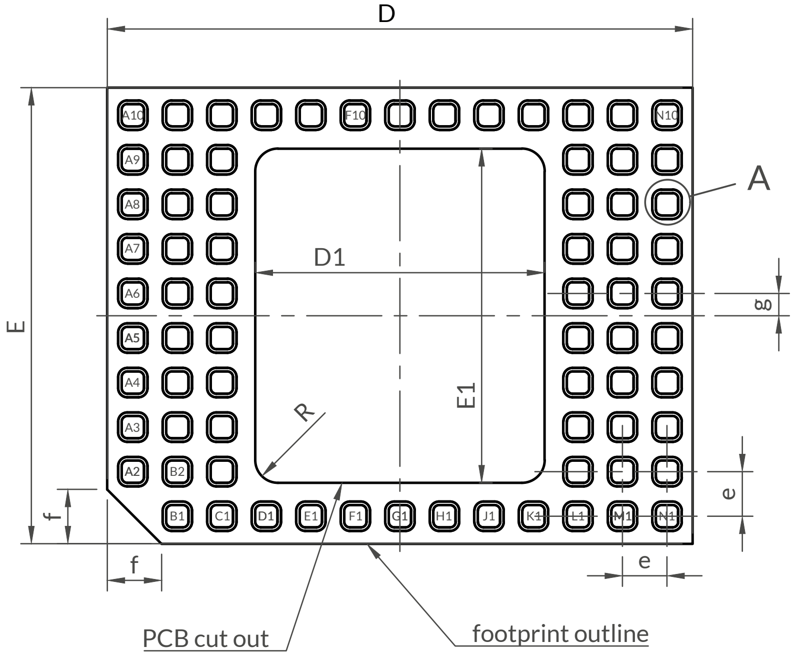
(a) Top view
(b) Side view
(c) Bottom view

Figure 10: Mechanical dimensions

Symbol	Min.	Typ.	Max.
A1	1.1	1.2	1.3
A2	0.9	1.0	1.1
A3	3.8	3.9	4.0
E	12.8	13.0	13.2
D	16.5	16.7	16.9
e		1.27 BSC	

Table 3: periCORE dimensions (in mm)

3.2 Recommended Footprint



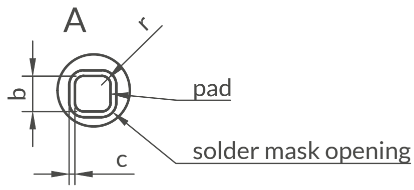


Figure 11: Recommended footprint (top view) for the periCORE module.

Symbol	Min.	Nom.	Max.
D	—	16.7	—
D1	8.255	—	8.355
E	—	13.0	—
E1	9.525	—	9.625
R	0.635	—	0.735
f	—	1.55	—

(a) Framing

Symbol	Min.	Nom.	Max.
r	0.125	0.16	0.195
b	0.6	0.635	0.67
c	0.065	0.1	0.135
e		1.27 BSC	
g	0.6	0.635	0.67

(b) Pads

Table 4: periCORE footprint dimensions in mm.

4 Electrical Specification

4.1 Signal Types

Type Name	Description
A	Analog pin type
D	Digital pin type
GND	Ground
I/O	Bidirectional
I	Input directional pin
O	Output directional pin
PD	With internal pull-down
PU	With internal pull-up
PWR	Power supply pin
PWRO	Power supply output pin
CFG	Configuration pin

Table 5: Signal Type Definitions

4.2 Pad Configuration and Functions

The configuration and functions of the pads of the periCORE are described in Table 6.

Pad	Signal	Type	Description
A2	P4_BI_DA-/TX-	A, I/O	Port 4, 100BASE-T1/100BASE-TX(TX) PHY Differential pair negative terminal
A3	P4_BI_DA+/TX+	A, I/O	Port 4, 100BASE-T1/100BASE-TX(TX) PHY Differential pair positive terminal
A4	P1_BI_DA-	A, I/O	Port 1, 100BASE-T1 PHY Differential pair negative terminal
A5	P1_BI_DA+	A, I/O	Port 1, 100BASE-T1 PHY Differential pair positive terminal
A6	P0_BI_DA+	A, I/O	Port 0, 100BASE-T1 PHY Differential pair positive terminal
A7	P0_BI_DA-	A, I/O	Port 0, 100BASE-T1 PHY Differential pair negative terminal
A8	24VDC_IN	PWR	24V DC input power
A9	Reserved		Do not connect
A10	GND	GND	electrical Ground
B1	GND	GND	electrical Ground
B2	Reserved		Connect to GND

Pad	Signal	Type	Description
B3	Reserved		Connect to GND
B4	Reserved		Connect to GND
B5	GND	GND	electrical Ground
B6	GND	GND	electrical Ground
B7	Reserved		Do not connect
B8	Reserved		Do not connect
B9	Reserved		Do not connect
B10	GND	GND	electrical Ground
C1	P4_N.C./RX+	A, I	Port 4, 100BASE-TX Receive (RX) Differential pair positive terminal; floating if port 4 is configured for 100BASE-T1
C2	GND	GND	electrical Ground
C3	Reserved		Do not connect
C4	JTAG_TDI	D, I, PD	JTAG Test Data In (TDI)
C5	JTAG_TCK	D, I, PD	JTAG Test Clock (TCK)
C6	JTAG_TRST_B	D, I, PD	JTAG Test Reset (TRST); Active low
C7	JTAG_TDO	D, O, PD	JTAG Test Data Out (TDO)
C8	JTAG_TMS	D, I, PD	JTAG Test Mode Select (TMS)
C9	Reserved		Do not connect
C10	Reserved		Do not connect
D1	P4_N.C./RX-	A, I	Port 4, 100BASE-TX Receive (RX) Differential pair negative terminal; floating if port 4 is configured for 100BASE-T1
D10	Reserved		Do not connect
E1	GND	GND	electrical Ground
E10	LED_DATA/JTAG_EN	D, O, PD, CFG	serial LED data output; JTAG enable configuration pin (sampled on power on reset (POR))
F1	Reserved		Do not connect
F10	LED_CLK	D, O, PD	serial LED clock output
G1	Reserved		Do not connect
G10	GND	GND	electrical Ground
H1	GND	GND	electrical Ground
H10	Reserved		Do not connect
J1	Reserved		Do not connect
J10	Reserved		Do not connect
K1	Reserved		Do not connect
K10	Reserved		Do not connect
L1	GND	GND	electrical Ground

Pad	Signal	Type	Description
L2	Reserved		Do not connect
L3	Reserved		Do not connect
L4	Reserved		Do not connect
L5	Reserved		Do not connect
L6	Reserved		Do not connect
L7	Reserved		Do not connect
L8	Reserved		Do not connect
L9	Reserved		Do not connect
L10	Reserved		Do not connect
M1	GND	GND	electrical Ground
M2	GPIO2	D, I/O, PD	General Purpose I/O;internal pull-down
M3	Reserved		Do not connect
M4	Reserved		Do not connect
M5	GND	GND	electrical Ground
M6	Reserved		Do not connect
M7	Reserved		Do not connect
M8	Reserved		Do not connect
M9	Reserved		Do not connect
M10	GND	GND	electrical Ground
N1	GND	GND	electrical Ground
N2	GPIO1	D, I/O, PD	General Purpose I/O
N3	DEBUG_EN	D, I/O, PD	Debug enable signal pin
N4	nRESET	D, I, PU	Active low; Reset signal to put periCORE into Reset state (see Section 5.1)
N5	3V3_OUT	PWRO	3.3V output power
N6	I2C_SCL	D, I/O, PU	I2C Serial Clock Line (SCL)
N7	I2C_SDA	D, I/O, PU	I2C Serial Data Line (SDA)
N8	UART_TX	D, I/O, PD	UART Transmit Data
N9	UART_RX	D, I/O, PU	UART Receive Data
N10	GND	GND	electrical Ground

Table 6: periCORE Pad mapping.

4.3 Absolute Maximum Ratings

Parameter	Min	Max	Unit
Supply voltage (`24VDC_IN`)	0	33	V
Output current (`3V3_OUT`)	0	(t.b.c.) 100	mA
Storage temperature	-40	+85	°C
`D, I/O`	-0.5	+3.63	V

Table 7: Absolute maximum ratings

Warning: Exceeding the specified absolute maximum ratings may damage the periCORE device.

The periCORE has limited built-in ESD protection. The device should be placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.

4.4 Operating Conditions

Parameter	Min	Typ.	Max	Unit
Supply voltage (`24VDC_IN`)	(t.b.c.) 10	24	30	V
Current consumption (`24VDC_IN` = 24V, I_{3V3_OUT} = 0)	—	(t.b.c.) 27	—	mA
Output current (`3V3_OUT`)	—	—	100	mA
Ambient Temperature	-40	—	+85	°C

Table 8: Recommended operating conditions

4.5 Electrical Characteristics

Symbol	Parameter	Conditions	Min	Typ.	Max	Unit
24VDC_IN						
V_{UVDT}	Under voltage detection threshold		2.3	—	—	V
3V3_OUT						
ΔV_{p-p}	Voltage ripple		—	40	—	mV
I_{3V3_OUT}	Output current		—	—	100	mA
C_L	Capacitive load on `3V3_OUT`		—	—	400	µF
GPIO						
V_{OH}	High-Level output voltage	IOH = - 8mA	2.4	3.0	—	V

Symbol	Parameter	Conditions	Min	Typ.	Max	Unit
V_{OL}	Low-Level output voltage	IOL = 8mA	–	0.3	0.5	V
V_{IH}	High-Level input voltage		2.0	–	–	V
V_{IL}	Low-Level input voltage		–	–	0.9	V
I_{IH}	High-Level input current		–	70	–	µA
I_{IL}	Low-Level input current		–	2.5	–	µA
I_o	Output current digital pads		–	–	5 (t.b.c.)	mA
R_{PD}	Pull-down resistance		40	56	106	kΩ
I^{2}C						
V_{OH}	High-Level output voltage	IOH = - 8mA	2.4	3.0	–	V
V_{OL}	Low-Level output voltage	IOL = 8mA	–	0.3	0.5	V
V_{IH}	High-Level input voltage		2.0	–	–	V
V_{IL}	Low-Level input voltage		–	–	0.9	V
I_{IH}	High-Level input current		–	70	–	µA
I_{IL}	Low-Level input current		–	0.7	–	mA
I_o	Output current digital pads		–	–	5 (t.b.c.)	mA
R_{PD}	Pull-up resistance		–	4.7	–	kΩ
C_b	Capacitive load on each `I2C_SDA` and `I2C_SCL` line		–	–	75	pF
UART						
V_{OH}	High-Level output voltage	`UART_TX`; IOH = - 8mA	2.4	3.0	–	V
V_{OL}	Low-Level output voltage	`UART_TX`; IOL = 8mA	–	0.3	0.5	V
V_{IH}	High-Level input voltage	`UART_RX`	2.0	–	–	V
V_{IL}	Low-Level input voltage	`UART_RX`	–	–	0.9	V
I_{IH}	High-Level input current	`UART_RX`	–	2.5	–	µA
I_{IL}	Low-Level input current	`UART_RX`	–	70	–	µA
I_o	Output current digital pads		–	–	5 (t.b.c.)	mA
R_{PD}	Pull-up resistance	`UART_RX`	24	45	113	kΩ
R_{PD}	Pull-down resistance	`UART_TX`	40	56	106	kΩ
Serial LED Interface						
V_{OH}	High-Level output voltage	IOH = - 8mA	2.4	3.0	–	V
V_{OL}	Low-Level output voltage	IOL = 8mA	–	0.3	0.5	V
R_{PD}	Pull-down resistance		40	56	106	kΩ
JTAG Interface						
V_{OH}	High-Level output voltage	IOH = - 8mA	2.4	3.0	–	V
V_{OL}	Low-Level output voltage	IOL = 8mA	–	0.3	0.5	V

Symbol	Parameter	Conditions	Min	Typ.	Max	Unit
V_{IH}	High-Level input voltage		2.0	—	—	V
V_{IL}	Low-Level input voltage		—	—	0.9	V
I_{IH}	High-Level input current	JTAG_TCK; JTAG_TDI; JTAG_TMS; JTAG_JTCE;	—	70	—	µA
I_{IH}	High-Level input current	JTAG_TRST_B	—	0.7	—	mA
I_{IL}	Low-Level input current	All except JTAG_TDO	—	2.5	—	µA
I_o	Output current digital pads				5 (t.b.c.)	mA
R_{PD}	Pull-down resistance	All except JTAG_TRST_B	40	56	106	kΩ
R_{PD}	Pull-down resistance	JTAG_TRST_B	—	4.7	—	kΩ
DEBUG_EN, nRESET						
V_{OH}	High-Level output voltage	IOH = - 15mA	2.85	—	—	V
V_{OL}	Low-Level output voltage	IOL = 15mA	—	—	1.3	V
V_{IH}	High-Level input voltage		2.31	—	—	V
V_{IL}	Low-Level input voltage		—	—	0.99	V
I_{IH}	Input leakage current		-70	—	70	nA
R_{PU}	Pull-up resistance	nRESET; V_{IN} = 0 V	25	40	55	kΩ
R_{PD}	Pull-down resistance	DEBUG_EN; V_{IN} = 3.3 V	25	50	55	kΩ
C_{IO}	Input capacitance		—	5	—	pF
Port 0,1 and 4 MDI signals						
R_{in}	Integrated Termination on Differential MDI Pairs		—	100	—	Ω

Table 9: Electrical characteristics

5 Timing Specification

5.1 Power On/Off and Reset Sequence

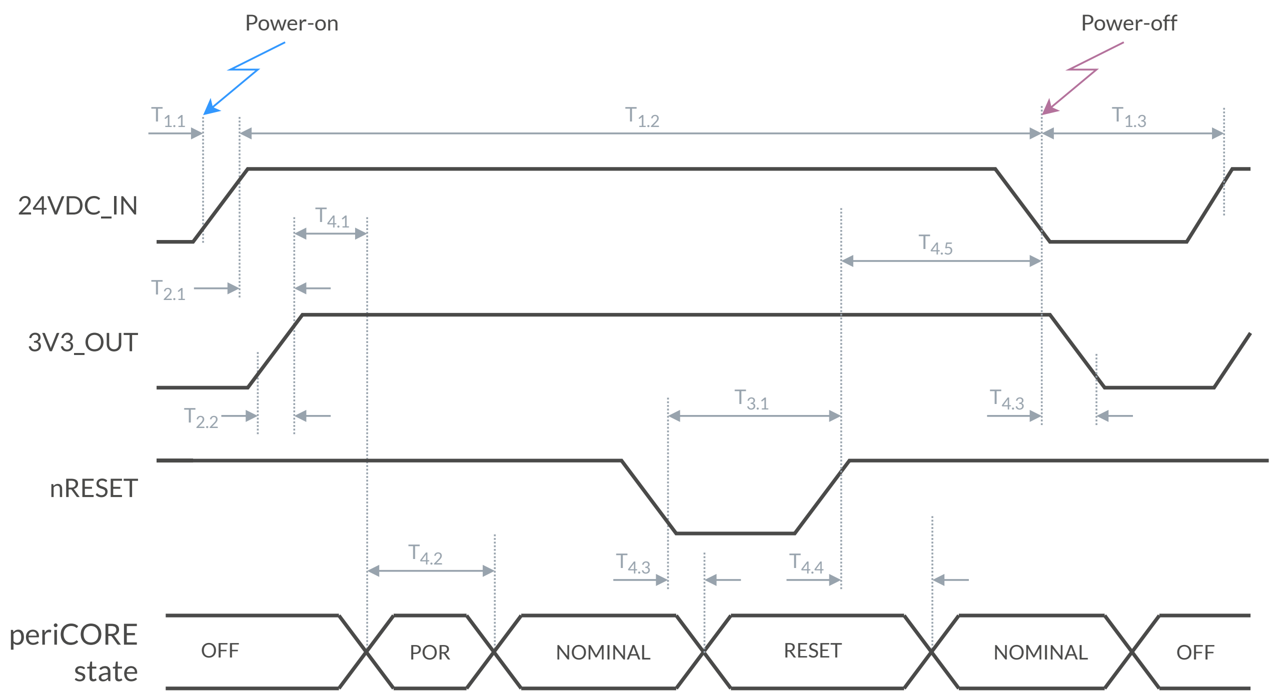


Figure 12: Power On/Off and Reset timing diagram.

Symbol	Parameter	Min	Typ.	Max	Unit
$T_{1.1}$	24VDC_IN rise time [2]	–	–	6	ms
$T_{1.2}$	Period after Power-on in which power supply (24VDC_IN) must not be interrupted [3].	310	–	–	ms
$T_{1.3}$	Time between Power-off and the next Power-on [4]	6	–	–	ms
$T_{2.1}$	From Power-on to stable 3V3_OUT (I_{3V3_OUT} = 0)	–	–	1.3	ms
$T_{2.2}$	3V3_OUT rise time[5]	–	–	2.5	ms
$T_{3.1}$	Duration of nRESET pulse	8	–	–	µs
$T_{4.1}$	Delay from Power-on to the POR	–	–	400	µs
$T_{4.2}$	Duration of POR	–	310	–	ms
$T_{4.3}$	Delay from nRESET activation to entering RESET state	–	–	8	µs

[2] This condition must not be violated
[3] By interrupting is assumed any excursion of the power supply voltage below V_{UVDT}
[4] This effectively means that 24VDC_IN must stay below V_{UVDT} in this time period.
[5] The external capacitive load on the pin 3V3_OUT must be taken into consideration in order not to violate this requirement.

Symbol	Parameter	Min	Typ.	Max	Unit
$T_{4.4}$	Delay from nRESET deactivation to entering NOMINAL mode	–	310	–	ms
$T_{4.5}$	Period after nRESET deactivation in which power supply must not be interrupted[3]	310	–	–	ms

Table 10: Power On/Off and Reset timing

5.2 Serial LED Interface

Timing specification regarding the serial LED interface is shown below.

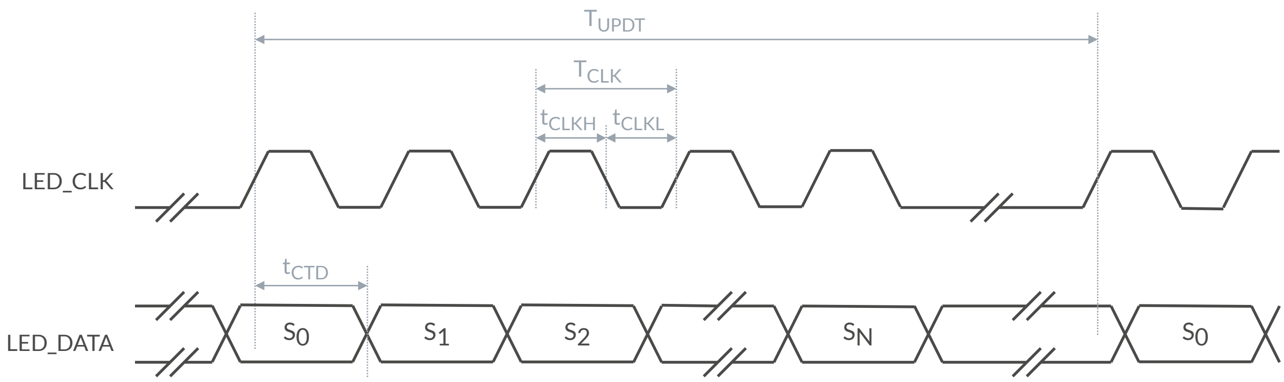


Figure 13: Serial LED Interface Timing Diagram

Parameter	Description	Min	Typ.	Max	Unit
T_{UPDT}	LED update cycle period	–	42	–	ms
T_{CLK}	LED_CLK period	–	320	–	ns
t_{CLKH}	LED_CLK high-pulse width	140	–	180	ns
t_{CLKL}	LED_CLK low-pulse width	140	–	180	ns
t_{CTD}	LED_CLK to LED_DATA output time	–	–	180	ns

Table 11: Serial LED Interface Timing

5.3 JTAG Interface

Timing specification of the JTAG Interface.

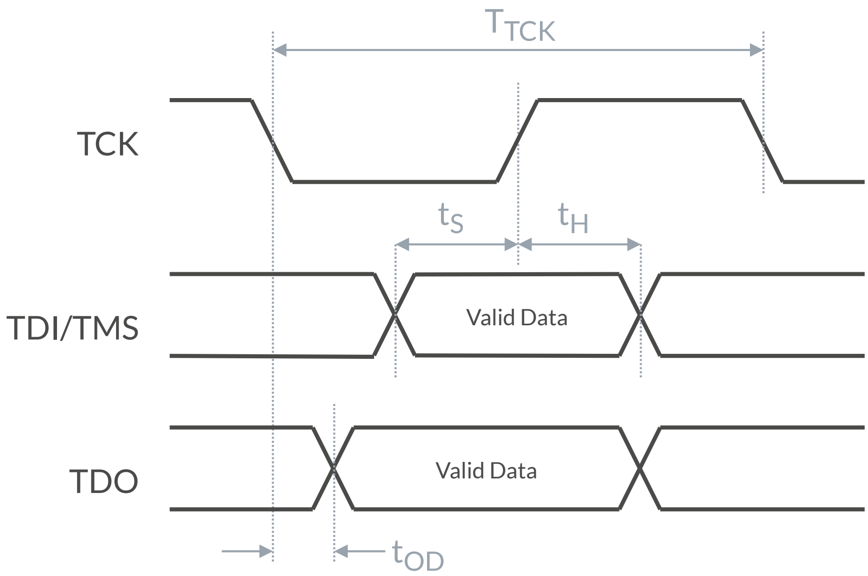


Figure 14: JTAG Interface Timing Diagram

Symbol	Description	Min	Typ.	Max	Unit
T_{TCK}	TCK period	40	–	–	ns
t_S	Input setup time	10	–	–	ns
t_H	Input hold time	10	–	–	ns
t_{OD}	TDO delay from TCK falling edge	–	–	14	ns

Table 12: JTAG Interface Timing

6 Factory Reset

A periCORE based product implements two methods to perform the factory reset. One method is invoked via the RESTful API (see Section 7.3.6) and the second method is performed with physical access to the device. The latter becomes necessary, when the access via the RESTful API has been lost, e.g. when the admin mTLS certificate has been lost.

The founding *libperiCORE* implements the functionality of the factory reset behaviour. To invoke the physical Factory Reset a periCORE based product needs to be powered (via `24VDC_IN`) but must not be able to establish any network link on any of the 3 network ports (Port 0, Port 1 as well as Port 4).

20 seconds after power up, the Factory Reset is executed, and the device is restored to factory defaults (see Section 7.3.6 for further details).

7 Firmware

The intent use of the periCORE module is within an end product of the OEM customer. As a white label product, the periCORE firmware architecture supports enhancements by the OEM customer. Therefore, Perinet provides the infrastructure for the software development, in form of the *periCORE Development Kit* [5] as well as an example application [6], which shows the rebranding towards a periCORE based product with dedicated adaptions to fulfil the customer needs in terms of functionality as well as appearance (see Section 7.8).

The source code of the firmware itself is written in standard C++20 [14] on top of a bare metal software environment. A minimal and simplified operating system (OS) [8] and the *libperiCORE* provide the basic functionality of the firmware. Both are Perinet deliverables and allow focusing on the application development during the implementation of the *periCORE based Custom Application*.

This section addresses the basic architecture, functional behaviour, production handling as well as rebranding possibilities of a periCORE module.

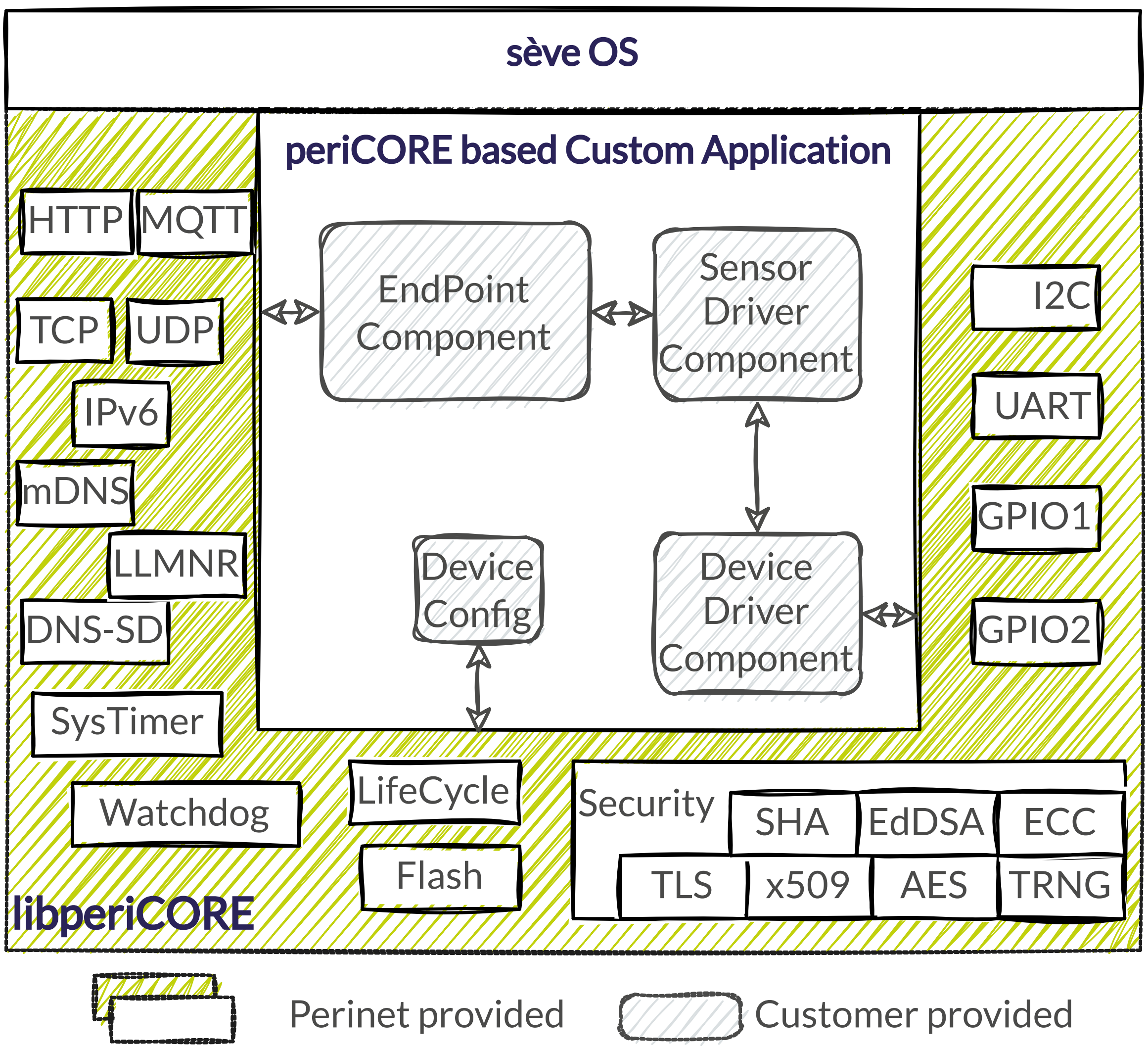


Figure 15: General periCORE based firmware architecture.

7.1 Firmware Architecture

The firmware for a periCORE based product is structured in the *sève OS*, the *libperiCORE* and the *periCORE based Custom Application* as shown in Figure 15.

periCORE based Custom Application The application level of a periCORE based firmware contains software to control and communicate to platform specific hardware. It implements also a value transformation from a implementation specific sensor measurement to a semantic metric, e.g the transformation of 0-10 V to a distance, as described in Perinets example application (periCORE Firmware Development Application Note [6].

An *periCORE based Custom Application* is implemented in standard C++, uses the sève OS as well as Perinets *libperiCORE*.

libperiCORE A substantial feature set (i.a. security, product life cycle function, network protocols, hardware dependent bus driver etc.) is provided as a software library. The library defines a standard C++20 conform interface and is supposed to be used in all periCORE based firmware variants. The details are described in Section 7.7.

sève OS An event-flow based operating system is used as foundation for the *periCORE based Custom Application* and for the *libperiCORE*. Its component based architecture provides a possibility for a well-structured application design. Components communicate exclusive via event channels among each other, which are implicitly synchronized against race conditions.

Event channels are the implementation of a dataflow model. They transport the information of the occurrence as well as the data of an event at the same time.

An event triggers the execution of an activity, which is implemented within components. Activities are executed cooperative after a FIFO scheme and are executed according to that scheme with a run to completion semantic.

The sève OS is Perinet technology. More details are provided with the periCORE sève Operating System Datasheet [9].

7.2 Persistent Memory Structure

The persistent memory of periCORE based product is divided into different logical segments. The segments are used to implement the current state of the product life cycle and are summarized in Figure 16.

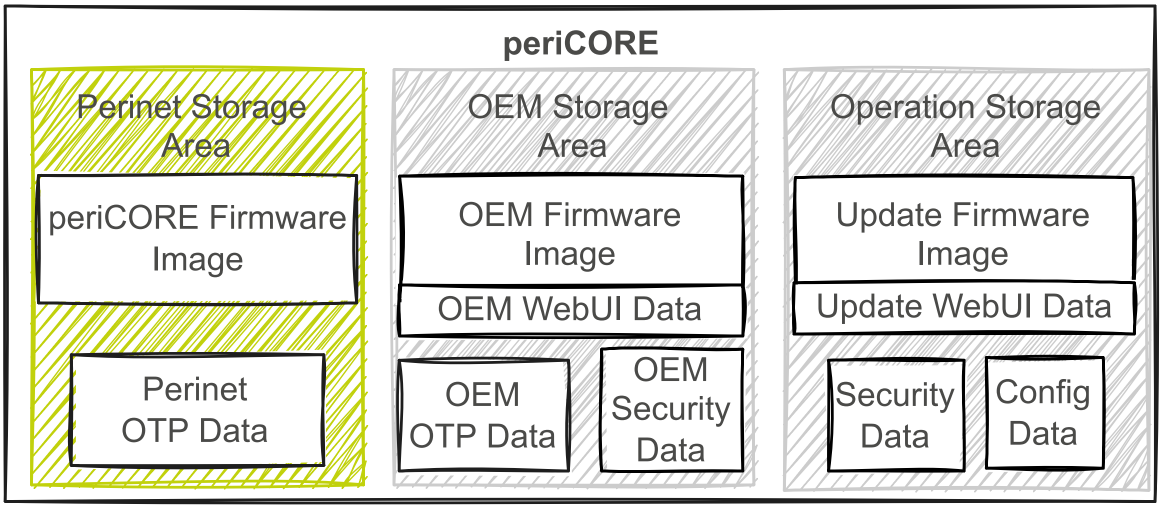


Figure 16: Structure of the persistent memory of a periCORE based product.

The periCORE module defines three logical storage areas (*Perinet Storage Area*, *OEM Storage Area* and *Operation Storage Area*) for different life states of a periCORE based product, *periCORE Production state*, *OEM Production state* and *Updated State*.

Perinet Storage Area During the manufacturing of the periCORE module, the production specific metadata (see Section 9.2) is stored in the OTP of the *Perinet Storage Area*. In addition, the general (*periCORE Firmware Image*) is stored in that region as well.

This memory region is not changed during the lifetime of a periCORE module.

OEM Storage Area The memory segment is used to store the factory default data at manufacturing time of the OEM customer. During a factory reset (see Figure 17) the in this segment stored data is restored.

This memory segment is also used to store the device attestation certificate (see Section 7.5).

This memory region cannot be changed during normal operations of the periCORE based product.

Operation Storage Area Persistent changes of a periCORE based product are stored in this memory segment. It is used to store the *Update Firmware Image*, runtime security data as well as configuration data.

Data stored in this memory segment is persistent for power cycles of the periCORE based product and is removed during a *Factory Reset* (see Listing 3).

7.3 Product Life Cycle

The Figure 17 summarizes the different states of a periCORE based product as well as the different state transitions.

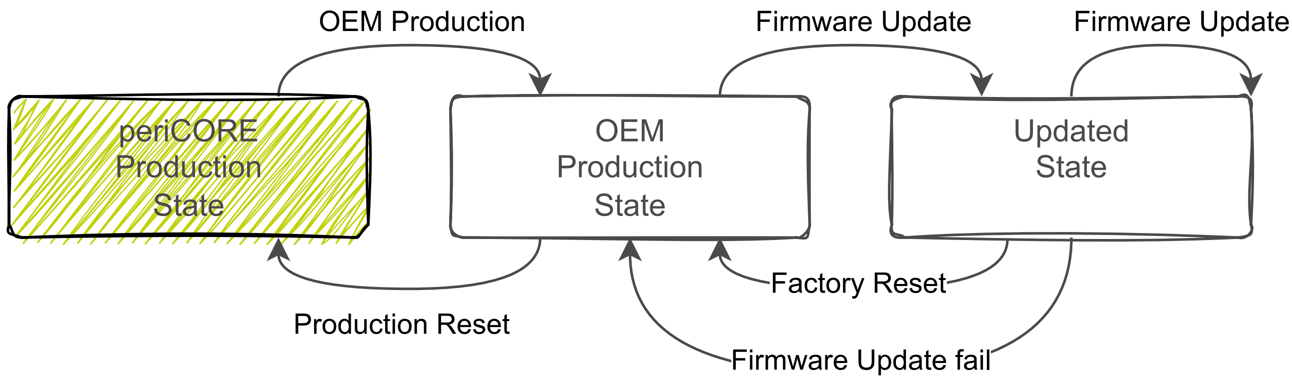


Figure 17: The Different life states of a periCORE based product.

7.3.1 periCORE Production State

A periCORE module leaves the Perinet manufacturing facility in the *periCORE Production State*. The following information are stored inside the *Perinet Storage Area*:

1. The *periCORE Firmware Image*
2. Unique production specific attributes (see Section 9.2), i.a. MAC-Address, serial number and production date
3. Unique attestation certificate information, to identify a valid periCORE module, that has been produced by an accredited manufacturing service.

Note: The periCORE module is delivered with a general purpose firmware image, the *periCORE Firmware Image*. Its intent use is for production use cases only. Perinet does not recommend using the *periCORE Firmware Image* within end customer applications.

The *periCORE Firmware Image*, is suited for production usage only of the OEM manufacturing facility. The following summarized features are provided with the periCORE Firmware Image:

- Attestation of an authentic periCORE module, which proofs to be produced by Perinet approved manufacturing services (see Section 7.5).
- Deploying of Security Information (X.509 based certificates).
- Deploying of the *OEM firmware image*.
- Access to all peripherals via network services.
- Various networking features, IPv6, DNS-SD, MDNS, HTTP, TLS, MQTT, etc.

Note: The periCORE module attestation information are deleted during the *OEM Production* procedure.

7.3.2 OEM Production State

The *OEM Production State* is the delivery state of a periCORE based product. The following attributes are stored during the production step *OEM Production*:

1. Product specific authentication information (see Section 7.5).
2. *OEM Firmware Image*.
3. Product specific device information (see Section 9.2), i.a. product name, host name as well as product serial number.
4. Product specific configuration values, e.g. sensor calibration data.

Note: No access control (RBAC) has been activated in that state by default. All configuration as well as security settings are accessible and writable without any client authentication. The RBAC has to be configured and activated additionally.

7.3.3 Updated State

In the *Update State* a periCORE based product is operated by the *Update Firmware Image*. Once a *Firmware Update* has been performed, a periCORE based product will always boot the *Update Firmware Image*.

To restore the OEM Production State, an explicit Factory Reset (see Section 7.3.6) can be performed.

Note: A failure during a Firmware Update procedure, e.g. caused by network interruption or a corrupted Update Firmware Image, leads to a changed life state of a periCORE based product. The resulting state in that case is the *OEM Production State*.

7.3.4 Firmware Update

A Firmware Update procedure can only be performed from either the *OEM Production State* or the *Updated State*. The resulting state of the periCORE based device will be the *Updated State*. The procedure is triggered via the RESTful API (see Section 7.4 for further details) and its behaviour is idempotent.

```
curl --data-binary '@<filename>' \
-H "Content-Type: application/octet-stream" \
-X PUT https://periNODE-sernm.local/update
```

Listing 1: An example Firmware Update deployment with the `curl` command line tool.

The RESTful API call will exchange the *Update Firmware Image* inside the *Operation Storage Area* (see Figure 16). This has no effect on the currently executed firmware image, even when the

Firmware Update is performed from the life state *Updated State*. In order to execute the newly stored *Update Firmware Image*, an additional reboot is necessary, which can be performed via a power cycle or via the RESTful API (see Section 7.4).

```
curl -X PATCH https://periNODE-sernm.local/reboot
```

Listing 2: An example for triggering the reboot command with the `curl` command line tool.

No changes to the persistent stored data of the *Security Data* nor the *Config Data* segments (see Figure 16) are made during the Firmware update procedure. A reset of one or both sections, if necessary, needs to be performed manually via the RESTful API (see Section 7.4).

```
curl -X PATCH https://periNODE-sernm.local/config/reset
```

Listing 3: An example for configuration reset performed with the `curl` command line tool.

Note: A failure during a Firmware Update procedure, e.g. caused by network interruption or a corrupted Update Firmware Image, leads to a changed life state of a periCORE based product. The resulting state is the *OEM Production State*.

7.3.5 OEM Production

In order to reach the state *OEM Production State*, the following steps need to be performed:

1. Authenticate the periCORE module (see Section 7.5).
2. Deploy product specific *OEM Firmware Image*.
3. Deploy device specific *OEM Security Data*.
4. Deploy product specific *OEM OTP Data*.
5. Reboot the device.

Note: The above-mentioned steps of the OEM Production can only be performed from the *periCORE Production State*.

periCORE Authentication

The device attestation is performed implicitly with each RESTful API (see Section 7.4) request. An explicit device authentication is described in (see Section 7.5).

Note: The product specific authentication information are deleted when during the *OEM Production* procedure.

Deploy *OEM Firmware Image*

The deployment of the *OEM Firmware Image* will be performed via RESTful API (see Section 7.4) call. The *Firmware Image* (see Section 7.4.2) is transported via *PUT* request. In the following excerpt an example is shown how to write the *OEM Firmware Image*:

```
curl --data-binary '@<filename>' \
-H "Content-Type: application/octet-stream" \
-X PUT https://periCORE-sernm.local/production/oem-firmware
```

Listing 4: Deploying the *OEM Firmware Image* with the `curl` command line tool.

Note: With the deployment of the *OEM Firmware Image* the *OEM Security Data* is invalidated. That includes the periCORE authentication information. After this step, a periCORE based product will not be able to provide a unique authentication for itself. It is strongly recommended, to store proper *OEM Security Data* into the device, after performing this step.

Deploy *OEM Security Data*

```
curl -X PATCH --data-binary @perinode-host-cert-bundle.pem \
    https://periNODE-sernm.local/security/perinode-host-cert
```

Listing 5: Deploying the host certificate of the *OEM Security Data* with the `curl` command line tool.

```
curl -X PATCH --data-binary @root-ca-cert.crt \
    https://periNODE-sernm.local/security/perinet-root-ca-cert
```

Listing 6: Deploying the root CA certificate of the *OEM Security Data* with the `curl` command line tool.

Deploy *OEM OTP Data*

```
cat <<EOF | curl -X PATCH --data-binary @-
    ↪ https://periNODE-sernm.local/nodeInfo
{
    "hostname": "periNODE-sernm",
    "manufacturer": "Example Inc.",
    "product_charge": "1",
    "product_name": "periNODE example",
    "product_part_number": "MPN.0123.4567",
    "product_serial": "sernm",
    "product_version": "1"
}
EOF
```

Listing 7: Deploying the *OEM OTP DATA* with the `curl` command line tool.

7.3.6 Factory Reset

A Factory Reset does invalidate the whole *Operational Storage Area*. This sets a periCORE based product into to the *OEM Production State*. The following properties are removed or reset:

***Config Data* reset:** Any persistent runtime configuration data is overwritten with the firmware specific default values. That includes network configuration settings like `mqtt_broker_name` or `application_name`.

***Security Data* reset:** Any configured security information are set back to the default settings, the *OEM Security Data*. The mTLS is disabled and therefore the client authentication (RBAC) (see Section 7.6) is disabled.

***Update Firmware Image* invalidation:** The *Update Firmware Image* from the persistent *Operational Storage Area* is invalidated. After the next boot, the periCORE based product is executing the *OEM Firmware Image*.

A *Factory Reset* can be performed either via the RESTful API (see Section 7.4) as shown in the following excerpt or via the *Physical Factory Reset* (see Section 6).

```
curl -X PATCH https://periNODE-sernm.local/reset
```

Listing 8: Invoke a *Factory Reset* via the RESTful API.

Note: Changes to *Security Data* as well as *Config Data* does only take effect after a reboot of the device. Since all security settings are reset during a Factory Reset, it is strongly recommended, to redo the security configuration.

7.3.7 Production Reset

The *Production Reset* procedure, brings the device into the *periCORE Production State*, which is operated by the general purpose *periCORE Firmware Image*. The access to the procedure is only active for 60sec on the first boot after a performed *Physical Factory Reset* (see Section 6).

Accessing the *periCORE Production State* is neither invalidating the *OEM Firmware Image* nor the OEM Data (*OEM Security Data OEM OTP Data*). With the start of the

The following excerpt shows how to activate entering the *periCORE Firmware Image Production Reset* via the RESTful API (see Section 7.4 for further details).

```
curl -X PATCH https://periNODE-sernm.local/production/reset
```

Listing 9: Invoke a *Production Reset* via the RESTful API.

Note: The *Production Reset* procedure, will permanently invalidate the *OEM Firmware Image* as well as the OEM Data (*OEM Security Data OEM OTP Data*) and will be in effect after a reboot of the periCORE based product.

7.4 RESTful API

A periCORE based product implements access to a RESTful [4] API. The hostname of your periCORE is `periCORE-`*`sernm`*`.local`, where *`sernm`* is generated from the serial number as described in Section 9.2. The following routes are available (see Table 13) and will be described in the following subsections.

URL Resource	Description
/info	See Section 7.4.1
/config	
/config/reset	
/update	See Section 7.4.2
/reboot	
/reset	
/production/oem-firmware	
/production/reset	
/security	See Section 7.4.3
/security/host-cert	
/security/root-cert	
/security/client-cert	
/security/reset	

Table 13: RESTful API routes overview

7.4.1 Info Service

/info periCORE based product information object (see Section 7.4.1).

`GET` expects empty body.

`200` Return `NodeInfo` object.

`204` Return empty body. No data is available.

`401` Unauthorized access, returns empty body

`500` Internal server error on unexpected error, returns empty body.

/config periCORE based product configuration object (see Section 7.4.1).

`GET` expects an empty body.

`200` Return `NodeConfig`.

`204` Return empty body. No data is available.

`401` Unauthorized access, returns empty body

`500` Internal server error on unexpected error, returns empty body.

`PATCH` expects a complete or partial `NodeConfig` object.

`204` The Request has been accepted but was not processed yet. The object will be deserialized and merged. Given *keys* will be overwritten. The object will be stored persistently.

`401` Unauthorized access, returns empty body

`500` Internal server error on unexpected error, returns empty body.

/config/reset periCORE based product configuration object reset (see Section 7.4.1).

`PATCH` expects an empty body.

`204` The Request has been accepted but was not processed yet. The default content of the object will be restored and the object will be stored persistently.

`401` Unauthorized access, returns empty body

`500` Internal server error on unexpected error, returns empty body.

Node Info

```
syntax="proto3";
package perinet.api;

message SwVersion {
  uint32 api = 1; // API compatibility incarnation
  uint32 build = 2; // build iteration
  uint32 version_number = 3; // firmware feature level incarnation
}

message NodeInfo {
    VersionInfo version_info = 1; // firmware version information
    string manufacturer = 2; // manufacturer identifier
    string hostname = 3; // host network identification name, e.g.
   ↪ periNODE-<id>.local periCORE-<id>.local
    string mac_address = 4; // unique mac address
    string product_charge = 5; // production batch identifier
    string product_part_number = 6; // product part identifier
    string product_serial = 7; // serial number of the product
    string product_name = 8; // calling name of the product
    string product_version = 9; // version of the product at production time
    string pericore_charge = 10; // batch identifier of the included periCORE
    string pericore_part_number = 11; // periCORE part identifier
    string pericore_serial = 12; // periCORE serial number
    string pericore_version = 13; // periCORE version identifier
}
```

Listing 10: periCOREs *NodeInfo* object definition

Node Config

```
syntax="proto3";
package perinet.api;
option go_package = "perinet/api";

message NodeConfig {
    message Interface {
        google.protobuf.Any type = 1; //the type of interface, the
    ↪ particular interface has been configured to. Implementation specific.
        string element_name = 2; // element name of a particular interface,
    ↪ like an sensor source or an actuator sink.
        float period_seconds = 3; // in seconds, 0 means only event based
    ↪ (triggered) publishing
        uint32 samples_per_period = 4;  //defines how many values shall be
    ↪ sampled within a period,
                                        //the published value will be the
    ↪ rounded average
        // repeated Trigger trigger = 5;
    }
    string application_name = 1; // identification of the application the
    ↪ periCORE based node is assigned to
    string mqtt_broker_name = 2; // URI of the MQTT broker, the periCORE
    ↪ based node shall be connected to
    repeated Interface configs = 3;
}
```

Listing 11: periCOREs *NodeConfig* object definition

7.4.2 Life Cycle Service

/update periCORE based product *Update Firmware Image* (see Section 7.4.2).

- `PUT` expects octet-stream body.
  - `204` Returns empty body. The Request has been received and is being processed.
  - `401` Unauthorized access, returns empty body.
  - `500` Internal server error on unexpected error, returns empty body.
  - `404` Resource is not available in this life state, e.g for *periCORE Production State* (see Section 7.3).

/reboot periCORE based product configuration object.

- `PATCH` expects an empty body.
  - `204` Returns an empty body. The request is being processed and the device is performing a software reboot.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

/reset periCORE based product remote factory reset (see Section 7.3).

`PATCH` expects an empty body.

`204` The Request has been accepted and is being processed. All persistent data is reset to its default state. (see Section 7.3)

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

/production/oem-firmware periCORE based product *OEM Firmware Image* (see Section 7.3).

`PUT` expects octet-stream body (see Section 7.4.2).

`204` Returns empty body. The Request has been received and is being processed.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

`404` Resource is not available in this life state, e.g for *OEM Production State* (see Section 7.3).

/production/reset periCORE based product *Production Reset* (see Section 7.3).

`PATCH` expects an empty body.

`204` The Request has been accepted and is being processed. All persistent data is reset to its default state. (see Section 7.3)

`404` Internal server error on unexpected error, returns empty body.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

Firmware Image

The expected *Firmware Image* from the Life Cycle Service is a signed binary format. Further details can be found in the periCORE Firmware Development Application Note [6].

7.4.3 Security Service

/security/host-cert periCORE based product host certificate object (see Section 7.6).

`GET` expects an empty body.

`200` Return the host public certificate.

`401` Unauthorized access, returns empty body.

`500` Internal server error on unexpected error, returns empty body.

`PATCH` expects a text/plain-encoded body containing the certificate.

- `204` The request has been accepted and processed. Returns empty bod
- `401` Unauthorized access, returns empty body.
- `500` Internal server error on unexpected error, returns empty body.

/security/root-cert periCORE based product root certificate object (see Section 7.6).

`GET` expects an empty body.

- `200` Return the root public certificate.
- `401` Unauthorized access, returns empty body.
- `500` Internal server error on unexpected error, returns empty body.

`PATCH` expects a text/plain-encoded body containing the root certificate.

- `204` The request has been accepted and processed. Returns empty body.
- `401` Unauthorized access, returns empty body.
- `500` Internal server error on unexpected error, returns empty body.

/security/client-cert periCORE based product client certificate object (see Section 7.6).

`GET` expects an empty body.

- `200` Return the client public certificate.
- `401` Unauthorized access, returns empty body.
- `500` Internal server error on unexpected error, returns empty body.

`PATCH` expects a text/plain-encoded body containing the certificate.

- `204` The request has been accepted and processed. Returns empty body.
- `401` Unauthorized access, returns empty body.
- `500` Internal server error on unexpected error, returns empty body.

/security/reset periCORE based product security configuration reset object. (see Section 7.6)

`PATCH` expects an empty body.

- `204` Security configuration is reset to factory defaults: root and host certificates are replaced with the factory certificates, the client certificate is deleted, and the mTLS feature is disabled.
- `401` Unauthorized access, returns empty body.
- `500` Internal server error on unexpected error, returns empty body.

/security periCORE based product mTLS configuration object (see Section 7.6).

`GET` expects an empty body.

- 200 Returns a JSON-encoded object containing the key `enable_user_role` and it's value.
- 401 Unauthorized access, returns empty body.
- 500 Internal server error on unexpected error, returns empty body.

PATCH Expects a JSON-encoded object containing the key `enable_user_role` and it's value.

- 204 The request has been accepted and processed. Returns empty body.
- 401 Unauthorized access, returns empty body.
- 500 Internal server error on unexpected error, returns empty body.

7.5 Device Attestation

The periCORE module and all periCORE based products can be authenticated implicitly during an HTTP based access to the RESTful API (see Section 7.4). Since the access is secured via TLS, the received X.509 based certificate is used to authenticate the device.

The periCORE can be authenticated with the trust anchor Perinets Root CA certificate (*perinet-ecc-root-ca.crt* [7]).

7.6 Security

Certificate Type	Description
Root Certificate	The certificate that identifies the root CA, which is trusted by the periCORE. It is used by the periCORE to authenticate remote clients before they connect to the periCORE, when mTLS has been enabled.
Host Certificate	A unique certificate used by the periCORE to authenticate itself when communicating with a remote client via the HTTP Server. It holds the host name and for consistency purposes, it has been issued by a trusted root CA. This certificate proves the originality of the device (the host) itself.
Client Certificate	The certificate is used by a periCORE to authenticate itself when connecting as a client to a remote server. It would be used when the periCORE would try to connect to a MQTT broker, for instance.

Table 14: Security Resources

The *host and client certificates* depend on their *private key* associated. While the `GET`-method does not expose the private key, when a `PATCH`-request sends a new certificate to the device, it requires that the private key is attached in the certificate file.

To illustrate, a valid format of a *X.509 base64-encoded* certificate, concatenated with a private key, is shown below:


```
-----BEGIN CERTIFICATE-----
MIIB+TCCAZ6gAwIBAgIRAJ+o3YOez5YfOYAE9dnZ1uswCgYIKoZIzj0EAwIwSzEL
MAkGA1UEBhMCREUxFTATBgNVBAoMDFBlcmluZXQgR21iSDElMCMGA1UEAwwcRGV2
IFBlcmluZXQgQmF0Y2ggT3JkZXIgQ0EgMTAeFw0yMTAxMTMxMjI4MjhaFw0yMzA3
MDIxMjI4MjhaMEsxCzAJBgNVBAYTAkRFMRUwEwYDVQQKDAxQZXJpbmV0IEdtYkgx
JTAjBgNVBAMMHERldiB5Ix7978JIAAETBCTgSEWVIDIwMjEtMDEwWTATBgcqhkjO
PQIBBggqhkjOPQMBBwNCAASUbDFum+1dxJQLIEkydVBLzgA0Sfm1/9INfZ4yscgG
khEEk8mkfxkqebTZiKcRaHU8UnRs+ynM/P0LVbSnLsCBo2MwYTAOBgNVHQ8BAf8E
BAMCAQYwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQUbqaoSPeKEM5ixQr8QfB4
chSiW/YwHwYDVR0jBBgwFoAU3tPoJRtPDxs0J8RMeQNLS4lpsjAwCgYIKoZIzj0E
AwIDSQAwRgIhAMCOmfqVvHAm5ytSiJbpFIFWRkItpirWgWIZjUIrIgt2AiEA66bJ
DmI73vqSjhIRcBKAP0e7WaTAdeuhw0It+/BihBE=
-----END CERTIFICATE-----
-----BEGIN EC PARAMETERS-----
```

```
BggqhkjOPQMBBw==
-----END EC PARAMETERS-----
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIHhPsOT+I20b+j5Ix7978JItxV7y8ST23JYPko+I7Eo5Ix7978JISM49
AwEHoUQDQg5Ix7978JIaaZAecRQUbAOGMYPSkYNxPsDQcvUP0EYnTO0FxivaOYkR
ygaqR6hhxWJh+UKEjq5Tt7N6PY64qqp4eQ==
-----END EC PRIVATE KEY-----
```


The following excerpt shows how to update the *host certificate* with the `curl` command line tool. This example can be used for the root and client certificate as well, just change the URL and the file accordingly:

```
curl --data-binary @<certificate-file> \
 -H "Content-Type: text/plain" \
 -X PATCH https://periCORE-sernm.local/security/host-cert
# reboot device
curl -X PATCH https://periCORE-sernm.local/reboot
```

Note: Access to the periCORE is only possible incorporating the *always-on* TLS security. Clients, like web browsers or tools (e.g. `curl`), can be instructed to ignore the check of the CA of the server (periCORE). The connection is always encrypted, but this allows for cases where trust is guaranteed by the user (e.g. the periCORE is to be the only network member). In any case, the base URL will have the following form: `https://periCORE-sernm.local/`.

The *root certificate* is the public certificate of the trusted authority (by default referring to the Perinet ECC Root CA), in the form of a single X.509 base64-encoded certificate without the private key.

A periCORE can optionally enable *mTLS* with included *Role Based Access Control (RBAC)*. When enabled, the connecting remote client must authenticate itself by sending its client certificate before the connection can be established. The client certificate will be verified with and is expected to be signed by the stored root certificate and it is expected. Additionaly the client certificate carries a user role.

The roles ***admin***, ***super***, and ***reader*** are supported by the periCORE.

admin: full read/write access to the periCORE with no restrictions.

super: read and write to any resources with the exception of */security* and */update*.

reader: no write access to any resource.

Enabling *mTLS/RBAC* can make the periCORE's HTTP-server inacessible when no valid certificates are deployed, or the client trying to connect does not provide the expected signed certificate. In case of attempts with invalid client certificate, the error response **401 Unauthorized** is expected.

Through the `/security` resource, you can acquire the status of *mTLS/RBAC* in the form of a JSON-encoded object that carries the parameter `enable_user_role`. An example how to do this is shown below:

```
curl -X GET https://periCORE-sernm.local/security
# default response: {"enable_user_role":false}
```

To enable mTLS/RBAC via the RESTful API, it is possible to use the following excerpt:

```
curl --data='{"enable_user_role":true}' \
  -X PATCH https://periCORE-sernm.local/security
# reboot device
curl -X PATCH https://periCORE-sernm.local/reboot
```

The following excerpt shows how to reset the security configuration:

```
curl -X PATCH https://periCORE-sernm.local/security/reset
```

When the periCORE is not accessible, e.g. due to the loss of the *client certificate*, please refer to the factory reset procedure in section Section 6.

The security configuration is persistent, but requires a reboot of the device to be applied. This can be done through the RESTFul API described in Section 7.4 or via a power cycle.

7.7 *libperiCORE* Software Library

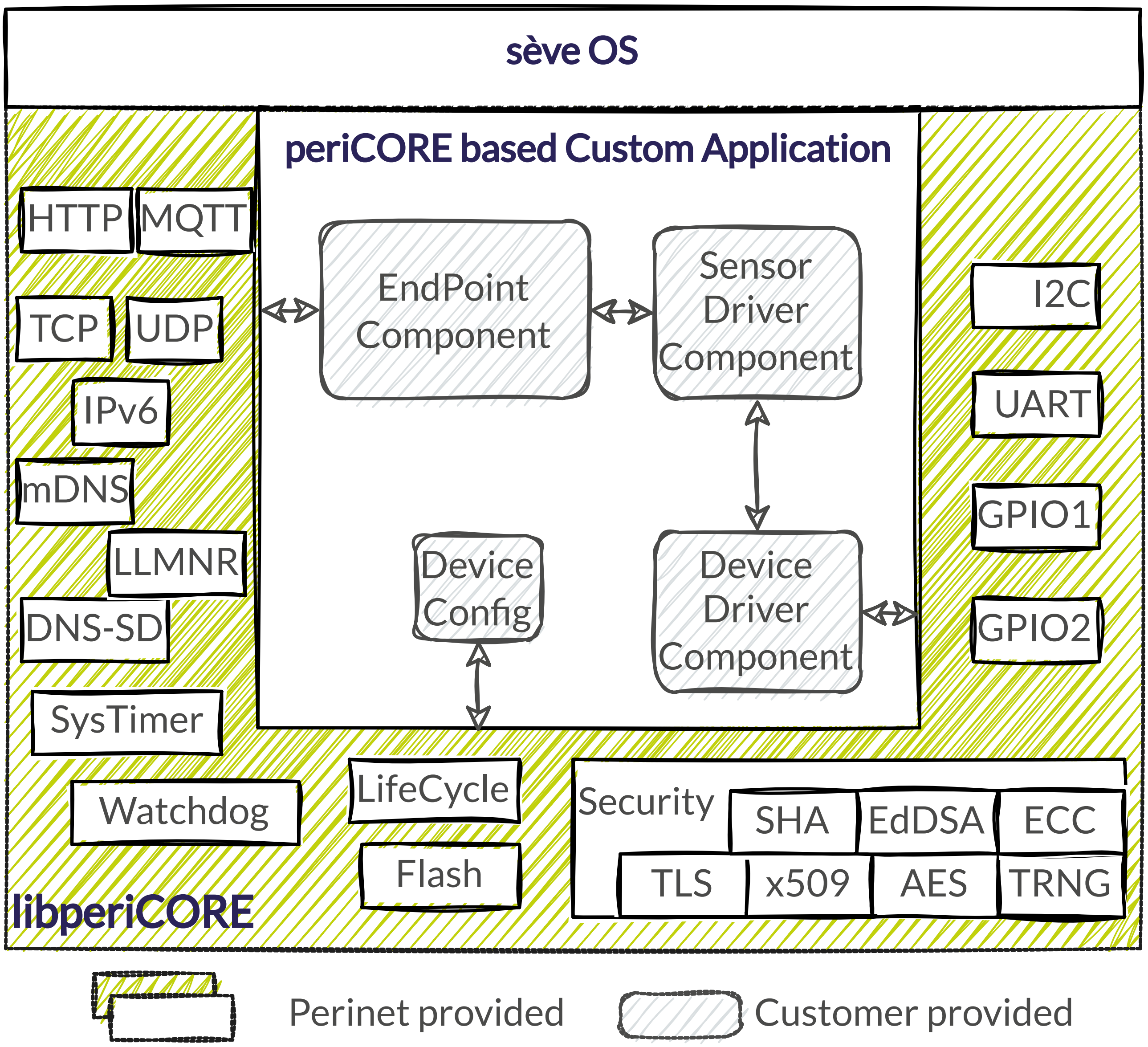


Figure 18: Overview of the *libperiCORE* structure.

7.8 Web User Interface

As shown prior in Figure 16, the *OEM Storage Area* and the *Operation Storage Area* contain a partition for a web user interface (WebUI).

When the periCORE based product is in *OEM Production State* or *Updated State* (referring to Figure 17), the WebUI is served by the firmware's integrated web server via the URL `https://periCORE-`*`sernm`*`.local` (refer to Section 9.2 for information on the generation of *`sernm`*).

An exemplary, expandable WebUI-template is part of the *periCORE SDK*. When the firmware is compiled with the *update_image* target, the resulting firmware binary will contain the WebUI. In the compilation process, all `.css`-, `.js`- and `.html`-files are minified, and `.png`-files are optimized.

Note: Be advised to keep file sizes low as for the *WebUI Data*-segments, a maximum of 256kB must be considered.

Updating the firmware with that binary, by means of the RESTful API (see Section 7.3.4), will then implicitly write the *WebUI Data*-segment of the *OEM Storage Area* or *Update Storage Area* respectively in the process. After a reboot, which is necessary for the firmware to update, the WebUI becomes available.

Due to the requirements of a client's system to support DNS-SD or mDNS, it is recommended at this point in time to access the `.local`-URL through the following web browsers:

- Windows Edge
- Safari

7.8.1 WebUI Template

The periCORE SDK comes with an expandable single-page web application template. One of the main purposes of it is the ability to read and write most of the services offered by its RESTful API (see Section 7.4), by means of a modern, secure web user interface.

This section shows the structure of the WebUI-template and how to customize it.

By default the following pages with the respective intended purposes exist:

Page	**Intended Purpose**
Home	Dashboard; display elements for *sensors* and control elements for *actuators*
Information	View for periCORE based product information (see Section 7.4.1)
Configuration	Inputs for periCORE based product configuration (see Section 7.4.1)
Security	Inputs for periCORE based security configuration (see Section 7.4.3)
Firmware Update	Upload of a new firmware image
API documentation	Download of the API documentation as an `api.proto`-file
Online documentation	Link to the documentation on the internet

Table 15: Security Resources

Example: changing the corporate identity

Figure 19 shows a mock-up image of the website, that contains header and footer but no content.

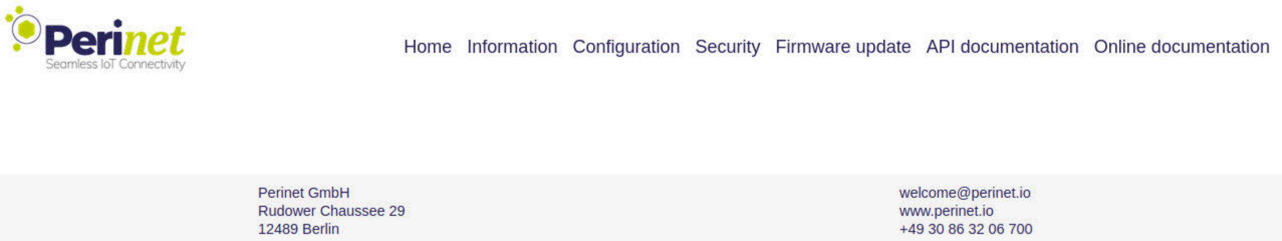


Figure 19: WebUI's corporate identity

Here, the logo and the information in the footer are subjects to a change of corporate identity. To carry out those changes, we need to modify the file `webui/fs_periCORE_product/js/peri_base.js`.

Figure 20 shows the periCORE SDK, with the structure of the WebUI-related files in the directory tree on the left, and the mentioned source file opened.

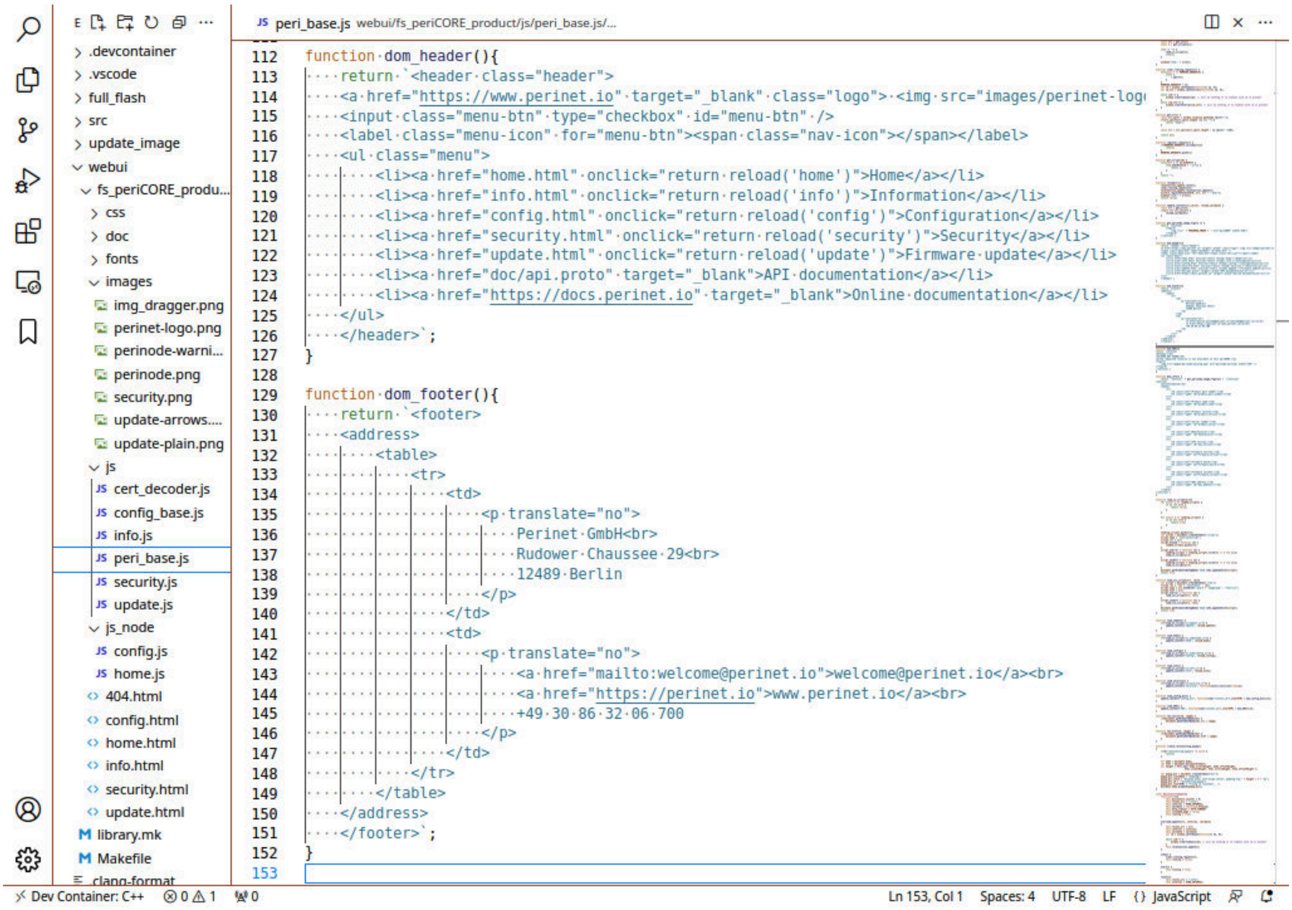


Figure 20: Source code for the WebUI's corporate identity in the periCORE SDK

To change the logo, refer to function *dom_header()* (line 112 in the figure), where the HTML image element's `src`-tag is by default set to the file `images/perinet-logo.png`.

To change the content of the footer, refer to the function *dom_footer()* (line 129 in the figure), and modify the HTML paragraphs.

Example: adding information to the *Home* page

Depending on the kind of sensor or actuator, and the implementation of the application specific sample and config objects (refer to Section 7.4.1), different HTML-elements can be considered feasible and can be freely chosen, to represent values or interactively display the status in case of a sensor peripheral, or in case of an actuator peripheral, to control it.

Figure 21 shows the *Home* page of a periCORE based product with actuator peripherals. In this case, the peripheral devices are two GPIO ports, with *GPIO 1* configured as an output (actuator) and *GPIO 2* configured as an input (sensor). Hence the purpose here is to offer a HTML input element that can control output *GPIO 1*. In case of input *GPIO 2*, we are simply disabling the same HTML input element for now.

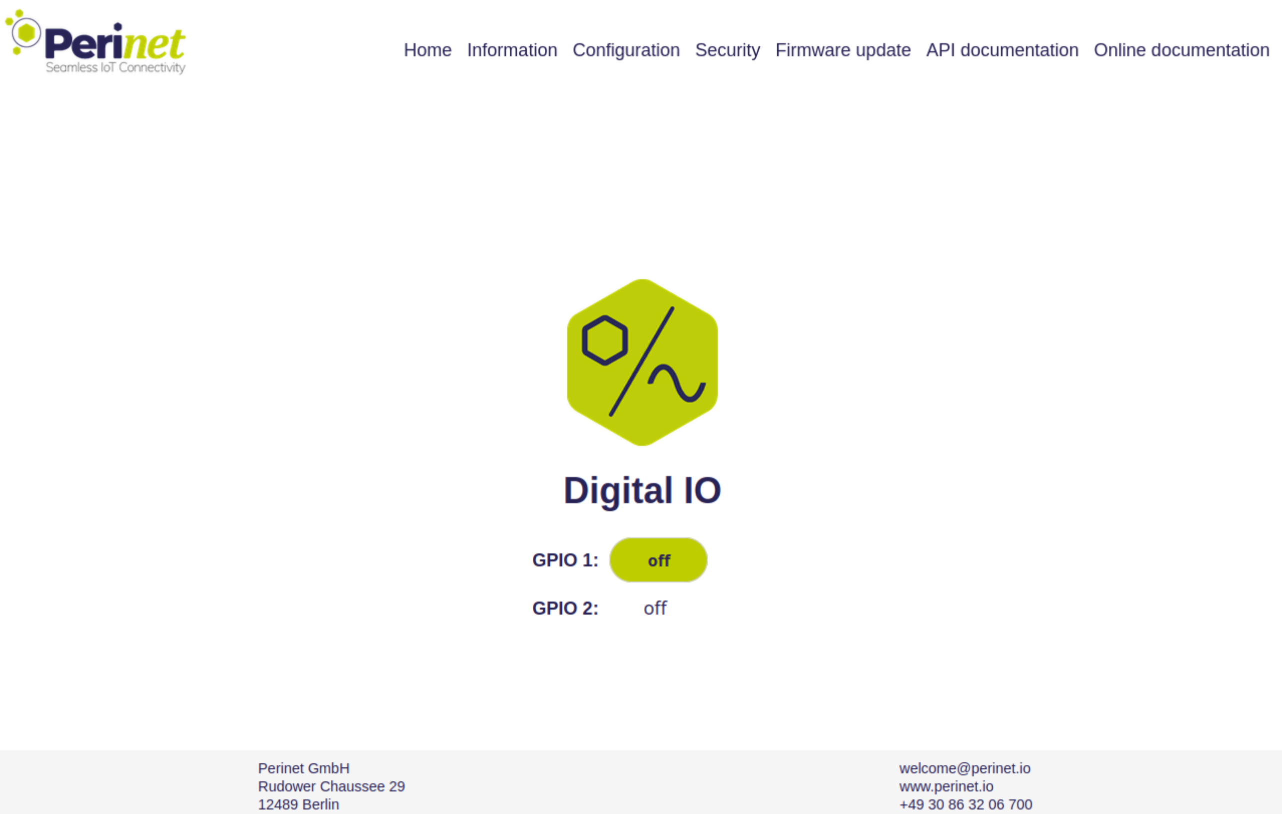


Figure 21: WebUI *Home* page for an actuator

The whole content of this page is derived from the source file `webui/fs_periCORE_product/js_node/home.js`.

Figure 22 shows the periCORE SDK with the mentioned source file opened.

Figure 22: source code for the WebUI's *Home* page in the in periCORE SDK

Its HTML is located at the top in function *dom_home()* and is manipulated by the functions below it.

Pre-defined functions are in place to facilitate the necessary updates that need to be made, to reflect the live status of the sensor or actuator in the user interface.

A function *evaluate_sample()* continuously provides the response of the RESTful API's sample object , in the form of its JSON-body that can be parsed and acted upon.

A function *evaluate_config()* continuously provides the response of the RESTful API's configuration object (refer to Section 7.4.1), in the form of its JSON-body as well.

The default period for repeating the requests is 1 second, as is set within the function *reload_home()* (not visible in the figure).

The action that occurs when an actuator element needs to be controlled can be freely implemented.

For example, the function *switch_if_1()* (not visible in the figure), that is connected to a click event on the input element for *GPIO 1*, will send a patch request to the RESTful API's configuration object (see Section 7.4.1), that results in toggling of the output-port.

Feedback about the success of this operation would be implemented through the mentioned `evaluate_sample()` function (visible in Figure 22), that updates the input element (e.g. change its text from "off" to "on").

8 Product Handling

The periCORE module is delivered in Tape and Reel with *500*pcs reel.

8.1 Reel Information

A reel of type A is used.

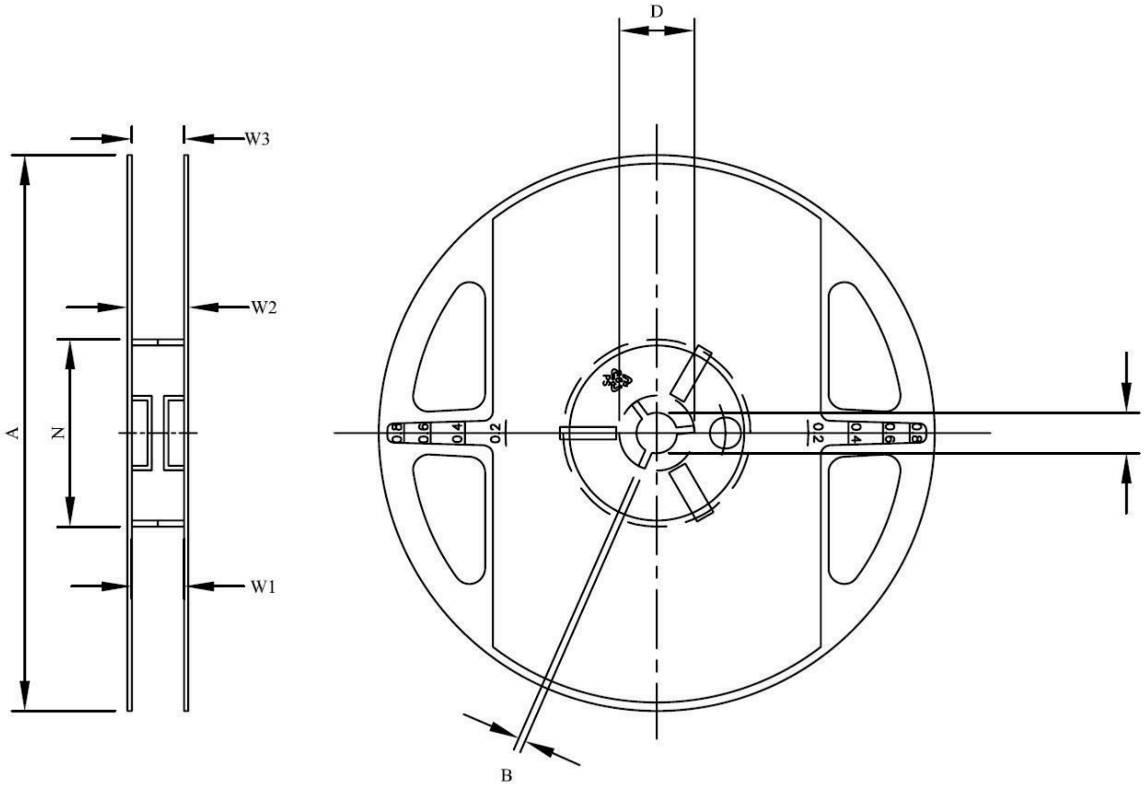


Figure 23: The dimensions of the Reel, used to deliver the periCORE modules.

Dimension	Value
A	33.00 mm
B	2.00 mm
C	12.82 mm
D	25.52 mm
N	100.00 mm
W1	30.00 mm
W2	36.00 mm
W3	30.00 mm

Table 16: Dimension of the reel.

8.2 Tape Information

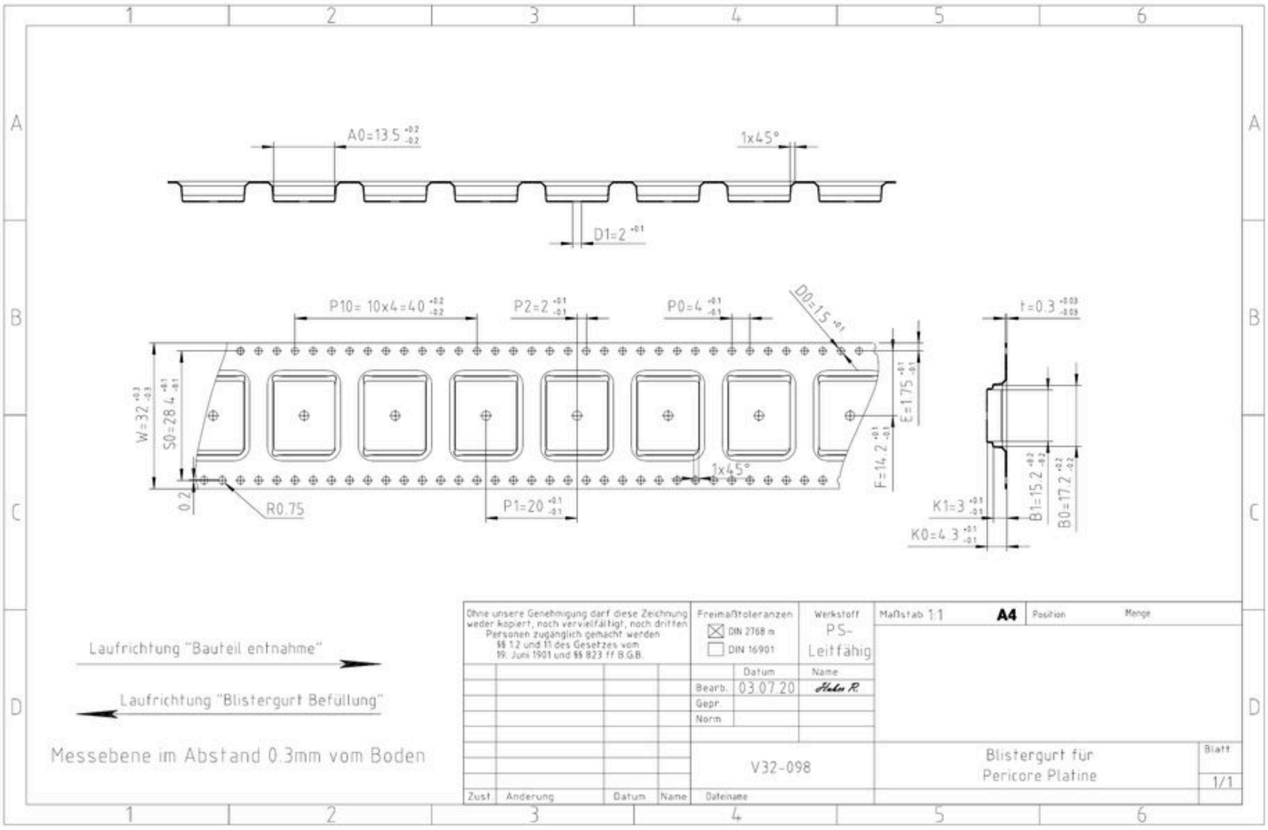


Figure 24: Tape and Reel packaging specification for the periCORE module.

8.3 Moisture Sensitivity Levels

The periCORE modules are rated at moisture sensitivity level 3 (MSL3). The reel dry bag is labelled with a moisture sensitive warning label with detailed information. With opening of the dry bag, the periCORE modules must be mounted within 168 hours while staying surrounded in factory conditions of maximum 30°C/60%RH.

During a potential storing, e.g. when the modules cannot be mounted within 168 hours after opening the dry bag, the storage condition must not exceed 10%RH.

The periCORE modules require baking/tampering if the humidity indicator card shows more than 10% when read at +23±5°C or if the conditions mentioned above are not met. Please refer to JEDEC J-STD-033B standards for more details about the bake procedure.

8.4 Pick and Place, Soldering

The module is suitable for pick and place machines. The surface area of the information adhesive on the main IC on top of the modules offers a good interface to vacuum driven pick up nozzles.

The periCORE is designed to be processed with reflow soldering.

8.5 ESD Handling Precautions

The periCORE module has limited built-in ESD (electro-static discharge) protection. Therefore, it is advised to always take reasonable precautions:

- Use a well grounded anti-static wrist strap when handling the module
- Discard the module only on anti-static surfaces
- Touch the module only at its edges or at the marking adhesive on the main IC on top of the module. Those are non-conductive. Never touch the pads or other components on the module
- The module should never get in touch with fabrics like clothing etc.

Note: By disobeying accepted ESD handling practices, the module may be damaged. The warranty may be void, if the module is damaged by ESD.

9 Product Marking

9.1 Optical Product Marking

The periCORE module is marked with a two-dimensional Data Matrix code according to ISO/IEC 16022 [1]. An example is shown in Figure 25.

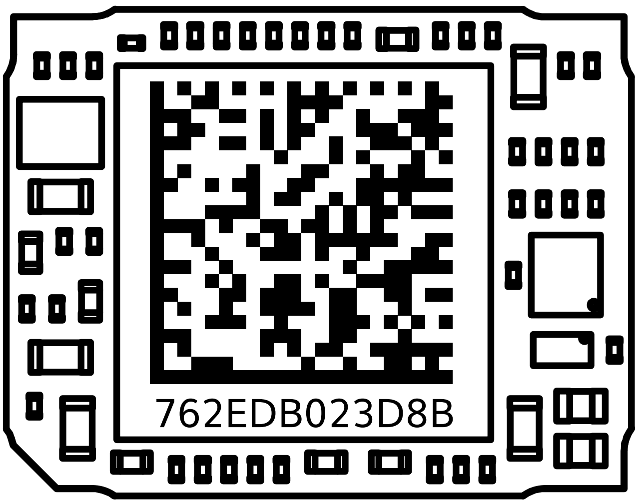


Figure 25: Visual marking via data matrix code of the periCORE module.

The sticker on the central IC shows the data matrix code and the 12-digit alphanumeric serial number. The information in the data matrix code is listed below:

- `pericore_serial`
- `pericore_part_number`

- `pericore_version`
- `pericore_charge`

```
762EDB023D8B\n
PRN000001\n
pC 4.1\n
2022-08
```

Listing 12: The example content of the data matrix code for a periCORE module.

9.2 Electronic Product Marking

Via the network interface, product identification parameters can be read via the RESTful API (see Section 7.4.1):


```
{
  "hostname": "periCORE-sernm",
  "mac_address": "76:2E:DB:02:3D:8B",
  "manufacturer": "Perinet GmbH",
  "pericore_charge": "1",
  "pericore_part_number": "PRN.000.001",
  "pericore_serial": "762EDB023D8B",
  "pericore_version": "5",
  "product_charge": "",
  "product_name": "",
  "product_part_number": "",
  "product_serial": "",
  "product_version": "",
  "version_info": {
    "firmware_variant": "periCORE",
    "firmware_version": {
      "api": 12,
      "build": 52,
      "version_number": 14
    }
  }
}
```


Listing 13: The periCORE NodeInfo object received via the resource /info.

Note: The above listed parameters are provided as an example and do variate for different periCORE modules.

9.3 Unique Serial Number

A periCORE module is identified by a 12-digit unique serial number. The globally unique serial number is deployed to each periCORE module during the production. It is available on the periCORE module as part of the optical marking (see Section 9.1) as well as part of the electronic marking (see Section 9.2).

9.4 Unique Hostname

During the production, a periCORE module is configured with a unique *hostname*. The hostname is generated out of two parts, the static *prefix* and a dynamically generated *suffix*. During *OEM Production* the hostname can be overwritten via the RESTful API (see Section 7.3.5 and Section 7.4).

periCORE-sernm.local the *prefix* part is set to *periCORE* for all periCORE modules.

periCORE-sernm.local the 5-digit alphanumeric *suffix* was generated from a unique serial number (see Section 9.3) by using a Base32 conversion algorithm, which is described in Table 17.

Step	Instruction	Example
1	Take the *last 6 digits* of the serial number	Serial number `762EDB023D8B` ⇒ `023D8B`
2	Convert them to binary	`023D8B`$_{(16)}$ ⇒ `0000 0010 0011 1101 1000 1011`$_{(2)}$
3	Enqueue a 0 to the most significant position, if the serial number starts with `742EDB`. Enqueue an 1 otherwise.	`0000 0010 0011 1101 1000 1011` ⇒ `1 0000 0010 0011 1101 1000 1011`
4	Order the sequence into sets of 5	`1 0000 0010 0011 1101 1000 1011` ⇒ `10000 00100 01111 01100 01011`
5	Convert each set using the character definition from Table 18	`10000 00100 01111 01100 01011` ⇒ `s e r n m`

Table 17: Algorithm to generate the hostname *suffix* from the serial number

A unique part of the hostname is generated with the conversion of the serial number to a Base32 encoded alphanumeric representation. The Base32 encoded representation was selected to discard ambiguous characters.

Value$_2$	00000	00001	00010	00011	00100	00101	00110	00111	01000	01001	01010	01011	01100	01101	01110	01111	10000	10001	10010	10011	10100	10101	10110	10111	11000	11001	11010	11011	11100	11101	11110	11111
Value$_{10}$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Value$_{32}$	a	b	c	d	e	f	g	h	i	j	k	m	n	p	q	r	s	t	u	v	w	x	y	z	2	3	4	5	6	7	8	9

Table 18: The Base32 conversion table to generate the hostname *suffix*.

A periCORE based product supports the mDNS based name resolution protocol. The libperiCORE (see Section 7) implements an mDNS responder. Therefore, the hostname of a peri-

CORE based product can be resolved via the mDNS protocol with its link local FQDN, e.g. the `periCORE-sernm.local` resolves to the following Link Local Address (IPv6 LLA):

`fe80:0000:0000:0000:762e:db02:3d8b:0000`

9.5 Unique IPv6 Link Local Address

The periCORE's IPv6 Link Local Address (IPv6 LLA) is formed from the serial number, that can be retrieved from the optical (see Section 9.1) or electronic (see Section 9.2) product marking. It is therefore MAC-address based as well (see Section 9.6) and is composed as follows:

$$\texttt{fe80:0000:0000:0000:}xxxx:xxxx:xxxx\texttt{:0000}$$

where each x represents one digit of the serial number following its ordering, e.g. `fe80::762e:db02:3d8b:0000` represents the IPv6 LLA for the serial number `762EDB023D8B`.

9.6 MAC Address

The periCORE's MAC-address is based on the serial number, that can be retrieved from the optical (see Section 9.1) or electronic (see Section 9.2) product marking. It is composed as shown in the following excerpt:

$$xx:xx:xx:xx:xx:xx$$

where each x represents one of the digits of the serial number in order, e.g. `76:2E:DB:02:3D:8B` represents the MAC-address for the serial number `762EDB023D8B`.

10 Ordering Information

Ordering Code	Product Name	Description
PRN.000.001	periCORE	periCORE single pair ethernet communication module.
PRN.000.019	periCORE Development Board	Minimal firmware development setup.
PRN.000.020	periCORE Development Kit	Full featured firmware development setup.

11 Contact & Support

For customer support, please call us at **+49 30 863 206 701** or send an e-mail to *support@perinet.io*.

For complete contact information visit us at `www.perinet.io`

A List of Figures

B List of Listings

C List of Tables

D Glossary

100BASE-T1 A Ethernet Standard where two endpoints are connected by a single twisted pair cable. It is one of the so-called Single Pair Ethernet (SPE) standards. It operates in full duplex with a data rate of 100 MBit per second. Furthermore, it uses PAM-3 modulation with a voltage level from -1 to +1V, differentially on the two wires. 7, 8

100BASE-TX A Ethernet Standard where two twisted pairs with differential signals are used, one for each direction. The data rate is 100 MBit per second. It is also called "Fast Ethernet". 7, 8

API Application Programming Interface. 35, 54, 55

BSC Basic Spacing Between Centers. 15, 16

CA Certification Authority, a trusted entity which is represented by a certificate that is used to verify the signature on a certificate issued by that authority (trust anchor). 41–43

CSMA/CD Carrier-sense multiple access with collision detection. 9

DNS-SD DNS Service Discovery [2] is a way of using standard DNS programming interfaces, servers and packet formats to browse the network for services. 6, 30, 46

ESD Electro Static Discharge. 3, 53

FIFO First In First Out, where the first in is the first out. For resource management, like for the scheduling of tasks it is also known as a first-come, first-served (FCFS) and specifies the order of the execution of a task in regard to his occurrence over time. 28

FQDN Fully Qualified Domain Name, sometimes also referred to as an absolute domain name, is a domain name that specifies its exact location in the tree hierarchy of the Domain Name System (DNS). 56

GPIO General-Purpose Input/Output. 7, 13

HTTP Hypertext Transfer Protocol is an application-layer protocol for transmitting hypermedia documents, such as HTML. 30, 41

I2C Inter-Integrated Circuit is a synchronous, multi-master, multi-slave, packet switched, single-ended, serial bus. 7

IC Integrated Circuit. 1, 52, 53

idempotent Idempotence is the property of certain operations in mathematics and computer science whereby they can be applied multiple times without changing the result beyond the initial state. 31

IIoT Industrial Internet of Things. 1, 7

IPv6 Internet Protocol Version 6 [10], a communication protocol. 30

IPv6 LLA Internet Protocol Version 6 [10] link-local unicast address. 56

MAC Media Access Controller, component that handles concurrent access to a shared physical communication medium. 9, 30, 56

mDNS multicast Domain Name Service [3], a protocol that implements a local distributed name resolving mechanism. 6, 30, 46, 55, 56

MQTT Message Queuing Telemetry Transport is a lightweight, publish-subscribe based network protocol that transports messages between devices. 30

mTLS Mutual TLS extends the TLS protocol by requiring clients to pass certificates, allowing to provide authorization mechanisms of Application services. 26, 34, 40, 42–44

OEM Original Equipment Manufacturer is generally perceived as a company that produces parts and equipment that may be marketed by another manufacturer. 27, 29–31

OS Operating System, a system software that manages computer hardware and software resources. 27, 28

OTP One-Time Programmable memory. 29

PHY Physical layer. Lowest layer of the OSI model. 8

RBAC Role Based Access Control. 31, 34, 43, 44

SPE Single Pair Ethernet. 7

TLS Transport Layer Security Protocol. Used by Application Protocols like MQTT or HTTP to allow secure data transfer. 30, 41, 43

UART A Universal Asynchronous Receiver-Transmitter is a computer hardware device for asynchronous serial communication. 7

X.509 X.509 is an International Telecommunication Union (ITU) standard defining the format of public key certificates. 30, 41

E References

[1] ISO/IEC 16022:2006(E). *Information technology — Automatic identification and data capture techniques — Data Matrix bar code symbology specification*. Geneva, Switzerland, 2006.

[2] S. Cheshire and M. Krochmal. *DNS-Based Service Discovery*. RFC 6763. `http://www.rfc-editor.org/rfc/rfc6763.txt`. RFC Editor, Feb. 2013. URL: `http://www.rfc-editor.org/rfc/rfc6763.txt`.

[3] S. Cheshire and M. Krochmal. *Multicast DNS*. RFC 6762. `http://www.rfc-editor.org/rfc/rfc6762.txt`. RFC Editor, Feb. 2013. URL: `http://www.rfc-editor.org/rfc/rfc6762.txt`.

[4] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. RFC 7231. `http://www.rfc-editor.org/rfc/rfc7231.txt`. RFC Editor, June 2014. URL: `http://www.rfc-editor.org/rfc/rfc7231.txt`.

[5] Perinet GmbH. periCORE Development Kit User Guide. PRN.100.378. `https://docs.perinet.io/`.

[6] Perinet GmbH. periCORE Firmware Development Application Note. PRN.100.379. `https://docs.perinet.io/`.

[7] Perinet GmbH. Perinets public Root Certification Authority certificate. `perinet-ecc-root-ca.crt`. `https://docs.perinet.io`.

[8] Perinet GmbH. sève Operating System - A Component Oriented Event-Flow Model White Paper. PRN.100.533. `https://docs.perinet.io/`.

[9] Perinet GmbH. sève Operating System Datasheet. PRN.100.377. `https://docs.perinet.io/`.

[10] R. Hinden and S. Deering. *IP Version 6 Addressing Architecture*. RFC 4291. `http://www.rfc-editor.org/rfc/rfc4291.txt`. RFC Editor, Feb. 2006. URL: `http://www.rfc-editor.org/rfc/rfc4291.txt`.

[11] "IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and Metropolitan Area Networks - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Physical Layer Parameters and Specifications for 1000 Mb/s Operation over 4 pair of Category 5 Balanced Copper Cabling, Type 1000BASE-T". In: *IEEE Std 802.3ab-1999* (1999), pp. 1–144. DOI: `10.1109/IEEESTD.1999.90568`.

[12] "IEEE Standards for Local and Metropolitan Area Networks: Supplement - Media Access Control (MAC) Parameters, Physical Layer, Medium Attachment Units, and Repeater for 100Mb/s Operation, Type 100BASE-T (Clauses 21-30)". In: *IEEE Std 802.3u-1995 (Supplement to ISO/IEC 8802-3: 1993; ANSI/IEEE Std 802.3, 1993 Edition)* (1995), pp. 1–415. DOI: `10.1109/IEEESTD.1995.7974916`.

[13] "IEEE Standards for Local and Metropolitan Area Networks: Supplements to Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications - Specification for 802.3 Full Duplex Operation and Physical Layer Specification for 100 Mb/s Operation on Two Pairs of Category 3 Or Better Balanced Twisted Pair Cable (100BASE-T2)". In: *IEEE Std 802.3x-1997* (1997), pp. 1–415. DOI: `10.1109/IEEESTD.1997.95611`. URL: `https://standards.ieee.org/ieee/802.3x/1082/`.

[14] ISO/IEC. *Programming Languages — C++*. International Standard N4849. Dec. 2020. URL: `https://www.iso.org/standard/79358.html`.

[15] "ISO/IEC/IEEE International Standard - Part 3: Standard for Ethernet - Amendment 1: Physical Layer Specifications and Management Parameters for 100 Mb/s Operation over a Single Balanced Twisted Pair Cable (100BASE-T1)". In: *ISO/IEC/IEEE 8802-3:2017/Amd 1:2017(E)* (2018), pp. 1–92. DOI: `10.1109/IEEESTD.2018.8310988`.

[16] NXP Semiconductor. *UM10204*. https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

F Revision History

Revision	Date	Author(s)	Description
1	022-06-08	msen, dper, shoe, kwal, tkla, psil	Initial Release
2	July 22, 2022	shoe, psil, kwal, nsav	Update cover picture, Add missing LED interface description(Table 6), Add reel dimensions to Section 8, Add description for physical factory reset (Section 6), Update hardware sections (Section 2,Section 4,Section 3), Add Section 5, Update Section 10