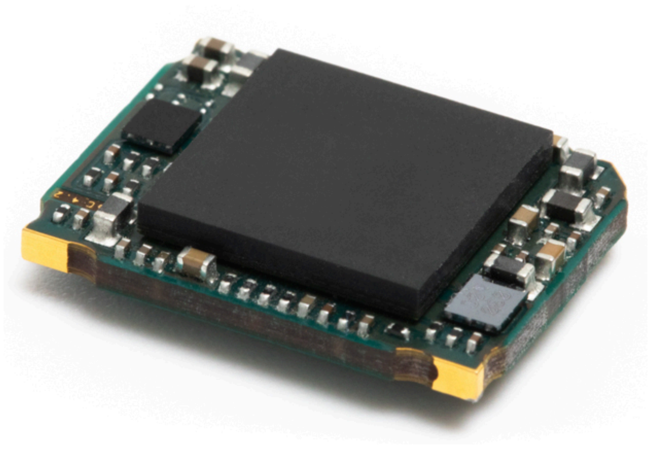
periCORE

Single Pair Ethernet communication module



Datasheet

Seamless IoT Connectivity

Abstract
This datasheet presents the *periCORE*, a versatile IIoT module designed for connecting sensors and actuators to a Local Area Network using Single Pair Ethernet. Its compact size makes it ideal for integration into sensor housings, linking them directly with the IT world. The module also supports multiple interfaces, accommodating a wide range of analog front-end ICs.

Doc-No.: PRN.100.375 rev: 3 2024-02-08

Document Information

Title	periCORE
Subtitle	Single Pair Ethernet communication module
Type	Datasheet
Status	Release
Version	3
Date	2024-02-08
Disclosure Restriction	

Contents

1 Overview

The periCORE module serves as an Ethernet communication tool, specifically created for use with sensor and actuator devices. It enhances these devices by adding network functionalities in both hardware and software aspects, facilitating easy integration. This module transforms passive sensors and actuators into smart devices capable of processing data and responding to events. It comes with built-in, advanced network features like state-of-the-art security and efficient firmware management. Additionally, periCORE supports easy rebranding and customization of devices, allowing anyone to develop customized firmware using the available 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 ± 20 %)
- 2 x GPIO

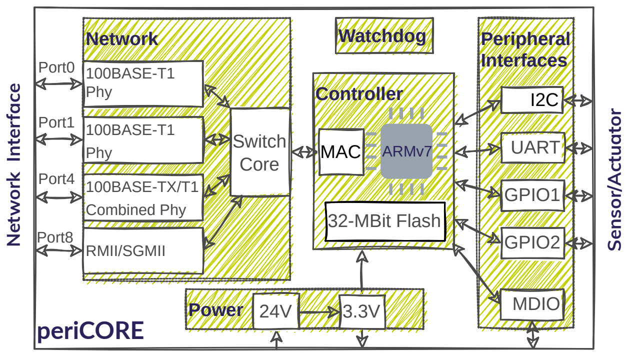


Figure 1: periCOREs hardware blocks.

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 3.9 mm (Figure 2)
Mounting: Solder pads, 73 LGA-Pads, Pattern 13 x 10, Pitch 1.27 mm

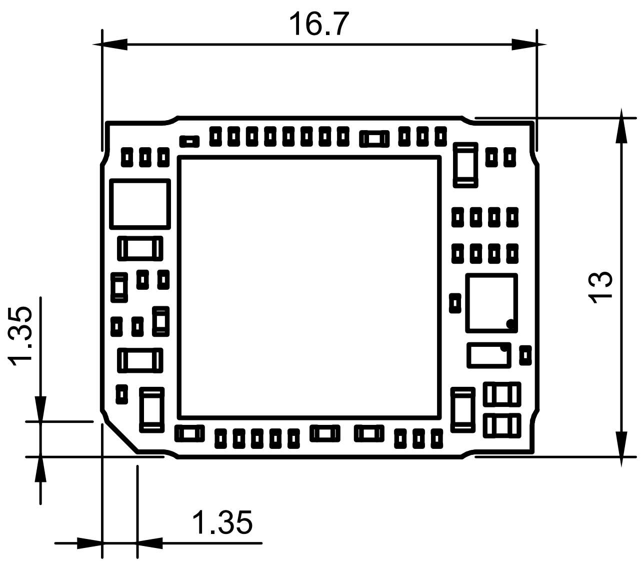


Figure 2: periCOREs dimensions in mm.

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

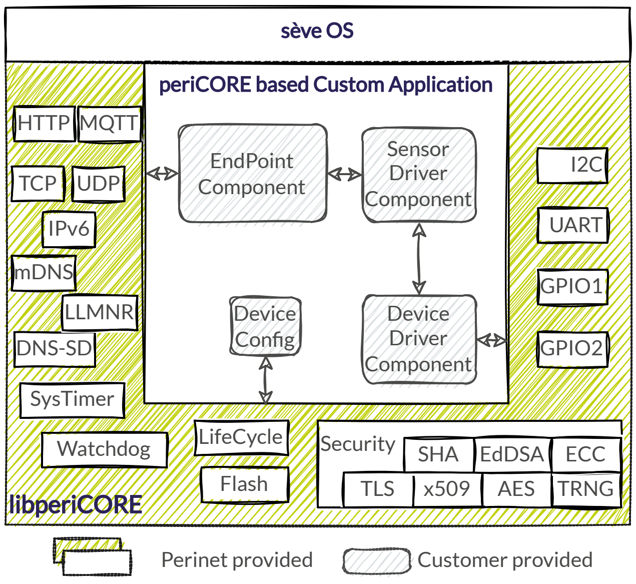


Figure 3: The software architecture with Custom Application template, provided by Perinet.

2 Module Architecture

The *periCORE* communcation module is designed with various external interfaces to meet the demands of an IIoT Industry 4.0 setting. These interfaces are organized into three main blocks, each linked to the module's internal controller. A detailed illustration of these interface blocks is available in Figure 4. The diagram also includes the Micro Controller Unit (MCU), depicted as the 'Controller' hardware block.

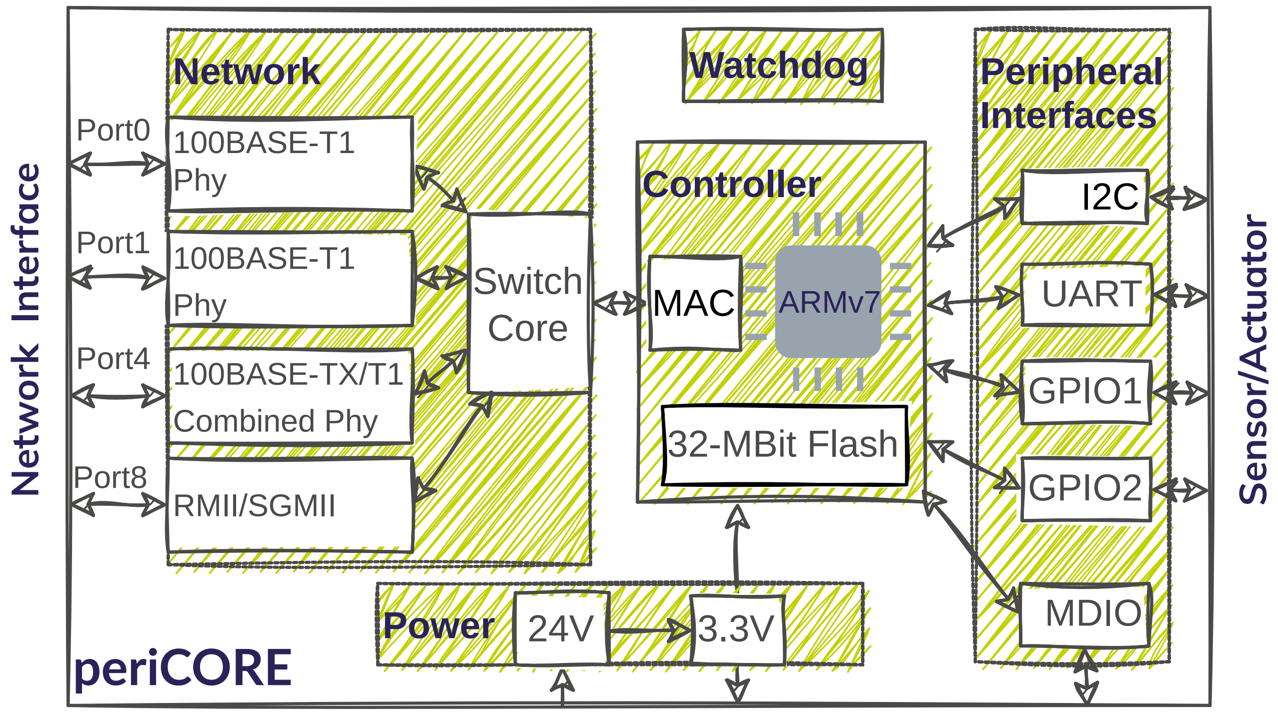


Figure 4: Included hardware blocks of the *periCORE* communcation module .

Network This includes three Ethernet communication ports compatible with 100BASE-T1 for SPE communication. Port 4 can be set to either 100BASE-T1 or 100BASE-TX. More information is available in Section 2.1.

Peripheral Interfaces The *periCORE* communcation module offers I2C, 2-wire UART, and two GPIO ports for connecting and controlling OEM sensor or actuator products. Detailed information on these interfaces is in Section 2.2.

Power Alongside the 24V input, a 3.3V output is provided to power peripheral circuits. Details on the power block are in Section 2.3.

Controller This Micro Controller Unit (MCU) runs platform-specific firmware for sensor/actuator applications. It manages sensor/actuator data collection and configures network interfaces, as discussed in Section 2.4.

Watchdog An independent circuit designed to reset the *periCORE* communcation module in case of errors. Further details are in Section 2.5.

2.1 Network Interfaces

The periCORE module is equipped with two Ethernet PHYs compliant with 100BASE-T1 (Port 0 and Port 1), and an adaptable PHY supporting both 100BASE-T1 and 100BASE-TX (Port 4). A 100BASE-T1 PHY can function in either master or slave mode, as defined by [17]. This physical layer creates a connection between a master mode 100BASE-T1 PHY and a slave mode 100BASE-T1 PHY. However, a link cannot be established if both PHYs are set to the same mode, either master or slave. Each port's mode (master or slave) is set during the bootup process and remains static. For port configuration details, refer to Table 1.

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

Automatic negotiation for the master or slave mode of the 100BASE-T1 PHY is not specified in the IEEE 802.3bw standard [17] and is not a feature of the *periCORE* communcation module module. Each port on the periCORE module is connected to a 100MBit full-duplex MAC, which is integrated with the switch core.

Note: Port 4 on the *periCORE* communcation module can operate as either 100BASE-T1 or 100BASE-TX. This port must be configured accordingly.

2.1.1 Ports 0, 1 and 4 - 100BASE-T1

A typical application circuit using Ports 0, 1, and 4 for 100BASE-T1 is presented in Figure 5. This circuit also demonstrates the concept of hybrid cabling, which uses a single cable with two pairs of wires—one pair for data transfer and the other for power.

In the circuit, Port 0 functions as a 100Base-T1 slave, while Ports 1 and 4 are set as 100Base-T1 masters. Notably, Port 4's default setting is 100Base-TX, requiring manual configuration to 100Base-T1. The slave port (Port 0) connects to an M8 Hybrid Male connector (style 6J-M8CI [5]), while the master ports (Ports 1 and 4) connect to M8 Hybrid Female connectors (style 6J-M8C [5]). This setup ensures that the direction of power flow and the master/slave configuration are determined by the connector type. Power flows from a master (M8H Female connector) to a slave (M8H Male connector). The standards for both male and female M8 Hybrid connectors and their mating faces (6P-M8CI and 6P-M8C) are defined in IEC 63171-6:2021 [5].

Note: The circuit design shown in Figure 5 does not provide a 2 kV isolation for the data lines as required by the product safety standard IEC 62368-1:2023.

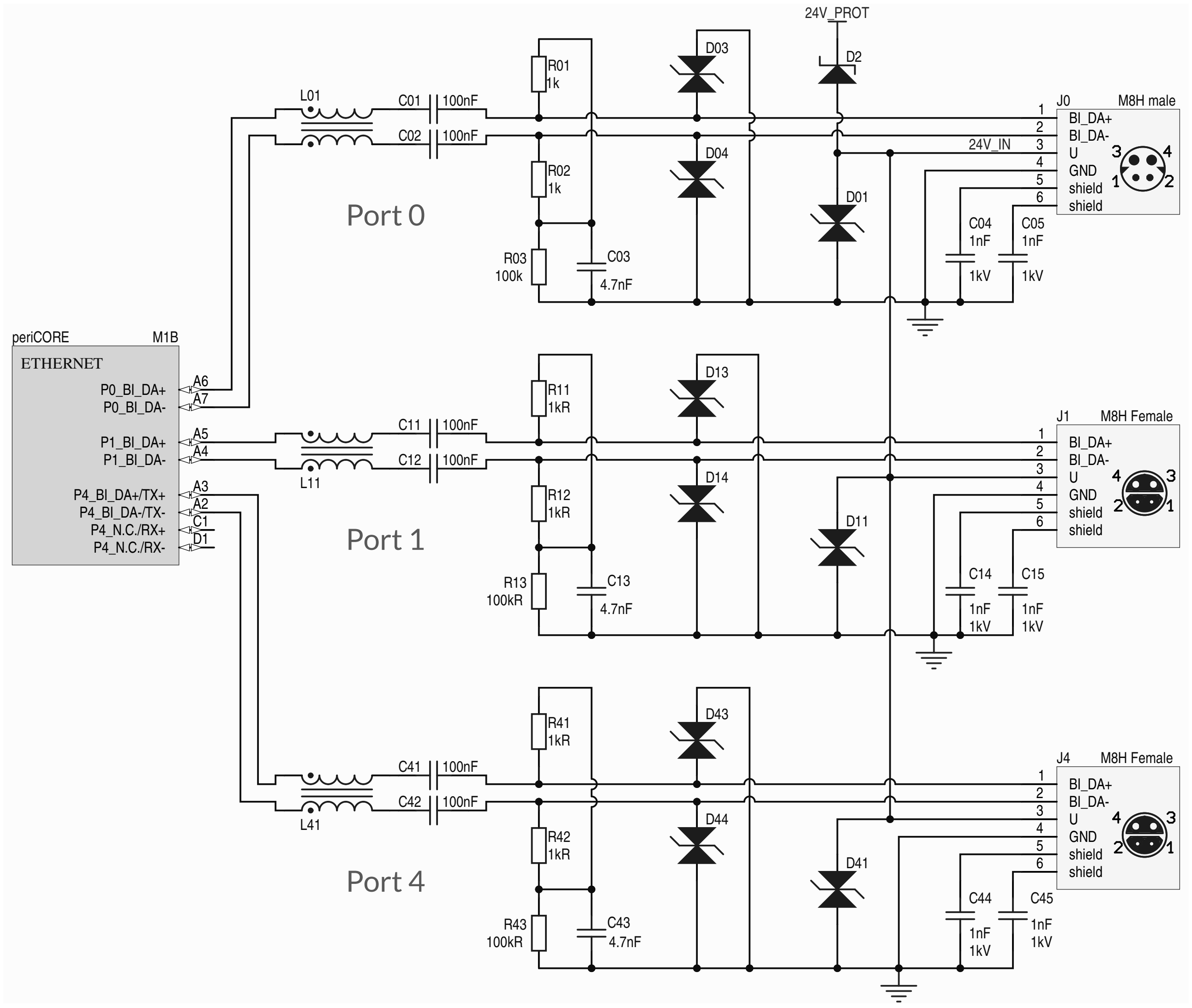


Figure 5: 100BASE-T1 typical application circuit

The circuit design for the *periCORE* communcation module module follows the recommendations of OPEN ALLIANCE [2]. It includes the following components for each port (where x denotes the port number: 0, 1, and 4):

- Cx1, Cx2 - DC blocking capacitors with minimum voltage rating of 50V and 10 % tolerance
- Lx1 - common mode choke. Recommended parts are: Murata DLW31SH222SQ2#, Bourns SRF3216-222Y and Wurth Elektronik 744232222
- Rx1, Rx2 - common mode RF currents termination with 1% tolerance and minimal power rating of 0.4 W
- Rx3 - discharge resistor. Power rating > 0.1 W
- Cx3 - HF bypass capacitor with voltage rating >= 100 V and not more than 10% tolerance
- Cx4, Cx5 - Shield isolation capacitors with 1kV voltage rating

- Dx1 - ESD protection for power supply lines. Recommended parts are: Littelfuse SD24C-01FTG and Bourns CDSOD323-T24SC
- D2 - Reverse polarity protection
- Dx3, Dx4 - ESD protection for the data lines with capacitance < 3.5 pF
- J1, J4 - M8 Hybrid Female connector (TE connectivity T4040110044-000)
- J0 - M8 Hybrid Male connector (TE connectivity T4041110044-000)

Placement and layout guidelines:

- Dx1, Dx3 and Dx4 should be placed as close as possible to the connector.
- MDI (BI_DA) differential pairs should be routed as coplanar waveguides with characteristic differential impedance of 100 Ω ± 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.
- We recommend using the minimum number of 4 layers when planning the stackup of the PCB

2.1.2 100BASE-T1/TX Port 4

Port 4 is set by default to operate in 100BASE-TX mode. It incorporates a media access controller (MAC) compliant with IEEE 802.3, IEEE 802.3u, and IEEE 802.3x standards (refer to [14] and [15]). The port's auto-negotiation feature, which follows IEEE 802.3u and IEEE 802.3ab specifications (see [13]), is enabled by default. In the 100BASE-TX mode, the MAC can automatically determine the most suitable speed (either CSMA/CD or full-duplex) based on the results of the PHY auto-negotiation. An example application circuit for Port 4 in 100BASE-TX mode can be found in Figure 6.

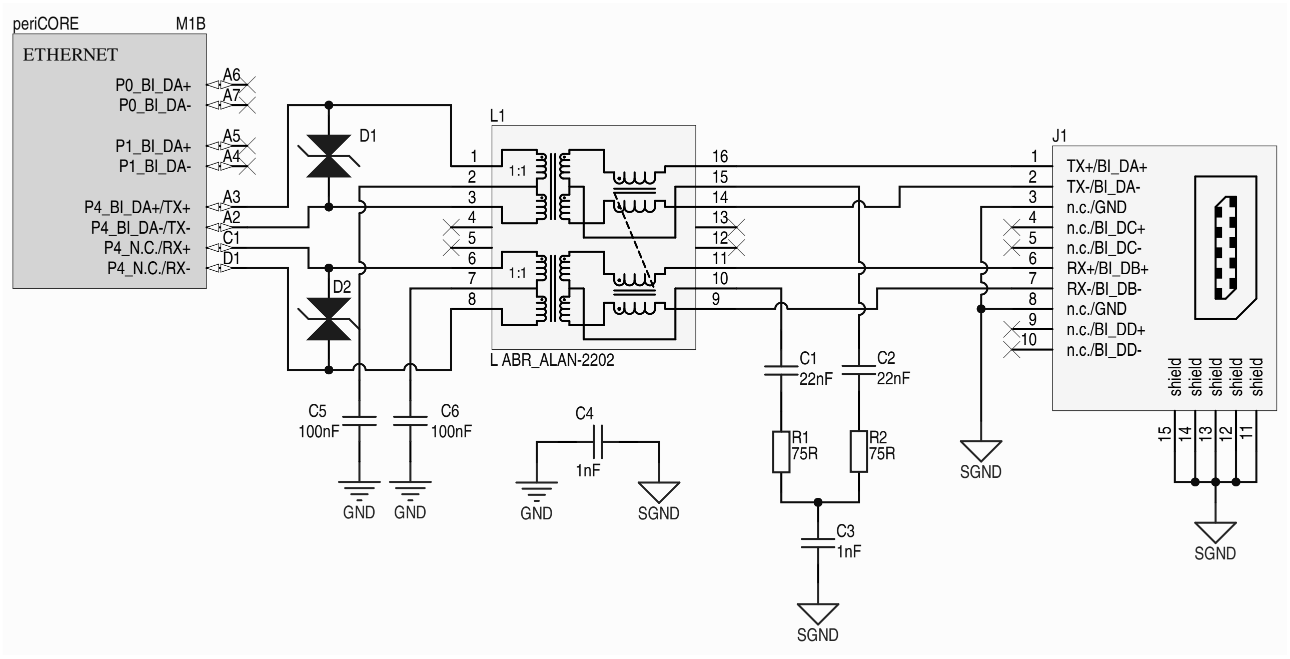


Figure 6: periCORE Port 4 100BASE-TX typical application circuit

The circuit consists of the following following parts:

- J1 - HARTING iX Industrial Ethernet connector (HARTING 09452812560)
- L1 - LAN transformer (ABRACON ALAN-2202 or similar)
- C1, C2 - Capacitors, 22 nF, voltage rating >= 100 V
- R1, R2 - Resistors, 75 Ω, power rating >= 0.1 W
- C3, C4 - Capacitor, 1 nF, voltage rating >= 2 kV
- C5, C6 - Capacitors, 100 nF, voltage rating >= 100 V
- D1, D2 - TVS diodes, 2.5 pF capacitance (Littelfuse SP4020-01FTG-C or similar)

C1, C2, R1, R2, and C3 are components of the Bob Smith termination network, which helps terminate common mode signals. C4 provides a low impedance path for these signals between the primary and secondary sides of the transformer. Additionally, C5 and C6 connect the transformer's secondary windings' center taps to the ground potential for AC signals.

When placing and laying out these components, consider the following guidelines:

- Place D1 and D2 as close to the transformer taps as possible.
- Route MDI differential pairs (from the connector to the transformer and from the transformer to the periCORE module) as coplanar waveguides. They should have a characteristic differential impedance of 100 Ω ± 10 %. Keep the traces symmetrical and the distance between them approximately equal to their width.
- Ensure a significant distance between the MDI pair and other traces or power planes. Ideally, maintain a spacing of at least 5 times the dielectric height (5H edge-to-edge).

- Design the traces from C3 and C4 to the shield to have minimal inductance.
- For PCB stackup, a minimum of 4 layers is recommended.

2.1.3 Serial LED Interface

The periCORE module uses its Serial LED Interface to provide link status information for Ports 0, 1, and 4. This interface operates with two wires: LED_CLK (for the clock signal) and LED_-DATA (for data).

Data is transmitted as a serial bit stream on the LED_DATA pin, synchronized with the clock signal on the LED_CLK pin. For detailed timing information, refer to Section 5.2.

The bit stream is composed of 24 bits, and each bit is active-low. For every port, there are 3 bits indicating different statuses:

- 100MBit/ACT - This bit is active (low) when a 100Mbit link is established and toggles with any link activity.
- ACT - This bit shows link activity and toggles with any activity on the link.
- LNK - This bit indicates the presence of a link, being low when the link is established.

The specific arrangement of these bits in the stream is outlined 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

To properly use the data provided by the Serial LED Interface, an external circuit equipped with shift registers is required. A diagram illustrating this circuit setup is available in Figure 7.

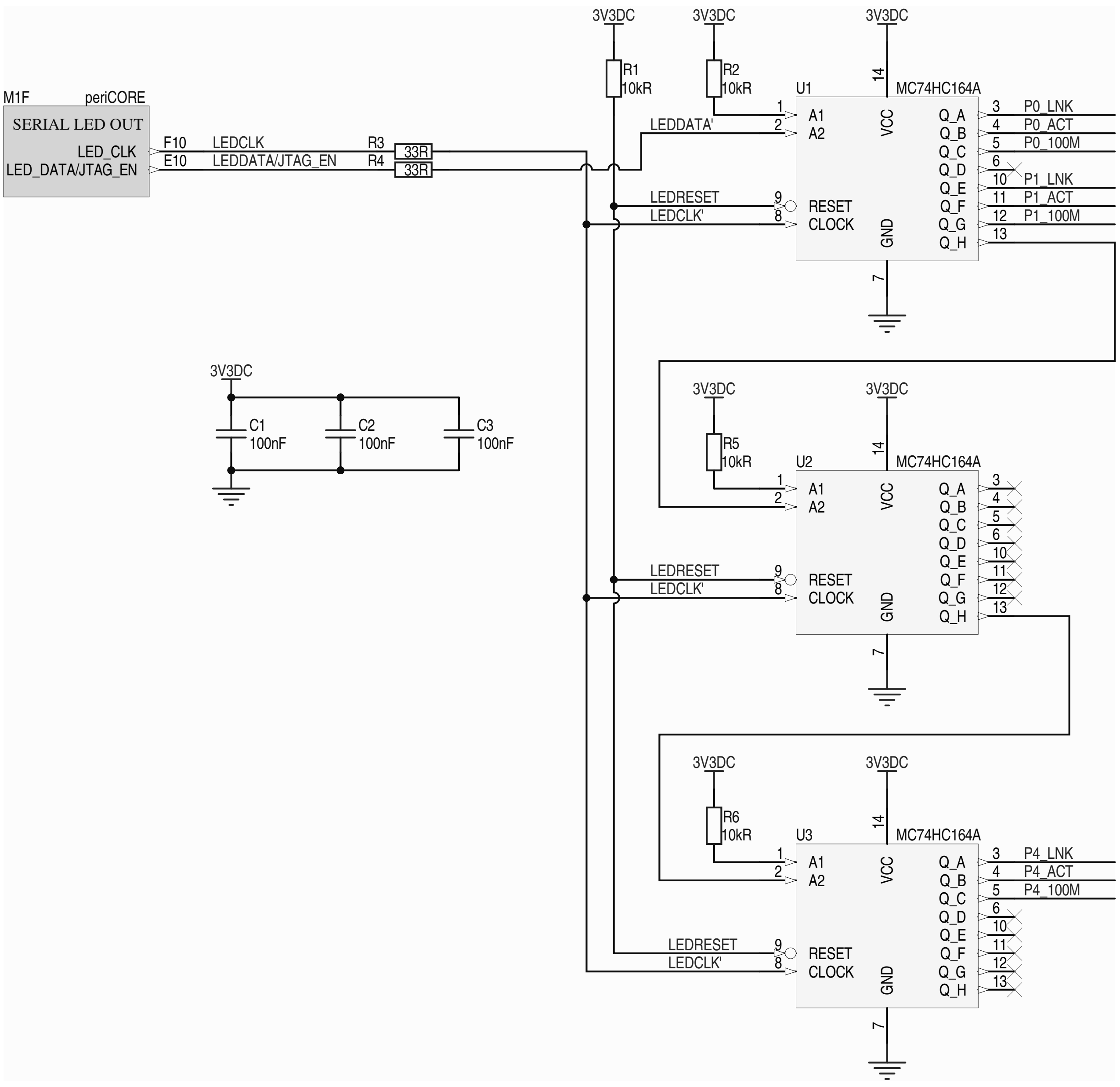


Figure 7: Serial LED interface typical application circuit

The circuit designed for the Serial LED Interface consists of three 8-bit shift registers linked in series. This setup allows the circuit to process the entire 24-bit stream from the interface. The link status bits provided by these shift registers can be utilized to create various LED indicators and blinking patterns, depending on specific application needs.

An example schematic is shown in Figure 8. This example demonstrates a setup with a green LED (D1) and a yellow LED (D2) programmed to behave as follows:

- D1 lights up when a network link is established and turns off otherwise, while D2 remains off.
- During network activity, both LEDs blink alternately - D1 turns on when D2 is off, and

vice versa.

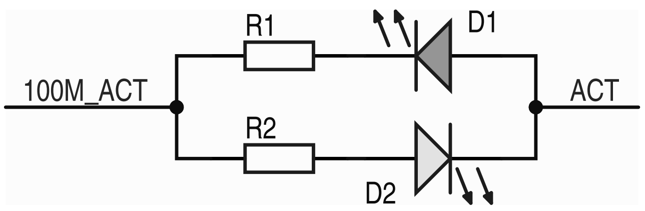


Figure 8: Link Status LED implementation example

2.2 Peripheral Interfaces

2.2.1 I2C

The Inter-Integrated Circuit (I^2C) interface in periCORE is a serial bus system using two wires: SDA (Serial Data Line) and SCL (Serial Clock Line). I^2C is ideal for communication with various peripheral devices such as serial EEPROMs, A/D & D/A Converters, and different sensors. For more details on I^2C, see the I^2C-bus specification [19].

The periCORE supports the I2C interface with these characteristics:

- Operates in Fast-mode with a fixed bit rate of 400 kbit/s ± 20 %.
- Functions in Master mode within a single master bus architecture.
- Utilizes 7-Bit device addressing.

An example application of this interface is shown below. It illustrates a 0-10 V sensor interface with M12 4-pin connector, periCORE, and an external ADC. The analog signal from pin 4 of the M12 connector is first attenuated and filtered through resistors R_1, R_2, and capacitor C_1. The ADC then converts this attenuated analog signal into digital samples. periCORE reads these digital samples from the ADC via the I^2C interface and makes them available over the network. Power for the sensor is supplied through pins 1 and 3 of the M12 connector. Diodes D_1 and D_2 provide ESD protection and should be positioned close to the connector pins.

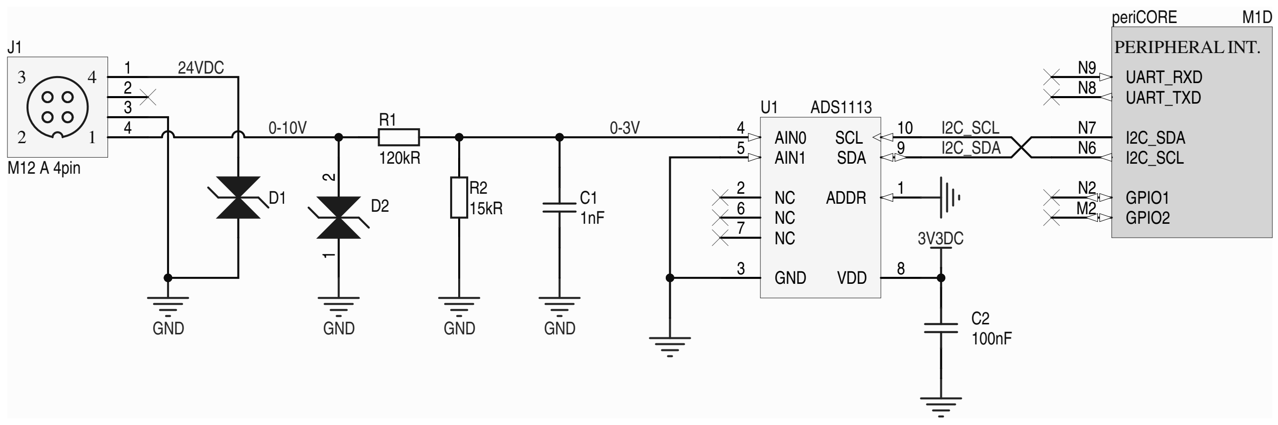


Figure 9: 0-10 V implementation example with I^2C

2.2.2 GPIO

The periCORE module features up to two General-Purpose I/O (GPIO) pins.

These GPIO pins can be set up either as digital outputs or digital inputs, depending on the requirement. Furthermore, they can function bidirectionally by alternating between output and input modes through software configuration.

2.2.3 UART

UART, or Universal Asynchronous Receiver-Transmitter, is a hardware component utilized for asynchronous serial communication. This process involves transmitting data bits sequentially over a single wire. Each communication sequence begins with a start bit, followed by the data bits sent with the least significant bit (LSB) first. This may be followed by an optional parity bit and concludes with at least one stop bit. A common communication frame structure for UART is illustrated below:

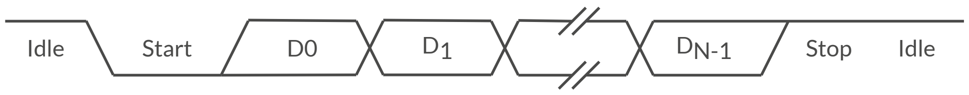


Figure 10: UART frame structure

The periCORE module includes support for a single UART, which consists of two I/O pins: TXD for transmitting data and RXD for receiving data. Data transmission on the TXD pin occurs with the least significant bit (LSB) sent first. RXD is an input pin where data is received. Hardware flow control is not supported by periCORE; if needed, it must be managed through software methods like XON/XOFF[1].

The default configuration for UART on periCORE is as follows:

- Baud rate: 115200 Baud
- 1 start bit
- 1 stop bit
- No parity

Additionally, the UART interface of the periCORE modules can support speeds up to 1.5 MBaud.

2.3 Power Block

2.3.1 Power Input

The *periCORE* communcation module module is equipped with a step-down DC-DC converter that powers all essential functions of the module. Details on the input voltage range, current consumption, and other electrical parameters are provided in Section 4.3.

[1]Software flow control is application specific and is not included in the *libperiCORE* package.

2.3.2 Power Output

The periCORE module also allows external peripheral circuits to utilize its internal DC-DC converter, providing a 3.3VDC output available at the `3V3_OUT` pin. This output includes over-current and short circuit protection. In the event of a short circuit, the 3.3V output will drop, rendering the periCORE non-functional until the short-circuit issue is resolved and the 3.3V supply is restored. Further details on the maximum output current and other electrical characteristics of the 3.3V output can be found in Section 4.3.

2.4 Controller Block

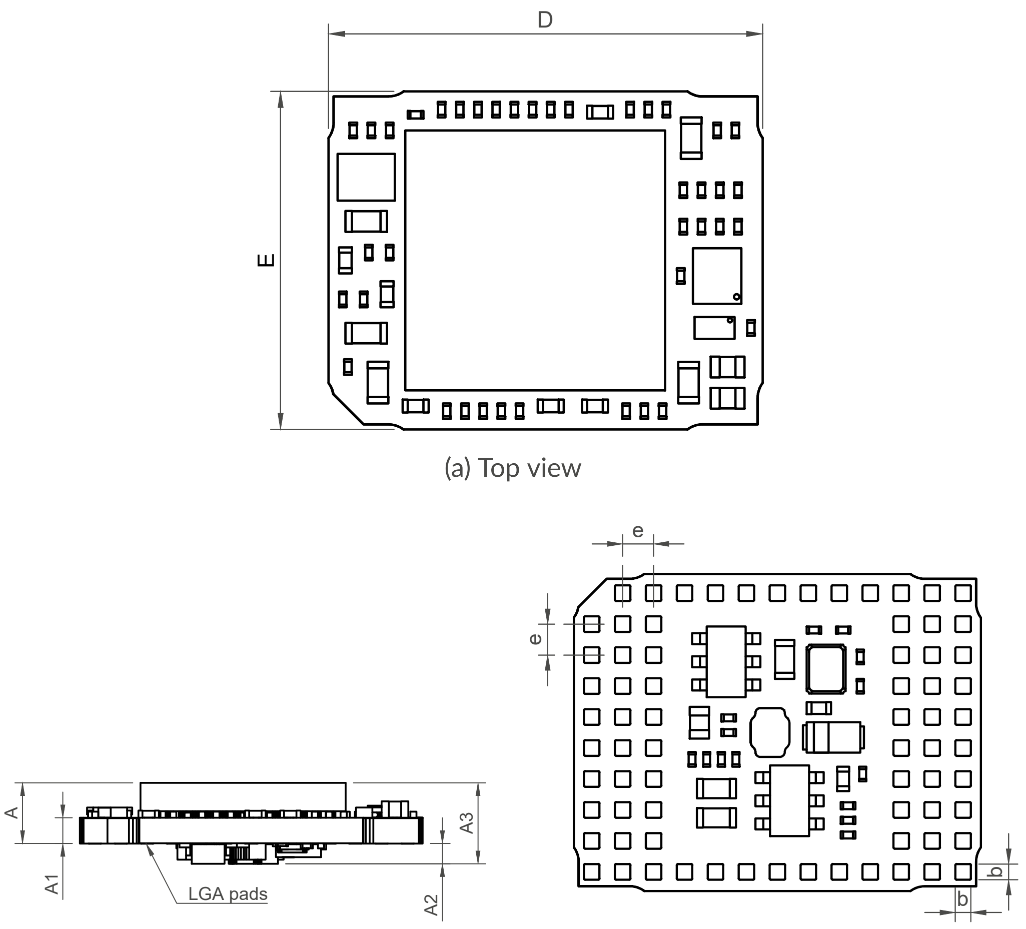
The firmware of the periCORE module operates on an ARMv7 MCU core. It utilizes a 32-MBit Flash memory for storing the periCORE firmware and other auxiliary data. The MCU operates at a clock frequency of 250 MHz and has 512 kb of available RAM.

2.5 Watchdog Block

The periCORE module includes a watchdog timer, designed to help the MCU recover from fault conditions. If the MCU fails to respond to the watchdog timer within a 5-second interval, the watchdog triggers a reset of the MCU. This 5-second detection window is fixed and cannot be altered.

3 Mechanical Specification

3.1 Module Dimensions



(a) Top view

(b) Side view

(c) Bottom view

Figure 11: Mechanical dimensions

Symbol	Min.	Typ.	Max.
A1	1.1	1.3	1.4
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
b		0.635	
e		1.27 BSC	

Table 3: periCORE dimensions (in mm)

3.2 Recommended Footprint

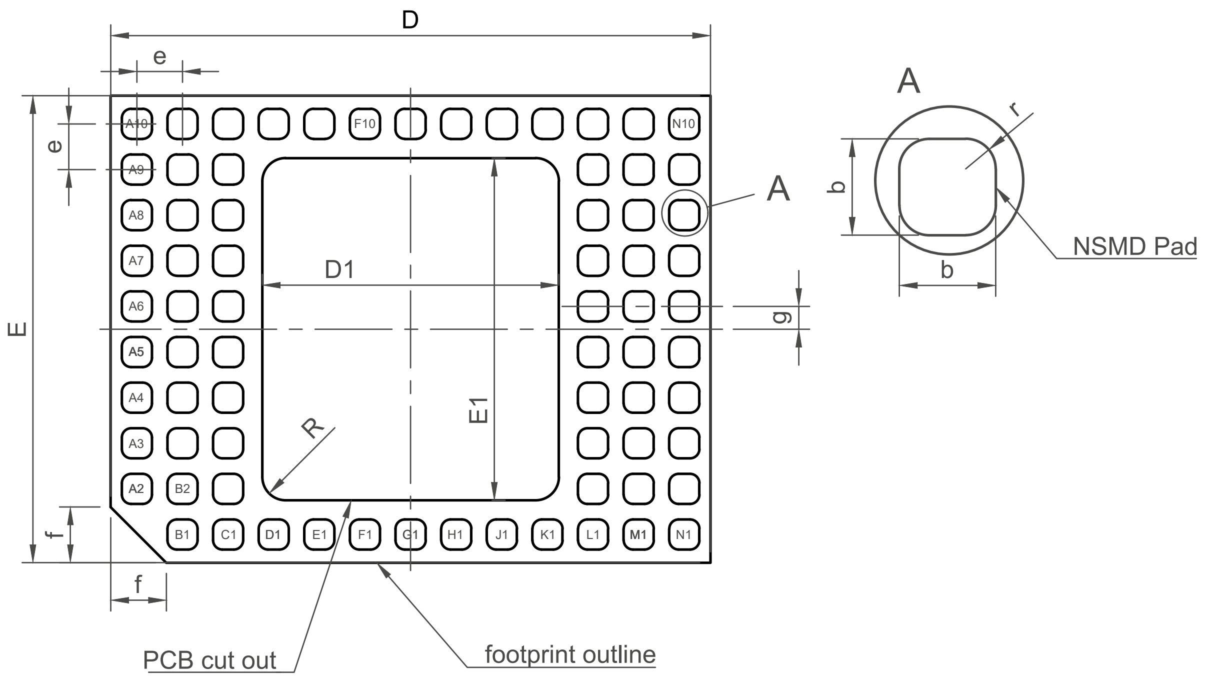


Figure 12: Recommended footprint (top view) for the periCORE module.

Symbol	Min.	Typ.	Max.
D	–	16.7	–
D1	7.9	–	8.3
E	–	13.0	–
E1	9.3	–	9.7
R	–	–	1.0
f	–	1.6	–

(a) Framing

Symbol	Value
r	50 %
b	0.635
e	1.27 BSC
g	0.635

(b) Pads

Table 4: periCORE footprint dimensions in mm.

All pads should be NSMD (Non Solder Mask Defined).

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		Connect to GND
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	100	mA
Storage temperature	-40	+85	°C
`A, D`	-0.5	+3.63	V

Table 7: Absolute maximum ratings

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

The periCORE module includes basic ESD (Electrostatic Discharge) protection features. However, to prevent electrostatic damage, especially to the MOS (Metal-Oxide-Semiconductor) gates, it is recommended to store and handle the device in conductive foam. This precaution helps safeguard the module from potential electrostatic damage.

4.4 Operating Conditions

Parameter	Min	Typ.	Max	Unit
Supply voltage (`24VDC_IN`)	10	24	27	V
Current consumption (`24VDC_IN` = 24V; I_{3V3_OUT} = 0)	—	30	—	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	Falling	2.3	—	—	V
V_{UVDT}	Under voltage detection threshold	Rising	—	—	2.9	V
C_{IN}	Input capacitance. [2]		—	10	—	µF
3V3_OUT						
ΔV_{p-p}	Voltage ripple	`24VDC_IN` = 27V; I_{3V3_OUT} = 100mA; C_L = 400 µF	—	—	30	mV

Symbol	Parameter	Conditions	Min	Typ.	Max	Unit
I_{3V3_OUT}	Output current		–	–	100	mA
C_L	Capacitive load on 3V3_OUT		–	–	400	µF
GPIO						
I_o	Output current		–	–	2	mA
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
R_{PD}	Pull-down resistance		40	56	106	kΩ
I^{2}C						
I_o	Output current		–	–	2	mA
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
R_{PD}	Pull-up resistance		–	4.7	–	kΩ
C_b	Capacitive load on each I2C_SDA and I2C_SCL line		–	–	75	pF
UART						
I_o	Output current	UART_TX	–	–	4	mA
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
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						
I_o	Output current		–	–	2	mA
R_{PD}	Pull-down resistance		40	56	106	kΩ
JTAG Interface						
I_o	Output current	JTAG_TDO	–	–	2	mA
V_{IH}	High-Level input voltage	All except JTAG_TDO	2.0	–	–	V
V_{IL}	Low-Level input voltage	All except JTAG_TDO	–	–	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
R_{PD}	Pull-down resistance	All except JTAG_TRST_B	40	56	106	kΩ

Symbol	Parameter	Conditions	Min	Typ.	Max	Unit
R_{PD}	Pull-down resistance	JTAG_TRST_B	—	4.7	—	kΩ
DEBUG_EN, nRESET						
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

[2]The inrush current is limited by the external resistance.

5 Timing Specification

5.1 Power On/Off and Reset Sequence

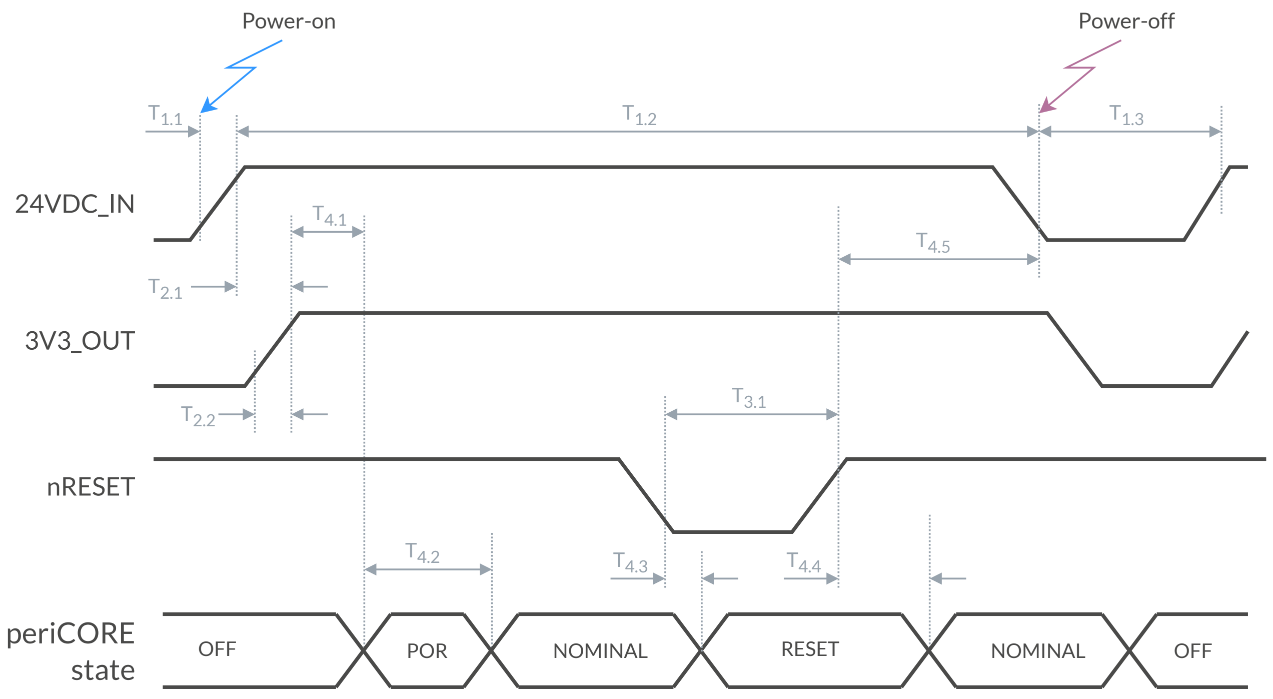


Figure 13: Power On/Off and Reset timing diagram.

Symbol	Parameter	Min	Typ.	Max	Unit
$T_{1.1}$	24VDC_IN slew rate [3]	–	–	8	V/ms
$T_{1.2}$	Period after Power-on in which power supply (24VDC_IN) must not be interrupted [4].	310	–	–	ms
$T_{1.3}$	Time between Power-off and the next Power-on [5]	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[6]	–	–	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 assertion to entering RESET state	–	–	8	µs
$T_{4.4}$	Delay from nRESET deassertion to entering NOMINAL mode	–	310	–	ms
$T_{4.5}$	Period after nRESET deassertion in which power supply must not be interrupted[4]	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.

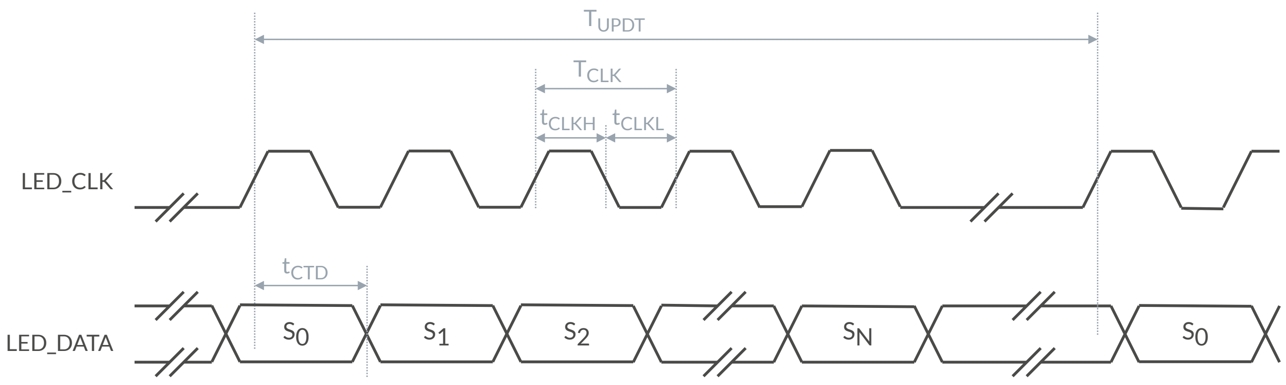


Figure 14: Serial LED Interface Timing Diagram

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

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

5.3 JTAG Interface

Timing specification of the JTAG Interface.

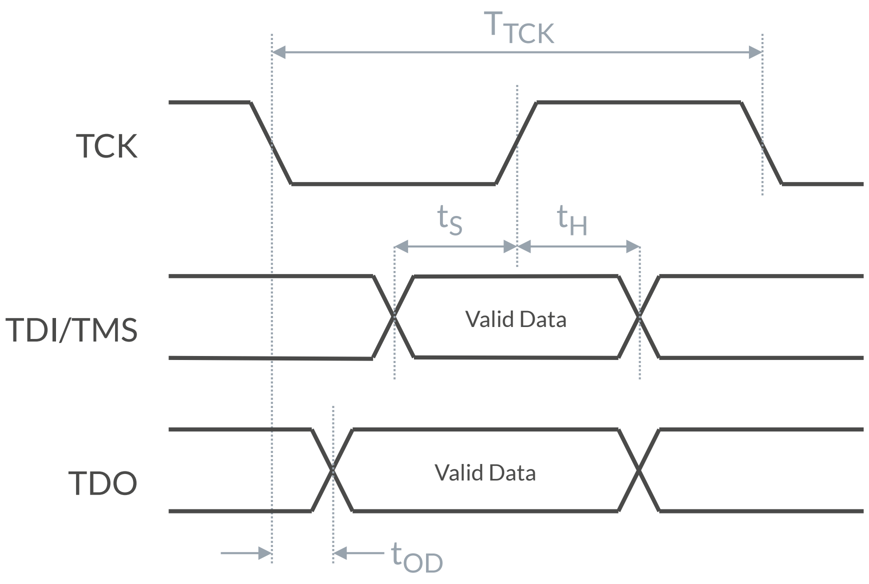


Figure 15: JTAG Interface Timing Diagram

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 can be reset to factory settings using two different methods. One method is through the RESTful API, as detailed in Section 7.3.6. The second method is a physical reset, which is necessary if RESTful API access is lost, such as in cases where the admin mTLS certificate is unavailable. The foundational *libperiCORE* library facilitates the factory reset process. To physically initiate a factory reset, ensure that the periCORE based product is powered on (using `24VDC_IN`) but is not connected to any network through its network ports (Port 0, Port 1, and Port 4). The factory reset will automatically occur 20 seconds after the device is powered up, returning the device to its original factory settings. For more details on the Factory Reset process, refer to Section 7.3.6.

7 Firmware

The periCORE module is primarily intended for integration into OEM customer products. As a white-label product, its firmware architecture is designed to support enhancements and customizations by OEM customers. Perinet facilitates this by providing the necessary software development infrastructure, including the *periCORE Development Kit* [7] and an example application [8]. These resources demonstrate how to rebrand and adapt a periCORE-based product to meet specific customer needs in terms of functionality and appearance (refer to Section 7.8).

The firmware's source code is written in standard C++20 [16] and operates on a bare metal software environment. The minimal and simplified operating system (OS) [10] and the *libperiCORE* form the foundation of the firmware, both provided by Perinet. These components allow developers to focus on application development while implementing a *periCORE-based Custom Application*.

This section outlines the basic architecture and functional behavior of the periCORE firmware, along with its production handling and rebranding options.

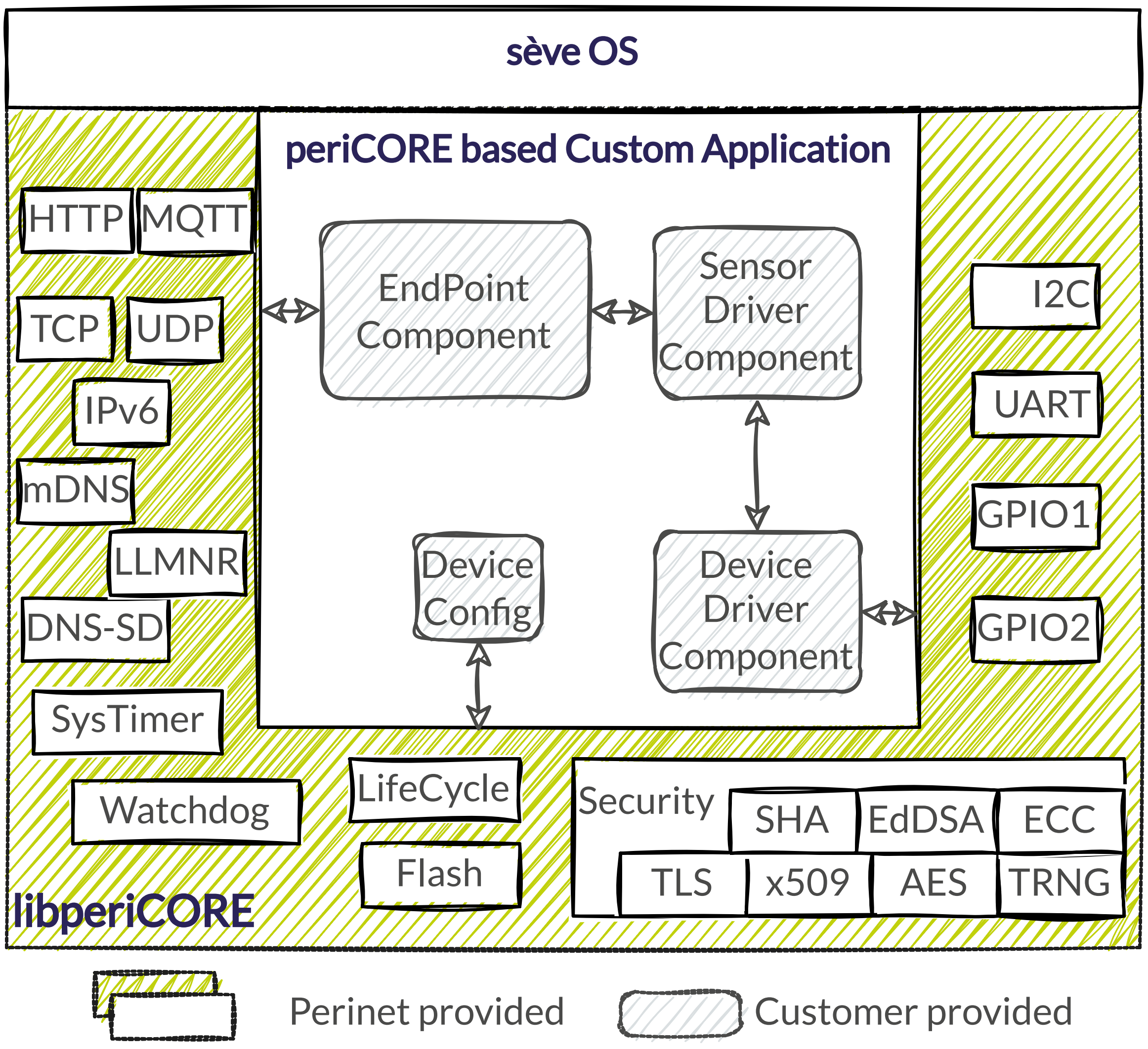


Figure 16: General periCORE based firmware architecture.

7.1 Firmware Architecture

The firmware for a periCORE-based product comprises three main components: the *sève OS*, *libperiCORE*, and the *periCORE based Custom Application*, as illustrated in Figure 16.

periCORE based Custom Application This layer contains the application-specific software that manages and communicates with platform-specific hardware. It also includes software for converting raw sensor data into meaningful metrics, such as translating a 0-10V signal into a distance measurement, as detailed in the periCORE Firmware Development Application Note [8]. Applications are developed in standard C++, utilizing the sève OS and Perinet's *libperiCORE*.

libperiCORE *libperiCORE* provides a comprehensive set of features (including security, product life cycle functions, network protocols, and hardware-dependent bus drivers) as a software library. It offers a standard C++20 compliant interface, intended for use in all periCORE firmware variants. More information on this library can be found in Section 7.7.

sève OS The sève OS serves as the foundational operating system for both the *periCORE based Custom Application* and *libperiCORE*. It features a component-based architecture, facilitating well-structured application design. Components communicate exclusively through event channels, which are protected against race conditions. These channels function as part of a dataflow model, conveying both event occurrence and associated data. Activities within components are triggered by events and are executed cooperatively following a FIFO scheme with a run-to-completion approach. The sève OS is a Perinet technology, with further details available in the periCORE sève Operating System Datasheet [11].

7.2 Persistent Memory Structure

The persistent memory in a periCORE-based product is organized into various logical segments. These segments play a crucial role in representing the current state of the product's life cycle. A comprehensive overview of these memory segments is provided in Figure 17.

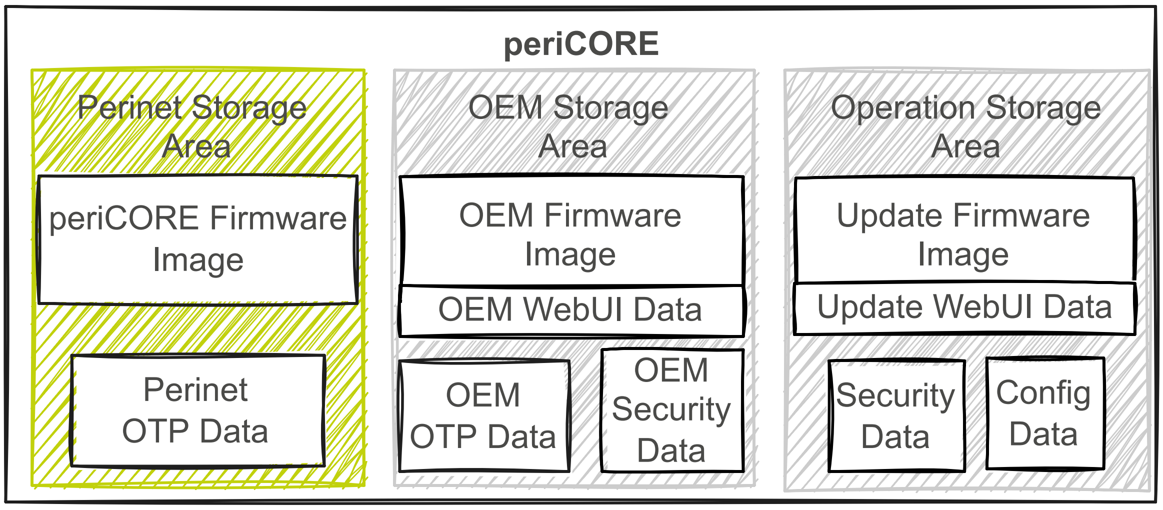


Figure 17: Structure of the persistent memory of a periCORE based product.

The periCORE module has three distinct logical storage areas for different life states of a periCORE based product: the *Perinet Storage Area*, *OEM Storage Area*, and *Operation Storage Area*. These areas correspond to the *periCORE Production state*, *OEM Production state*, and *Updated State*, respectively.

Perinet Storage Area In the manufacturing phase of the periCORE module, production-specific metadata (refer to Section 9.2) is saved in the OTP of this area. Additionally, the general *periCORE Firmware Image* is also stored here.

This memory area remains unchanged throughout the life of a periCORE module.

OEM Storage Area This segment holds the factory default data for the OEM customer, set during manufacturing. In case of a factory reset (see Figure 18), data from this segment is restored.

It also contains the device attestation certificate (see Section 7.5).

Changes to this memory area are not possible during the regular operation 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).

Operation Storage Area Persistent modifications to a periCORE based product, including the *Update Firmware Image*, runtime security data, and configuration settings, are saved in this area.

Data here is persistent across power cycles but is erased during a *Factory Reset* (refer to Listing 3).

7.3 Product Life Cycle

Figure 18 provides a summary of the various states and state transitions of a periCORE based product.

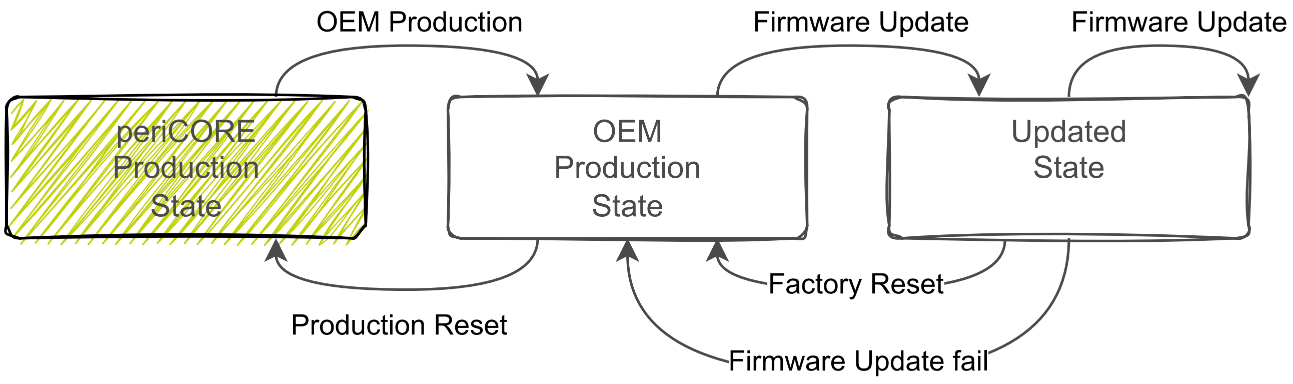


Figure 18: The Different life states of a periCORE based product.

7.3.1 periCORE Production State

When a periCORE module leaves the Perinet manufacturing facility, it is in the *periCORE Production State*. In this state, specific information is stored in the *Perinet Storage Area*, including:

1. The *periCORE Firmware Image*, designed primarily for production use.
2. Unique production-specific attributes such as MAC-Address, serial number, and production date (detailed in Section 9.2).
3. Unique attestation certificate information, confirming the module's authenticity and accreditation by Perinet manufacturing services.

Note: The general-purpose *periCORE Firmware Image* is intended for production scenarios only. Perinet advises against using this firmware image in end-customer applications.

The *periCORE Firmware Image* offers the following features for OEM manufacturing:

- Authentication of an authentic periCORE module, certified by Perinet (refer to Section 7.5).
- Deployment of Security Information, including X.509 certificates.
- Installation of the *OEM firmware image*.
- Network services access to all peripherals.
- Various networking capabilities, including IPv6, DNS-SD, MDNS, HTTP, TLS, MQTT, and more.

Note: Attestation information for the periCORE module is erased during the *OEM Production* process.

7.3.2 OEM Production State

The *OEM Production State* is the state in which a periCORE-based product is delivered. During the *OEM Production* process, the following key attributes are stored:

1. Authentication information specific to the product (details in Section 7.5).
2. The *OEM Firmware Image*.
3. Product-specific device information such as product name, host name, and serial number (see Section 9.2).
4. Configuration values unique to the product, like sensor calibration data.

Note: By default, no Role-Based Access Control (RBAC) is active in this state, meaning all configurations and security settings are accessible and modifiable without client authentication. Implementing and activating RBAC is an additional step that must be taken.

7.3.3 Updated State

In the *Updated State*, a periCORE-based product operates using the *Update Firmware Image*. After a *Firmware Update* is successfully completed, the product will boot from the *Update Firmware Image* going forward.

To revert to the OEM Production State, a Factory Reset must be explicitly performed (details in Section 7.3.6).

Note: If a Firmware Update procedure fails, for instance due to a network disruption or corrupted Update Firmware Image, the periCORE-based product will transition to the *OEM Production State.*

7.3.4 Firmware Update

The Firmware Update process for a periCORE-based device can be initiated only when the device is in either the *OEM Production State* or the *Updated State*. Successful completion of this update will transition the device into the *Updated State*. The update is initiated through the RESTful API, which is detailed in Section 7.4. Notably, the update process is designed to be idempotent, meaning it can be safely repeated without changing the result beyond the initial application.

```
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 facilitates the exchange of the *Update Firmware Image* within the *Operation Storage Area* of the periCORE module (refer to Figure 17). It's important to note that this exchange does not immediately affect the firmware image currently in operation. This is true even when the Firmware Update is carried out from the *Updated State*.

To activate and run the new *Update Firmware Image*, a reboot of the periCORE module is required. This reboot can be triggered either through a power cycle or remotely via the RESTful API (details in 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.

During a Firmware Update, there are no alterations made to the data stored in the *Security Data* and *Config Data* segments of the periCORE module's persistent memory (as detailed in Figure 17). If a reset of these sections is required, it must be done manually. This can be accomplished through specific commands available via the RESTful API, further described in 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: If a Firmware Update process is unsuccessful, for instance, due to a network disruption or a corrupted Update Firmware Image, it will result in a change of the device's operational state. Following such a failure, the periCORE based product will revert to the *OEM Production State*.

7.3.5 OEM Production

To transition a periCORE module to the *OEM Production State*, the following procedures should be completed:

1. Authenticate the periCORE module, as outlined in Section 7.5.
2. Install the product-specific *OEM Firmware Image*.
3. Implement device-specific *OEM Security Data*.
4. Input product-specific *OEM OTP Data*.
5. Reboot the device.

Note: These steps for the OEM Production process can only be executed from the *periCORE Production State*.

periCORE Authentication

Device attestation in the periCORE module occurs automatically with each RESTful API request, as detailed in Section 7.4. For information on explicit device authentication procedures, refer to Section 7.5.

Note: Product-specific authentication information is removed during the *OEM Production* process.

Deploy *OEM Firmware Image*

The process of deploying the *OEM Firmware Image* is carried out through a RESTful API call, as described in Section 7.4. This firmware image (detailed in Section 7.4.2) is uploaded using a *PUT* request. Below is an example demonstrating how to upload 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: Deploying the *OEM Firmware Image* invalidates the *OEM Security Data*, which encompasses the periCORE authentication information. As a result, a periCORE-based product will lose its ability to authenticate uniquely. Therefore, it is highly recommended to load appropriate *OEM Security Data* onto the device after completing 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

Performing a Factory Reset on a periCORE-based product clears the entire *Operational Storage Area*, returning the product to the *OEM Production State*. The reset process affects the following properties:

***Config Data* reset:** All persistent runtime configuration settings are reverted to their default firmware values. This includes network settings like `mqtt_broker_name` or `application_name`.

***Security Data* reset:** All security settings are reset to the default *OEM Security Data*. This disables mTLS and, consequently, client authentication (RBAC) as detailed in Section 7.6.

***Update Firmware Image* invalidation:** The *Update Firmware Image* stored in the *Operational Storage Area* is invalidated. Upon the next boot, the periCORE-based product will run the *OEM Firmware Image*.

A *Factory Reset* can be initiated through the RESTful API (see Section 7.4), as demonstrated in the example below, or via a *Physical Factory Reset* (refer to Section 6).

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

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

Note: Any modifications made to the *Security Data* and *Config Data* will only become effective after the device is rebooted. Given that all security settings are reset to their defaults during a factory reset, it is strongly advised to reconfigure the security settings post-reset for optimal security.

7.3.7 Production Reset

The *Production Reset* process transitions the device back to the *periCORE Production State*, where it operates using the general-purpose *periCORE Firmware Image*. This reset option is accessible for 60 seconds during the first boot following a *Physical Factory Reset* (details in Section 6).

Initiating the *Production Reset* does not erase the *OEM Firmware Image* or the OEM Data (including *OEM Security Data* and *OEM OTP Data*). The process to activate the *Production Reset* via the RESTful API is illustrated in the following excerpt (for more information, see Section 7.4).

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

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

Note: Performing the *Production Reset* will permanently invalidate both the *OEM Firmware Image* and the OEM Data (which includes *OEM Security Data* and *OEM OTP Data*). These changes will take effect after the periCORE-based product is rebooted.

7.4 RESTful API

A periCORE based product features a RESTful [6] API for user access. To connect, use the hostname `periCORE-`*`sernm`*`.local`, where *`sernm`* is a unique identifier based on the product's serial number as explained in Section 9. The API provides various routes, which are listed in Table 13. These routes and their functions will be detailed in the subsequent 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 Life Cycle Service of a periCORE based product requires the *Firmware Image* to be in a signed binary format. For more comprehensive details on this format and related procedures, refer to the periCORE Firmware Development Application Note [8].

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, along with all products based on it, features a built-in authentication mechanism. This authentication occurs automatically during access to the RESTful API via HTTP (detailed in Section 7.4). The secure TLS protocol used for this access ensures that the X.509 certificate received during the connection is employed to authenticate the device.

For device authentication, the periCORE uses Perinet's Root CA certificate (*perinet-ecc-root-ca.crt* [9]), serving as the trust anchor.

7.6 Security

Certificate Type	Description
Root Certificate	Identifies the root CA trusted by periCORE. It authenticates remote clients before their connection to periCORE when mTLS is enabled.
Host Certificate	A unique certificate that the periCORE uses to authenticate itself during communications with a remote client via HTTP Server. It includes the host name and is issued by a trusted root CA, verifying the device's originality.
Client Certificate	Enables the periCORE to authenticate itself as a client when connecting to remote servers. It is applicable in scenarios like connecting to an MQTT broker.

Table 14: Security Resources

Host and client certificates in the periCORE module are linked to their respective private keys. While using a `GET` method to retrieve these certificates does not reveal the private keys, updating them with a `PATCH` request requires the inclusion of the private key in the certificate file.

Below is an example showing a valid format of an *X.509 base64-encoded* certificate, which is concatenated with its private key:

```
-----BEGIN CERTIFICATE-----
MIICGjCCAb+gAwIBAgIRAMsuh27hf3EpZ5qTfkKDOpwwCgYIKoZIzj0EAwIwNjEa
MBgGA1UECgwRUEtJMmdvLW1pY2EtZ2t2bTkxGDAWBgNVBAMMD2V4YW1wbGUgUm9v
dCBDQTAeFw0yMjA3MjgxMjI3MzVaFw0yNDEwMzAxMjI3MzVaMDkxGjAYBgNVBAoM
EVBLSTJnby1taWNhLWdrdm05MRswGQYDVQQDDBJleGFtcGxlLWhvc3QubG9jYWww
WTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAQ13a55JkWRUGW2o30bU9Zqhw2e3gLn
wL0R6DOsyF118UITSyYjLuziahGaXWtqcx3YvuetBkWNe5/oYxuAmRQRo4GqMIGn
MA4GA1UdDwEB/wQEAwIFoDAJBgNVHRMEAjAAMB0GA1UdJQQWMBQGCCsGAQUFBwMB
BggrBgEFBQcDAjAdBgNVHQ4EFgQUO0UNscTx9QJG7JKks5gz+OCoDuQwHwYDVR0j
BBgwFoAUKLFmRdg8YvbYB5fNcBrsuJUh4h8wKwYDVR0RBCQwIoISZXhhbXBsZS1o
b3N0LmxvY2FsggxleGFtcGxlLWhvc3QwCgYIKoZIzj0EAwIDSQAwRgIhALqaMQpQ
E9h4voRoWtbLRweMpe82r+RIC4KB/RgIyrJvAiEArxjb08uoy4Q/pYn2CFrmh2T/
L95qNLuMNba9aE9n9EY=
-----END CERTIFICATE-----
-----BEGIN EC PARAMETERS-----
BggqhkjOPQMBBw==
-----END EC PARAMETERS-----
-----BEGIN EC PRIVATE KEY-----
MHcCAQEEIMG1qiAmJAgpAOBq+mJkAtNIyJH6qhd3GCrOTiHeuv12oAoGCCqGSM49
AwEHoUQDQgAENd2ueSZFkVBltqN9G1PWaocNnt4C58C9EegzrMhddfFJU0smIy7s
4moRml1ranMd2L7nrQZFjXuf6GMbgJkUEQ==
-----END EC PRIVATE KEY-----
```

The example below demonstrates how to update the *host certificate* using the `curl` command line tool. This method is also applicable for updating the root and client certificates. To do so, simply adjust the URL and the file in the command as needed:

```
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 module requires the use of TLS security, which is always enabled. Clients, such as web browsers or command line tools like `curl`, can bypass the server's (periCORE's) CA check if necessary. While this approach maintains encrypted connections, it's useful in scenarios where the user can guarantee trust, like when periCORE is the only device on the network. The base URL for accessing periCORE always follows this format: `https://periCORE-sernm.local/`.

The *root certificate* refers to the public certificate of a trusted authority, usually the Perinet ECC Root CA by default. It is provided as a single X.509 base64-encoded certificate, excluding the private key.

A periCORE module can be configured to enable *mTLS* along with *Role Based Access Control (RBAC)*. When mTLS is activated, a remote client must authenticate itself using a client certificate before establishing a connection. This certificate is verified against the stored root certificate and is expected to be signed by it. Additionally, the client certificate specifies a user role, determining the level of access.

The periCORE supports the following user roles:

***admin*:** Grants full read/write access without any restrictions.

***super*:** Allows reading and writing to all resources except */security* and */update*.

***reader*:** Provides read-only access, with no write permissions.

Activating *mTLS/RBAC* on the periCORE's HTTP server requires careful management of valid certificates. Without properly deployed certificates, or if a client fails to provide the correct signed certificate, the HTTP server may become inaccessible. Any connection attempts with an invalid client certificate typically result in a **401 Unauthorized** error response.

To check the status of *mTLS/RBAC*, you can use the `/security` resource. This resource provides a JSON-encoded object with the parameter `enable_user_role`, indicating whether mTLS/RBAC is enabled. An example of how to retrieve this information is provided below:

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

To activate mTLS/RBAC on the periCORE using the RESTful API, you can follow the command example provided below:

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

To reset the security configuration on the periCORE, refer to the command example provided below:

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

When the periCORE becomes inaccessible, for instance, due to a lost *client certificate*, refer to the factory reset procedure in Section 6 for resolution.

Note that changes to the security configuration are persistent. However, they only take effect after rebooting the device. Rebooting can be accomplished through the RESTful API, as detailed in Section 7.4, or by performing a power cycle.

7.6.1 Deployment of Client Certificates for periCORE

The periCORE's RBAC system utilizes the capability to embed strings in x.509 certificate version 3 using arbitrary extensions, as detailed in [18]. These extensions are crucial for periCORE to determine the `Role` assigned to each certificate, which in turn controls access to various operations.

Listing 12 showcases the configuration for these extensions. This configuration can be included in the CA configuration file used during the signing of the certificate sign request (CSR).

Below is the configuration for the three supported extensions. This configuration can be incorporated into the CA configuration file and applied when signing new certificates.

```
[ reader_user_ext ]
keyUsage                = critical,digitalSignature,keyEncipherment
basicConstraints        = CA:false
extendedKeyUsage        = clientAuth
1.2.3.001               = ASN1:UTF8String:reader_user

[ super_user_ext ]
keyUsage                = critical,digitalSignature,keyEncipherment
basicConstraints        = CA:false
extendedKeyUsage        = clientAuth
1.2.3.011               = ASN1:UTF8String:super_user

[ admin_user_ext ]
keyUsage                = critical,digitalSignature,keyEncipherment
basicConstraints        = CA:false
extendedKeyUsage        = clientAuth
1.2.3.111               = ASN1:UTF8String:admin_user
```

Listing 12: Configuration of extension for RBAC

To effectively deploy client certificates with these arbitrary extensions, it is recommended to use the *-extension* parameter in conjunction with the desired extension tag: *reader_user_ext*, *super_user_ext*, or *admin_user_ext*. The example command below demonstrates how to use the *admin_user_ext* extension tag with `openssl`:

```
 openssl ca -config ca-config-file -in csr-file -out crt-file -
extensions admin_user_ext
```

Listing 13: Example openssl command to sign and include admin_user extension

Once properly configured, the client certificate will be parsed as demonstrated in the example below:

```
    Certificate:
    Data:
      Version: 3 (0x2)
      Serial Number:
        cb:2e:87:6e:e1:7f:71:29:67:9a:93:7e:42:83:3a:9a
      Signature Algorithm: ecdsa-with-SHA256
      Issuer: O=PKI2go-mica-gkvm9, CN=example Root CA
      Validity
        Not Before: Jul 28 12:06:20 2022 GMT
        Not After : Oct 30 12:06:20 2024 GMT
      Subject: O=PKI2go-mica-gkvm9, CN=admin_pki_container
      Subject Public Key Info:
        Public Key Algorithm: id-ecPublicKey
          Public-Key: (256 bit)
          pub:
            04:7f:75:56:6d:33:6c:d5:70:a4:f9:64:3a:96:47:
            b6:10:30:91:ad:12:c8:bc:03:94:a6:5f:29:6e:f3:
            18:6b:2f:23:03:cf:3b:79:05:be:77:74:ba:16:33:
            1e:a0:0b:c3:e5:07:03:a3:74:b4:f1:b1:db:d7:cd:
            ea:61:e9:b7:cb
          ASN1 OID: prime256v1
          NIST CURVE: P-256
      X509v3 extensions:
        X509v3 Key Usage: critical
          Digital Signature, Key Encipherment
        X509v3 Basic Constraints:
          CA:FALSE
        X509v3 Extended Key Usage:
          TLS Web Client Authentication
        1.2.3.111:
          .
  admin_user
    Signature Algorithm: ecdsa-with-SHA256
       30:45:02:21:00:cc:08:15:e7:02:49:02:8e:88:af:ad:19:0e:
       a9:51:5c:f6:9c:64:6d:7a:aa:96:82:07:75:8b:da:f0:7b:5e:
       b6:02:20:29:7c:13:67:92:f0:41:cf:63:e7:71:2c:df:fd:1a:
       94:32:88:42:d1:17:88:85:78:cd:f0:5b:66:81:c3:e8:c6
```

Listing 14: Example Client Certificate Parsed

Table 15 provides a summary of the custom OIDs and their respective tags in ASN.1 format.

user	Custom OID	ASN.1 String Format
admin	1.2.3.111	UTF8String:admin_user
super	1.2.3.011	UTF8String:super_user
reader	1.2.3.001	UTF8String:reader_user

Table 15: ASN1 for RBAC certificate extension

7.7 *libperiCORE* Software Library

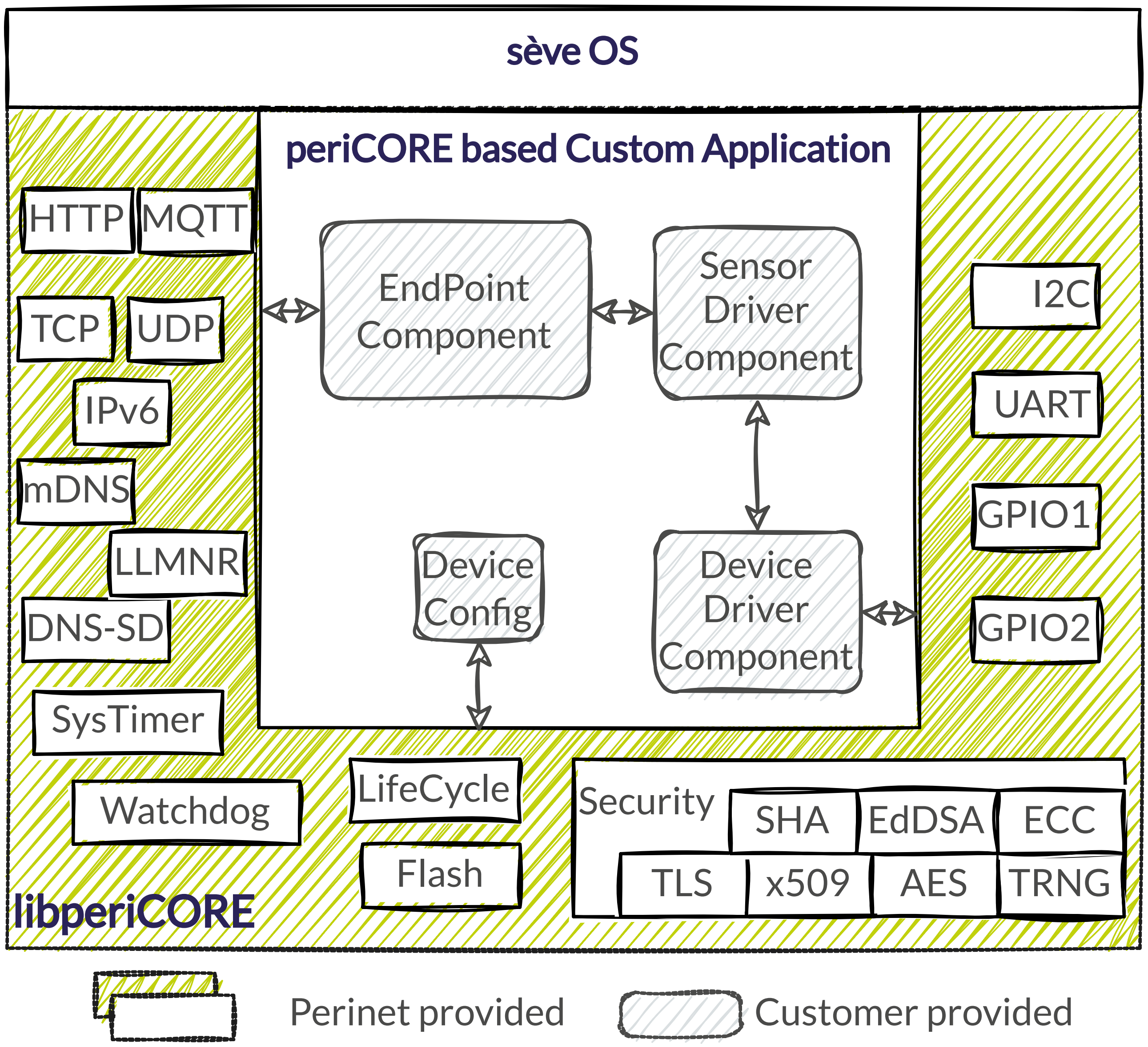


Figure 19: Overview of the *libperiCORE* structure.

7.8 Web User Interface

As illustrated in Figure 17, both the *OEM Storage Area* and the *Operation Storage Area* of the periCORE module include space allocated for a web user interface (WebUI).

When the periCORE based product is in *OEM Production State* or *Updated State* (referring to Figure 18), the WebUI can be accessed through 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`*).

The *periCORE SDK* includes an exemplary and customizable WebUI template. When compiling the firmware with the *update_image* target, the resulting binary will contain the WebUI. During the compilation, all `.css`, `.js`, and `.html` files are minified, and `.png` files are optimized.

Note: It is advisable to maintain low file sizes for the *WebUI Data*-segments, considering a maximum limit of 256kB.

Updating the firmware with the newly compiled binary via the RESTful API (refer to Section 7.3.4) writes the *WebUI Data*-segment into either the *OEM Storage Area* or the *Operation Storage Area*. A reboot is required for the firmware update to take effect and for the WebUI to become available.

Given the necessity for client systems to support DNS-SD or mDNS, currently recommended web browsers for accessing the `.local`-URL are:

- Windows Edge
- Safari

7.8.1 WebUI Template

The periCORE SDK includes an expandable template for a single-page web application. This template is designed primarily to facilitate the reading and writing of data via the periCORE's RESTful API (detailed in Section 7.4) through a modern and secure web interface.

This section outlines the structure of the WebUI template and provides guidance on how to customize it according to specific needs.

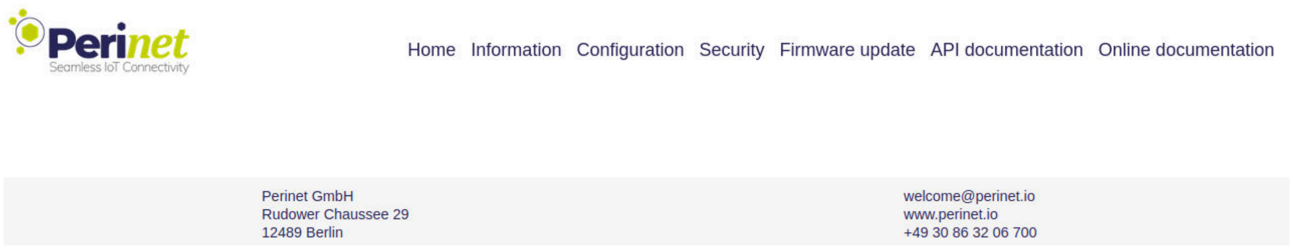
The template, by default, contains the following pages, each serving a distinct purpose:

Page	**Intended Purpose**
Home	Dashboard providing display elements for *sensors* and control elements for *actuators*.
Information	A view for periCORE based product information (refer to Section 7.4.1).
Configuration	Inputs for configuring periCORE based products (see Section 7.4.1).
Security	Inputs for configuring periCORE based security settings (refer to Section 7.4.3).
Firmware Update	Interface for uploading a new firmware image.
API documentation	Option to download the API documentation as an `api.proto`-file.
Online documentation	Link to the comprehensive online documentation.

Table 16: Security Resources

Example: Changing the Corporate Identity

Figure 20 displays a mock-up image of the website's layout, featuring a header and footer but excluding content.



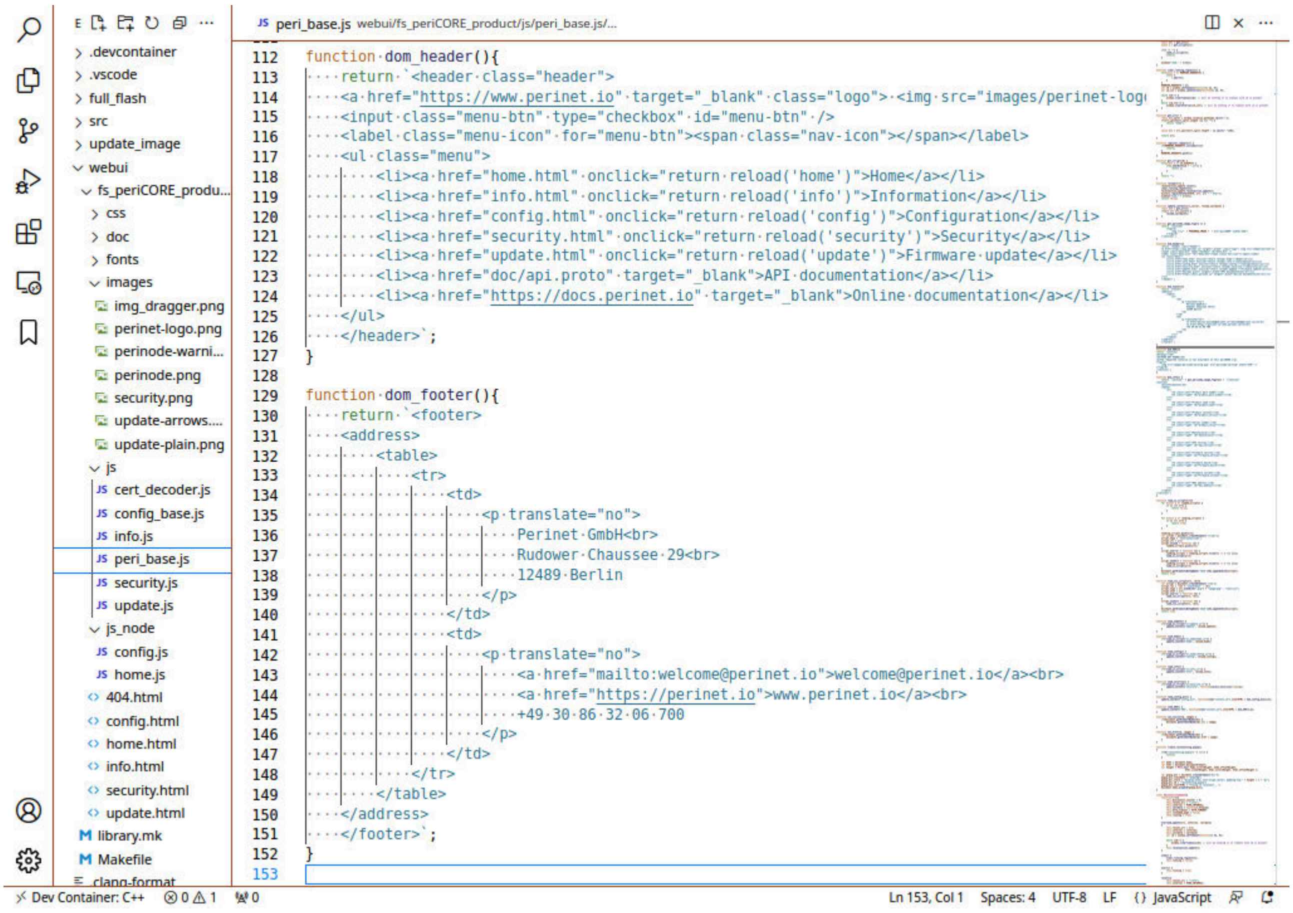
Perinet GmbH
Rudower Chaussee 29
12489 Berlin

welcome@perinet.io
www.perinet.io
+49 30 86 32 06 700

Figure 20: WebUI's corporate identity

In this scenario, alterations to the logo and footer information are required to reflect a change in corporate identity. To implement these modifications, the file `webui/fs_periCORE_product/js/peri_b` should be edited.

Figure 21 illustrates the periCORE SDK interface. It displays the structure of WebUI-related files in the directory tree on the left side, with the specific source file `peri_base.js` opened for editing.



```
function dom_header(){
    return `<header class="header">
    <a href="https://www.perinet.io" target="_blank" class="logo"> <img src="images/perinet-log
    <input class="menu-btn" type="checkbox" id="menu-btn" />
    <label class="menu-icon" for="menu-btn"><span class="nav-icon"></span></label>
    <ul class="menu">
        <li><a href="home.html" onclick="return reload('home')">Home</a></li>
        <li><a href="info.html" onclick="return reload('info')">Information</a></li>
        <li><a href="config.html" onclick="return reload('config')">Configuration</a></li>
        <li><a href="security.html" onclick="return reload('security')">Security</a></li>
        <li><a href="update.html" onclick="return reload('update')">Firmware update</a></li>
        <li><a href="doc/api.proto" target="_blank">API documentation</a></li>
        <li><a href="https://docs.perinet.io" target="_blank">Online documentation</a></li>
    </ul>
    </header>`;
}

function dom_footer(){
    return `<footer>
    <address>
        <table>
            <tr>
                <td>
                    <p translate="no">
                        Perinet GmbH<br>
                        Rudower Chaussee 29<br>
                        12489 Berlin
                    </p>
                </td>
                <td>
                    <p translate="no">
                        <a href="mailto:welcome@perinet.io">welcome@perinet.io</a><br>
                        <a href="https://perinet.io">www.perinet.io</a><br>
                        +49 30 86 32 06 700
                    </p>
                </td>
            </tr>
        </table>
    </address>
    </footer>`;
}

```

Figure 21: Source code for the WebUI's corporate identity in the periCORE SDK

To update the logo, access the *`dom_header()`* function (shown at line 112 in the figure). Here, the `src` attribute of the HTML image element is set to `images/perinet-logo.png` by default.

For modifying the footer's content, look into the *`dom_footer()`* function (line 129 in the figure) and adjust the HTML paragraphs as needed.

Example: Adding Information to the *Home* Page

The type of sensor or actuator, along with the implementation of the application-specific sample and config objects (see Section 7.4.1), dictates the selection of HTML elements. These elements can be chosen to best represent sensor values or display actuator status interactively.

Figure 22 presents the *Home* page of a periCORE-based product with actuator peripherals. In this example, the peripherals are two GPIO ports. *GPIO 1* is set as an output (actuator), and *GPIO 2* as an input (sensor). The goal is to provide an HTML input element to control the output *GPIO 1*. For the input *GPIO 2*, the same HTML input element is disabled for the time being.

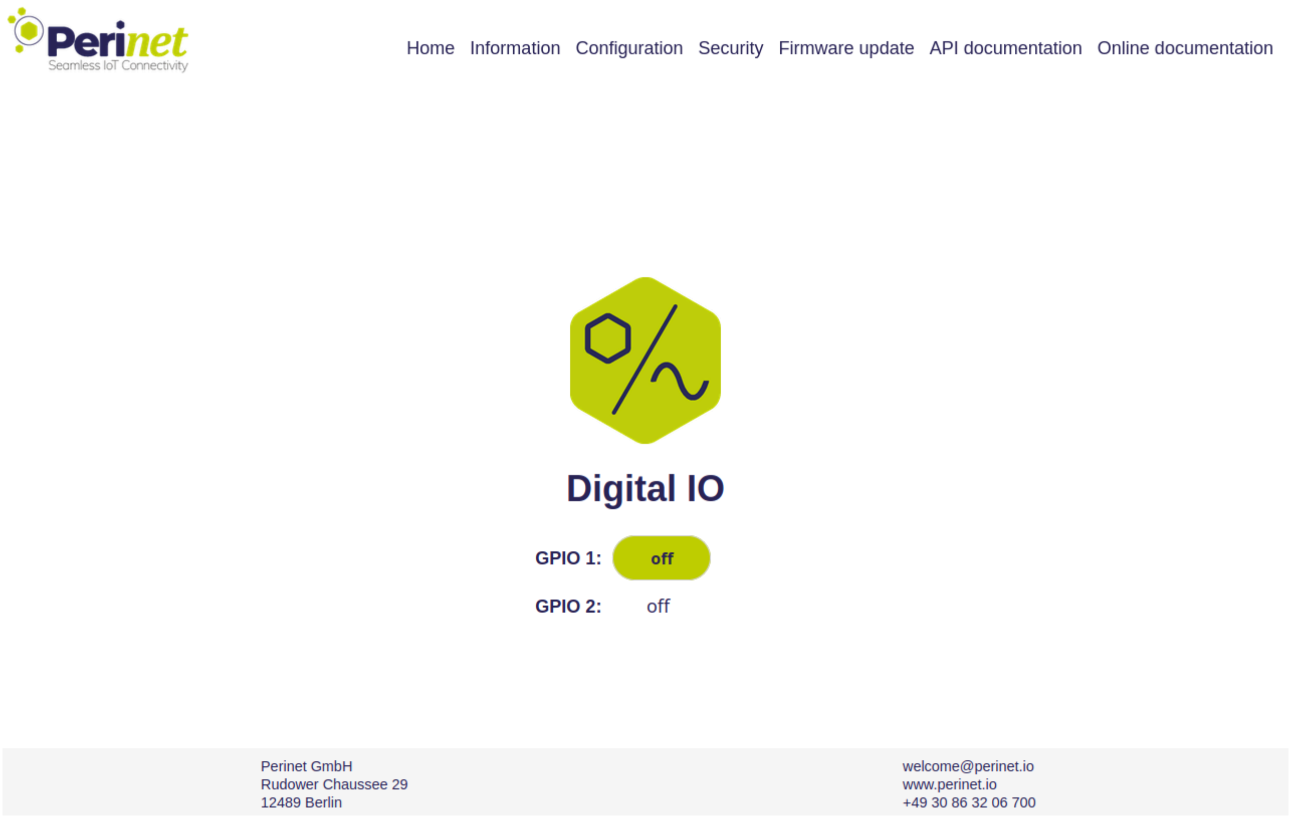
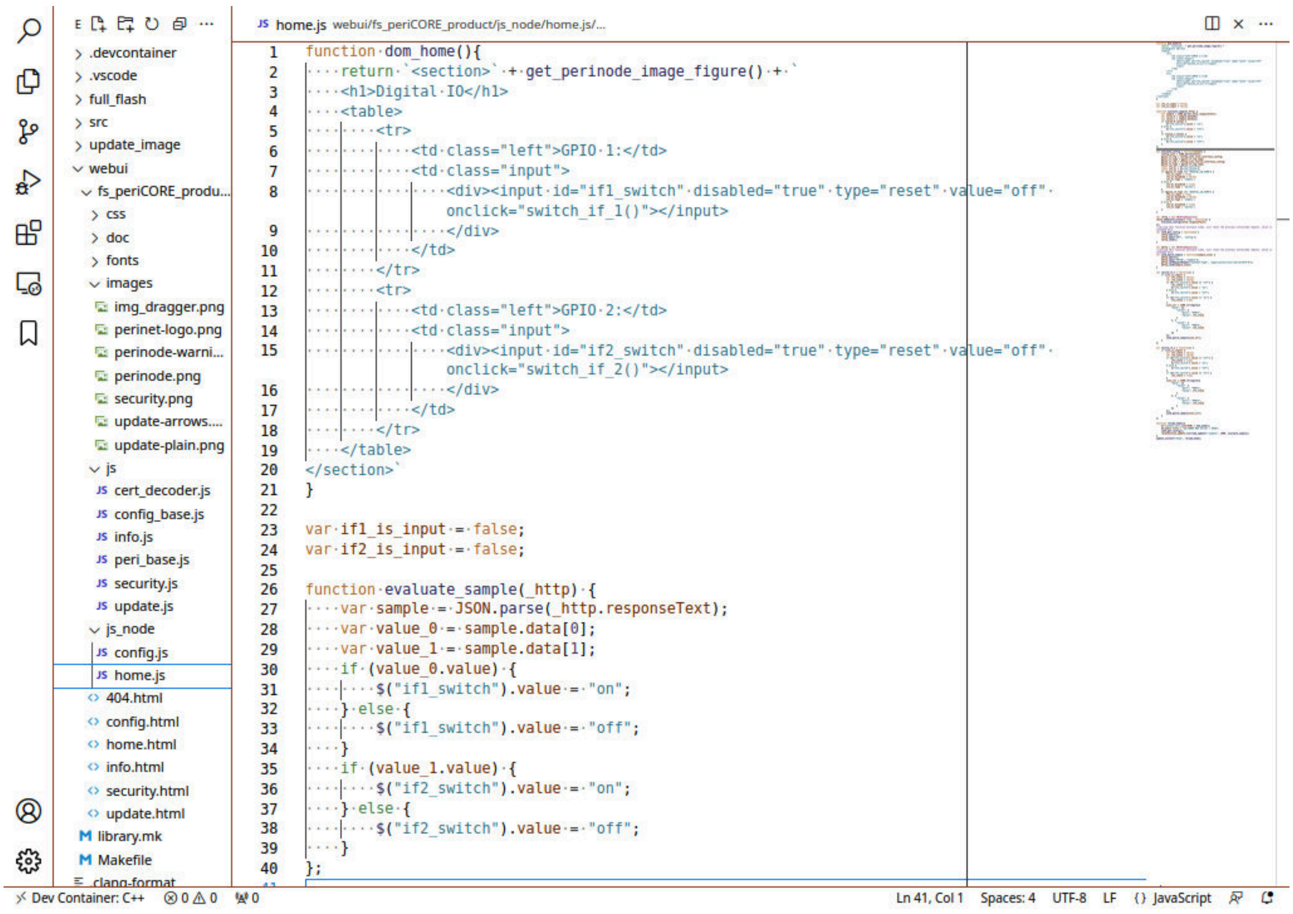


Figure 22: WebUI *Home* page for an actuator

The content of the *Home* page is sourced from the file `webui/fs_periCORE_product/js_-node/home.js`.

Figure 23 displays the periCORE SDK interface with the aforementioned source file open for editing.



home.js webui/fs_periCORE_product/js_node/home.js/...

```
function dom_home(){
    return `<section>` + get_perinode_image_figure() + `
    <h1>Digital IO</h1>
    <table>
        <tr>
            <td class="left">GPIO 1:</td>
            <td class="input">
                <div><input id="if1_switch" disabled="true" type="reset" value="off" 
                onclick="switch_if_1()"></input>
                </div>
            </td>
        </tr>
        <tr>
            <td class="left">GPIO 2:</td>
            <td class="input">
                <div><input id="if2_switch" disabled="true" type="reset" value="off" 
                onclick="switch_if_2()"></input>
                </div>
            </td>
        </tr>
    </table>
</section>`
}

var if1_is_input = false;
var if2_is_input = false;

function evaluate_sample(_http) {
    var sample = JSON.parse(_http.responseText);
    var value_0 = sample.data[0];
    var value_1 = sample.data[1];
    if (value_0.value) {
        $("if1_switch").value = "on";
    } else {
        $("if1_switch").value = "off";
    }
    if (value_1.value) {
        $("if2_switch").value = "on";
    } else {
        $("if2_switch").value = "off";
    }
};
```

Figure 23: source code for the WebUI's *Home* page in the in periCORE SDK

The HTML content for the *Home* page is defined in the `dom_home()` function, and subsequent functions manage its dynamic updates.

Pre-existing functions are designed to reflect the live status of sensors or actuators on the user interface. For instance, the `evaluate_sample()` function continuously retrieves the JSON-formatted response from the RESTful API's sample object for processing and display.

Similarly, `evaluate_config()` continuously fetches the JSON response from the RESTful API's configuration object (refer to Section 7.4.1).

The interval for these requests is set to 1 second in the `reload_home()` function (not shown in the figure).

The handling of actuator elements is customizable. For instance, the `switch_if_1()` function (not shown in the figure), linked to a click event on the *GPIO 1* input element, sends a patch request to the RESTful API's configuration object (see Section 7.4.1). This request toggles the state of the output port.

Feedback about the success of this operation would be implemented through the mentioned `evaluate_sample()` function (visible in Figure 23), 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.

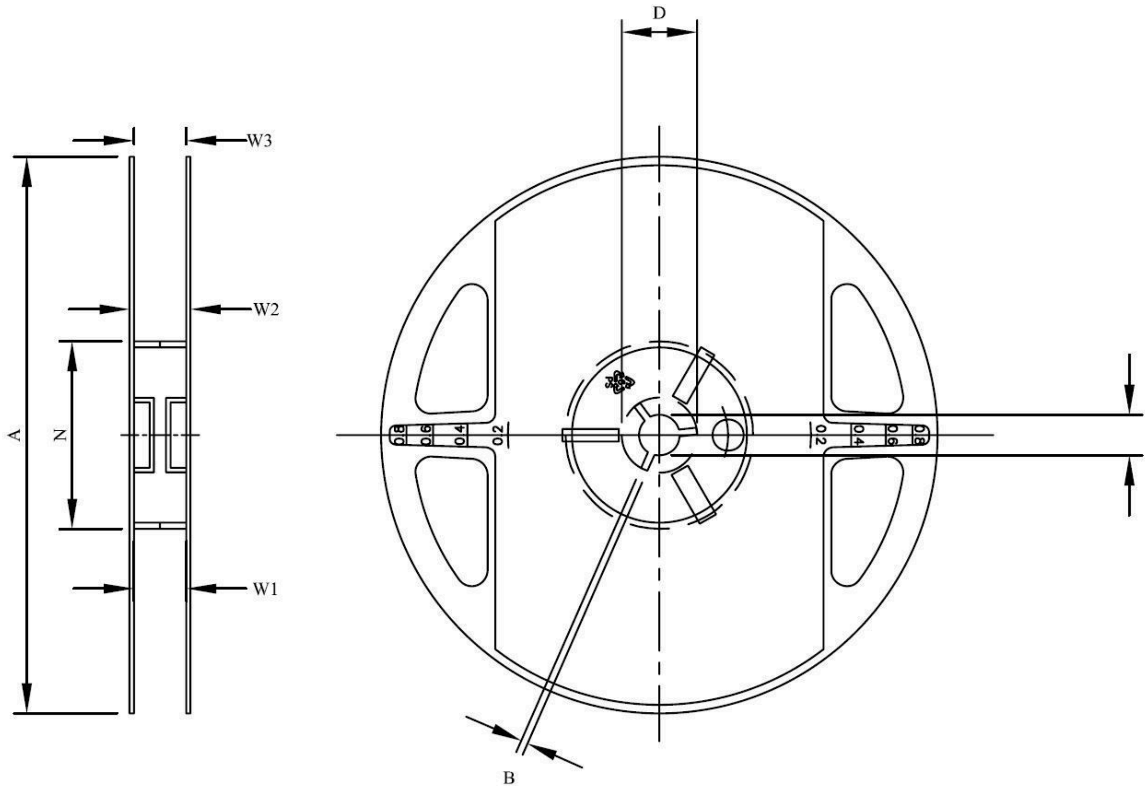


Figure 24: The dimensions of the Reel, used to deliver the periCORE modules.

Dimension	Value
A	330.00 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 17: Dimension of the reel.

8.2 Tape Information

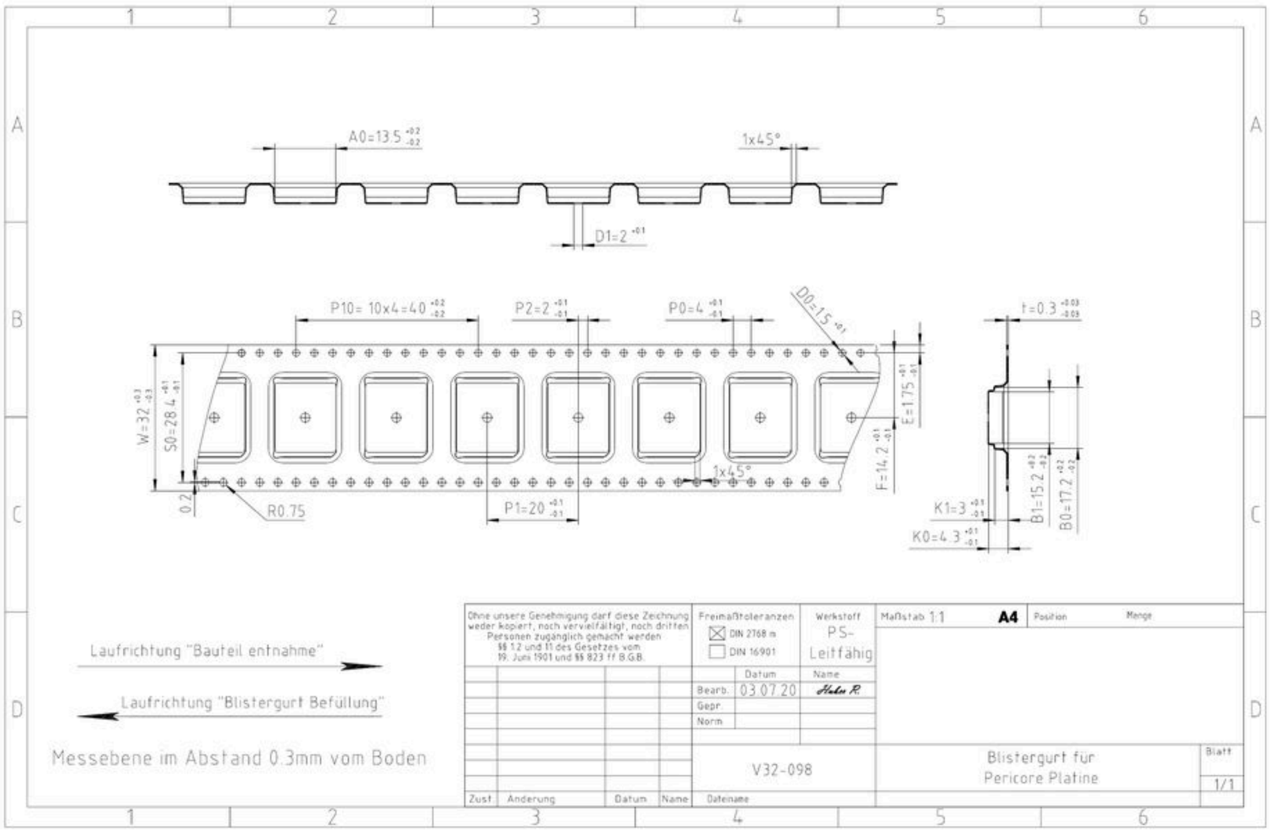


Figure 25: Tape and Reel packaging specification for the periCORE module.

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

The module is suitable for pick and place assembly 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.

8.5 Soldering

The periCORE is a surface mount module, produced on a 6-layer FR4 printed circuit board (PCB). It has gold plated pads and is suitable for a reflow soldering process with a lead-free soldering paste. For host boards with the periCORE module, only a single reflow soldering process is permitted.

The reflow profile to be used with the periCORE module depends on the thermal mass of the entire populated host PCB, the efficiency of the heat transfer of the reflow oven, and the characteristics of the used solder paste. For this reason, the optimum reflow profile should be defined for every design individually. The information given here are based on the IPC/JEDEC J-STD-020E standard.

The recommended reflow profile of the periCORE module is shown in Figure 26 and the values of the parameters are given in Table 18. The reflow profile is for the soldering process based on RoHS/Pb-free (Sn96.5/Ag3.0/Cu0.5) solder paste.

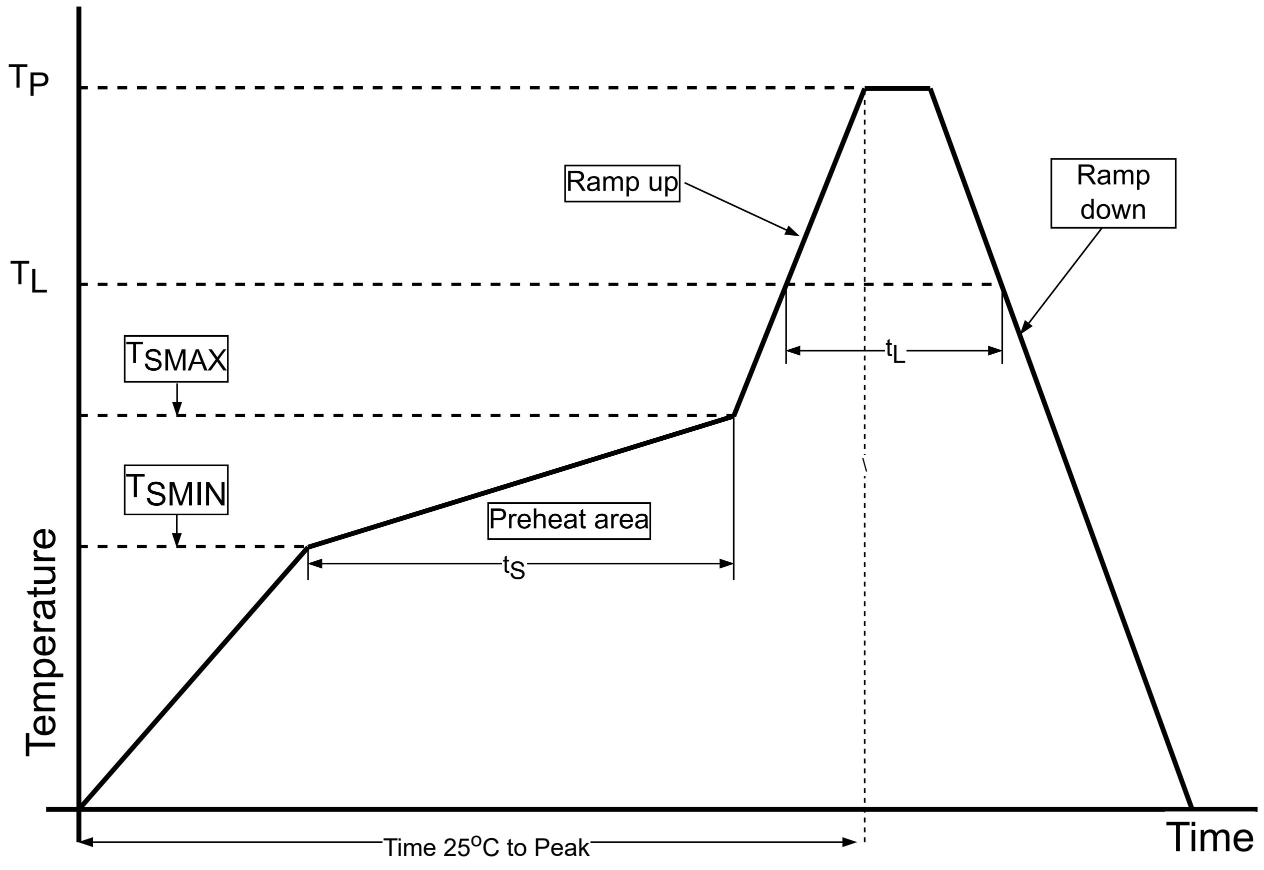


Figure 26: Recommended reflow profile

Symbol	Parameter	Value
–	Max Ramp up rate (T_L to T_P)	3 °C/s
T_{SMIN}	Minimum soak temperature	150 °C
T_{SMAX}	Maximum soak temperature	200 °C
t_S	Soak time	60 - 180 sec
T_L	Liquidus temperature	217 °C
t_L	Time above T_L	60 - 150 sec
T_P	Peak temperature	235 - 245 °C
t_P	Time within +0 / -5 °C of actual T_P	30 sec
–	Max Ramp down rate (T_P to T_L)	6 °C/s
–	Max Time from +25 °C to T_P	3 min

Table 18: Recommended reflow profile parameter values.

8.6 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 27.

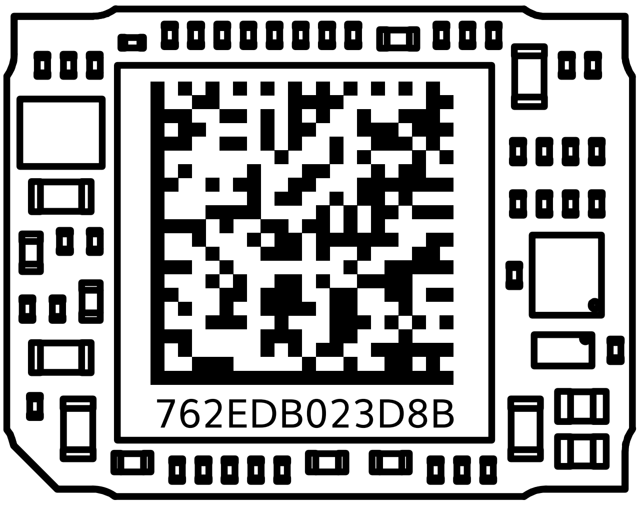


Figure 27: Visual marking via data matrix code of the periCORE module.

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 15: 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 16: 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 19.

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 20	`10000 00100 01111 01100 01011` ⇒ `s e r n m`

Table 19: 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 20: 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 periCORE 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:

fe80:0000:0000:0000:$xxxx$:$xxxx$:$xxxx$: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, 10

API Application Programming Interface. 37, 57, 58

ASN.1 Abstract Syntax Notation One. 47

BSC Basic Spacing Between Centers. 17, 18

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). 43, 44

CSMA/CD Carrier-sense multiple access with collision detection. 10

CSR Certificate Sign Request. 46

DNS-SD DNS Service Discovery [3] is a way of using standard DNS programming interfaces, servers and packet formats to browse the network for services. 6, 33, 49

ESD Electro Static Discharge. 3, 56

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. 31

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). 59

GPIO General-Purpose Input/Output. 7, 15

HTTP Hypertext Transfer Protocol is an application-layer protocol for transmitting hypermedia documents, such as HTML. 33, 43

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

IC Integrated Circuit. 1, 54, 56, 57

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. 34

IIoT Industrial Internet of Things. 1, 7

IPv6 Internet Protocol Version 6 [12], a communication protocol. 33

IPv6 LLA Internet Protocol Version 6 [12] link-local unicast address. 59

MAC Media Access Controller, component that handles concurrent access to a shared physical communication medium. 10, 32, 59, 60

MDI Media Dependent Interface, a Fast-Ethernet chipset component. 24

mDNS multicast Domain Name Service [4], a protocol that implements a local distributed name resolving mechanism. 6, 33, 49, 59

MQTT Message Queuing Telemetry Transport is a lightweight, publish-subscribe based network protocol that transports messages between devices. 33

mTLS Mutual TLS extends the TLS protocol by requiring clients to pass certificates, allowing to provide authorization mechanisms of Application services. 28, 36, 42, 43, 45

OEM Original Equipment Manufacturer is generally perceived as a company that produces parts and equipment that may be marketed by another manufacturer. 29, 31, 33

OID Object Identifier. 47

OS Operating System, a system software that manages computer hardware and software resources. 29–31

OTP One-Time Programmable memory. 31

PCB Printed Circuit Board. 55

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

RBAC Role Based Access Control. 33, 36, 45–47, 65

SPE Single Pair Ethernet. 7

TLS Transport Layer Security Protocol. Used by Application Protocols like MQTT or HTTP to allow secure data transfer. 33, 43, 44

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. 33, 43

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] OPEN ALLIANCE. *IEEE 100BASE-T1 System Implementation Specification*. URL: http://www.opensig.org/download/document/231/OA+100BASE-T1+system+implementation+specification_D1.0_final_18.pdf.

[3] 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.

[4] 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.

[5] "Connectors for electrical and electronic components - Product requirements - Part 6: Connectors - Detail specification for 2-way and 4-way (data/power), shielded, free and fixed connectors for transmission capability and power supply capability with frequencies up to 600 MHz". In: *IEC 63171-6* (2021). URL: https://webstore.iec.ch/publication/68051.

[6] 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.

[7] Perinet GmbH. periCORE Development Kit User Guide. PRN.100.378. https://docs.perinet.io/PRN100378-periCOREDevelopmentKitUserGuide.pdf.

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

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

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

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

[12] 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.

[13] "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.

[14] "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.

[15] "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/.

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

[17] "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.

[18] OpenSSL. *x509v3 Configuration - Arbitrary Extensions*. URL: https://www.openssl.org/docs/manmaster/man5/x509v3_config.html#ARBITRARY-EXTENSIONS.

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

F Revision History

Revision	Date	Author(s)	Description
1	2022-06-08	msen, dper, shoe, kwal, tkla, psil	Initial Release
2	2022-07-22	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
3	2024-02-08	shoe, nsav, tkla, clip	Update reel dimensions in Table 17, Update Table 6 where `M7` must be connected to GND. Update 100Base-T1 application circuit subsection Section 2.1, Add 100Base-TX application circuit subsection Section 2.1, Remove wrong specified $T_{4.3}$ for 3V3_OUT in Figure 13, Update max Supply Voltage to 27V (Table 8), Add Input capacitance Table 9, Update Output current Table 9, Update dimension precisions in Section 3.1, Update I2C interface performance parameter to *400 kbit/s ± 20 %* Section 2.2, Update UART interface performance parameters to *1.5 MBaud* Section 2.2, Add RMII interface in hardware block diagram Figure 4, editorial review.