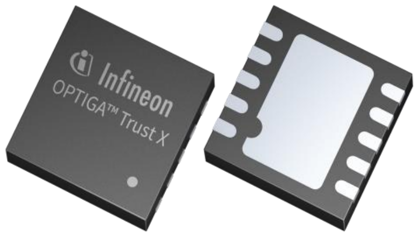
OPTIGA™ Trust X

Datasheet



Key Features

- High-end security controller
- Turnkey solution
- Mutual authentication using ECDSA
- DTLS client IETF standard RFC 6347
- Secure communication using DTLS
- Compliant with the USB Type-C™ Authentication standard
- I2C interface
- Up to 10 kB user memory
- Cryptographic support: ECC NIST P256 and P384, AES-128 (via DTLS client), SHA-256, TRNG, DRNG
- PG-USON-10-2 package (3 x 3 mm)
- Standard & extended temperature ranges
- Full system integration support with Host Software Library
- OPTIGA™ Trust X Software Framework on Github (https://github.com/infineon/optiga-trust-x)
- Common Criteria Certified EAL6+ (high) hardware
- Crypto ToolBox with ECC NIST P256, P384, SHA-256 (sign, verify, key generation, ECDH, key derivation)
- Device Security Monitor
- Lifetime for Industrial Automation and Infrastructure is 20 years and 15 years for other Application Profiles

Benefits

- Protection of IP and data
- Protection of business case
- Protection of corporate image
- Safeguarding of quality and safety

Applications

- Industrial control and automation
- Consumer electronics and Smart home
- Medical devices

About this document

Scope and purpose

This Datasheet provides information to enable integration of a security device, and includes package, connectivity and technical data.

About this document

Intended audience

This Datasheet is intended for device integrators and board manufacturers.

1 Introduction

As embedded systems (e.g. IoT devices) are increasingly gaining the attention of attackers, Infineon offers the OPTIGA™ Trust X as a turnkey security solution for industrial automation systems, smart homes, consumer devices and medical devices. This high-end security controller comes with full system integration support for easy and cost-effective deployment of high-end security for your assets.

1.1 Broad range of benefits

Integrated into your device, the OPTIGA™ Trust X supports protection of your brand and business case, differentiates your product from your competitors, and adds value to your product, making it stronger against cyberattacks.

1.2 Enhanced security

The OPTIGA™ Trust X is based on advanced security controller with built-in tamper proof NVM for secure storage and Symmetric/Asymmetric crypto engine to support ECC 256, AES-128(via DTLS client) and SHA-256. This new security technology greatly enhances your overall system security.

1.3 Fast and easy integration

The turnkey setup – with full system integration and all key/certificate material preprogrammed – reduces your efforts for design, integration and deployment to a minimum. As a turnkey solution, the OPTIGA™ Trust X comes with preprogrammed OS/Application code locked and with host-side modules to integrate with host micro controller software. The extended temperature range of −40°C to +105°C combined with a standardized I2C interface and the small PG-USON-10-2 footprint will facilitate onboarding in your existing ecosystem. Almost 30 years in a market-leading position with nearly 20 billion security controllers shipped worldwide are the result of Infineon's strong expertise and its commitment to make security a success factor for you.

1.4 Applications

The OPTIGA™ Trust X covers a broad range of use cases necessary for many types of applications that include the following:

a) Network node protection such as TLS or DTLS
b) Protect the Authenticity, Integrity and Confidentiality of your product, data and IP
c) Mutual Authentication
d) Secure Communication
e) Datastore Protection
f) Lifecycle Management
g) Platform Integrity Protection
h) Secure Updates

1.5 Device Features

The OPTIGA™ Trust X comes with upto 10kB user memory that can be used to store X.509 certificates. OPTIGA™ Trust X is based on Common Criteria Certified EAL6+ (high) hardware enabling it to prevent physical attacks on the device itself and providing high assurance that the keys or arbitrary data stored cannot be accessed by an unauthorized entity. The CC certificate can be found at www.bsi.bund.de by searching for BSI-DSZ-CC-0961-V2-2018 (Hardware Identifier IFX_CCI_00000Bh). OPTIGA™ Trust X supports a highspeed I2C communication interface of up to 1MHz (FM+).

Table 1 Products

Type	Description	Temperature range	Package
OPTIGA™ Trust X SLS 32AIA020X4	Embedded security solution for connected devices	−25°C to +85°C Standard Temperature Range (STR)	PG-USON-10-2
OPTIGA™ Trust X SLS 32AIA020X2	Embedded security solution for connected devices	−40°C to +105°C Extended Temperature Range (ETR)	PG-USON-10-2
Evaluation Kit	Includes host micro controller connected to OPTIGA™ Trust X with USB/Ethernet adapters to connect to external world which enables you to evaluate OPTIGA™ Trust X features and start the Design-In activity		Board

Infineon and its distribution partners offer a wide range of customization options (e.g. X.509 certificate generation and key provisioning) for the security chip.

Table 2 Abbreviations

Abbreviation	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
AUTH	Authentication
CA	Certification Authority
DTLS	Datagram Transport Layer Security
DRNG	Deterministic Random Number Generator
EAL	Evaluation Assurance Level
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
ETR	Extended Temperature Range
IETF	Internet Engineering Task Force
IOT	Internet of Things
IP	Intellectual Property
I2C	Inter-Integrated Circuit
NIST	National Institute of Standards and Technology
OCP	OPTIGA™ Crypto and Protected Communication
OS	Operating System
PAL	Platform Abstraction Layer
PKI	Public Key Infrastructure

Abbreviation	Definition
RFC	Request For Comments
TLS	Transport Layer Security
TRNG	True Random Number Generator
SHA	Secure Hash Algorithm
SKU	Stock Keeping Unit
STR	Standard Temperature Range
USB	Universal Synchronous Bus

2 System Block Diagram

The following figure depicts the system block diagram for OPTIGA™ Trust X.

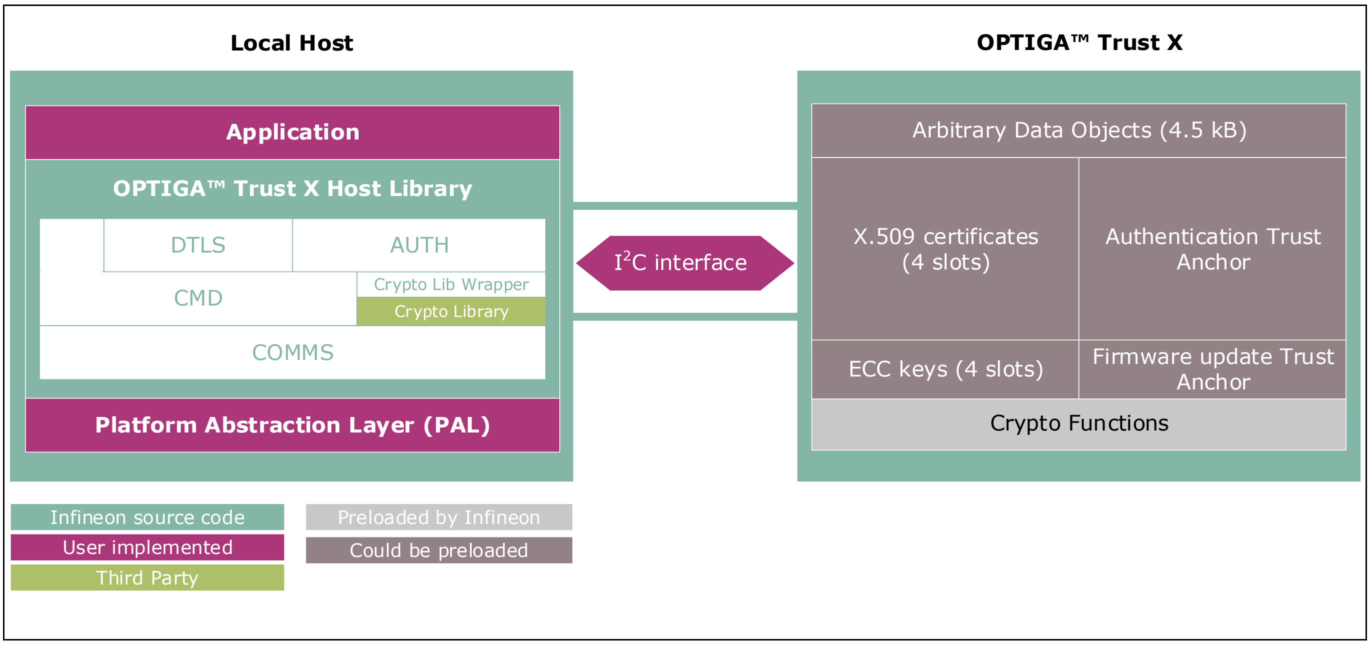


Figure 1 System Block Diagram

The System Block Diagram is explained below for each layer.

1. Local Host
   - Application – This is the target application which utilizes OPTIGA™ Trust X for its security needs
   - DTLS – DTLS client aka. OCP Library provides APIs for performing Mutual Authentication and Encrypted Communication using OPTIGA™ Trust X
   - AUTH – Authentication aka. Integration Library provides APIs for performing One Way Authentication for Brand Protection and IP Protection using OPTIGA™ Trust X
   - CMD – Provides APIs to send and receive commands to and from OPTIGA™ Trust X. Any TLS stack can be integrated to offload crypto operations to OPTIGA™ Trust X via this Command Library.
   - Crypto Lib Wrapper – Provides wrapper APIs for Third Party crypto library, mainly used in One Way Authentication
   - Crypto Library – External cryptographic software which is used for One Way Authentication
   - COMMS – Provides wrapper APIs for communication with OPTIGA™ Trust X which internally uses Infineon I2C Protocol (IFXI2C)
   - PAL – A layer that abstracts platform specific drivers (e.g. i2c, timer, gpio, sockets etc.)
2. OPTIGA™ Trust X
   - Arbitrary Data Objects – The target application can store upto 4.5kB (~4600 bytes) of data into OPTIGA™ Trust X
   - X.509 – Upto 4, X.509 based Certificates can be stored into OPTIGA™ Trust X
   - Keys – Upto 4, ECC based keys can be stored into OPTIGA™ Trust X

- Mutual Authentication Trust Anchor – Customer PKI domain Trust Anchor for Mutual Authentication using TLS/DTLS can be stored into OPTIGA™ Trust X
- Firmware Update Trust Anchor – Customer PKI domain Trust Anchor for Firmware Updates can be stored into OPTIGA™ Trust X
- Crypto Functions - OPTIGA™ Trust X provides cryptographic functions and protocols that can be invoked via local host

Note: Unique ECC private keys and X.509 Certificates – During production at Infineon fab, unique asymmetric keys (private and public) are generated. The public key is signed by customer specific CA and resulting X.509 certificate issued is securely stored on OPTIGA™ Trust X. Special measures are taken to prevent leakage and modification of private key material at the Common Criteria Certified production site

3 Interface and Schematics

This section explains the schematics of the product and gives some recommendations as to how the controller should be externally connected.

3.1 System Integration Schematics

Figure 1 illustrates how to integrate OPTIGA™ Trust X to your local host.

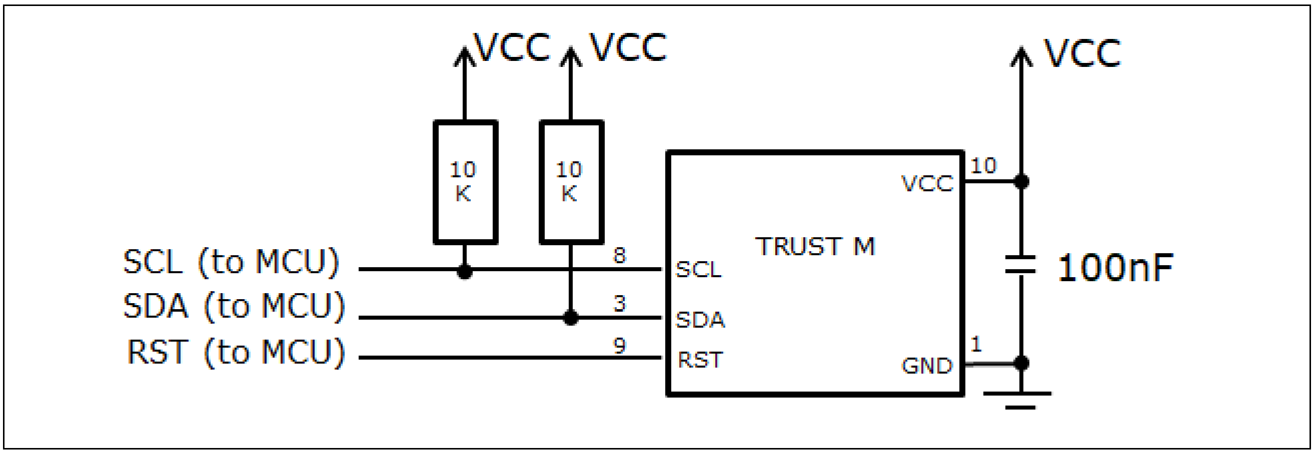


Figure 2 **System Integration Schematic Diagram**

Note: *Value of the pullup resistors depends on the target application circuit and the targeted I2C frequency.*

4 Description of packages

This chapter provides information on the package types and how the interfaces of each product are assigned to the package pins. For further information on compliance of the packages with European Parliament Directives, see "RoHS Compliance" on Page 30.

For details and recommendations regarding the assembly of packages on PCBs, please see the following: http://www.infineon.com/cms/en/product/technology/packages/

4.1 PG-USON-10-2

The package dimensions (in mm) of the controller in PG-USON-10-2 packages are given below.

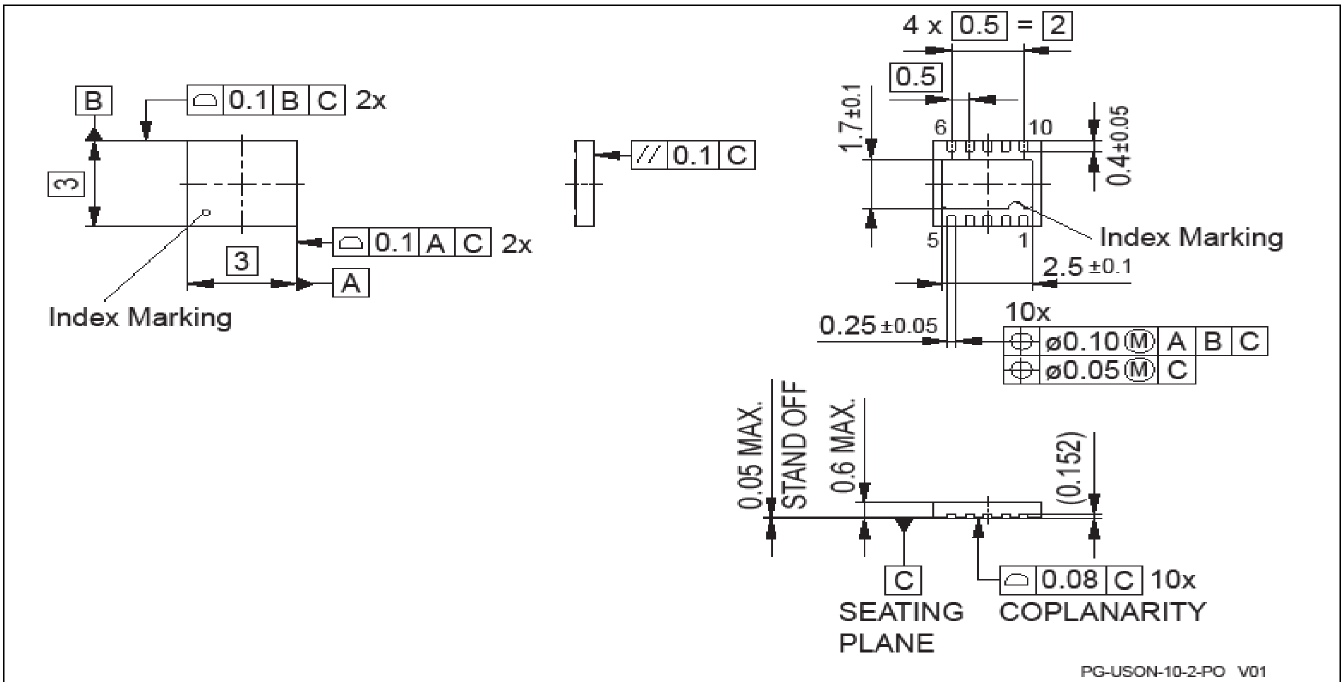


Figure 3 PG-USON-10-2 Package Outline

The following figure shows the footprint of the PG-USON-10-2 package:

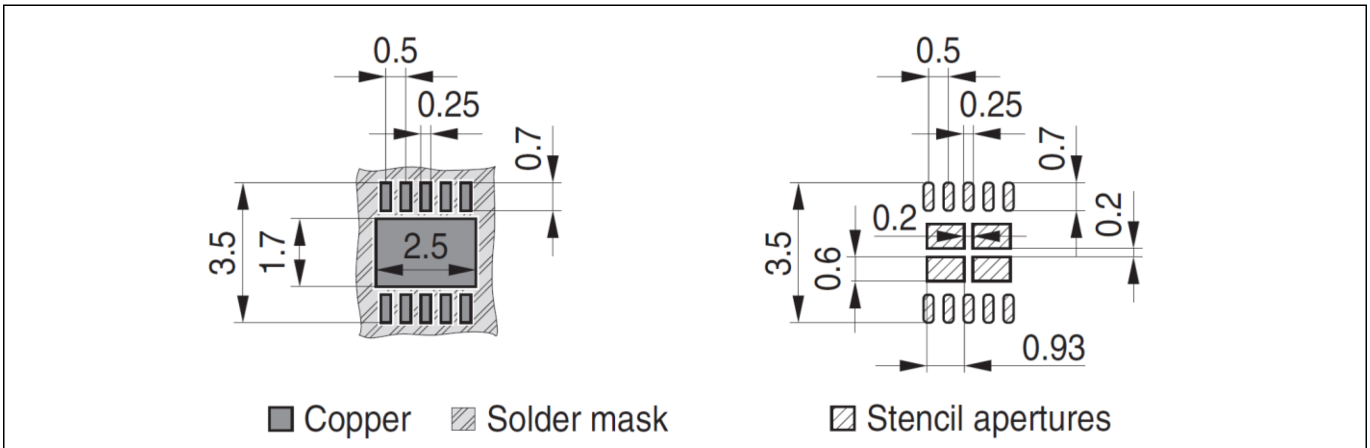


Figure 4 PG-USON-10-2 Package Footprint

The following figure shows the PG-USON-10-2 in top view:

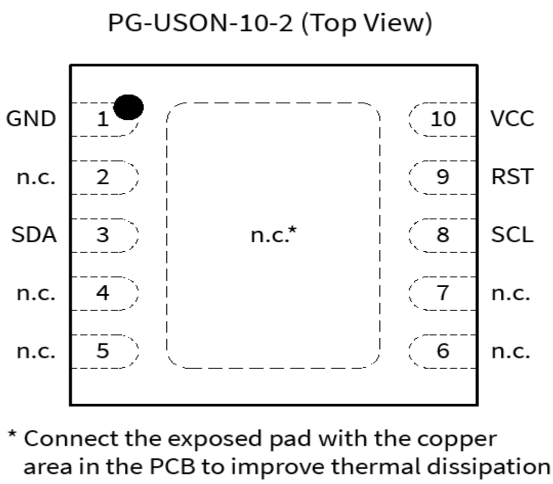


Figure 5 PG-USON-10-2 top view

4.2 Production sample marking pattern

The following figure describes the productive sample marking pattern on PG-USON-10-2.

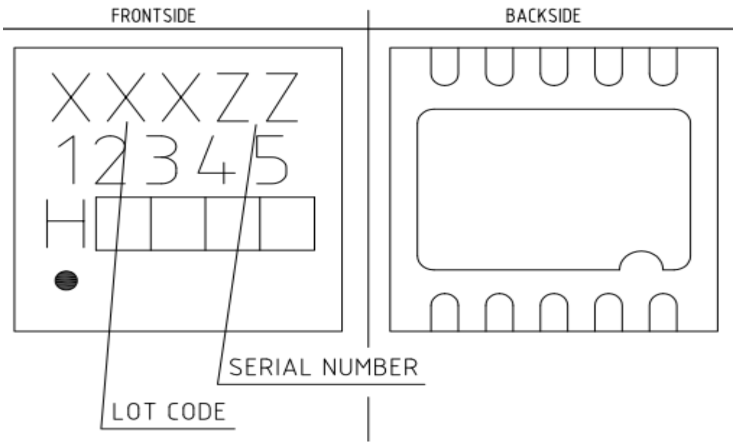


Figure 6 PG-USON-10-2 sample marking pattern

The black dot indicates pin 01 for the chip. The following table describes the sample marking pattern:

Table 3 Marking table for PG-USON-10-2 Packages

Indicator	Description
LOT CODE	Defined and inserted during fabrication
ZZ	Indicates the Certifying Authority Serial Number / SKU#, e.g. "00" would mean "SKU#00"
H/E	H = "Halogen-free", E = "Engineering samples" This indicator is followed by "YYWW", where YY is the "Year" and WW is the "Work Week" of the production. This is inserted during fabrication. Engineering samples have "E YYWW" and productive samples have "H YYWW"

Indicator	Description
12345	Convention: T&#$@ where: • The letter "T" indicates the OPTIGA Trust family • & indicates whether the product is a Trust X or Trust E controller • # indicates whether the controller is an ETR (E) or STR (S) variant • $ specifies the OPTIGA™ Trust X/E release version number • @ specifies the software version Example: "TXE10" means 'OPTIGA™ Trust X', 'ETR variant', 'release version 1', 'software version 0'

The contacts and their functionality are given in the table below.

Table 4 Contact Definitions and Functions of PG-USON-10-2 Packages

Pin	Type	Function
01	GND	Supply voltage (Ground)
02	NC	Not connected / Do not connect externally
03	I/O	Serial Data Line (SDA)
04	NC	Not connected / Do not connect externally
05	NC	Not connected / Do not connect externally
06	NC	Not connected / Do not connect externally
07	NC	Not connected / Do not connect externally
08	I/O	Serial Clock Line (SCL)
09	IN	Active Low Reset (RST)
10	PWR	Supply voltage (V_{CC})

5 Technical Data

This section summarizes the technical data of the product. It provides the operational characteristics as well as the electrical DC and AC characteristics.

5.1 I2C Interface Characteristics

Table 5 I2C Operation Supply and Input Voltages

Parameter	Symbol	Values			Unit	Note or Test Condition
		Min.	Typ.	Max.		
Supply voltage	V_{CC_I2C}	1.62	–	5.5	V	
SDA, SCL input voltage	V_{IN_I2C}	−0.3	–	V_{CC_I2C} + 0.5 or 5.5[1]	V	V_{CC_I2C} is in the operational supply range
		−0.3	–	5.5	V	V_{CC_I2C} is switched off

1) Whichever is lower

5.1.1 I2C Standard/Fast Mode Interface Characteristics

For operation of the I2C interface, the electrical characteristics are compliant with the I²C bus specification Rev. 4 for "standard-mode" (f_{SCL} up to 100 kHz) and "fast-mode" (f_{SCL} up to 400 kHz), with certain deviations as stated in the table below.

Note: T_A as given for the operating temperature range of the controller unless otherwise stated.

Table 6 I2C Standard Mode Interface Characteristics

Parameter	Symbol	Values			Unit	Note or Test Condition
		Min.	Typ.	Max.		
SCL clock frequency	f_{SCL}	0	–	100	kHz	
Input low-level	V_{IL}	−0.3	–	0.3 * V_{CC_I2C}	V	
Low-level output voltage	V_{OL1}	0	–	0.4	V	Sink current 3 mA; $V_{CC\_I2C} \geq 2.7$ V Sink current 2 mA; $V_{CC\_I2C} < 2.7$ V
Low-level output current	I_{OL}	3 2	–	–	mA	$V_{OL} = 0.4$ V; $V_{CC\_I2C} \geq 2.7$ V $V_{OL} = 0.4$ V; $V_{CC\_I2C} < 2.7$ V
Output fall time from V_{IHmin} to V_{ILmax} (at device pin)	t_{OF}	–	–	250	ns	$C_b \leq 400$ pF; $V_{CC\_I2C} \geq 2.7$ V $C_b \leq 200$ pF; $V_{CC\_I2C} < 2.7$ V
Capacitive load for each bus line	C_b	–	–	400 200	pF	$V_{CC\_I2C} \geq 2.7$ V $V_{CC\_I2C} < 2.7$ V

Table 7 I2C Fast Mode Interface Characteristics

Parameter	Symbol	Values			Unit	Note or Test Condition
		Min.	Typ.	Max.		

Parameter	Symbol	Values			Unit	Note or Test Condition
		Min.	Typ.	Max.		
SCL clock frequency	f_{SCL}	0	–	400	kHz	
Input low-level	V_{IL}	−0.3	–	0.3 * V_{CC_I2C}	V	
Low-level output voltage	V_{OL1}	0	–	0.4	V	Sink current 3 mA; $V_{CC\_I2C} \geq 2.7$ V Sink current 2 mA; $V_{CC\_I2C} < 2.7$ V
Low-level output current	I_{OL}	3 2	–	–	mA	$V_{OL} = 0.4$ V; $V_{CC\_I2C} \geq 2.7$ V $V_{OL} = 0.4$ V; $V_{CC\_I2C} < 2.7$ V
Output fall time from V_{IHmin} to V_{ILmax} (at device pin)	t_{OF}	20 * V_{CC_I2C} / 5.5 V[1]	–	250	ns	$C_b \leq 400$ pF; $V_{CC\_I2C} \geq 2.7$ V $C_b \leq 200$ pF; $V_{CC\_I2C} < 2.7$ V
Capacitive load for each bus line	C_b	15[2]	–	400 200	pF	$V_{CC\_I2C} \geq 2.7$ V $V_{CC\_I2C} < 2.7$ V

1) A min. capacitive load is necessary to reach t_{OF}

2) A min. capacitive load is necessary to reach t_{fmin}

5.1.2 I2C Fast Mode Plus Interface Characteristics

For operation of the I2C interface, the electrical characteristics are compliant with the I^2C bus specification Rev. 4 for "fast mode plus" (f_{SCL} up to 1 MHz), with certain deviations as stated in the table below.

Note: T_A as given for the operating temperature range of the controller unless otherwise stated.

Table 8 I2C Fast Mode Plus Interface Characteristics

Parameter	Symbol	Values			Unit	Note or Test Condition
		Min.	Typ.	Max.		
SCL clock frequency	f_{SCL}	0	–	1000	kHz	
Input low-level	V_{IL}	−0.3	–	0.3 * V_{CC_I2C}	V	
Low-level output voltage	V_{OL1}	0	–	0.4	V	Sink current 3 mA; $V_{CC\_I2C} \geq 2.7$ V Sink current 2 mA; $V_{CC\_I2C} < 2.7$ V
Low-level output current	I_{OL}	3 2	–	–	mA	$V_{OL} = 0.4$ V; $V_{CC\_I2C} \geq 2.7$ V $V_{OL} = 0.4$ V; $V_{CC\_I2C} < 2.7$ V
Output fall time from V_{IHmin} to V_{ILmax} (at device pin)	t_{OF}	20 * V_{CC_I2C} / 5.5 V[1]	–	120	ns	$C_b \leq 150$ pF
Capacitive load for each bus line	C_b	15[1]	–	150	pF	

1) A min. capacitive load is necessary to reach t_{OF}

5.1.3 Electrical Characteristics

Note: *T_A as given for the operating temperature range of the controller unless otherwise stated. All currents flowing into the controller are considered positive.*

5.1.4 DC Electrical Characteristics

T_A as given for the controller's operating ambient temperature range unless otherwise stated.
All currents flowing into the controller are considered positive.

Table 9 Electrical Characteristics

Parameter	Symbol	Values			Unit	Note or Test Condition
		Min.	Typ.	Max.		
Supply voltage	V_{CC}	1.62	–	5.5	V	Overall functional range
	V_{CC_I2C}	1.62	–	5.5	V	Supply voltage range for operation of I2C
Supply current[1]	I_{CCAVG}	–	20.0	–	mA	While running a typical authentication profile T_A = 25°C; V_{CC} = 5.0 V
Supply current, in *sleep mode*	I_{CCS3}	–	70	100	µA	T_A = 25°C; V_{CC_I2C} = 3.3 V; I2C ready for operation (no bus activity), all other inputs at V_{CC}, no other interface activity
RST input low voltage	V_{IL}	–0.3	–	0.2 * V_{CC}	V	I_{IL} = –50 µA to +20 µA
RST input high voltage	V_{IH}	0.7 * V_{CC}	–	V_{CC} + 0.3	V	I_{IL} = –50 µA to +20 µA

1) Supply current can be limited from 6mA to 15mA by software commands.

5.1.5 AC Electrical Characteristics

T_A as given for the controller's operating ambient temperature range unless otherwise stated.
All currents flowing into the controller are considered positive.

Table 10 AC Characteristics

Parameter	Symbol	Values			Unit	Note or Test Condition
		Min.	Typ.	Max.		
V_{CC} rampup time	t_{VCCR}	1	–	1000	µs	400 mV to 90% of V_{CC} target voltage ramp

The V_{CC} ramp is depicted in Figure 7. 90% of the target supply voltage must be reached within t_{VCCR} after it has exceeded 400 mV. Moreover, its variation must be kept within a ±10% range.

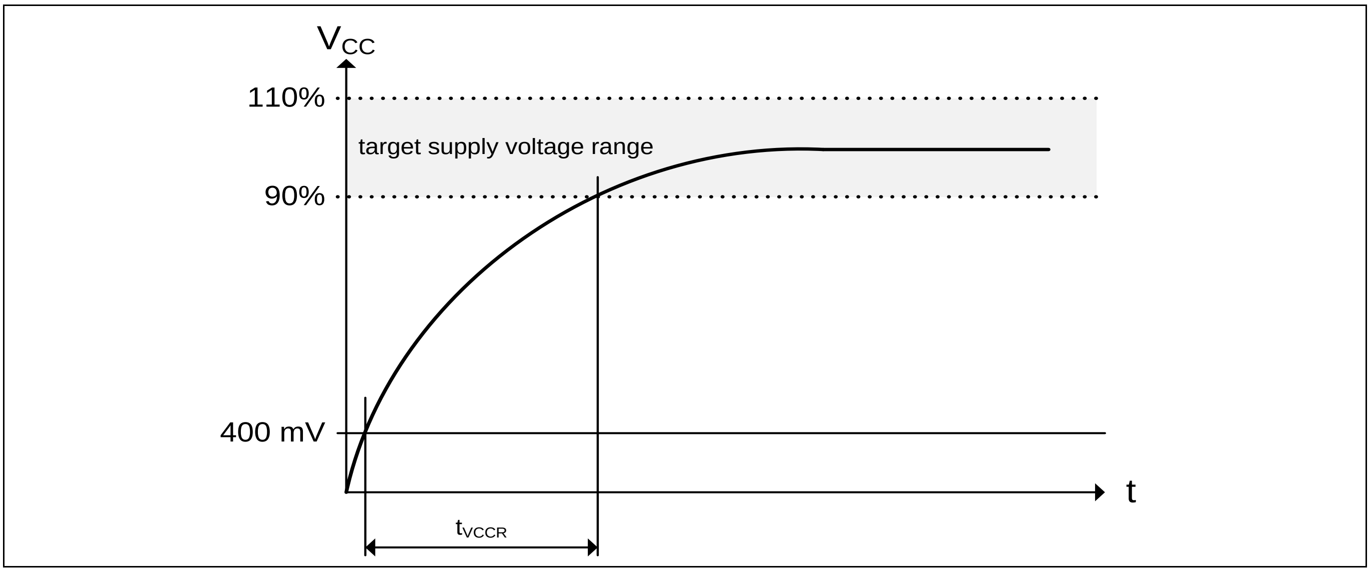


Figure 7 V_{cc} **Rampup**

5.1.6 Start-Up of I2C Interface

There are 2 variants possible for performing the startup procedure:

- Startup after power-on
- Startup for warm resets

5.1.6.1 Startup after Power-On

The activation of the I2C interface after power-on needs the following reset procedure.

- VCC is powered up and the state of the SDA and SCL line are set to high level during power-up
- The first transmission may start at the earliest $t_{STARTUP}$ after power-up of the device

The following figure shows the startup timing of the I2C interface for this case.

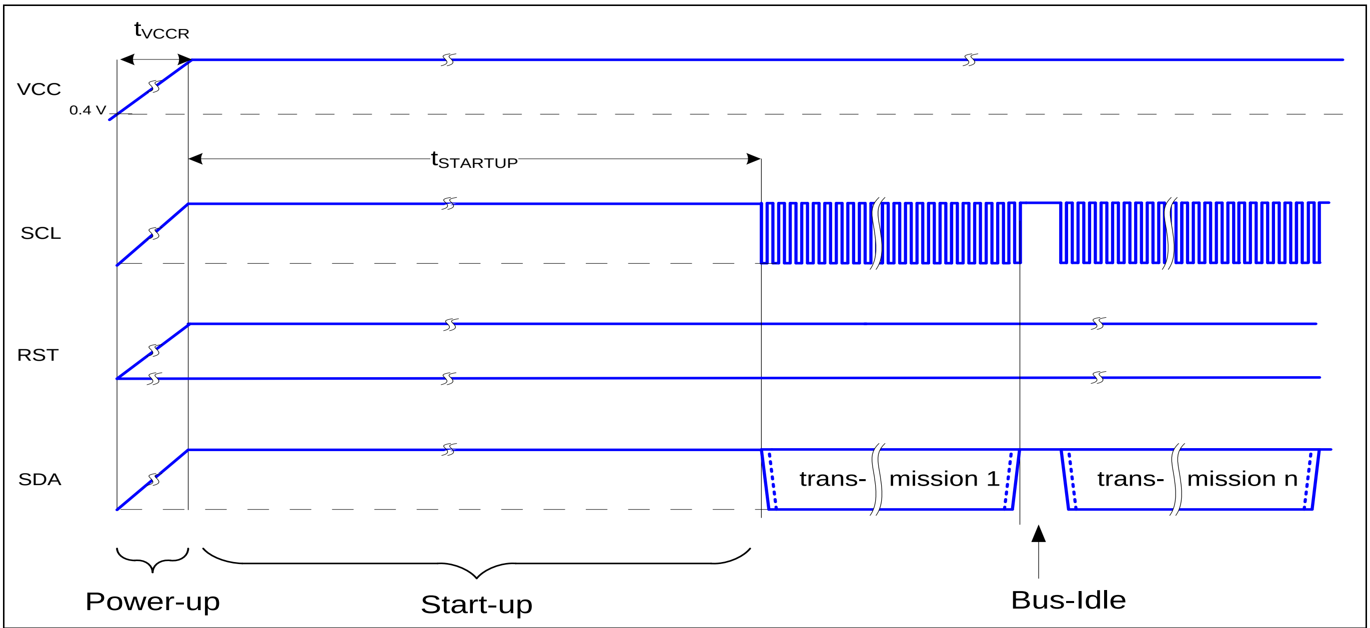


Figure 8 Startup of I2C Interface after Power-On

Table 11 Startup of I2C Interface After Power-On

Parameter	Symbol	Values			Unit	Note or Test Condition
		Min.	Typ.	Max.		
Startup time	$t_{STARTUP}$	10			ms	

5.1.6.2 Startup for Warm Resets

When using the reset signal for triggering a warm reset after power-on, the activation of the I2C interface needs the following reset procedure

- VCC remains powered up.
- The terminal stops I2C communication. SDA and SCL lines are set to high level before RST is set to low level.
- After its falling edge, RST has to be kept at low level for at least *t1*. At the latest *t2* after the falling edge of RST, the terminal must set RST to high level.
- The first transmission may start at the earliest $t_{STARTUP}$ after the rising edge of RST

The following figure shows the timing for this startup case.

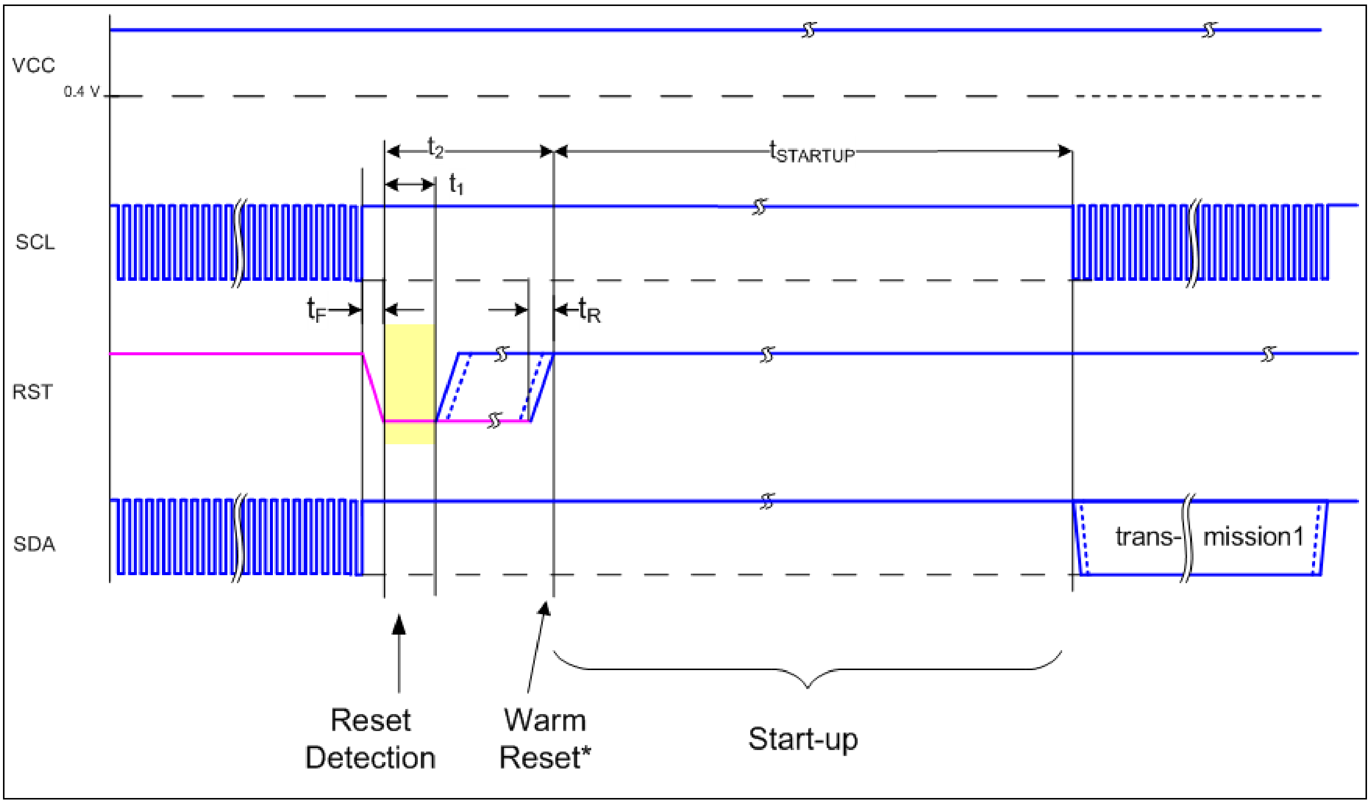


Figure 9 Startup of I2C Interface for Warm Resets

Note: If NVM programming was requested prior to the reset, $t_{STARTUP}$ will be extended from a typical value of 10 ms to a maximum of 12 ms.

Table 12 Startup of I2C Interface for Warm Resets[1]

Parameter	Symbol	Values			Unit	Note or Test Condition
		Min.	Typ.	Max.		
Startup time	$t_{STARTUP}$	10			ms	
Rise time	t_R			1	µs	From 10% to 90% of signal amplitude
Fall time	t_F			1	µs	From 10% to 90% of signal amplitude
Reset detection	t_1	10			µs	
Reset low		10		2500	µs	

1) Reset triggered by software (without power off/on cycle)

6 Connecting to Host

6.1 OPTIGA™ Trust X Host Software Architecture

The OPTIGA™ Trust X Host Library layers were explained in the System Block Diagram in Figure 1. In following sections, we will cover how to communicate with OPTIGA™ Trust X using I2C.

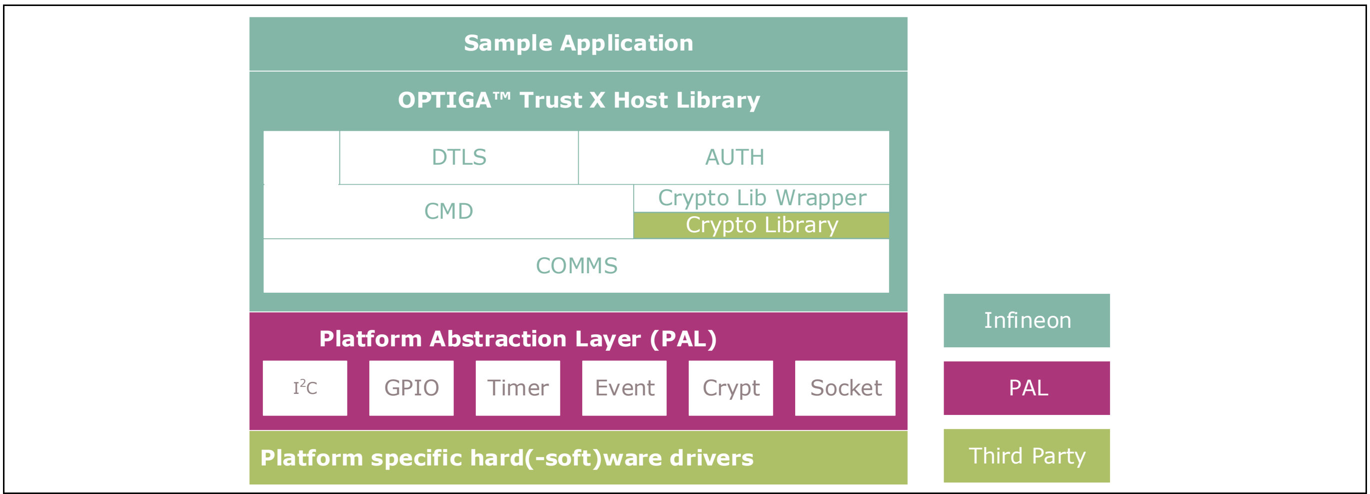


Figure 10 OPTIGA™ Trust X Host Software Architecture

6.2 Release Package Folder Structure

The following figure shows the release package structure when OPTIGA™ Trust X is extracted on PC.

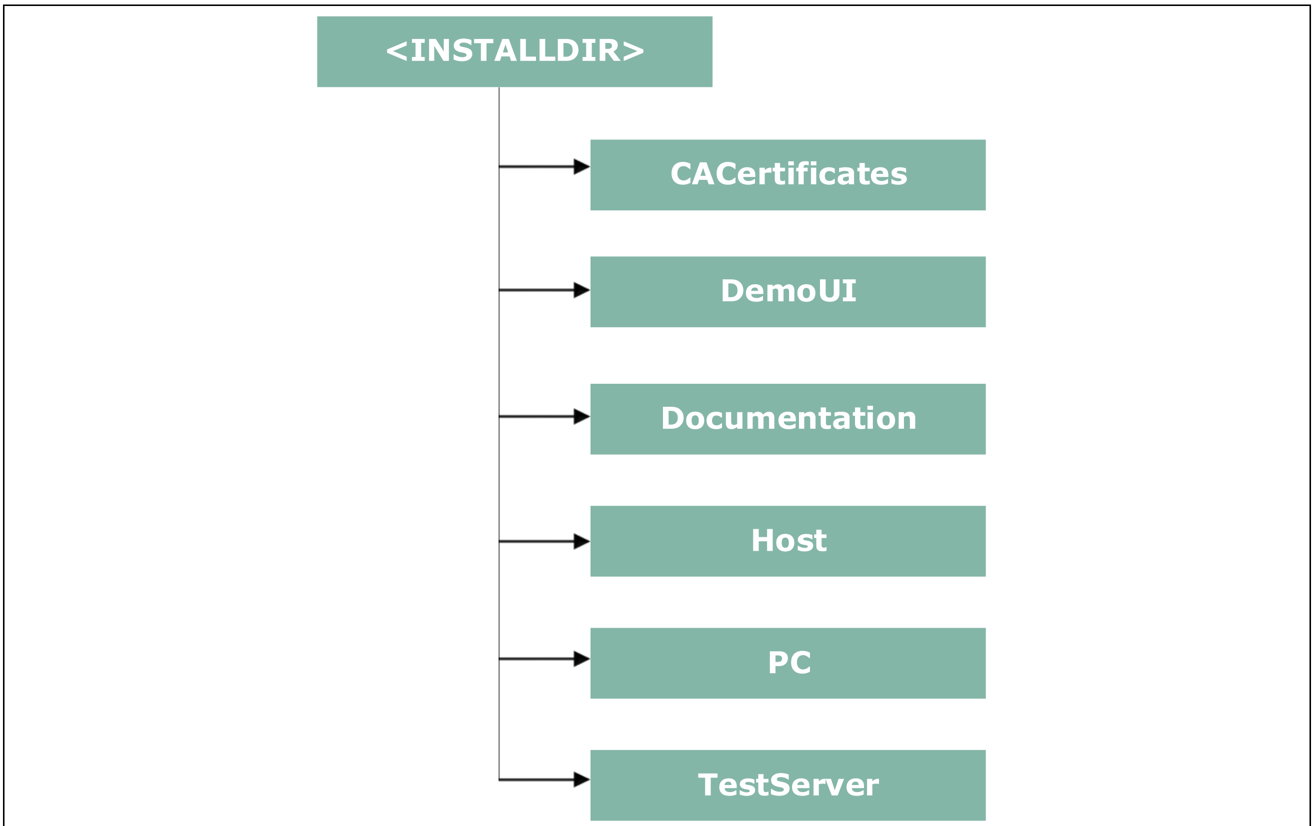


Figure 11 Release Package Folder Structure

1. <INSTALLDIR> is the root directory to which the release contents are extracted. The content of each subdirectory under extracted directory <INSTALLDIR> is explained below.
2. CACertificates
   This directory contains OPTIGA™ Trust X Test and Productive Trust-Anchor/CA certificates.
3. DemoUI
   This directory contains binaries and Demo UI Application for OPTIGA™ Trust X.
4. Documentation
   This directory contains all common OPTIGA™ Trust X documentation.
5. Host
   This directory contains source files, header files, binaries, documents, API as compiled help (CHM) and sample application for OPTIGA™ Trust X Host Software.
6. PC
   This directory contains source files, header files, binaries and sample application for OPTIGA™ Trust X PC Software.
7. TestServer
   This directory contains Sample Test Server Application and Test certificates required for DTLS client feature demonstration

6.3 Host Software Folder Structure

The following figure shows the Host Software folder structure when OPTIGA™ Trust X is extracted on PC.

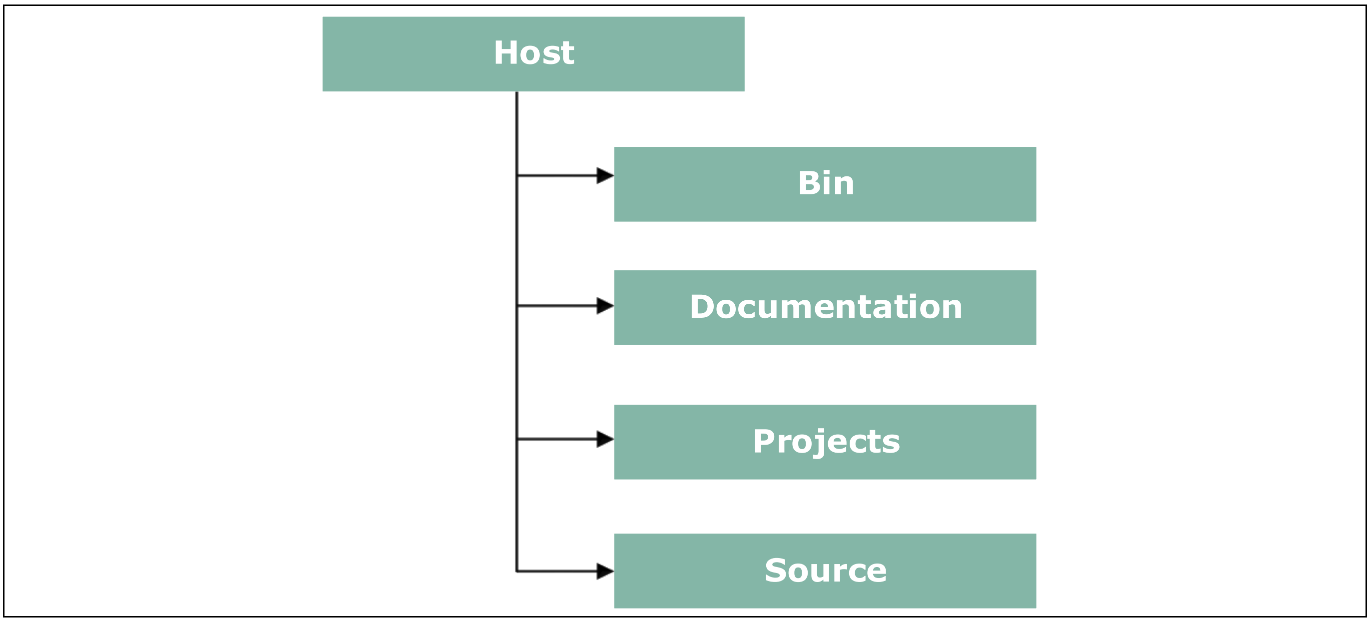


Figure 12 Host Software Folder Structure

1. Bin

This directory contains prebuilt binaries for Eval Kit based on XMC4500 Relax Kit v1 that communicates with OPTIGA™ Trust X.

2. Documentation

This directory contains documentation outlining software for Eval Kit based on XMC4500 Relax Kit v1.

3. Projects

This directory contains project files for Eval Kit based on XMC4500 Relax Kit v1.

4. Source

This directory contains all source files for OPTIGA™ Trust X Host Software Library.

Further the following figure elaborates the Host Software source folder structure.

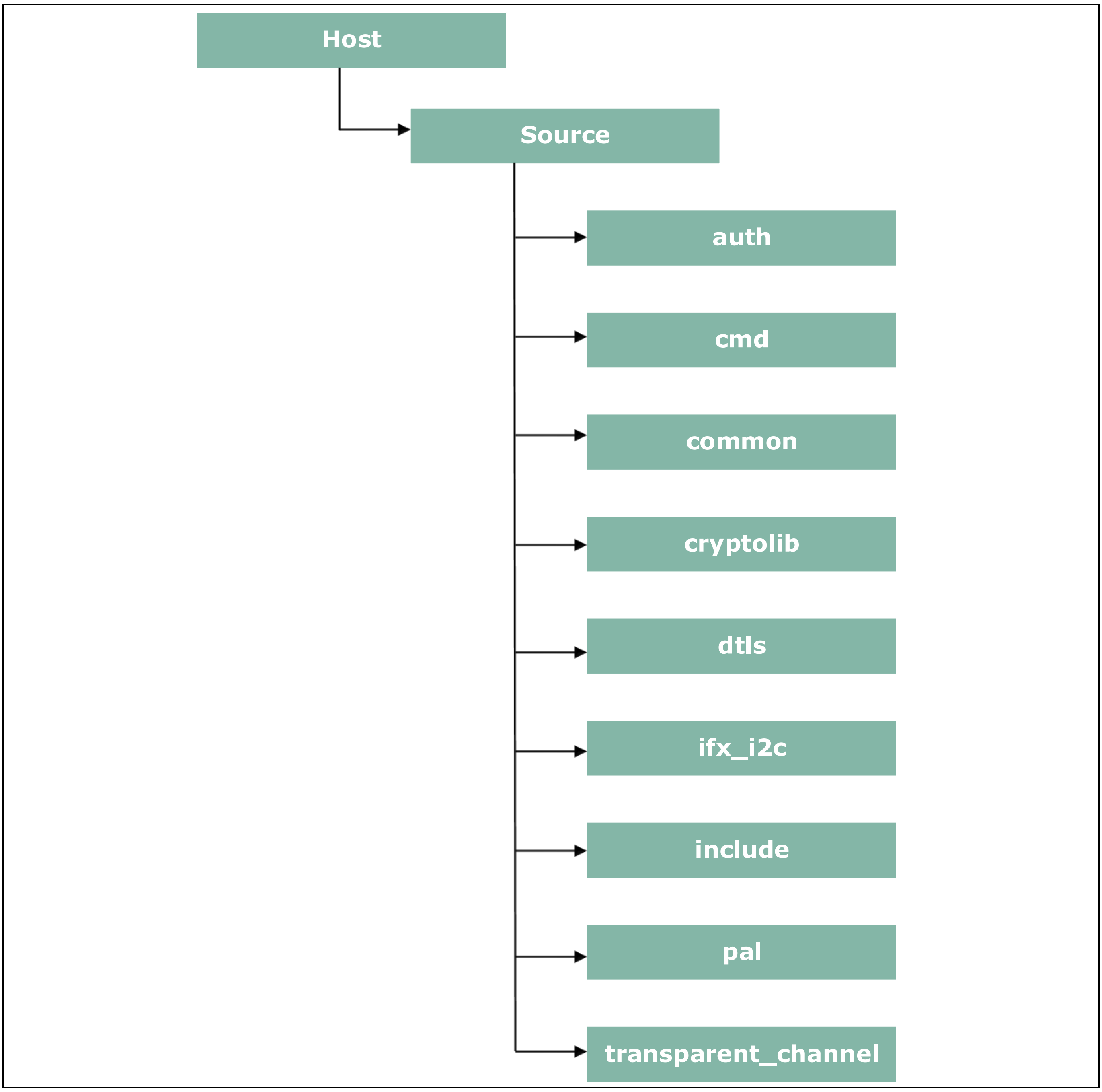


Figure 13 Host Source Folder Structure

1. auth – This folder contains sources for One Way Authentication which are platform independent. The layer is also known as Integration Library.
2. cmd – This folder contains sources for all OPTIGA™ Trust X commands which are platform independent.

3. common – This folder contains sources that are common for all functionality (e.g. utilities).
4. cryptolib – This folder contains binaries for crypto library wrapper which is platform independent.
5. dtls – This folder contains sources for Mutual Authentication and Encrypted Communication using DTLS client, which are platform independent. The layer is also known as OCP Library.
6. ifx_i2c – This folder contains sources for Infineon protocol over I2C (aka IFX I2C).
7. include – This folder contains header files for all Host Software.
8. pal – This folder contains all the platform dependent code.
9. transparent_channel – This folder contains transparent channel communication mainly used for Eval Kit.

6.4 Porting Notes

The Platform Abstraction Layer (PAL) APIs have to be updated to integrate the OPTIGA™ Trust X host libraries in the local host target platform.

The PAL reference code for the XMC4500 Relax kit is provided as part of package which can be used. The implementation can be referred in *"<INSTALLDIR>/Host/Source/pal/xmc4500"* and the header files are available in *"<INSTALLDIR>/Host/Source/Include"* with the required APIs used by upper layers. The header files are platform agnostic and would not require any change.

6.5 Communication with OPTIGA™ Trust X

The hardware/platform resource configuration with respect to I2C master and GPIOs (Vdd and Reset) are to be updated in *pal_ifx_i2c_config.c*. These configurations are used by the IFX I2C implementation to communicate with OPTIGA™ Trust X.

1. Update I2C master platform specific context[e.g. (void*)&i2c_master_0]

```
/**
 * \brief PAL I2C configuration for OPTIGA
*/
pal_i2c_t optiga_pal_i2c_context_0 =
{
    /// Pointer to I2C master platform specific context
    (void*)&i2c_master_0,
    /// Slave address
    0x30,
    /// Upper layer context
    NULL,
    /// Callback event handler
    NULL
};
```

2. Update platform specific context for GPIOs (Vdd and Reset) [e.g. (void*)&pin_3_4]

```
/**
 * \brief Vdd pin configuration for OPTIGA
*/
pal_gpio_t optiga_vdd_0 =
{
    // Platform specific GPIO context for the pin used to toggle Vdd
```

```
            (void*)&pin_3_4
        };

        /**
         * \brief Reset pin configuration for OPTIGA
        */
        pal_gpio_t optiga_reset_0 =
        {
            // Platform specific GPIO context for the pin used to toggle
Reset
            (void*)&pin_3_3
        };
```

3. Update PAL I2C APIs [*pal_i2c.c*] to communicate with OPTIGA™ Trust X

The pal_i2c is expected to provide the APIs for I2C driver initialization, de-initialization, read, write and set bitrate kind of operations

a) *pal_i2c_init*

b) *pal_i2c_deinit*

c) *pal_i2c_read*

d) *pal_i2c_write*

e) *pal_i2c_set_bitrate*

In few target platforms, the I2C master driver initialization (*pal_i2c_init*) is done during the platform start up. In such an environment, there is no need to implement *pal_i2c_init* and *pal_i2c_deinit* functions. Otherwise, these (*pal_i2c_init* & *pal_i2c_deinit*) functions must be implemented as per the upper layer expectations based on the need. The details of these expectations are available in the Host library API documentation (chm).

The reference implementation of PAL I2C based on XMC4500 Relax kit does not need to have the platform I2C driver initialization explicitly done as part of *pal_i2c_init* as it is taken care by the DAVE library initialization. Hence *pal_i2c_init* & *pal_i2c_deinit* are not implemented.

In addition to the above specified APIs, the PAL I2C must handle the events from the low level I2C driver and invoke the upper layer handlers registered with PAL I2C context for the respective transaction as shown in the below example.

```
        //I2C driver callback function when the transmit is completed
successfully
        void i2c_master_end_of_transmit_callback(void)
        {
            invoke_upper_layer_callback(gp_pal_i2c_current_ctx,
                                (uint8_t)PAL_I2C_EVENT_TX_SUCCESS);
        }
```

In above example the I2C driver callback, when transmit is successful invokes the handler to inform the result.

4. Update PAL GPIO [*pal_gpio.c*] to power on and reset the OPTIGA™ Trust X

a) *pal_gpio_set_high*

b) *pal_gpio_set_low*

5. Update PAL Timer [*pal_os_timer.c*] to enable timer
   a) *pal_os_timer_get_time_in_milliseconds*
   b) *pal_os_timer_delay_in_milliseconds*
6. Update Event management for the asynchronous interactions for IFX I2C [*pal_os_event.c*]
   a) *pal_os_event_register_callback_oneshot*
   b) *scheduler_timer_isr*

The *pal_os_event_register_callback_oneshot* function is expected to register the handler and context provided as part of input parameters and triggers the timer for the requested time.

```
void pal_os_event_register_callback_oneshot(
                                          register_callback callback,
                                          void* callback_args,
                                          uint32_t time_us)
{
    callback_registered = callback;
    callback_ctx = callback_args;

    //lint --e{534} suppress "Return value is not required to be checked"
    TIMER_SetTimeInterval(&scheduler_timer , (time_us*100));

    TIMER_Start(&scheduler_timer);
}
```

And the handler registered must be invoked once the timer is elapsed as shown in *scheduler_timer_isr*

```
void scheduler_timer_isr(void)
{
    TIMER_ClearEvent(&scheduler_timer);
    //lint --e{534} suppress "Return value is not required to be checked"
    TIMER_Stop(&scheduler_timer);
    TIMER_Clear(&scheduler_timer);

    if (callback_registered)
    {
        callback_registered((void*)callback_ctx);
    }
}
```

6.6 Reference code on XMC4500 for communicating with OPTIGA™ Trust X

```
static volatile uint32_t optiga_pal_event_status;
static void optiga_pal_i2c_event_handler(void* upper_layer_ctx,
                                                  uint8_t event);

pal_i2c_t optiga_pal_i2c_context_0 =
{
    /// Pointer to I2C master platform specific context
    (void*)&i2c_master_0,
    /// Slave address
```

```
            0x30,
            /// Upper layer context
            NULL,
            /// Callback event handler
            NULL,
        };

        // OPTIGA pal i2c event handler
        static void optiga_pal_i2c_event_handler(void* upper_layer_ctx,
                                                  uint8_t event)
        {
            optiga_pal_event_status = event;
        }

        /* Function to verify I2C communication with OPTIGA */
        pal_status_t test_optiga_communication(void)
        {
            pal_status_t pal_return_status;
            uint8_t data_buffer[10] = {0x82};

            // set callback handler for pal i2c
            optiga_pal_i2c_context_0.upper_layer_event_handler =
                                  optiga_pal_i2c_event_handler;

            // Send 0x82 to read I2C_STATE from optiga
            do
            {
                 optiga_pal_event_status = PAL_I2C_EVENT_BUSY;
                 pal_return_status =
                            pal_i2c_write(&optiga_pal_i2c_context_0,
                                          data_buffer,
                                          1);
                if (PAL_STATUS_FAILURE == pal_return_status)
                {
                  // Pal I2C write failed due to I2C busy is in busy
                   // state or low level driver failures
                    break;
                }

            // Wait until writing to optiga is completed
            }  while (PAL_I2C_EVENT_SUCCESS != optiga_pal_event_status);


            // Read the I2C_STATE from OPTIGA
            do
            {
                optiga_pal_event_status = PAL_I2C_EVENT_BUSY;
                pal_return_status =
                             pal_i2c_read(&optiga_pal_i2c_context_0 ,
                                          data_buffer ,
                                          4);
               // Pal I2C read failed due to I2C busy is in busy
              // state or low level driver failures
               if (PAL_STATUS_FAILURE == pal_return_status)
                {
                    break;
```

```
            }
        // Wait until reading from optiga is completed
        } while (PAL_I2C_EVENT_SUCCESS != optiga_pal_event_status);

        return pal_return_status;
    }

    /* Main Function */
    int32_t main(void)
    {
        DAVE_STATUS_t status;
        pal_status_t pal_return_status;

        // Initialisation of DAVE Apps
        status = DAVE_Init();

        // Stop if DAVE init fails
        if (DAVE_STATUS_FAILURE == status)
        {
            while (1U)
            {;}
        }
        pal_return_status = test_optiga_communication();

        return (int32_t)pal_return_status;
    }
```

7 OPTIGA™ Trust X External Interface

7.1 Commands

This section provides short description of the commands exposed by the OPTIGA™ Trust X secuirty chip and mapping of these commands w.r.t Use Cases.

Table 13 OPTIGA™ Trust X command table

Command Name	Description
OpenApplication	Command to launch an application
GetDataObject	Command to get (read) a data object
SetDataObject	Command to set (write) a data object
GetRandom	Command to generate a random stream
SetAuthScheme	Command to set the authentication scheme which gets used subsequently
GetAuthMsg	Command to get (receive from OPTIGA™ Trust X) an authentication message
SetAuthMsg	Command to set (send to OPTIGA™ Trust X) an authentication message
ProcUpLinkMsg	Command to process an up-link message for DTLS(receive from OPTIGA™ Trust X)
ProcDownLinkMsg	Command to process a down-link message for DTLS (send to OPTIGA™ Trust X)
CalcHash	Command to calculate a Hash
CalcSign	Command to calculate a signature
VerifySign	Command to verify a signature
CalcSSec	Command to execute a Diffie-Hellmann key agreement
DeriveKey	Command to derive keys
GenKeyPair	Command to generate public/private key pairs

Table 14 Mapping of commands with Use cases

Use Case	OPTIGA™ Trust X commands used
Mutual Authentication using DTLS	SetAuthScheme, ProcUpLinkMsg & ProcDownLinkMsg
One Way Authentication	GetRandom, GetDataObject, SetAuthScheme, SetAuthMsg & GetAuthMsg
Crypto Toolbox	GetRandom, SetAuthScheme, SetAuthMsg, GetAuthMsg, CalcHash, CalcSign, VerifySign, CalcSSec, DeriveKey, GenKeyPair
Datastore (user memory ~ 10kB)	GetDataObject and SetDataObject

7.2 Crypto Performance

The performance metrics for various schemes are provided by the Table 15 below. If not particularly mentioned, the performance is measured @ OPTIGA™ Trust X I/O interface with:

- I2C FM (400KHz)
- Without power limitation
- @ 25°C

- VCC = 3.3V

Table 15 Crypto performance

Scheme	Algorithm	Performance in ms[1]	Notes
Calculate signature	FIPS 186-3	~ 60	Doesn't include message hashing before calling a toolbox function
Verify signature	FIPS 186-3	~ 85	
Diffie-Hellman key agreement	NIST 256	~ 65	Based on ephemeral key pair
Key pair generation	NIST 256	~ 80	Generate 256 bit ECC key pair
Key Derivation	TLS PRF SHA 256	~ 140	
Random Number Generation		~ 5 - ~ 7	16 bytes of true randome number generation
Write Data		~ 15	100 bytes of data
Read Data		~ 9	100 bytes of data
Hash calculation	SHA 256	~ 5 Kbyte/s	In blocks of 500 bytes

[1]Minimum Execution of the entire sequence in milli seconds, except the External World timings

8 Security Monitor

The Security Monitor is a central component which enforces the security policy of the OPTIGA™ Trust X. It consumes security events sent by security aware parts of the OPTIGA™ Trust X embedded SW and takes actions accordingly as specified in Security Policy below.

8.1 Security Events

The events below actively influence the security monitor.

Table 16 Security Events

Event	Description
Decryption Failure	This event occurs in case a decryption and/ or integrity check of provided data lead to an integrity failure.
Private Key Use	This event occurs in case the internal services are going to use an OPTIGA™ Trust X hosted private key.
Suspect System Behavior	This event occurs in case the embedded software detects inconsistencies with the expected behavior of the system. Those inconsistencies might be redundant information which doesn't fit to their counterpart.

8.2 Security Policy

Security Monitor judges the notified security events regarding the number of occurrence over time and in case those violate the permitted usage profile of the system takes actions to throttle down the performance and thus the possible frequency of attacks.

The permitted usage profile is defined as:

1. t_{max} is set to 5 seconds (± 5%)
2. A Suspect System Behavior event is never permitted and will cause setting the Security Event Counter (SEC) to its maximum (= 255).
3. One protected operation (refer to Table 16)) events per t_{max} period.

In other words it must not allow more than one out of the protected operations per t_{max} period (worst case, ref to bullet 1. above). This condition must be stable, at least after 500 uninterrupted executions of protected operations.

For more information, please refer to Solution Reference Manual document available as part of the package.

9 RoHS Compliance

On January 27, 2003 the European Parliament and the council adopted the directives:

- 2002/95/EC on the Restriction of the use of certain Hazardous Substances in electrical and electronic equipment ("RoHS")
- 2002/96/EC on Waste Electrical and Electrical and Electronic Equipment ("WEEE")

Some of these restricted (lead) or recycling-relevant (brominated flame retardants) substances are currently found in the terminations (e.g. lead finish, bumps, balls) and substrate materials or mold compounds.

The European Union has finalized the Directives. It is the member states' task to convert these Directives into national laws. Most national laws are available, some member states have extended timelines for implementation. The laws arising from these Directives have come into force in 2006 or 2007.

The electro and electronic industry has to eliminate lead and other hazardous materials from their products. In addition, discussions are on-going with regard to the separate recycling of ceratin materials, e.g. plastic containing brominated flame retardants.

Infineon Technologies is fully committed to giving its customers maximum support in their efforts to convert to lead-free and halogen-free[1] products. For this reason, Infineon Technologies' "Green Products" are ROHS-compliant.

Since all hazardous substances have been removed, Infineon Technologies calls its lead-free and halogen-free semiconductor packages "green." Details on Infineon Technologies' definition and upper limits for the restricted materials can be found here.

The assembly process of our high-technology semiconductor chips is an integral part of our quality strategy. Accordingly, we will accurately evaluate and test alternative materials in order to replace lead and halogen so that we end up with the same or higher quality standards for our products.

The use of lead-free solders for board assembly results in higher process temperatures and increased requirements for the heat resistivity of semiconductor packages. This issue is addressed by Infineon Technologies by a new classification of the Moisture Sensitivity Level (MSL). In a first step the existing products have been classified according to the new requirements.

[1]Any material used by Infineon Technologies is PBB and PBDE-free. Plastic containing brominated flame retardants, as mentioned in the WEEE directive, will be replaced if technically/economically beneficial.

10 Appendix A – Infineon I2C Protocol Registry Map

OPTIGA™ Trust X supports IFX I2C v1.65 and is implemented as I2C slave, which uses different address locations for status, control and data communication registers. These registers with description are outlined below in the following table.

Table 17 IFX I2C Registry Map Table

Register Address	Name	Size in Bytes	Description	Master Access
0x80	DATA	DATA_REG_LEN	This is the location where data shall be read from or written to the I2C slave	Read / Write
0x81	DATA_REG_LEN	2	This register holds the maximum data register (Addr 0x80) length. The allowed values are 0x0010 up to 0xFFFF. After writing the new data register length it becomes effective with the next I2C master access. However, in case the slave could not accept the new length it indicates its maximum possible length within this register. Therefore it is recommended to read the value back after writing it to be sure the I2C slave did accept the new value. Note: the value of MAX_PACKET_SIZE is derived from this value or vice versa (MAX_PACKET_SIZE= DATA_REG_LEN-5)	Read / Write
0x82	I2C_STATE	4	Bits 31:24 of this register provides the I2C state in regards to the supported features (e.g. clock stretching …) and whether the device is busy executing a command and/or ready to return a response etc. Bits 15:0 defining the length of the response data block at the physical layer.	Read only
0x83	BASE_ADDR	2	This register holds the I2C base address as specified by Table 18. If not differently defined by a particular project the default value at reset is 0x30. After writing a different address the new address become effective with the next I2C master access. In case the bit 15 is set in addition to the new address (bit 6:0) it becomes the new default address at reset (persistent storage).	Write only
0x84	MAX_SCL_FREQU	4	This register holds the maximum clock frequency in KHz supported by the I2C slave. The value gets adjusted to the register I2C_Mode setting. Fast Mode (Fm): The allowed values are 50 up to 400. Fast Mode (Fm+): The allowed values are 50 up to 1000.	Read
0x85	GUARD_TIME[1]	4	For details refer to Table 21	Read only

[1] In case the register returns 0xFFFFFFFF the register is not supported and the default values specified in Table 'List of protocol variations' shall be applied.

Register Address	Name	Size in Bytes	Description	Master Access
0x86	TRANS_TIMEOUT[1]	4	For details refer to Table 21	Read only
0x88	SOFT_RESET	2	Writing to this register will cause a device reset. This feature is optional	Write only
0x89	I2C_MODE	2	This register holds the current I2C Mode as defined by Table 19. The default mode is SM & FM (011B).	Read / Write

Table 18 Definition of BASE_ADDR

Fields	Bits	Value	Description
DEF_ADDR	15	0 1	Volatile address setting by bit 6:0, lost after reset. Persistent address setting by bit 6:0, becoming default after reset.
BASE_ADDR	6:0	0x00-0x7F	I²C base address specified by Table 17

15	14	13	12	11	10	9	8
DEF_ADDR	RFU						
7	6	5	4	3	2	1	0
RFU	BASE_ADDR						

15	14	13	12	11	10	9	8
DEF_MODE	RFU						
7	6	5	4	3	2	1	0
RFU					Mode		

Table 19 Definition of I2C_MODE

Fields	Bits	Value	Description
DEF_MODE	15	0 1	Volatile mode setting by bit 2:0, lost after reset. Persistent mode setting by bit 2:0, becoming default after reset. This bit is always read as 0.
MODE[2]	2:0	001 010 011 100 other values	Sm Fm SM & Fm (fab out default) Fm+ not valid; writing will be ignored

[1] In case the register returns 0xFFFFFFFF the register and its functionality is not supported

[2] This mode defines the adherence of the bus signals to the electrical characteristics according standard I2C bus specification

31	30	29	28	27	26	25	24
BUSY	RESP_RDY	RFU		SOFT_RESET	CONT_READ	REP_START	CLK_STRETCHING
23	22	21	20	19	18	17	16
RFU							
15-0							
Length of data block to be read							

Table 20 Definition of I2C_STATE

Field	Bit(s)	Value	Description
BUSY	31	0 1	Device is not busy Device is busy executing a command
RESP_RDY	30	0 1	Device is not ready to return a response Device is ready to return a response
SOFT_RESET	27	0 1	SOFT_RESET not supported SOFT_RESET supported
CONT_READ	26	0 1	Continue Read not supported Continue Read supported
REP_START	25	0 1	Repeated start not supported Repeated start supported
CLK_STRETCHING	24	0 1	Clock stretching not supported Clock stretching supported

10.1 IFX I2C Protocol Variations

To fit best to application specific requirements the protocol might be tailored by specifying a couple of parameters which is described in the following table.

Table 21 List of Protocol Variations

Parameter	Default Value	Description
MAX_PACKET_SIZE	0x110	Maximum packet size accepted by the receiver. The protocol limits this value to 0xFFFF, but there might be project specific requirements to reduce the transport buffers size for the sake of less RAM footprint in the communication stack. If shortened, it could be statically defined or negotiated at the physical layer.
WIN_SIZE	1	Window size of the sliding windows algorithm. The value could be 1 up to 2.
MAX_NET_CHAN	1	Maximum number of network channels. The value could be 1 up to 16. One indicates the OSI Layer 3 is not used and the CHAN field of the PCTR must be set to 0000.
CHAINING	TRUE	Chaining on the transport layer is supported (TRUE) or not (FALSE)
TRANS_TIMEOUT	10 ms	(Re) transmission timeout specifies the number of milliseconds to be elapsed until the transmitter considers a frame transmission is lost and retransmits the non-acknowledged

Parameter	Default Value	Description
		frame. The Timer gets started as soon as the complete frame is transmitted. The value could be 1 up to 1000. However, as higher the number as longer does it take to recover from a frame transmission error. *Note: The acknowledge timeout on the receiver side must be shorter than the retransmission timeout to avoid unnecessary frame repetitions.*
TRANS_REPEAT	3	Number of transmissions to be repeated until the transmitter considers the connection is lost and starts a re-synchronization with the receiver. The value could be 1 up to 4.
BASE_ADDR	0x30	I2C (base) address. This address could be statically defined or dynamically negotiated by the physical layer. If not different specified the default value is 0x30.
MAX_SCL_FREQU	1000 kHz	Maximum SCL clock frequency in kHz.
GUARD_TIME	50 µs	Minimum time to be elapsed at the I2C master measured from read data (STOP condition) until the next write data (Start condition) is allowed to happen. *Note 1: For two consecutive accesses on the same device GUARD_TIME re-specifies the value of t_{BUF} as specified by [I2Cbus].* *Note 2: Even if another I2C address is accessed in between GUARD_TIME has to be respected for two consecutive accesses on the same device.*
SOFT_RESET		Any write attempt to the SOFT_RESET register will trigger a warm reset (reset w/o power cycle). This register is optional and its presence is indicated by the I2C_STATE register's "SOFT_RESET" flag.

11 Appendix B - OPTIGA™ Trust X Command/Response I2C Sample Logs

The default I2C slave address for the OPTIGA™ Trust X is 0x30 [I2C_ADDR]. All the values in this section are specified in decimal form unless stated otherwise.

11.1 Sequence of commands to read Coprocessor UID from OPTIGA™ Trust X

Pre-requisites

1. Ensure that the security device is powered up
2. The OPTIGA™ Trust X will not acknowledge the slave address sent by a host if it is either busy or in idle state. Hence the host must retry or repeat the transaction until it is successful or timed out for 100 milliseconds (extreme case).
3. The specified guard time must be applied between each attempt of write / read operation by the Host I2C driver.
4. The log information for OPTIGA™ Trust X commands specified in below Tables contains the [IFX I2C] protocol information which comprises sequence numbers and checksum of the transactions.
    a. A sequence of commands must be strict for the OPTIGA™ Trust X (e.g. OpenApplication followed by GetDataObject to read a Coprocessor UID)
    b. A checksum in the data depends on the data received or sent via write/read operations. So any data change in the transaction is reflected in the check sum. Otherwise the write data transaction will not be accepted/acknowledged by the OPTIGA™ Trust X.

11.1.1 Check the status [I2C_STATE]

This is a very basic register read operation which ensures the behavior of the read/write operations of the local host I2C driver.

Table 22 Check I2C_STATE Register of OPTIGA™ Trust X

I2C_ADDR	Transaction Type	Data [values in hexadecimal]
30	Write [01 Bytes]	82
30	Read [04 Bytes]	08 00 00 00

11.1.2 Issue OpenApplication command

Before issuing any application specific command; e.g. read Coprocessor UID using GetDataObject, it is a must to send the OpenApplication command to initialize the application on the OPTIGA™ Trust X as shown below.

Table 23 OpenApplication on OPTIGA™ Trust X

I2C_ADDR	Transaction Type	Data [values in hexadecimal]
Step 1: Send OpenApplication command to initiate the application context on the OPTIGA™ Trust X		
30	Write [27 Bytes]	80 03 00 15 00 **70 00 00 10 D2 76 00 00 04 47 65 6E 41 75 74 68 41 70 70** 6C 04
Step 2: Read the I2C_STATE register [*Repeat this step until the read contains the data as specified below*]		
30	Write [01 Bytes]	82

I2C_ADDR	**Transaction Type**	**Data [values in hexadecimal]**
30	Read [04 Bytes]	C8 00 00 05
Step 3: Read the DATA register [*Acknowledgment from* OPTIGA™ Trust X *for the last data transacation*]		
30	Write [01 Bytes]	80
30	Read [05 Bytes]	80 00 00 0C EC
Step 4: Read the I2C_STATE register [*Repeat this step until the read contains the data as specified below*]		
30	Write [01 Bytes]	82
30	Read [04 Bytes]	48 00 00 0A
Step 5: Read the DATA register which contains the response for the command issued		
30	Write [01 Bytes]	80
30	Read [10 Bytes]	00 00 05 00 **00 00 00 00** 14 87
Step 6: Send an acknowlegment for the data read		
30	Write [06 Bytes]	80 80 00 00 0C EC

11.1.3 Read Coprocessor UID

The Coprocessor UID contains the OPTIGA™ Trust X unique ID and the build information details. The GetDataObject command is used to read the Coprocessor UID information.

Table 24 Read Coprocessor UID

I2C_ADDR	**Transaction Type**	**Data [values in hexadecimal]**
Step 1: Send the GetDataObject command to read the Coprocessor UID		
30	Write [17 Bytes]	80 04 00 0B 00 **01 00 00 06 E0 C2 00 00 00 64** F0 9F
Step 2: Read the I2C_STATE register [*Repeat this step until the read contains the data as specified below*].		
30	Write [01 Bytes]	82
30	Read [04 Bytes]	48 00 00 25
Step 3: Read the DATA register which contains the response for the command issued.		
30	Write [01 Bytes]	80
30	Read [37 Bytes]	05 00 20 00 **00 00 00 1B CD XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX XX YY YY** ZZ ZZ Notes: a. XX is the unique ID part of the co-processor UID b. "YY YY" is the OPTIGA™ Trust X build number in BCD (Binary Coded Decimal) format c. ZZ ZZ is the checksum of the transaction
Step 4: Send an acknowlegment for the data read		
30	Write [06 Bytes]	80 81 00 00 56 30

12 Appendix B – Power Management

When operating, the power consumption of OPTIGA™ Trust X is limited to meet the requirements regarding the power limitation set by the Host. The power limitation is implemented by utilizing the current limitation feature of the underlying hardware device in steps of 1mA from 6mA to 15 mA with a precision of ±5%.

12.1 Low Power Sleep Mode

The OPTIGA™ Trust X automatically enters a low-power mode after a configurable delay. Once it has entered Sleep mode, the OPTIGA™ Trust X resumes normal operation as soon as its address is detected on the I2C bus. In case no command is sent to the OPTIGA™ Trust X it behaves as shown in Figure 14.

1. As soon as the OPTIGA™ Trust X is idle it starts to count down the "delay to sleep" time (t_{SDY}).
2. In case this time elapses the device enters the "go to sleep" procedure.
3. The "go to sleep" procedure waits until all idle tasks are finished (e.g. counting down the SEC). In case all idle tasks are finished and no command is pending, the OPTIGA™ Trust X enters sleep mode.

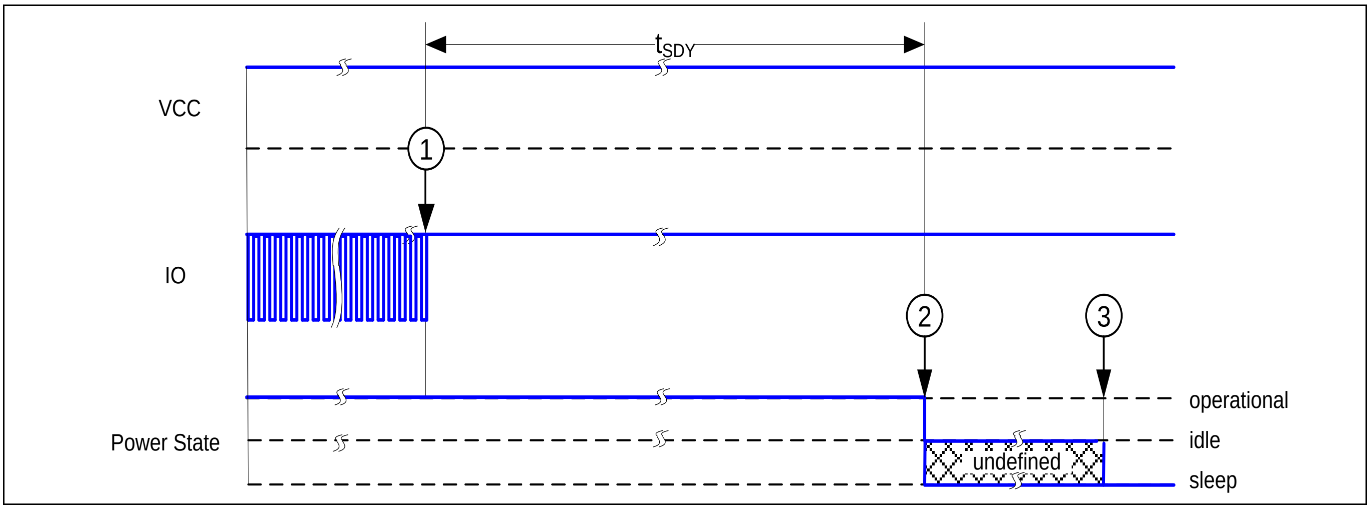


Figure 14 Go-to-Sleep Diagram

Revision history

Document version	Date of release	Description of changes
2.70	26.02.2019	Feedback incorporation. Updated diagrams. Added link to CC certification.
2.6	08.02.2019	Updated PG-USON10-2 foot print
2.5	31.01.2018	Feedback incorporation from all internal regions
2.4	11.01.2018	Feedback incorporation from all internal regions
2.3	01.01.2018	Feedback incorporation from all internal regions
2.2	12.12.2017	Feedback from all internal regions
2.1	23.06.2017	Updated Key features and Enhanced Security
2.0	08.06.2017	Updated Key features and Enhanced Security
1.4	22.02.2017	First version release
1.3		Internal review
1.2		Internal review
1.1		Internal review
1.0		Internal review

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2019-02-26

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email:
security.chipcard.ics@infineon.com

Document reference