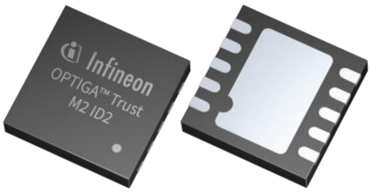
OPTIGA™ Trust M2 ID2

Product Version: V2

Key Features



- High-end security controller
- Common Criteria Certified EAL6+ (high) hardware
- Turnkey solution
- Up to 10kB user memory
- PG-USON-10-2,-4 package (3 x 3 mm)
- Standard & Extended temperature ranges
- I2C interface with Shielded Connection (encrypted communication)
- Cryptographic support:
  - RSA® up to 2048,
  - AES key up to 256 , HMAC up to SHA512,
  - TLS v1.2 PRF and HKDF up to SHA512
- OPTIGA™ Trust M2 ID2 Software Framework on Github - https://github.com/Infineon/optiga-trust-m2-id2
- Crypto ToolBox commands for SHA-256, RSA® Feature, AES, HMAC and Key derivation
- Alibaba Cloud IoT connectivity
- Configurable device security monitor, 4 Monotonic up counters
- Protected(integrity and confidentiality) update of data, key and metadata objects
- Hibernate for zero power consumption[1]
- 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 and corporate image
- Safeguarding of quality and safety

Applications

- Industrial control and building automation
- Consumer electronics, Smart Home and Drones

About this document

Scope and purpose

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

Intended audience

This Datasheet is intended for device integrators and board manufacturers.

[1] Leakage current < 2.5µA only

Table of Contents

Table of Contents

1 Introduction

As embedded systems (e.g. IoT devices) are increasingly gaining the attention of attackers, Infineon offers the OPTIGA™ Trust M2 ID2 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 M2 ID2 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 M2 ID2 is based on an advanced security controller with built-in tamper proof NVM for secure storage and Symmetric/Asymmetric crypto engines to support RSA® up to 2048, AES key up to 256, HMAC up to SHA512, HKDF up to SHA512 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 M2 ID2 comes with preprogrammed OS/Application code locked and with host-side modules to integrate with host micro controller software. The temperature range of −40°C to +105°C combined with a standardized I2C interface and the small PG-USON-10-2,-4 footprints 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 results of Infineon's strong expertise and its commitment to make security a success factor for you.

1.4 Applications

The OPTIGA™ Trust M2 ID2 covers a broad range of use cases necessary for many types of applications that include the following:

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

1.5 Device Features

The OPTIGA™ Trust M2 ID2 comes with up to 10kB of user memory that can be used to store X.509 certificates and data. OPTIGA™ Trust M2 ID2 is based on Common Criteria (CC) 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 (Hardware Identifier IFX_CCI_00000Bh) and referring to the latest CC certificate. OPTIGA™ Trust M2 ID2 supports a highspeed I2C communication interface of up to 1MHz (FM+).

Table 1 Products

Sales Code	Temperature range	Package	Description	Evaluation Kit
OPTIGA™ Trust M2 ID2 SLS 32AIA010I2	−40°C to +105°C Extended Temperature Range (ETR)	PG-USON-10-2,-4	Embedded security solution for connected devices	ESP32-DevKitC V4 connected to the OPTIGA™ Trust M2 ID2 to connect to the outside world.
OPTIGA™ Trust M2 ID2 SLS 32AIA010I3	−25°C to +85°C Standard Temperature Range (STR)	PG-USON-10-2,-4		

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 Features

Features	Supported Curve/Algorithm	ToolBox commands
RSA®	RSA® 1024/2048	Sign, Verify, Key generation, Encrypt and Decrypt
Key Derivation	TLS v1.2 PRF SHA 256/384/512	TLS PRF using SHA 256/384/512
	HKDF SHA-256/384/512	HKDF using SHA256/384/512
AES	Key size - 128/192/256 (ECB, CBC, CBC-MAC, CMAC)	Key generation, Encrypt and Decrypt
Random generation	TRNG, DRNG, Pre-Master secret for RSA® Key exchange	Generate random
HMAC	HMAC with SHA256/384/512	HMAC generation and Verification
Hash	SHA 256	Hash generation
Protected Data/key/metadata update (Integrity and/or confidentiality)	ECC NIST P256/384/521 ECC Brainpool P256/384/512 r1 RSA® 1024/2048 Signature scheme as ECDSA FIPS 186-3/RSA SSA PKCS#1 v1.5 without hashing	Secure data/key object update and metadata update for Data/key object

Table 3 Abbreviations

Abbreviation	Definition
AES	Advanced Encryption Standard
API	Application Programming Interface
CA	Certification Authority
CC	Common Criteria
DRNG	Deterministic Random Number Generator
DTLS	Datagram Transport Layer Security
EAL	Evaluation Assurance Level
ECB	Electronic Code Book
ETR	Extended Temperature Range
CBC	Cipher block chaining
CBC-MAC	Cipher block chaining message authentication code
CMAC	Cipher-based message authentication code
HKDF	Hash-based key derivation function
I2C	Inter-Integrated Circuit
IETF	Internet Engineering Task Force
IFX	Infineon
IOT	Internet of Things
IP	Intellectual Property
NIST	National Institute of Standards and Technology
OS	Operating System
PAL	Platform Abstraction Layer
PKI	Public Key Infrastructure
RFC	Request For Comments
SHA	Secure Hash Algorithm
SKU	Stock Keeping Unit
STR	Standard Temperature Range
TLS	Transport Layer Security
TRNG	True Random Number Generator
USB	Universal Serial Bus
HMAC	Hash based Message Authentication Code

2 System Block Diagram

The following figure depicts the system block diagram for OPTIGA™ Trust M2 ID2.

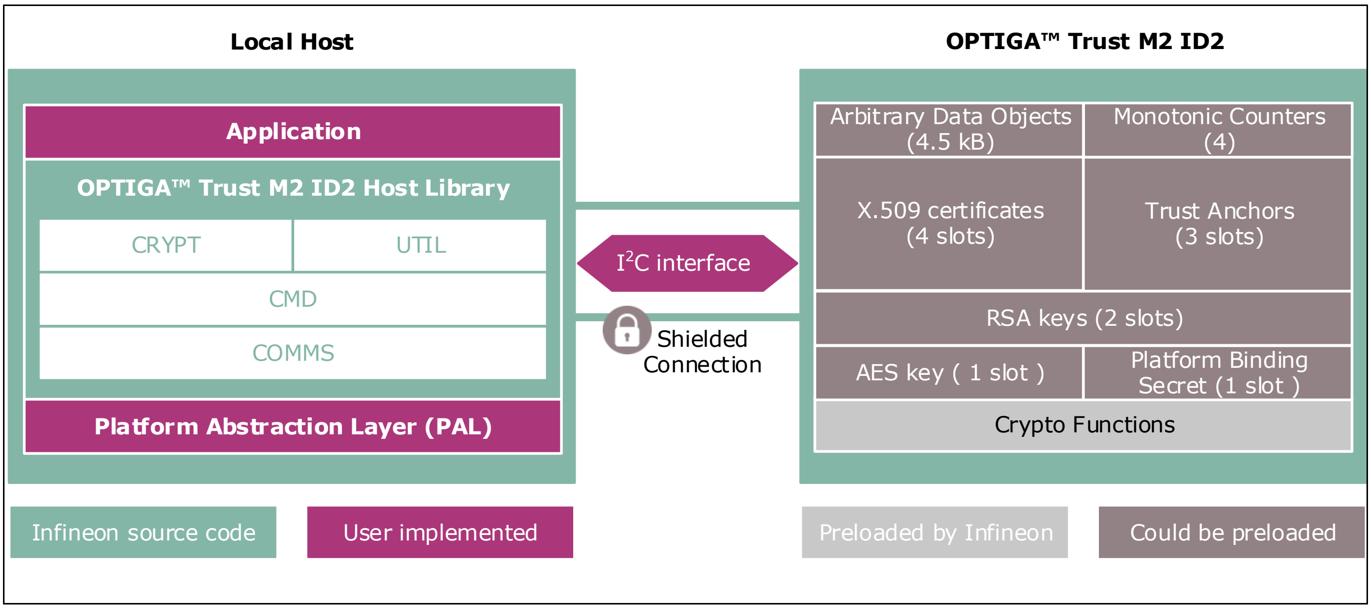


Figure 1 System Block Diagram

The System Block Diagram is explained below for each layer.

1. Local Host
   - Local Host Application – This is the target application which utilizes OPTIGA™ Trust M2 ID2 for its security needs
   - OPTIGA™ Trust M2 ID2 Host Library
     - CRYPT – Provides APIs to perform cryptographic functionalities. Any TLS stack can be integrated on Local Host as part of 3rd party Crypto Library to offload crypto operations to OPTIGA™ Trust M2 ID2.
     - UTIL – Provides APIs such as read/write, protected update of data, metadata, key objects and open/close application (e.g. Hibernate)
     - CMD – Provides APIs to send and receive commands (Section 7) to and from OPTIGA™ Trust M2 ID2
     - COMMS – Provides wrapper APIs for communication (optional encrypted communication using Shielded Connection) with OPTIGA™ Trust M2 ID2 which internally uses Infineon I2C Protocol (IFX I2C)
   - PAL – A layer that abstracts platform specific drivers (e.g. I2C, Timer, GPIO, platform crypto library etc.)
2. OPTIGA™ Trust M2 ID2
   - Arbitrary Data Objects – The target application can store up to 4.5kB (~4600 bytes) of data into OPTIGA™ Trust M2 ID2. The data could be additional Trust Anchors, certificates and shared secret.
   - Monotonic Counters - Provides 4 monotonic counting data objects (up counters). These can be used as general purpose counter or as linked counter to other objects.

System Block Diagram

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

- X.509 – Up to 4 X.509 based Certificates can be stored
- Keys – Up to 2 RSA and 1 AES based keys can be stored
- Secret – 1 Platform binding secret can be stored
- Trust Anchors – 3 slots, for Mutual Authentication (TLS/DTLS) and Firmware Updates can be stored
- Crypto Functions - OPTIGA™ Trust M2 ID2 provides cryptographic functions that can be invoked via local host

Note: *Unique AES key, RSA private keys and X.509 Certificates – During production at Infineon fab, unique asymmetric keys (private and public) are generated and symmetric key/shared secrets are provisioned. The public key is signed by customer specific CA and the resulting X.509 certificate issued is securely stored in the OPTIGA™ Trust M2 ID2. Special measures are taken to prevent the leakage and modification of private key/shared secret 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

The following figure illustrates how to integrate OPTIGA™ Trust M2 ID2 with your local host.

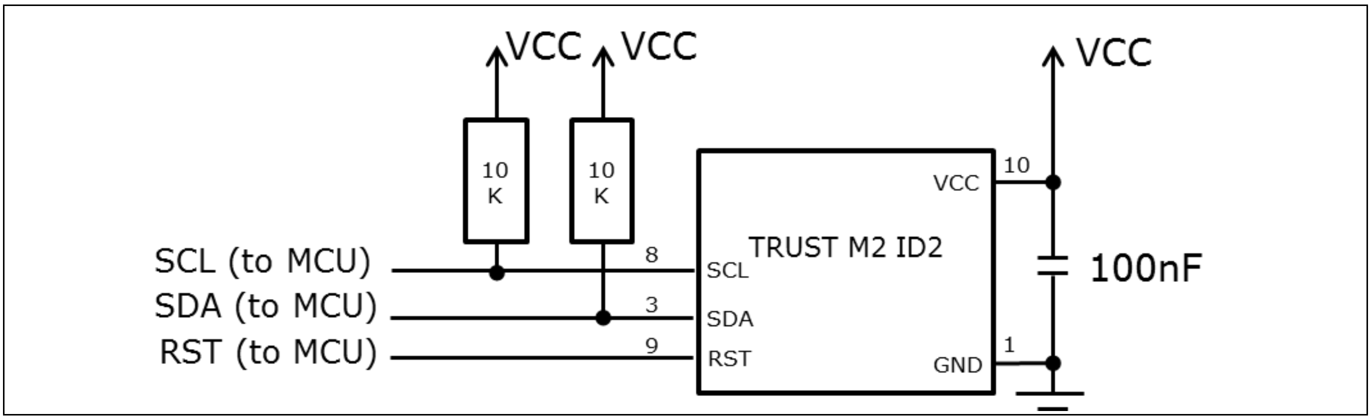


Figure 2 System Integration Schematic Diagram

Note: Value of the pullup resistors depend on the target application circuit and the target I2C frequency.

3.2 System Integration Schematics with Hibernation support

The following figure illustrates how to integrate OPTIGA™ Trust M2 ID2 with hibernation, with your local host.

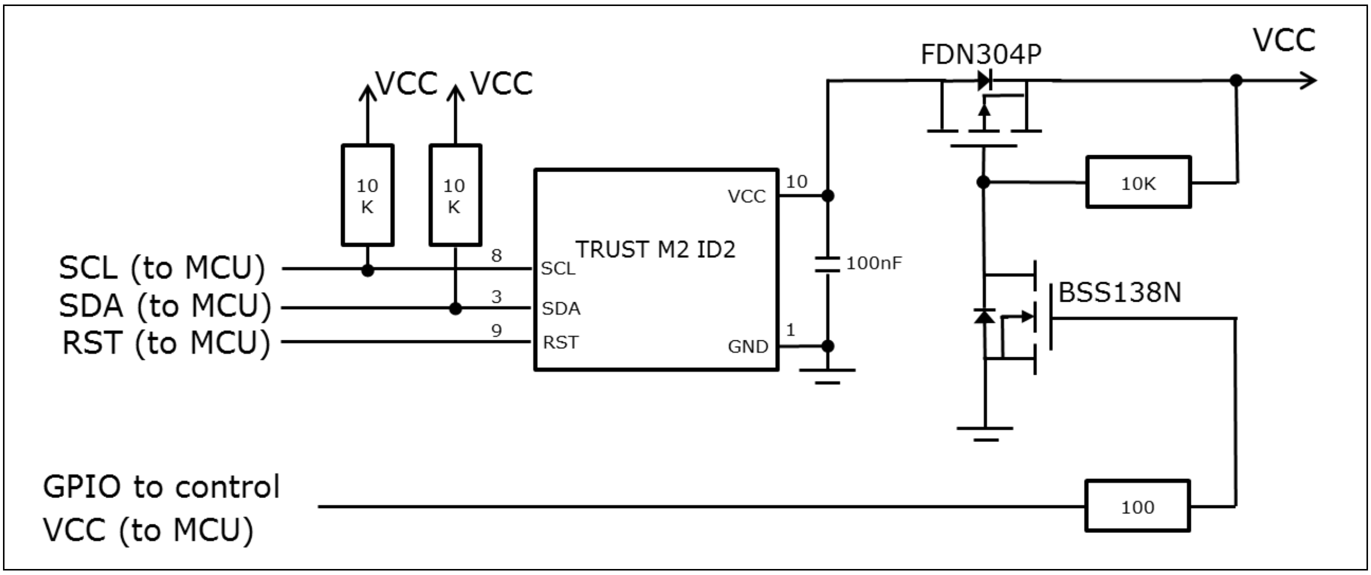


Figure 3 System Integration Schematic Diagram with Hibernation

Note: Value of the pullup resistors depend on the target application circuit and the target I2C frequency.

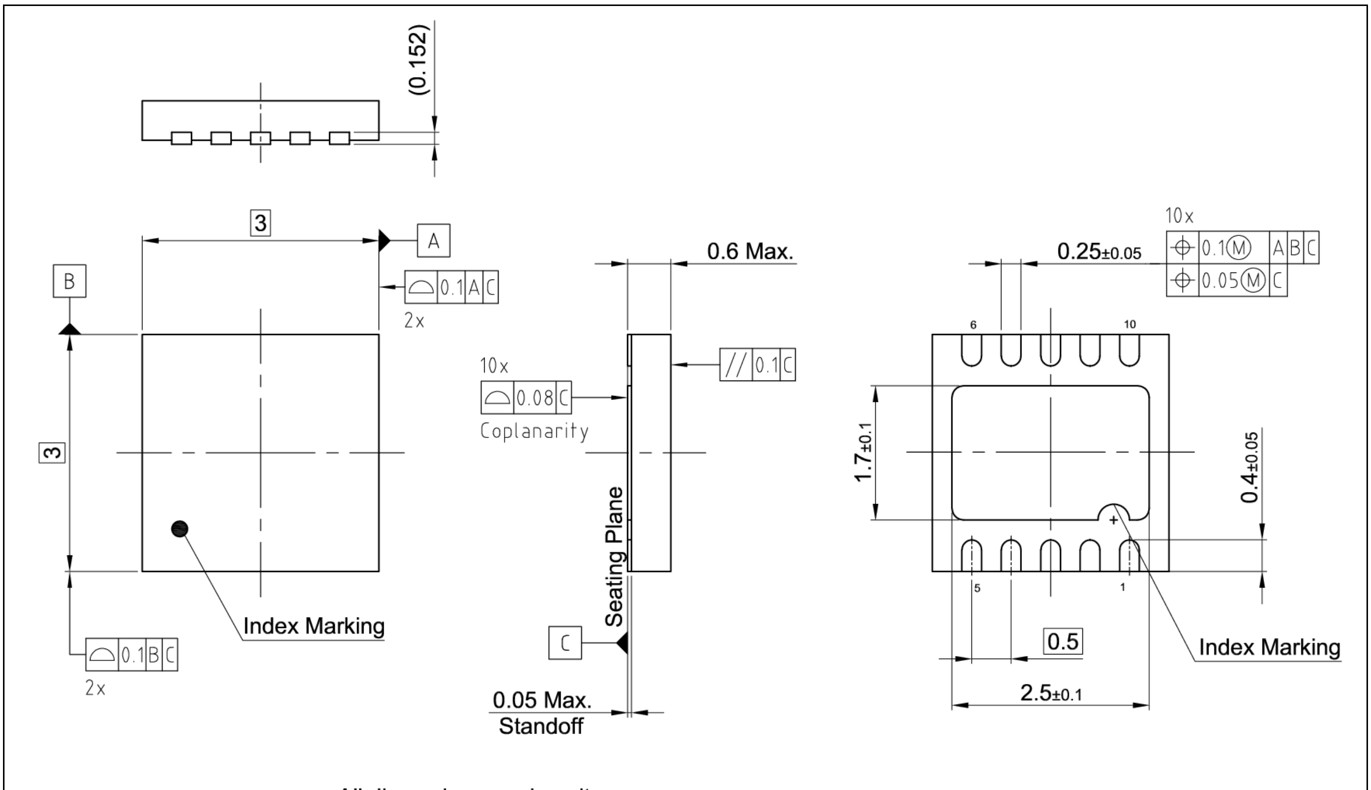
4 Description of packages

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,-4

The package dimensions (in mm) of the controller in PG-USON-10-2,-4 packages are given below.



All dimensions are in units mm
The drawing is in compliance with ISO 128-30, Projection Method 1 [⊲⊕]

Figure 4 PG-USON-10-2,-4 Package Outline

The following figure shows the PG-USON-10-2,-4 in top view:

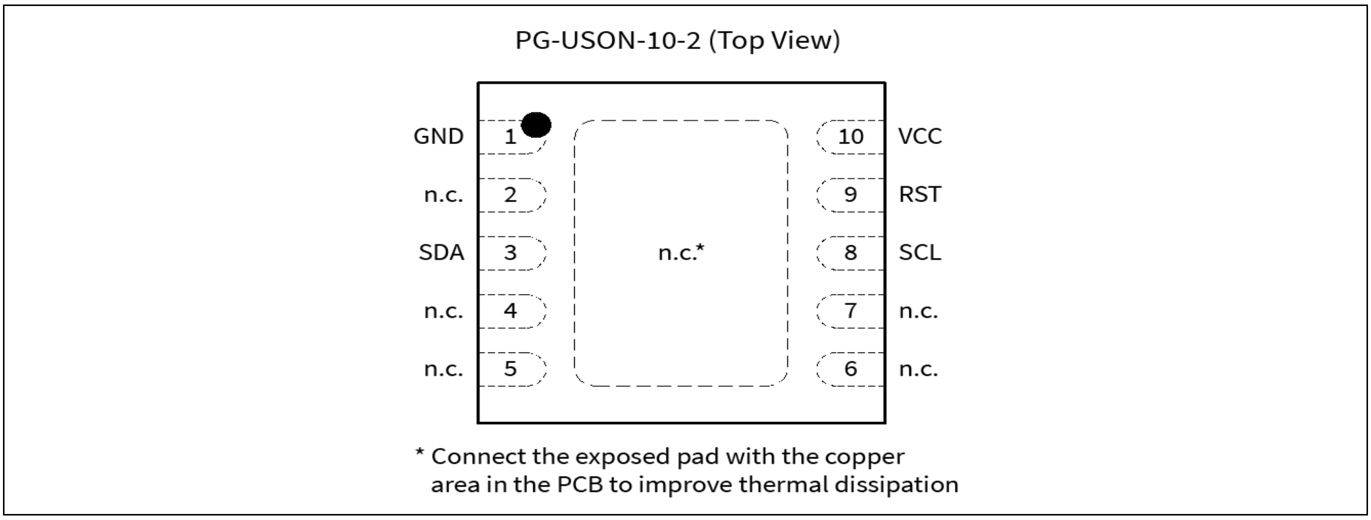

PG-USON-10-2 (Top View)
GND
1
n.c.
2
SDA
3
n.c.
4
n.c.
5
n.c.*
10
VCC
9
RST
8
SCL
7
n.c.
6
n.c.
* Connect the exposed pad with the copper area in the PCB to improve thermal dissipation

Figure 5 **PG-USON-10-2,-4 top view**

4.2 Production sample marking pattern

The following figure describes the productive sample marking pattern on PG-USON-10-2,-4.

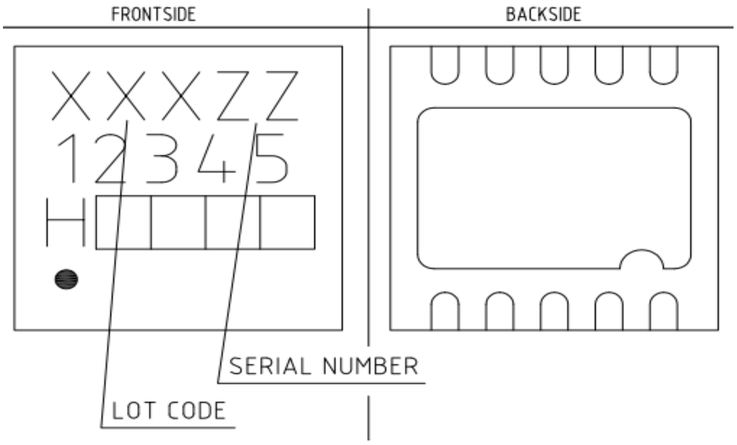


Figure 6 **PG-USON-10-2,-4 sample marking pattern**

The black dot indicates pin 01 for the chip. The following Table 4 describes the sample marking pattern:

Table 4 **Marking table for PG-USON-10-2,-4 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 the product is a Trust M controller • # indicates the controller is a STR (S) variant • $ specifies the OPTIGA™ Trust M2 ID2 release version number • @ specifies the software version Example: "TMS20" means 'OPTIGA™ Trust M2 ID2', 'STR variant', 'release version 1', 'software version 0'

The contacts and their functionality are given in the Table 5 below.

Table 5 Contact definitions and functions of PG-USON-10-2,-4 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 6 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 I^2C 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 7 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 8 I2C Fast Mode Interface Characteristics

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 9 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.*

5.1.3.1 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 10 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}	–	14.0	–	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	–	0.3 * 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
Hibernate current	–	–	< 2.5	–	µA	V_{cc} = 0 V, GND = 0 V, RST = 0 V, SCL= 3.3 V and SCL = 3.3 V

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

5.1.3.2 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 11 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.

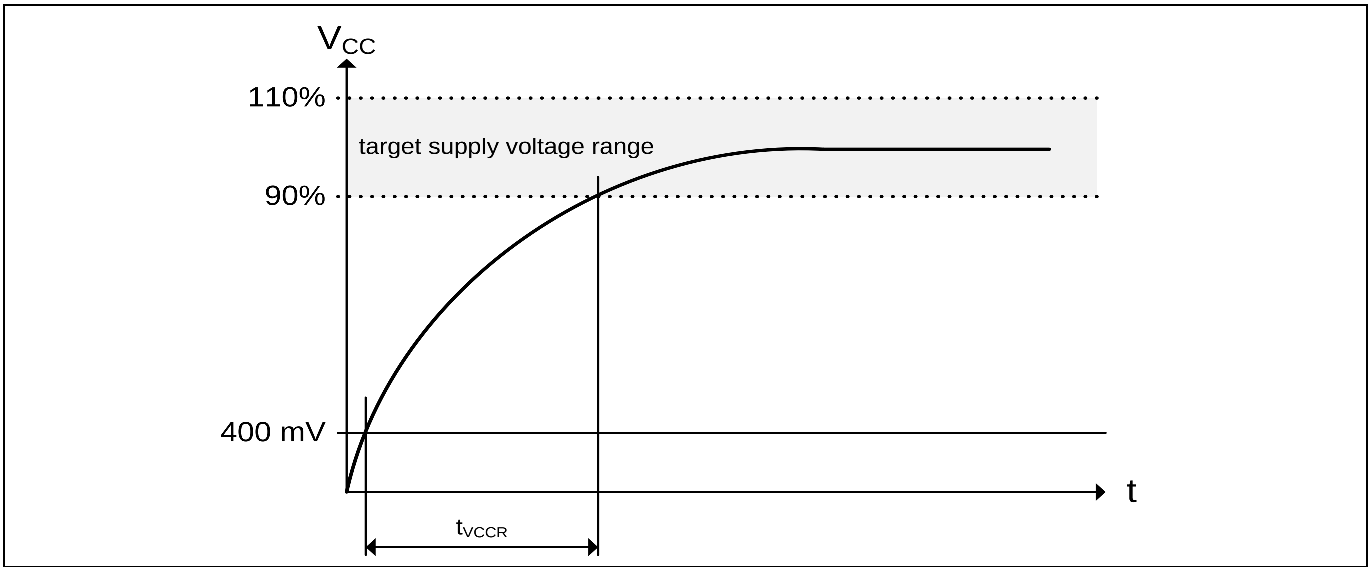


Figure 7 V_{cc} **Rampup**

5.1.4 Start-Up of I2C Interface

There are 2 variants possible for performing the startup procedure:

- Startup after power-on
- Startup for warm resets

5.1.4.1 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.

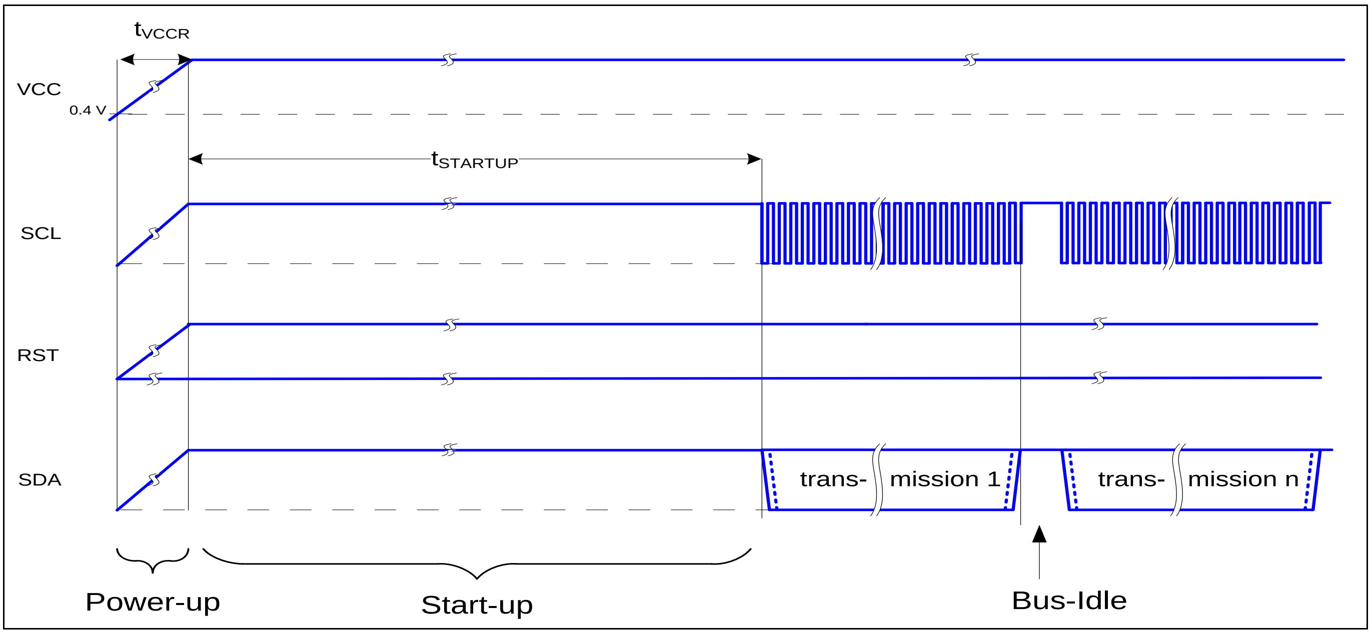


Figure 8 Startup of I2C Interface after Power-On

Table 12 Startup of I2C Interface After Power-On

Parameter	Symbol	Values			Unit	Note or Test Condition
		Min.	Typ.	Max.		
Startup time	$t_{STARTUP}$	15	–	–	ms	

5.1.4.2 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 t_1. At the latest t_2 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.

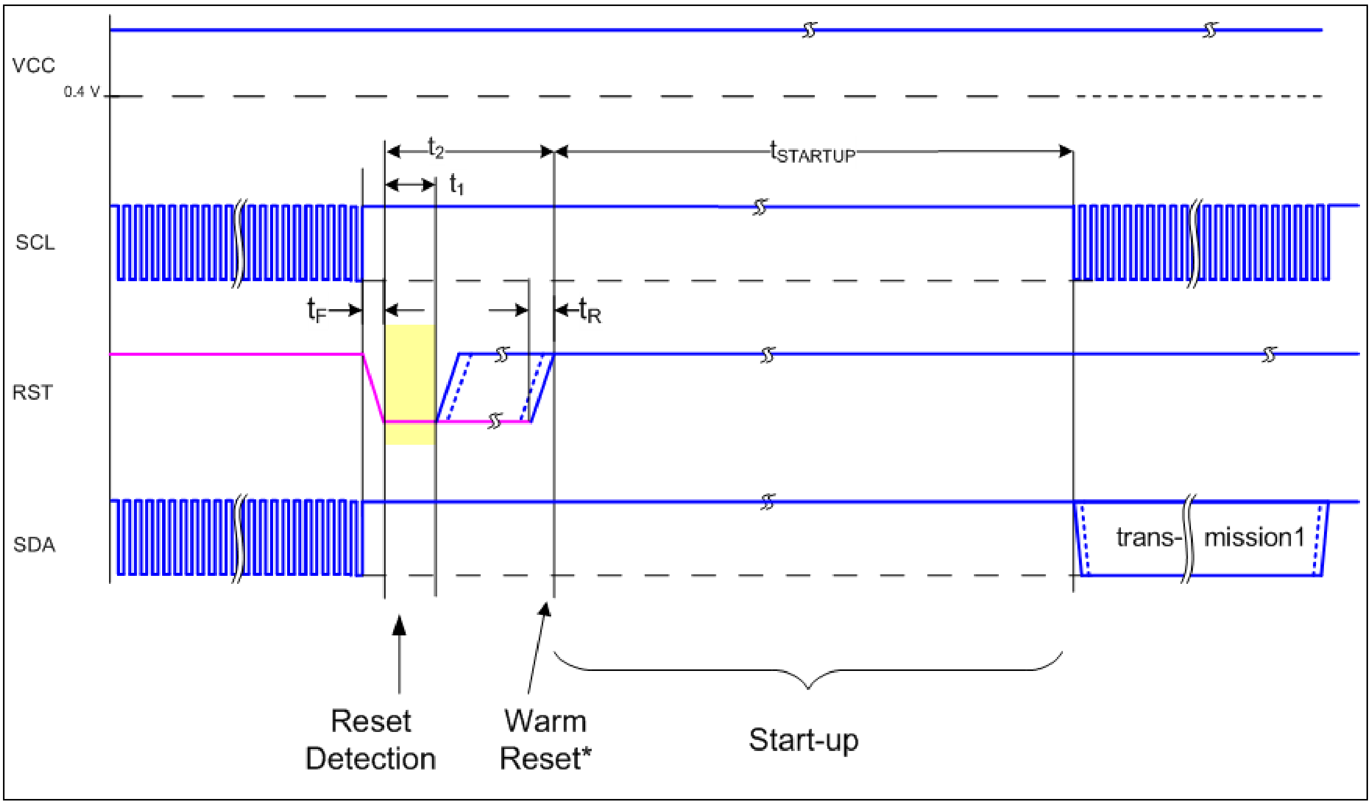


Figure 9 **Startup of I2C Interface for Warm Resets**

Note: If NVM programming was requested prior to the reset, $t_{STARTUP}$ will be extended from a typical value of 15 ms to a maximum of 20 ms.

Table 13 Startup of I2C Interface for Warm Resets[1]

Parameter	Symbol	Values			Unit	Note or Test Condition
		Min.	Typ.	Max.		
Startup time	$t_{STARTUP}$	15	–	–	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 M2 ID2 Host Software Architecture

The OPTIGA™ Trust M2 ID2 Host Library layers were explained in System Block Diagram Figure 1. In following sections, we will cover how to communicate with OPTIGA™ Trust M2 ID2 using I2C.

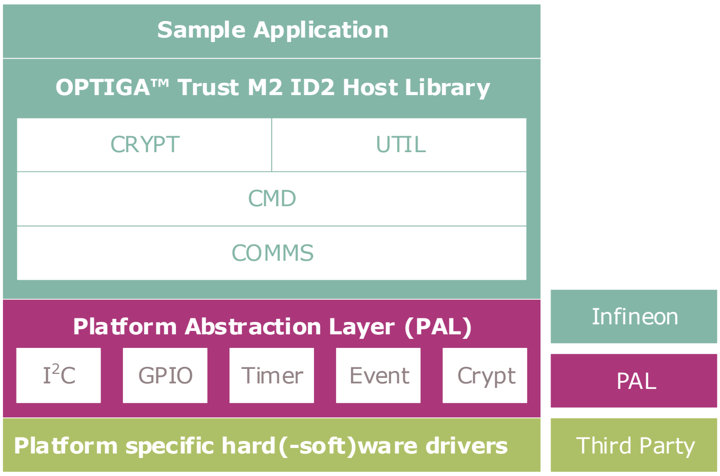

Sample Application
OPTIGA™ Trust M2 ID2 Host Library
CRYPT
UTIL
CMD
COMMS
Platform Abstraction Layer (PAL)
I²C
GPIO
Timer
Event
Crypt
Platform specific hard(-soft)ware drivers
Infineon
PAL
Third Party

Figure 10 OPTIGA™ Trust M2 ID2 Host Software Architecture

6.2 Release Package Folder Structure

The following figure shows the release package structure when OPTIGA™ Trust M2 ID2 is installed/extracted on PC.

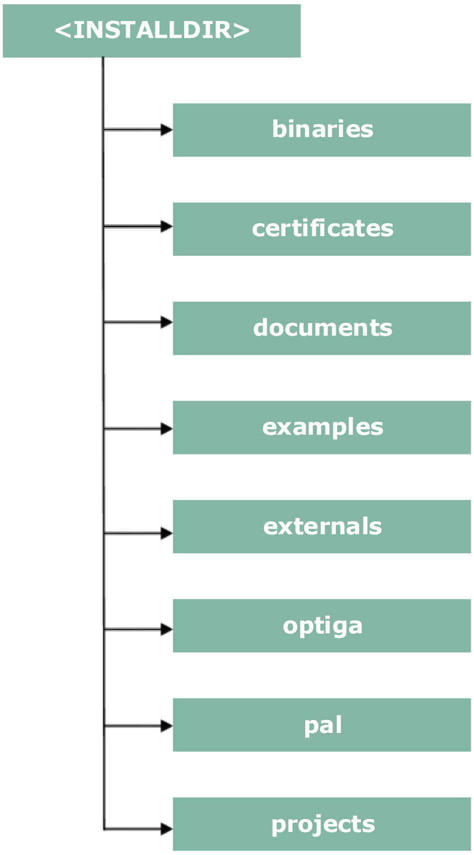


Figure 11 Release Package Folder Structure

<*INSTALLDIR*> is the root directory to which the release package contents are extracted. The following section explains the contents of each subdirectory under installed directory:

1. binaries

   This directory contains binaries for OPTIGA™ Trust M2 ID2 sample application.

2. certificates

   This directory contains OPTIGA™ Trust M2 ID2 certificates.

3. documents

   This directory contains all relevant OPTIGA™ Trust M2 ID2 documentation.

4. examples

   This directory contains example usecases for Toolbox features, AliOS irot HAL integration example files and a tool for generation of manifest for secure update of data and key object feature.

5. externals

   This directory contains mbedtls software crypto libraries and AliOS irot HAL integration file.

6. optiga

   This directory contains OPTIGA™ Trust M2 ID2 libraries.

7. pal

   This directory contains PAL for ESP32-DevKitC V4 and for mbedtls.

8. projects

   This directory contains the batch script to clone AliOS-Things github and copy the OPTIGA™ host library to AliOS workspace. It also conatins OPTIGA™ shell application patch for AliOS-Things

Further the following figure elaborates the OPTIGA™ Trust M2 ID2 Host Software folder structure.

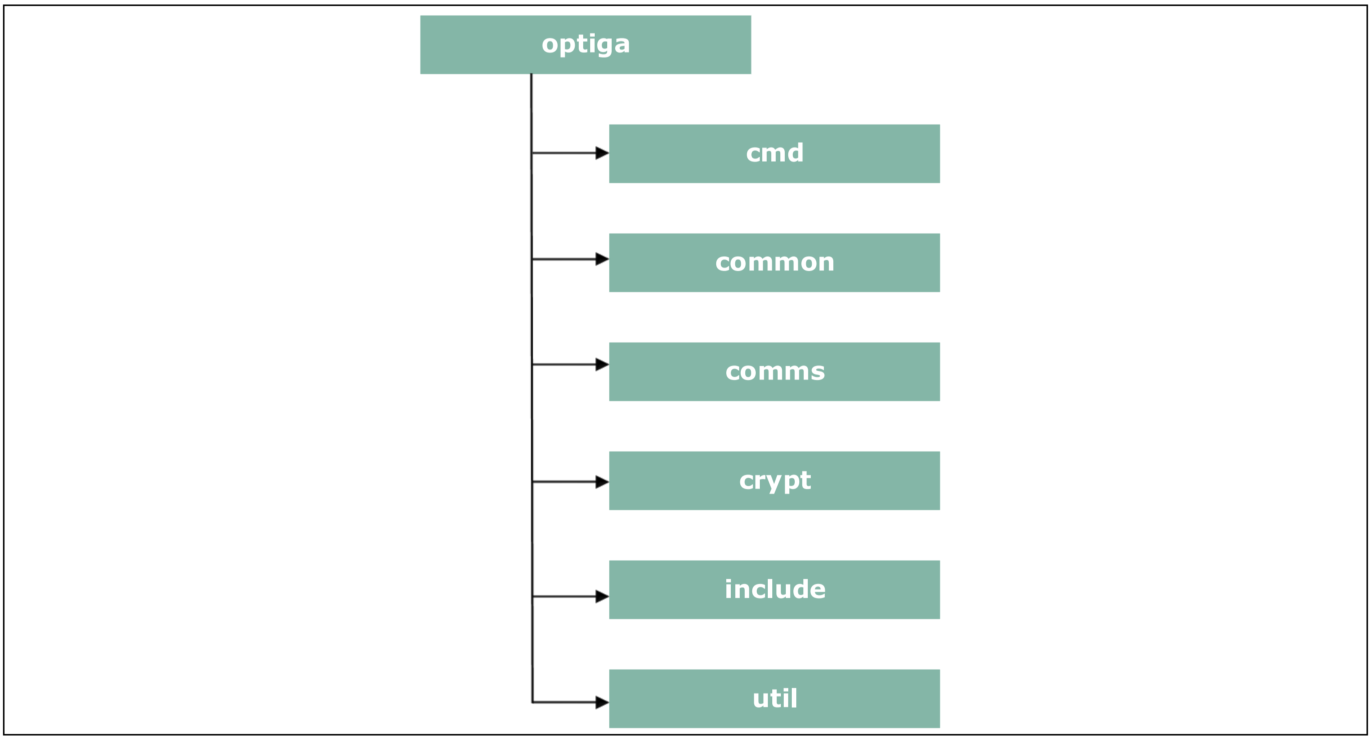


Figure 12 **Host Source Folder Structure**

1. cmd – This folder contains sources for all OPTIGA™ Trust M2 ID2 commands
2. common – This folder contains the common functions used across all the modules
3. comms – This folder contains the driver to communicate with OPTIGA™ Trust M2 ID2
4. crypt – This folder contains sources for cryptographic functionalities
5. include – This folder contains header files for all OPTIGA™ Trust M2 ID2 Host Software
6. util – This folder contains utility functions e.g. read/write and open/close application

6.3 Porting Notes

The implementation of Platform Abstraction Layer (PAL) needs to be updated in order to migrate to a new target platform.

The PAL reference code for the ESP32-DevKitC V4 is provided as part of package which can be used. The implementation can be found in *"<INSTALLDIR>/pal/ esp32_devkitc_alios"* and the header files are available in *"<INSTALLDIR>/optiga/include"* with the required APIs used by upper layers. The header files are platform agnostic and would not require any changes.

6.4 Communication with OPTIGA™ Trust M2 ID2

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 M2 ID2.

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_0,
    /// Slave address
    0x30,
    /// Upper layer context
    NULL,
    /// Callback event handler
    NULL
};
```

2. Update platform specific context for GPIOs (Vdd and Reset)

```
/**
 * \brief Vdd pin configuration for OPTIGA
*/
pal_gpio_t optiga_vdd_0 =
{
    // Platform specific GPIO context for the pin used to toggle Vdd
    (void*)&gpio_vdd
};

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

3. Update PAL I2C APIs [*pal_i2c.c*] to communicate with OPTIGA™ Trust M2 ID2

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*

A 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 ESP32-DevKitC V4 need to have the platform I2C driver initialization explicitly done as part of *pal_i2c_init* as it is not taken care by AliOS-Things initialization.

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 transmission is successful invokes the handler to inform the result.

4. Update PAL GPIO [*pal_gpio.c*] to power on and reset the OPTIGA™ Trust M2 ID2
   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) *pal_os_event_trigger_registered_callback*
   c) *queue_recv_task*

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. The p_pal_os_event is an event instance created using *pal_os_event_create*. First time, when this api in invoked a task and queue is created. During the subsequent api call the registered call back function gets added in the queue.

```
void pal_os_event_register_callback_oneshot(
                            pal_os_event_t * p_pal_os_event,
                            register_callback callback,
                            void * callback_args,
                            uint32_t time_us)
{
    p_pal_os_event->callback_registered = callback;
    p_pal_os_event->callback_ctx = callback_args;
    uint32_t time_ms;
    espos_task_t new_task_handle;


    if(create_state)
    {
        // !!!OPTIGA_LIB_PORTING_REQUIRED
        // The following steps related to TIMER must be taken
        // care while porting to different platform
        espos_mutex_create(&global_mutex, ESPOS_MUTEX_NORMAL);
        espos_queue_create(&global_queue,
                            sizeof(pal_os_event_clbs_t), 10);
        aos_task_new("queue_recv_task", queue_recv_task, 0,
                        8192);
        espos_timer_create(&scheduler_timer,
                            TIMER_NAME,
                            timer_handler,
                            NULL,
                            2,
                            ESPOS_TIMER_NO_AUTO_RUN);
        create_state = 0;

    }
    else
    {
        espos_timer_set_period(scheduler_timer, 2);
    }

    espos_timer_start(scheduler_timer);
}
```

The handler registered must be invoked once the timer has elapsed as shown in *pal_os_event_trigger_registered_callback.* The *pal_os_event_trigger_registered_callback* is to be registered with event timer interrupt to get trigerred when the timer expires. The pal_os_event_0 is the instance in the pal_os_event used store the registered callback and context.

```
void pal_os_event_trigger_registered_callback(void)
{
    register_callback callback;
    pal_os_event_clbs_t clbs;
    uint8_t exit = 1;

    // !!!OPTIGA_LIB_PORTING_REQUIRED
    // The following steps related to TIMER must be taken care
    // while porting to different platform

    // lint --e{534} suppress "Error handling is not required
    //so return value is not checked"
    while (exit)
    {
        if (0 == espos_mutex_lock(global_mutex,
                                    ESPOS_MAX_DELAY))
        {
            espos_timer_stop(scheduler_timer);
            clbs.clb_ctx = pal_os_event_0.callback_ctx;
            clbs.clb = pal_os_event_0.callback_registered;

            espos_queue_send(global_queue, &clbs,
                                ESPOS_MAX_DELAY);
            exit = 0;
        }
    }
}
```

In this function, the task which is created in *pal_os_event_register_callback_oneshot* function is running continuously to check for any entry is available in the queue for execution.

```
static void queue_recv_task(void *p)
{
    register_callback callback;
    pal_os_event_clbs_t clbs;
      ///If callback_ctx is NULL then callback function will
    //have unexpected behavior
    while(1)
    {
        if ( 0 == espos_queue_recv(global_queue, &clbs,
                            ESPOS_NO_DELAY) )
        {
            callback = clbs.clb;
            clbs.clb = NULL;
            callback((void * )clbs.clb_ctx);
        }
    }
}
```

6.5 Reference code on ESP32-DevKitC V4 for communicating with OPTIGA™ Trust M2 ID2

Connect the VCC pin of Security chip to 3.3V pin of ESP32-DevKitC V4 for reference code execution.

```
i2c_dev_t i2c_0 = {
    .port                  = 0,
    .config.address_width  = I2C_HAL_ADDRESS_WIDTH_7BIT,
    .config.freq           = I2C_BUS_BIT_RATES_400K,
    .config.mode           = I2C_MODE_MASTER,
    .config.dev_addr       = 0x30 << 1
};

/**
 * \brief PAL I2C configuration for OPTIGA.
 */
pal_i2c_t optiga_pal_i2c_context_0 =
{
    /// Pointer to I2C master platform specific context
    &i2c_0,
    /// Slave address
    0x30,
    /// Upper layer context
    NULL,
    /// Callback event handler
    NULL
};

static volatile uint32_t optiga_pal_event_status;
static void optiga_pal_i2c_event_handler(void* upper_layer_ctx,
                                            uint8_t event);
// OPTIGA pal i2c event handler
static void optiga_pal_i2c_event_handler(void* upper_layer_ctx,
                                          uint8_t event)
{
    optiga_pal_event_status = event;
}
pal_status_t test_optiga_communication(void)
{
    pal_status_t pal_return_status;
    uint8_t data_buffer[10] = {0x84};

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

```

        pal_return_status = pal_i2c_init(&optiga_pal_i2c_context_0);
        if (PAL_STATUS_FAILURE == pal_return_status)
        {
            // Pal I2C Init failed
            // state or low level driver failures
        }

        // 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
            }
            pal_os_timer_delay_in_milliseconds(2);
        // 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);
            if (PAL_STATUS_FAILURE == pal_return_status)
            {
                // Pal I2C read failed due to I2C busy is in busy
                // state or low level driver failures
            }
            pal_os_timer_delay_in_milliseconds(2);
        // Wait until reading from optiga is completed
        } while (PAL_I2C_EVENT_SUCCESS != optiga_pal_event_status);

        return pal_return_status;
    }

    int application_start(int argc, char *argv[])
    {
        test_optiga_communication();
    }
```

7 OPTIGA™ Trust M2 ID2 External Interface

7.1 Commands

This section provides short description of the commands exposed by the OPTIGA™ Trust M2 ID2 secuirty chip and mapping of these commands w.r.t Use Cases.

Table 14 Command table

Command Name	Description
OpenApplication	Command to launch an application
CloseApplication	Command to close/hibernate an application
GetDataObject	Command to get (read) a data object
SetDataObject	Command to set (write) a data object
SetObjectProtected	Command to set (write) data/key objects and its metadata protected (integrity protection, confidentiality)
GetRandom	Command to generate a random stream
CalcHash	Command to calculate a Hash
CalcSign	Command to calculate a signature
VerifySign	Command to verify a signature
DeriveKey	Command to derive keys
GenKeyPair	Command to generate public/private key pairs
EncryptAsym	Command to encrypt (Asymmetric) a message
DecryptAsym	Command to decrypt (Asymmetric) a message
EncryptSym	Command to encrypt (Symmetric) a message
DecryptSym	Command to decrypt (Symmetric) a message
GenSymKey	Command to generate a symmetric key

Table 15 Mapping of commands with Use cases

Use Case	OPTIGA™ Trust M2 ID2 commands used
Secure Communication with (D)TLS	GetRandom, CalcHash, CalcSign, VerifySign, DeriveKey, GenKeyPair, EncryptAsym and DecryptAsym,
Symmetric Key Attestation, Security Tokens	EncryptSym and DecryptSym
Datastore (user memory ~ 4.5kB)	GetDataObject and SetDataObject
Secure Firmware Update	VerifySign and DeriveKey
Secure update of Trust Anchors and Keys on Security Chip	SetObjectProtected command

7.2 Crypto Performance

The performance metrics for various schemes are provided by the Table 16 below. If not particularly mentioned, the performance is measured @ OPTIGA™ Trust M2 ID2 I/O interface with:

- I2C FM (400KHz)
- Without power limitation
- @ 25°C
- VCC = 3.3V

- RSA Signature scheme: RSA SSA PKCS#1 v1.5 without hashing
- Encryption/Decryption scheme: RSAES PKCS#1 v1.5
- Hash scheme: SHA256
- Key Derivation scheme: TLS v1.2 PRF SHA256, HKDF SHA256
- RSA Key size: 2048 bits
- AES Key size: 128 bits

Table 16 Crypto performance

Scheme	Algorithm	Performance in ms[1]	Performance with Shielded Connection in ms[1]	Notes
Calculate signature	RSA®	~ 310	~ 320	• 2048 bit exponentical • No data hashing
Verify signature	RSA®	~ 40	~ 50	• 2048 bit exponentical provided by external world • No data hashing
Key pair generation	RSA®	~ 2900[2]	~ 2910	Generate 2048 bit RSA key pair
Encryption	RSA®	~ 40	~ 50	Encrypt 127 bytes
Decryption	RSA®	~ 315	~ 325	Decrypt 127 bytes
Encryption	AES-128	~ 28	~ 35	Encrypt 256 bytes, ECB mode
Decryption	AES-128	~ 35	~ 42	Decrypt 256 bytes, ECB mode
Key derivation	PRF as per TLS v1.2	~ 50	~ 55	• To derive a key of 40 bytes • Shared secret (32 bytes) from session context and • The input key derivation data size is 48 bytes
Key derivation	HKDF with SHA256	~ 130	~ 135	Using a pre-shared secret from a data object
HMAC	HMAC with SHA256	~ 90	~ 95	Using a pre-shared secret from a data object and 128 bytes of input data
Hash calculation	SHA256	~ 15 Kbyte/s	~ 14 Kbyte/s	In blocks of 1280 bytes

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

[2] RSA key pair generation performance is not predictable and typically have a variation in performance. This could be significantly higher or lower as the one specified in the table which is an average value over collected samples.

8 Security Monitor

The Security Monitor is a central component which enforces the security policy of the OPTIGA™ Trust M2 ID2. It consumes security events sent by security aware parts of the OPTIGA™ Trust M2 ID2 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 17 Security Events

Event	Description
Decryption Failure	This event occurs in case a decryption and/or integrity check of provided data lead to a failure during protected update
Key Derivation	This event occurs in case the DeriveKey command gets applied on a persistent data object (not volatile data object as session context). In that case the persistent data object gets used as pre-shared secret.
Private Key Use	This event occurs in case the internal services are going to use an OPTIGA™ Trust M2 ID2 hosted private key.
Secret Key Use	This event occurs in case the internal services are going to use a OPTIGA™ hosted secret (symmetric) key (once per respective command), except temporary keys from session context are used.
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 17) 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 3. 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 M2 ID2 supports IFX I2C v2.01 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 18 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 19. Default value 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 22	Read only
0x86	TRANS_TIMEOUT[1]	4	For details refer to Table 22	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.

Appendix A – Infineon I2C Protocol Registry Map

Register Address	Name	Size in Bytes	Description	Master Access
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 20. The default mode is SM & FM (011B).	Read / Write

Table 19 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	I^{2}C base address specified by Table 18

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 20 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
PRESENT_LAYER	RFU						
15-0							
Length of data block to be read							

Table 21 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
PRESENT_LAYER	23	0 1	Presentation Layer not supported Presentation Layer supported

10.1 Infineon 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 22 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

Appendix A – Infineon I2C Protocol Registry Map

Parameter	Default Value	Description
		transmission is lost and retransmits the non-acknowledged frame. The Timer gets started as soon as the complete frame is transmitted. The value could be 1 up to 1000. However, the higher the number, the longer it takes 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.
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	1	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.
PRESENT_LAYER	1	This flag at the I2C_STATE register indicates the optional availability of the presentation layer, which is providing confidentiality and integrity protection of payloads (APDUs) transferred across the I2C interface. The presentation layer is used as part of Shielded Connection.

11 Appendix B - OPTIGA™ Trust M2 ID2 Command/Response I2C Sample Logs

The default I2C slave address for the OPTIGA™ Trust M2 ID2 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 M2 ID2

Pre-requisites

1. Ensure that the security device is powered up
2. The OPTIGA™ Trust M2 ID2 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 M2 ID2 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 M2 ID2 (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 M2 ID2.
5. The logs specified below are without the presentation layer (used for the Shielded Connection) of [IFX I2C]

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 23 Check I2C_STATE Register of OPTIGA™ Trust M2 ID2

I2C_ADDR	Transaction Type	Data [values in hexadecimal]
30	Write [01 Bytes]	82
30	Read [04 Bytes]	08 80 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 M2 ID2 as shown below.

Table 24 OpenApplication on OPTIGA™ Trust M2 ID2

I2C_ADDR	Transaction Type	Data [values in hexadecimal]
Step 1: Send OpenApplication command to initiate the application context on the OPTIGA™ Trust M2 ID2		
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**

I2C_ADDR	Transaction Type	Data [values in hexadecimal]
		70 70 6C 04 1A
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]	C8 80 00 05
Step 3: Read the DATA register [*Acknowledgment from* OPTIGA™ Trust M2 ID2 *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 80 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 M2 ID2 unique ID and the build information details. The GetDataObject command is used to read the Coprocessor UID information.

Table 25 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 80 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 M2 ID2 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 C – Power Management

When operating, the power consumption of OPTIGA™ Trust M2 ID2 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 Hibernation

This maximizes power saving (zero power consumption[1]), while the I2C bus stays connected. In this case OPTIGA™ Trust M2 ID2 saves the application context before power-off (switching off V_{CC}) and restores it after power-up. After power-up the application continues seamlessly from the state before hibernate.

12.2 Low Power Sleep Mode

The OPTIGA™ Trust M2 ID2 automatically enters a low-power mode after a configurable delay. Once it has entered Sleep mode, the OPTIGA™ Trust M2 ID2 resumes normal operation as soon as its address is detected on the I2C bus.

In case no command is sent to the OPTIGA™ Trust M2 ID2 it behaves as shown in Figure 13.

1. As soon as the OPTIGA™ Trust M2 ID2 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 M2 ID2 enters sleep mode.

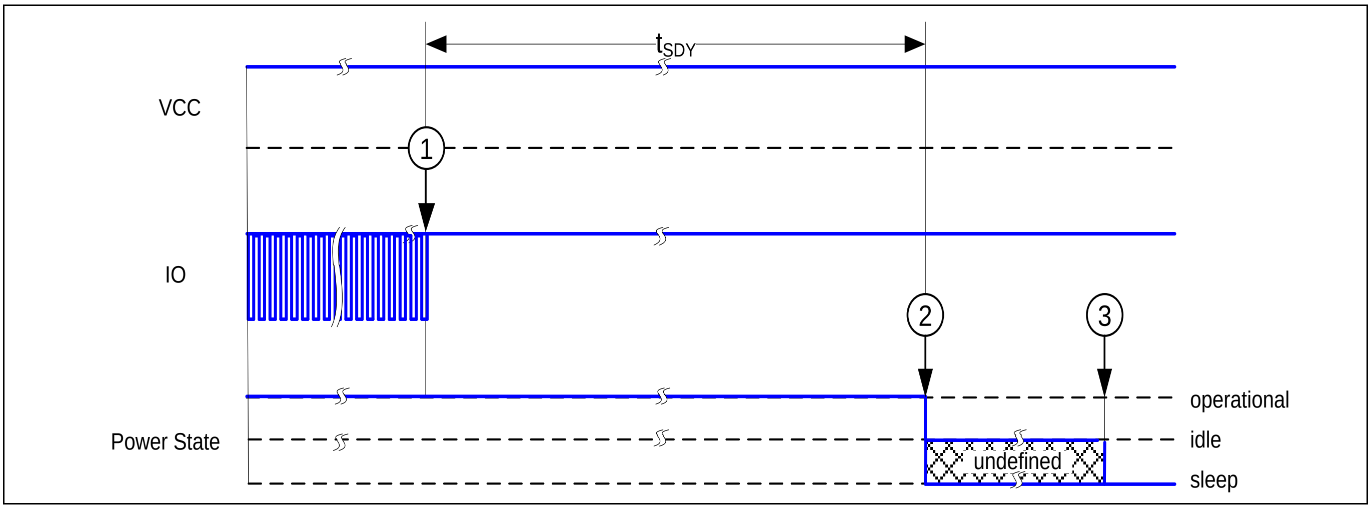


Figure 13 Go-to-Sleep Diagram

[1] Leakage current < 2.5µA only

Revision history

Document version	Date of release	Description of changes
2.10	2020-09-29	Release to Production version
2.00	2020-06-29	Fixed internal review comments
0.70	2020-05-23	Initial version updated for ES Release

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

Edition 2020-09-29

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email:
CSSCustomerService@infineon.com

Document reference

IMPORTANT NOTICE
The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics (“Beschaffenheitsgarantie”) .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer’s technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies’ products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.