OPTIGA™ Trust M

About this document

Scope and purpose

The purpose of this document is to guide a beginner to use the OPTIGA™ Trust M XMC4800 IoT Connectivity kit. The scope is limited to OPTIGA™ Trust M XMC4800 IoT Connectivity kit and its hardware and software components.

Intended audience

This document addresses: customers, solution providers and system integrators.

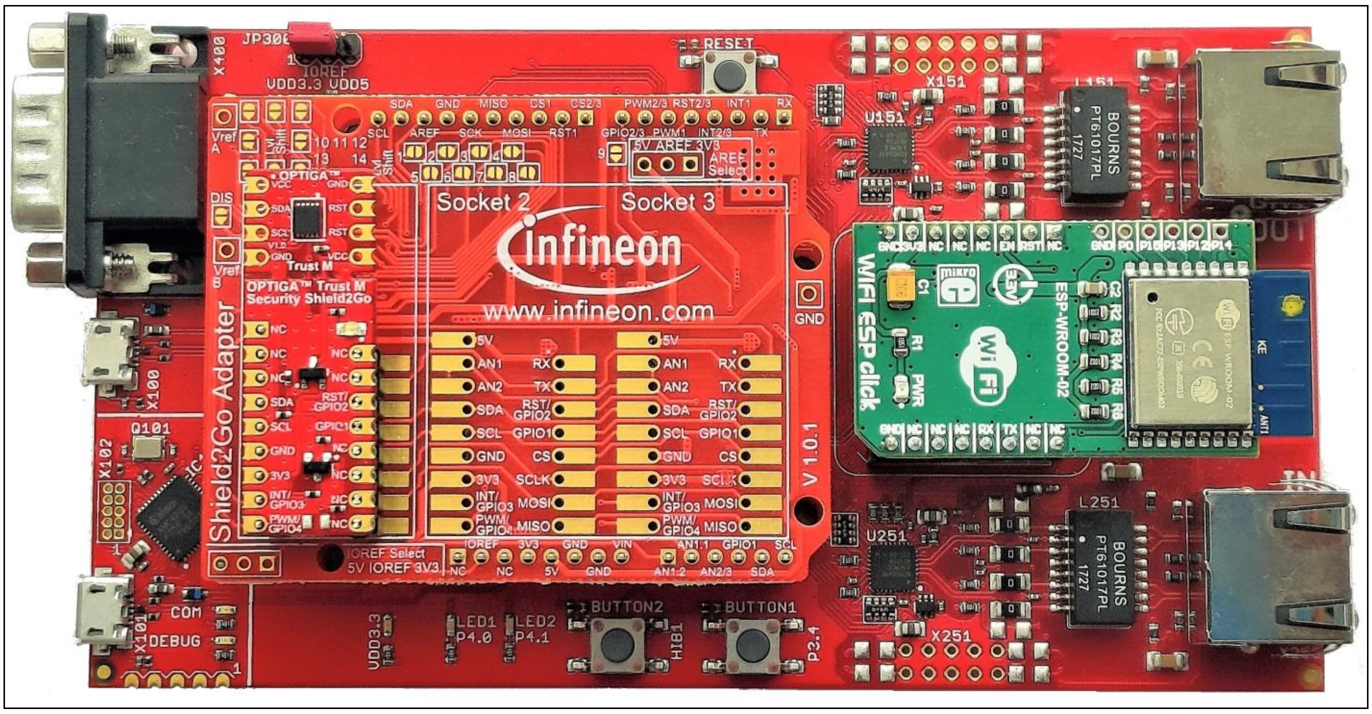


Table of Contents

1 Introduction

This document describes how to setup the environment to run OPTIGA™ Trust M application and use the provided binaries.

1.1 References

Table 1 References

Definition	Source
[1] xmc4800_IOTkit_usermanual	Infineon
[2] Infineon_I2C_Protocol	Infineon

1.2 Abbreviations

Table 2 Abbreviations

Abbreviation	Definition
API	Application Programming Interface
CA	Certificate Authority
CHM	Microsoft Compiled HTML Help
CMOS	Complementary Metal Oxide Semiconductor
DAVE	Digital Application Virtual Engineer
ECC	Elliptic Curve Cryptography
HTML	Hyper Text Markup Language
HW	Hardware
I2C	Inter Integrated Circuit
IDE	Integrated Development Environment
IoT	Internet of Things
NIST	National Institute of Standards and Technology
OS	Operating System
PAL	Platform Abstraction Layer
RSA	Rivest-Shamir-Adleman
PC	Personal Computer
RST	Reset
SCL	Serial Clock
SDA	Serial Data
SW	Software
TTL	Transistor Transistor Logic
USB	Universal Serial Bus
XMC	XMC4800 IoT Amazon FreeRTOS Connectivity Kit WIFI with EtherCAT Kit-V1.1

2 OPTIGA™ Trust M

OPTIGA™ Trust M is a security solution with a pre-programmed security controller with wide range of security features.

It supports secure data, key and metadata object update, hibernate and cryptographic toolbox functionalities, secure communication, platform integrity, data store protection and lifecycle management for Connected Device Security.

2.1 OPTIGA™ Trust M XMC4800 IoT Connectivity kit

OPTIGA™ Trust M XMC4800 IoT Connectivity kit is designed to provide all the components required to setup the environment to demonstrate the features of the OPTIGA™ Trust M.

2.1.1 Evaluation Kit Components

Table 3 Evaluation Kit contents

No.	Item	Description
1	XMC4800 board	Hardware Evaluation board for XMC4800 microcontroller from Infineon. More details can be found on Infineon website.
2	My IoT Adapter	Arduino compatible connector to add Shield2Go board on XMC4800 IoT Connectivity Kit.
3	OPTIGA™ Trust M Security Shield2Go	Shield2Go board contains OPTIGA™ Trust M chip. It is compatible with Infineon's My IoT adapter.
4	WIFI ESP Click	WIFI hardware module, which can be integrated with XMC4800 microcontroller.
5	Micro USB to USB cable	The cable provides DC supply to XMC4800 IoT Connectivity Kit and to flash software.

2.2 Installed Software Components

The installed directory structure of OPTIGA™ Trust M setup software is shown below:

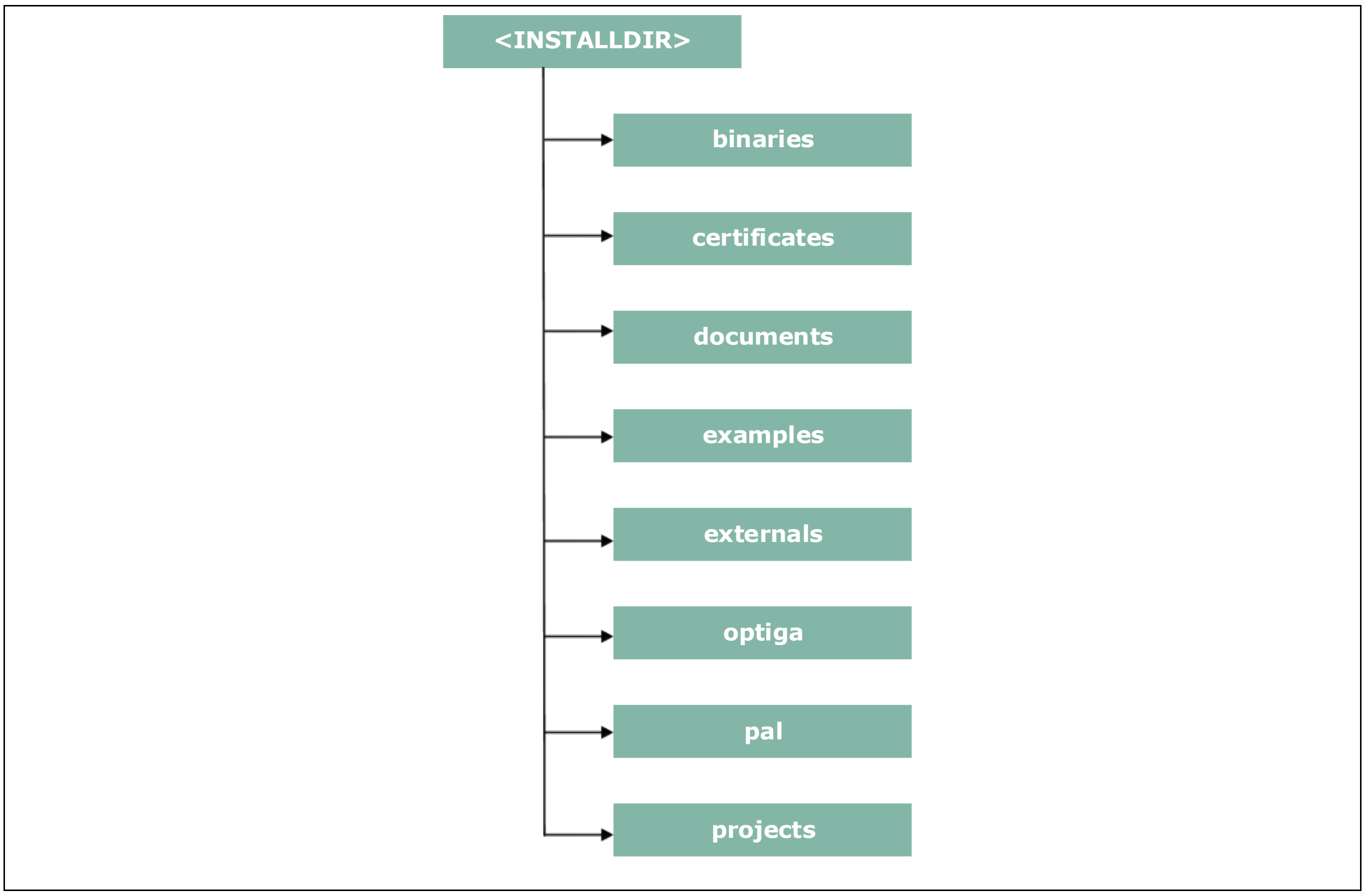


Figure 1 Installed directory structure

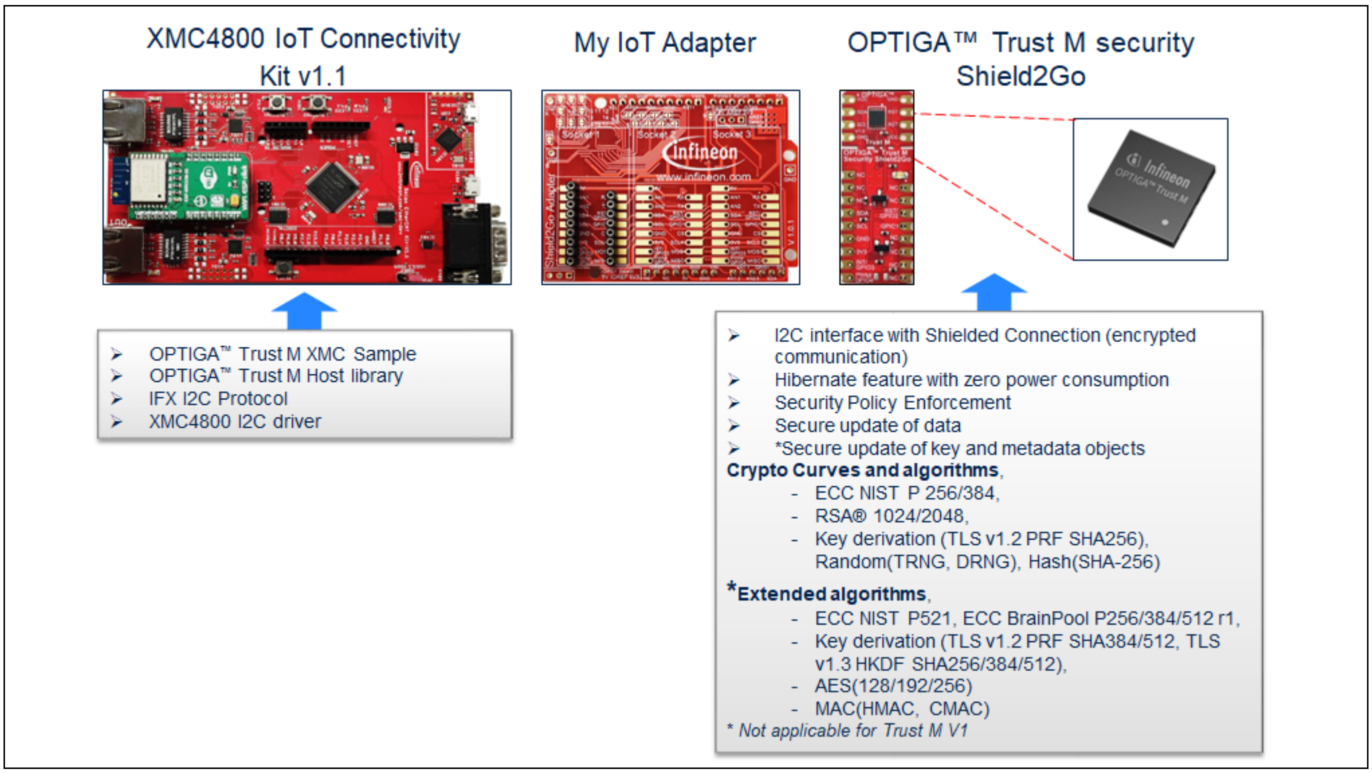
<*INSTALLDIR*> is the root directory to which the release package contents are extracted. The following section explains the contents of each subdirectory under installed directory:

1. **binaries** -- binaries for OPTIGA™ Trust M example application.
2. **certificates** -- OPTIGA™ Trust M certificates.
3. **documents** -- Relevant OPTIGA™ Trust M documentation.
4. **examples** – example use cases for Toolbox features and a tool for generation of manifest and fragment for protected update feature.
5. **externals** -- mbedtls software crypto library.
6. **optiga** -- OPTIGA™ Trust M libraries.
7. **pal** -- PAL for XMC4800 device and PAL for mbedtls software crypto library.
8. **projects** -- XMC4800 device example project in DAVE workspace.

3 System Setup

This section explains the basic components required for system setup.

3.1 System Overview


XMC4800 IoT Connectivity Kit v1.1
My IoT Adapter
OPTIGA™ Trust M security Shield2Go

- OPTIGA™ Trust M XMC Sample
- OPTIGA™ Trust M Host library
- IFX I2C Protocol
- XMC4800 I2C driver

- I2C interface with Shielded Connection (encrypted communication)
- Hibernate feature with zero power consumption
- Security Policy Enforcement
- Secure update of data
- *Secure update of key and metadata objects

Crypto Curves and algorithms,
- ECC NIST P 256/384,
- RSA® 1024/2048,
- Key derivation (TLS v1.2 PRF SHA256), Random(TRNG, DRNG), Hash(SHA-256)

***Extended algorithms**,
- ECC NIST P521, ECC BrainPool P256/384/512 r1,
- Key derivation (TLS v1.2 PRF SHA384/512, TLS v1.3 HKDF SHA256/384/512),
- AES(128/192/256)
- MAC(HMAC, CMAC)

** Not applicable for Trust M V1*

Figure 2 System Overview

This system consists of the following components:

1. XMC4800 IoT Connectivity Kit v1.1 from Infineon
    - The XMC4800 IoT Connectivity Kit is an evaluation board with XMC4800 Microcontroller from Infineon. For more information refer document [1].
    - It can connect to a WiFi access point using WiFi ESP click module.
    - It is used as a reference platform to simulate the Host.
    - It interacts with secure element via I2C.
2. My IoT Adapter
    - It acts as a gateway to add Shield2Go boards onto XMC4800 IoT Connectivity Kit V1.1.
3. OPTIGA™ Trust M Security Shield2Go
    - Shield2Go board contains OPTIGA™ Trust M chip. It is compatible with Arduino Uno along with Infineon's My IoT adapter.

The following interface/connection is done among the above components:

- Micro USB data cable (with Data line) from PC is connected to XMC to supply power.

3.2 Hardware Setup

The hardware required to run OPTIGA™ Trust M setup is described in this section.

3.2.1 XMC4800 IoT Connectivity Kit

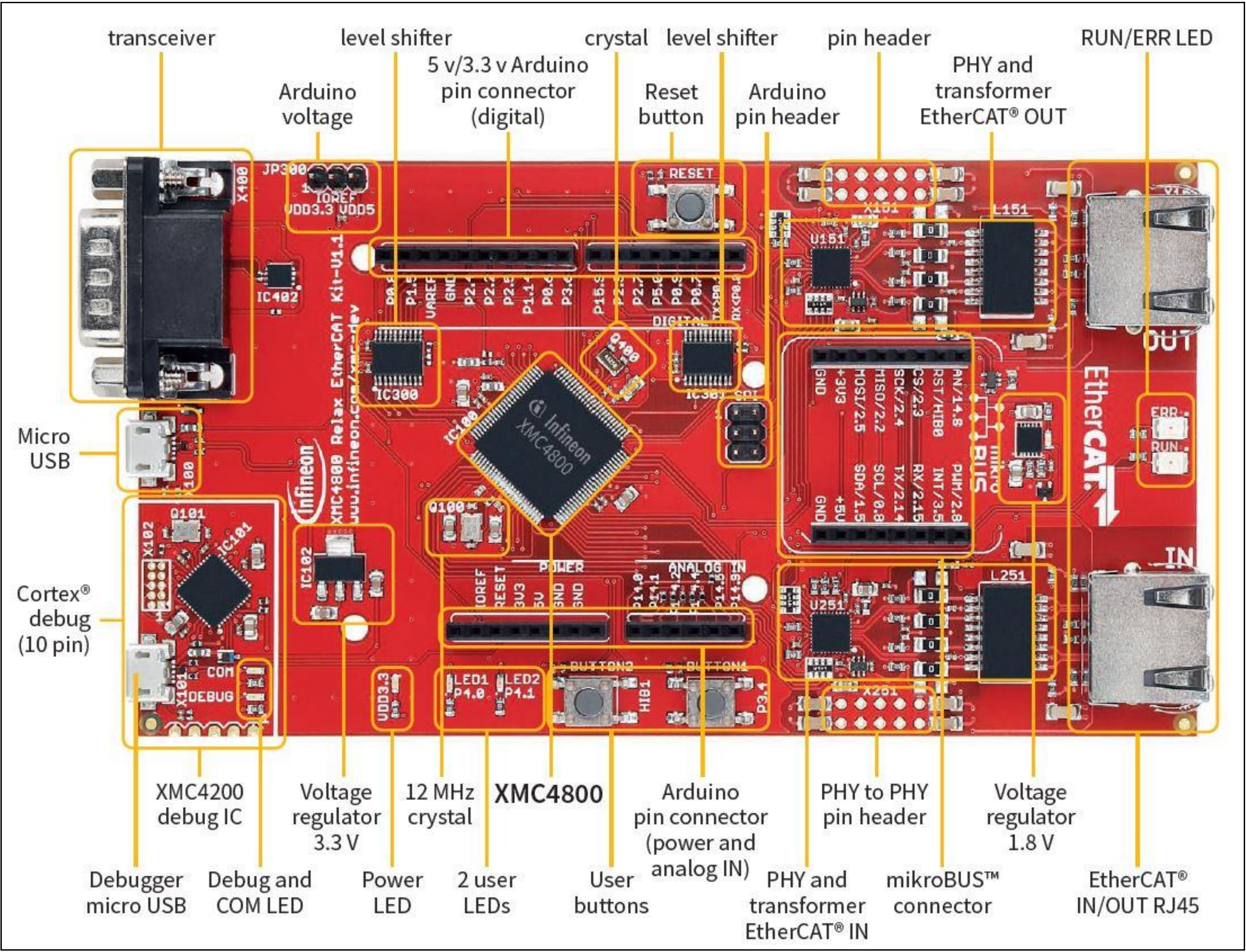


Figure 3 **XMC4800 IoT Connectivity kit**

Table 4 **XMC4800 IoT Connectivity kit connection**

No.	Item	Description
1	Debugger micro USB	Power supply of 5V is provided by connecting to Micro USB cable.
2	Arduino pin connector	External interface to connect to Arduino compatible Shields.
3	mikroBUS connector	Socket to connect WiFi ESP click module from MikroElektronica.
4	Micro USB	UART communication for debugging and logging purposes (VCOM).

The pin headers for Arduino shields can be used for GPIOs or signal interface as well. Arduino compatible connector supports I2C, UART and SPI interfaces among others.

Table 5 **XMC4800 IoT Connectivity kit Pin Information**

No.	Description	Pin
1	I2C SCL	P0.8
2	I2C SDA	P1.5
3	RST	P0.6
4	VCC	P2.7
5	GND	GND

For more information about pin details of Arduino shield, refer document [1].

For more information about the XMC Specification, Architecture and Design/Schematic, refer document [1]

3.2.2 My IoT Adapter

The My IoT adapter is an evaluation board that allows users to easily combine different Shield2Go boards to Arduino compliant ecosystem, for fast evaluation of IoT systems. With its solderless connectors, it allows users to easily stack Shield2Go boards instead of soldering it. The shield design is derived from XMC2Go evaluation board.

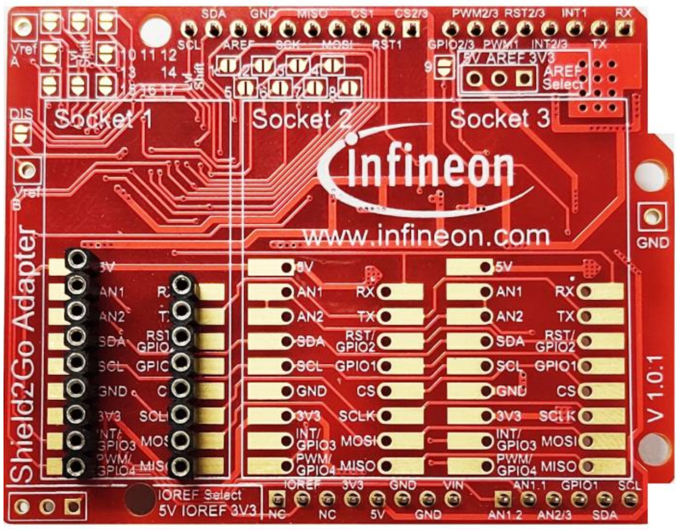


Figure 4 My IoT adapter

My IoT adapter features are as follows:

- Provide power supply and connectivity for Shield2Go boards.
- Level shifting handling capabilities between CMOS 3.3V and TTL 5V.
    - Solder bridges to selectively deactivate level shifting.
    - Additional pins enable setting the reference voltages for level shifting.
- Separate power control switches for Socket 1 and Socket 2. Socket 1 is independently controllable while Socket 2 and 3 share pins to underlying control board.

More information is available at Infineon website.

3.2.3 Shield2Go Security OPTIGA™ Trust M

Shield2Go boards are equipped with featured Infineon ICs and provide a standardized form factor and pin layout, allowing a 'plug and play' approach for easy prototyping.

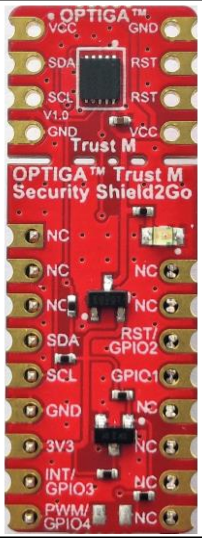


Figure 5 **OPTIGA™ Trust M Shield2Go**

The OPTIGA™ Trust M Shield2Go is equipped with OPTIGA™ Trust M security chip. It allows users to develop system solutions by combining Shield2Go with My IoT adapter and XMC.

Note: Ensure no voltage supplied to any of the pins exceeds the absolute maximum rating of V_{cc} + 0.3 V.

3.3 Software Setup

This section describes the software used in XMC to run the OPTIGA™ Trust M setup.

3.3.1 Software Components

All the software components required on XMC are explained in the following sections.

3.3.1.1 XMC4800 IoT Connectivity Kit

1. OPTIGA™ Trust M Host Library consists of the following:
   - Service Layer
     The layers (Util and Crypt) provide APIs to interact with OPTIGA™ for various use-case functionalities.
   - Access Layer
     This layer manages the access to the command interface of OPTIGA™ security chip. It also provides the communication interface to the OPTIGA™.
   - Platform Abstraction Layer
     This layer provides platform agnostic interfaces for the underlying HW and SW platform functionalities used by OPTIGA™ libraries.
   - Platform Layer
     This layer provides the platform specific components and libraries for the supported platforms.
2. IFX I2C Protocol
   This is an implementation as per document [2].
3. XMC4800 I2C Driver
   These are low level I2C device driver for I2C communication from XMC to OPTIGA™ Trust M Security chip.
4. OPTIGA™ Trust M XMC Example

This Example Application demonstrates Secure Data Object, Key and Metadata update, pairing of host and OPTIGA using Pre-shared secret, Hibernate feature, Cryptographic Toolbox Functionalities and Read/Write General Purpose Data use cases.

Note: The binaries and the example application provided with the application note are meant for the XMC4800 IoT Connectivity Kit v1. These binaries may not work as expected if executed on a different platform.

3.3.2 PC Requirements and Configurations

3.3.2.1 PC Requirement

A 32-bit or 64-bit PC with Windows 7/10 Operating System with the below requirements need to be used for setting up the OPTIGA™ Trust M setup:

1. One USB port.
2. DAVE 4.4.2 and device feature 2.2.4, which can be downloaded from Infineon website.
   Link to download DAVE 4.4.2: Dave Download
3. Segger J-Link tool v6.00 or greater for flashing software on XMC.
   Link to download Segger: J-Link tool Download
   Link to download manual: J-Link manual Download

Note: The path where DAVE tool is extracted is henceforth referred to as <DAVE_PATH> in the document.

1. Completion of user registration is must for DAVE download.
2. Re-enter the registered account data on Dave Download page and it takes to DAVE download link.

Note: All the tools mentioned in the above list are intended to be used with the binaries or source code given in the release package.

4 Using OPTIGA™ Trust M

4.1 Quick Setup

This section explains the steps to run OPTIGA™ Trust M example application.

4.1.1 Running OPTIGA™ Trust M Example Application

1. Make the connections among XMC4800 IoT Connectivity Kit, My IoT Adapter and OPTIGA™ Trust M Shield2Go as shown below

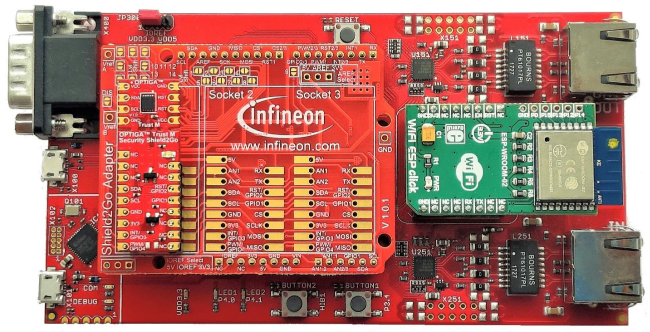


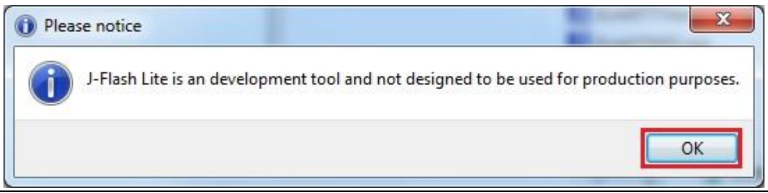
Figure 6 XMC4800 IoT Connectivity Kit, My IoT Adapter and OPTIGA™ Trust M Shield2Go connection

2. Power up the kit by connecting Micro USB cable between PC and Debugger micro USB. For placement of Debugger micro USB refer Figure 3.
3. Download the OPTIGA™ Trust M example application using JFlashLite tool as described in section 4.1.2.1. Hex file location is <INSTALLDIR>\binaries\xmc4800_iot_kit\dave4\xmc4800_optiga_example.hex.
4. OPTIGA™ Trust M example application uses USBD_VCOM for logging, refer section 4.1.3 for logging details.

4.1.2 Steps to download example hex file to XMC4800 Connectivity IoT Kit

4.1.2.1 Using JFlashLite tool

1. Run JFlashLite.exe from JLink installation folder. It shows a notice window. Click OK.

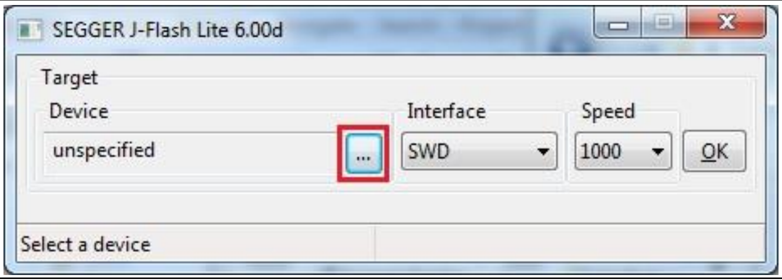


Please notice

J-Flash Lite is an development tool and not designed to be used for production purposes.

Figure 7 JFlashLite launch window

2. Click on Device to select a target device.



SEGGER J-Flash Lite 6.00d

Target

Device: unspecified

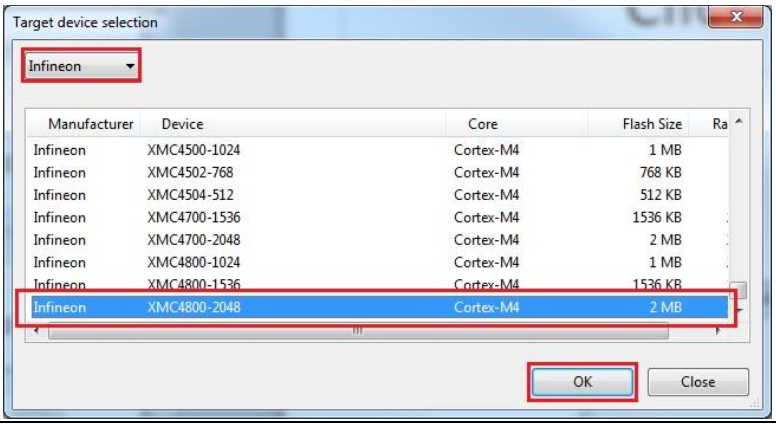
Interface: SWD

Speed: 1000

Select a device

Figure 8 JFlashLite select a device

3. Select Infineon as Manufacturer and Device as XMC4800-2048, and then click OK.

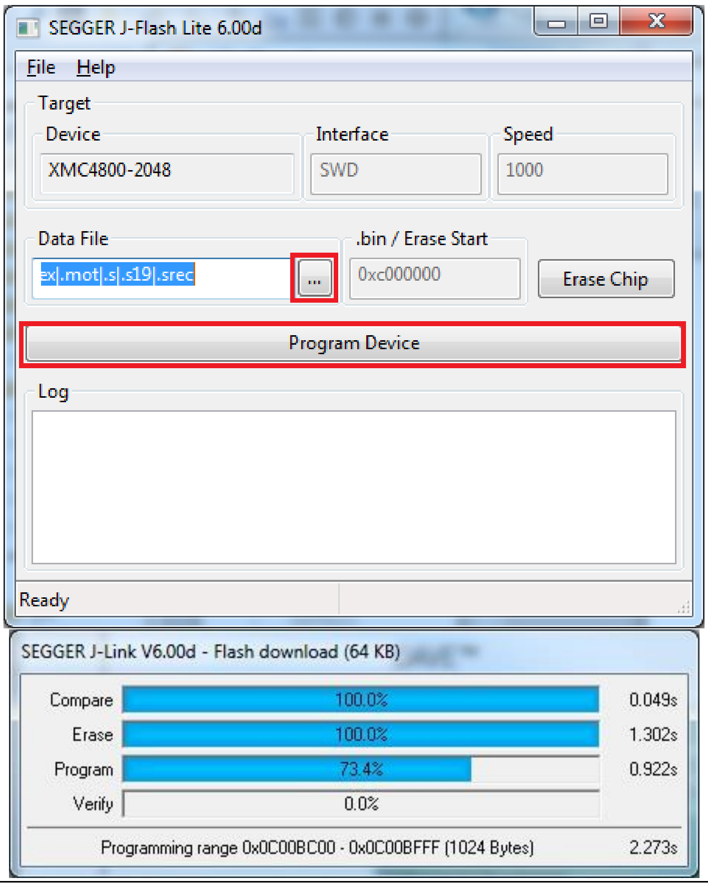

Target device selection

Infineon

Manufacturer	Device	Core	Flash Size	Ra
Infineon	XMC4500-1024	Cortex-M4	1 MB	
Infineon	XMC4502-768	Cortex-M4	768 KB	
Infineon	XMC4504-512	Cortex-M4	512 KB	
Infineon	XMC4700-1536	Cortex-M4	1536 KB	
Infineon	XMC4700-2048	Cortex-M4	2 MB	
Infineon	XMC4800-1024	Cortex-M4	1 MB	
Infineon	XMC4800-1536	Cortex-M4	1536 KB	
Infineon	XMC4800-2048	Cortex-M4	2 MB	

Figure 9 JFlashLite Target device selection

4. After target device selection, click OK on window shown in Figure 9.
5. Select hex file to be flashed under Data File and click on Program Device. It then shows the programming progress window.


SEGGER J-Flash Lite 6.00d

Target
Device: XMC4800-2048
Interface: SWD
Speed: 1000

Data File: ex|.mot|.s|.s19|.srec
.bin / Erase Start: 0xc000000

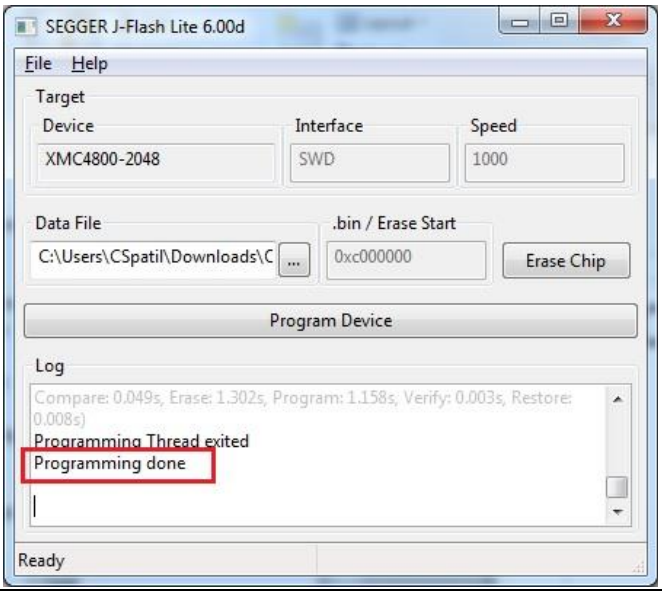
Ready

SEGGER J-Link V6.00d - Flash download (64 KB)

```
Compare   100.0%   0.049s
Erase     100.0%   1.302s
Program    73.4%   0.922s
Verify      0.0%
Programming range 0x0C00BC00 - 0x0C00BFFF (1024 Bytes)   2.273s
```

Figure 10 JFlashLite Hex file selection and programming progress window

6. Flash download completed.


SEGGER J-Flash Lite 6.00d

Target
Device: XMC4800-2048
Interface: SWD
Speed: 1000

Data File: C:\Users\CSpatil\Downloads\C
.bin / Erase Start: 0xc000000

```
Compare: 0.049s, Erase: 1.302s, Program: 1.158s, Verify: 0.003s, Restore:
0.008s)
Programming Thread exited
Programming done
```

Ready

Figure 11 JFlashLite programming completion window

4.1.3 Logger

4.1.3.1 Logger setup

1. Connect the micro USB cable between PC and micro USB. For placement of micro USB refer Figure 3.
2. Reset the XMC4800 by pressing the reset button.
3. Select the COM port with name “Communications Port”.

Note: *For binding the Windows serial driver(usbser.sys) with USBD_VCOM device user has to point to the driver.inf file in the folder path: <INSTALLDIR>\projects\xmc4800_iot_kit\common\Dave\Generated\USBD_VCOM\inf\*

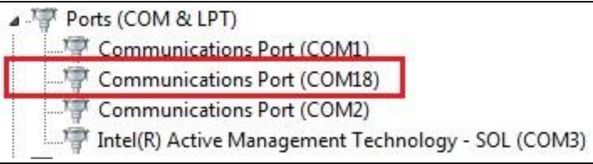
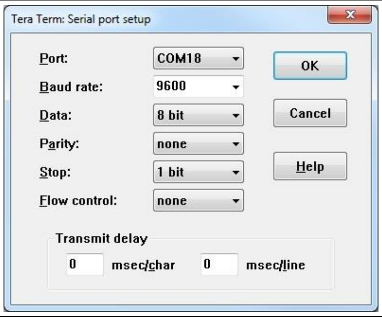

Ports (COM & LPT)
Communications Port (COM1)
Communications Port (COM18)
Communications Port (COM2)
Intel(R) Active Management Technology - SOL (COM3)

Figure 12 Discovery of USB Serial Device COM port

4. Configure COM port with 9600 8N1.


Tera Term: Serial port setup

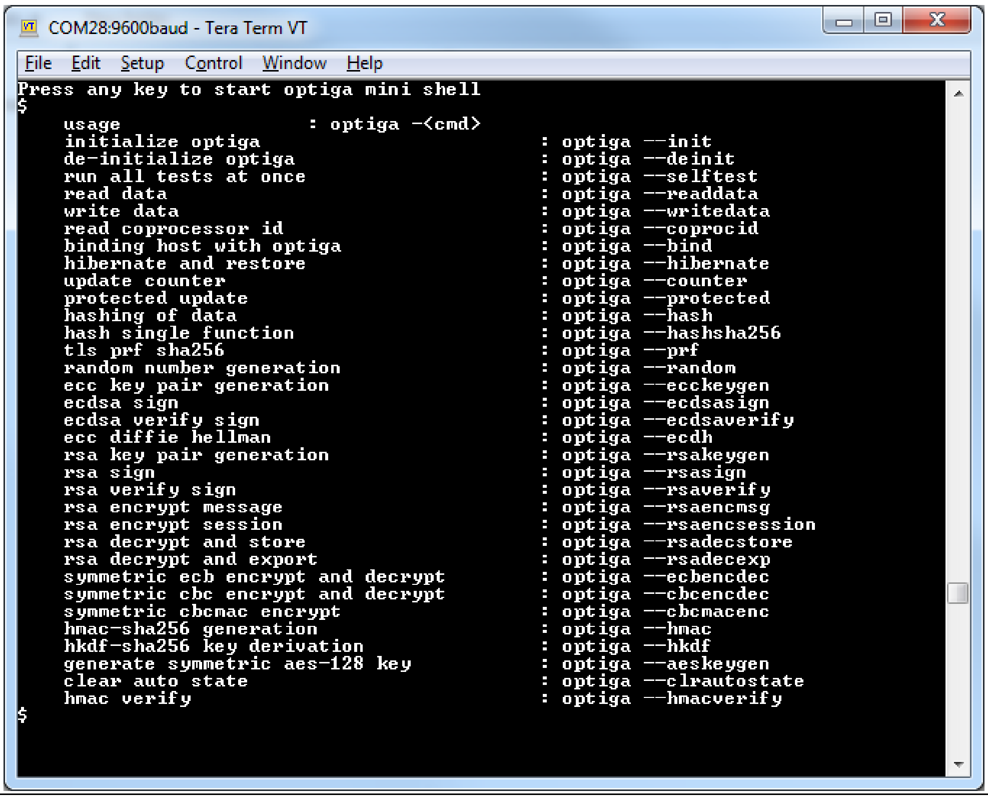
Port: COM18
Baud rate: 9600
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
0 msec/char 0 msec/line

Figure 13 TeraTerm terminal serial configuration

Using OPTIGA™ Trust M

5. Once connected, the terminal displays the text “Press any key to start optiga mini shell”.
6. It will list down the available OPTIGA™ commands.

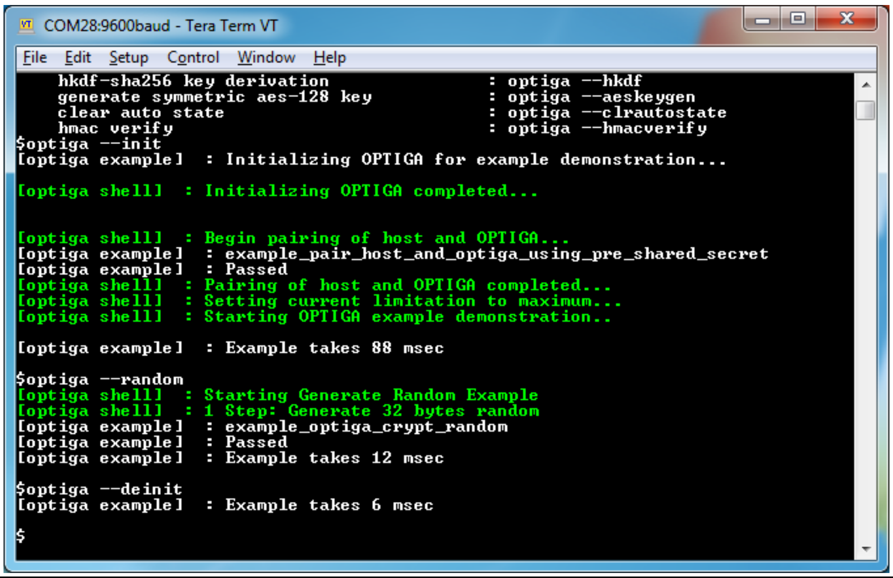

COM28:9600baud - Tera Term VT
File Edit Setup Control Window Help

```
Press any key to start optiga mini shell
$
    usage                    : optiga -<cmd>
    initialize optiga                           : optiga --init
    de-initialize optiga                        : optiga --deinit
    run all tests at once                       : optiga --selftest
    read data                                   : optiga --readdata
    write data                                  : optiga --writedata
    read coprocessor id                         : optiga --coprocid
    binding host with optiga                    : optiga --bind
    hibernate and restore                       : optiga --hibernate
    update counter                              : optiga --counter
    protected update                            : optiga --protected
    hashing of data                             : optiga --hash
    hash single function                        : optiga --hashsha256
    tls prf sha256                              : optiga --prf
    random number generation                    : optiga --random
    ecc key pair generation                     : optiga --ecckeygen
    ecdsa sign                                  : optiga --ecdsasign
    ecdsa verify sign                           : optiga --ecdsaverify
    ecc diffie hellman                          : optiga --ecdh
    rsa key pair generation                     : optiga --rsakeygen
    rsa sign                                    : optiga --rsasign
    rsa verify sign                             : optiga --rsaverify
    rsa encrypt message                         : optiga --rsaencmsg
    rsa encrypt session                         : optiga --rsaencsession
    rsa decrypt and store                       : optiga --rsadecstore
    rsa decrypt and export                      : optiga --rsadecexp
    symmetric ecb encrypt and decrypt           : optiga --ecbencdec
    symmetric cbc encrypt and decrypt           : optiga --cbcencdec
    symmetric cbcmac encrypt                    : optiga --cbcmacenc
    hmac-sha256 generation                      : optiga --hmac
    hkdf-sha256 key derivation                  : optiga --hkdf
    generate symmetric aes-128 key              : optiga --aeskeygen
    clear auto state                            : optiga --clrautostate
    hmac verify                                 : optiga --hmacverify
$
```

Figure 14 **Optiga available commands for shell application**

7. Enter optiga command in format of “optiga --<cmd>”.

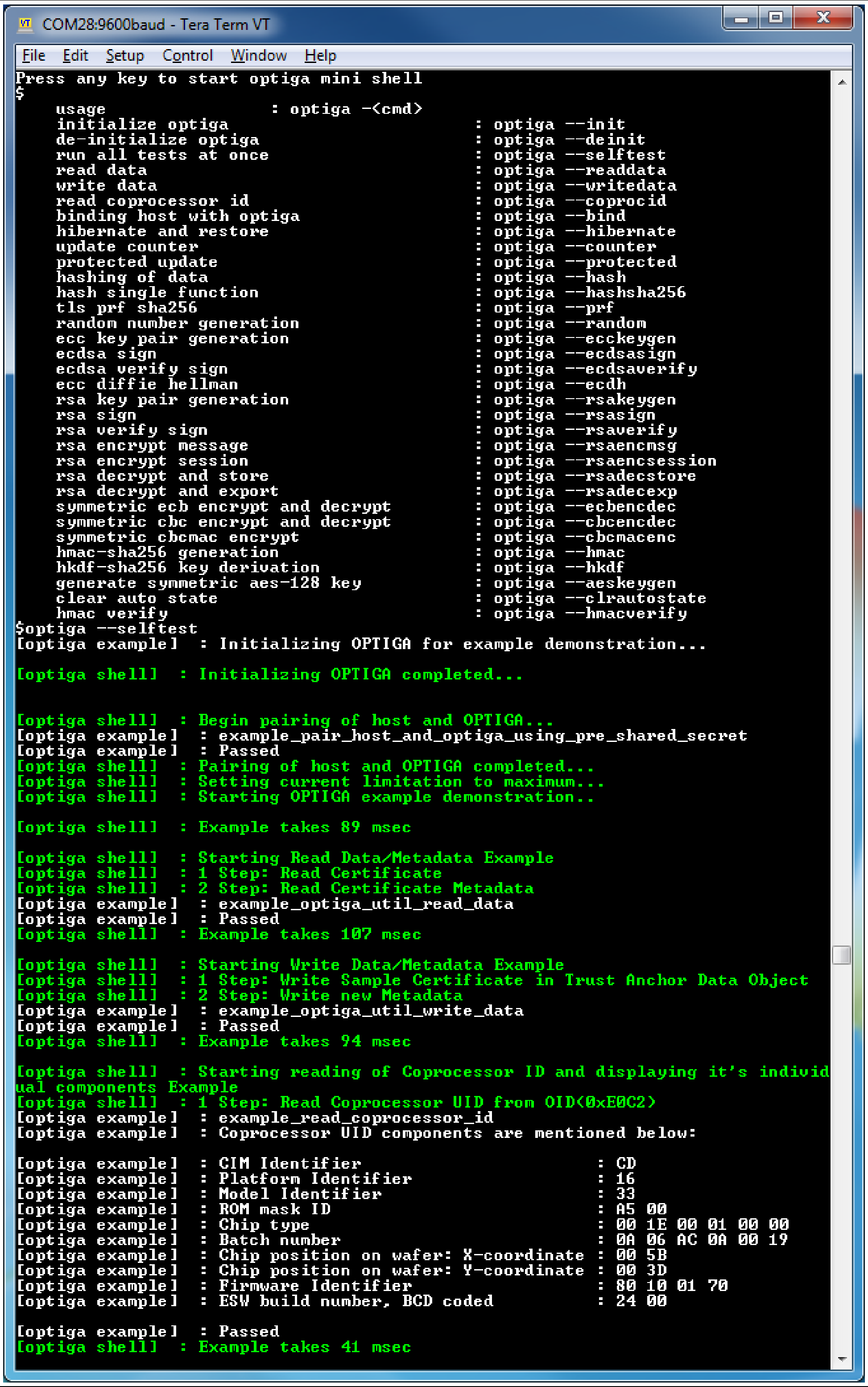
Note: *Execution of optiga --init is must before executing any other commands. Except for optiga --selftest command.*


COM28:9600baud - Tera Term VT
File Edit Setup Control Window Help

```
    hkdf-sha256 key derivation                  : optiga --hkdf
    generate symmetric aes-128 key              : optiga --aeskeygen
    clear auto state                            : optiga --clrautostate
    hmac verify                                 : optiga --hmacverify
$optiga --init
[optiga example]  : Initializing OPTIGA for example demonstration...

[optiga shell]  : Initializing OPTIGA completed...


[optiga shell]  : Begin pairing of host and OPTIGA...
[optiga example]  : example_pair_host_and_optiga_using_pre_shared_secret
[optiga example]  : Passed
[optiga shell]  : Pairing of host and OPTIGA completed...
[optiga shell]  : Setting current limitation to maximum...
[optiga shell]  : Starting OPTIGA example demonstration..

[optiga example]  : Example takes 88 msec

$optiga --random
[optiga shell]  : Starting Generate Random Example
[optiga shell]  : 1 Step: Generate 32 bytes random
[optiga example]  : example_optiga_crypt_random
[optiga example]  : Passed
[optiga example]  : Example takes 12 msec

$optiga --deinit
[optiga example]  : Example takes 6 msec

$
```

Figure 15 **Optiga random command execution using shell application**

8. The logs of the example execution are displayed along with status of each example as Passed or Failed.



COM28:9600baud - Tera Term VT

File Edit Setup Control Window Help

```
Press any key to start optiga mini shell
$
    usage                    : optiga -<cmd>
    initialize optiga                              : optiga --init
    de-initialize optiga                           : optiga --deinit
    run all tests at once                          : optiga --selftest
    read data                                      : optiga --readdata
    write data                                     : optiga --writedata
    read coprocessor id                            : optiga --coprocid
    binding host with optiga                       : optiga --bind
    hibernate and restore                          : optiga --hibernate
    update counter                                 : optiga --counter
    protected update                               : optiga --protected
    hashing of data                                : optiga --hash
    hash single function                           : optiga --hashsha256
    tls prf sha256                                 : optiga --prf
    random number generation                       : optiga --random
    ecc key pair generation                        : optiga --ecckeygen
    ecdsa sign                                     : optiga --ecdsasign
    ecdsa verify sign                              : optiga --ecdsaverify
    ecc diffie hellman                             : optiga --ecdh
    rsa key pair generation                        : optiga --rsakeygen
    rsa sign                                       : optiga --rsasign
    rsa verify sign                                : optiga --rsaverify
    rsa encrypt message                            : optiga --rsaencmsg
    rsa encrypt session                            : optiga --rsaencsession
    rsa decrypt and store                          : optiga --rsadecstore
    rsa decrypt and export                         : optiga --rsadecexp
    symmetric ecb encrypt and decrypt              : optiga --ecbencdec
    symmetric cbc encrypt and decrypt              : optiga --cbcencdec
    symmetric cbcmac encrypt                       : optiga --cbcmacenc
    hmac-sha256 generation                         : optiga --hmac
    hkdf-sha256 key derivation                     : optiga --hkdf
    generate symmetric aes-128 key                 : optiga --aeskeygen
    clear auto state                               : optiga --clrautostate
    hmac verify                                    : optiga --hmacverify
$optiga --selftest
[optiga example]  : Initializing OPTIGA for example demonstration...

[optiga shell]  : Initializing OPTIGA completed...


[optiga shell]  : Begin pairing of host and OPTIGA...
[optiga example]  : example_pair_host_and_optiga_using_pre_shared_secret
[optiga example]  : Passed
[optiga shell]  : Pairing of host and OPTIGA completed...
[optiga shell]  : Setting current limitation to maximum...
[optiga shell]  : Starting OPTIGA example demonstration..

[optiga shell]  : Example takes 89 msec

[optiga shell]  : Starting Read Data/Metadata Example
[optiga shell]  : 1 Step: Read Certificate
[optiga shell]  : 2 Step: Read Certificate Metadata
[optiga example]  : example_optiga_util_read_data
[optiga example]  : Passed
[optiga shell]  : Example takes 107 msec

[optiga shell]  : Starting Write Data/Metadata Example
[optiga shell]  : 1 Step: Write Sample Certificate in Trust Anchor Data Object
[optiga shell]  : 2 Step: Write new Metadata
[optiga example]  : example_optiga_util_write_data
[optiga example]  : Passed
[optiga shell]  : Example takes 94 msec

[optiga shell]  : Starting reading of Coprocessor ID and displaying it's individ
ual components Example
[optiga shell]  : 1 Step: Read Coprocessor UID from OID(0xE0C2)
[optiga example]  : example_read_coprocessor_id
[optiga example]  : Coprocessor UID components are mentioned below:

[optiga example]  : CIM Identifier                           : CD
[optiga example]  : Platform Identifier                      : 16
[optiga example]  : Model Identifier                         : 33
[optiga example]  : ROM mask ID                              : A5 00
[optiga example]  : Chip type                                : 00 1E 00 01 00 00
[optiga example]  : Batch number                             : 0A 06 AC 0A 00 19
[optiga example]  : Chip position on wafer: X-coordinate     : 00 5B
[optiga example]  : Chip position on wafer: Y-coordinate     : 00 3D
[optiga example]  : Firmware Identifier                      : 80 10 01 70
[optiga example]  : ESW build number, BCD coded              : 24 00

[optiga example]  : Passed
[optiga shell]  : Example takes 41 msec
```

Figure 16 TeraTerm log of example application

Note: If the symmetric key is written in 0xE200 OID, then generate symmetric aes-128 key example will return success without generation of the key. Modify file example_optiga_crypt_symmetric_generate_key.c to allow further generation of key.

4.1.3.2 Logger control

By default only logging from example is enabled in the release package.

Further control for OPTIGA™ Trust M host code logging is available in product specific config file (eg. optiga_lib_config_m_v3.h) under optiga_lib_config.h.

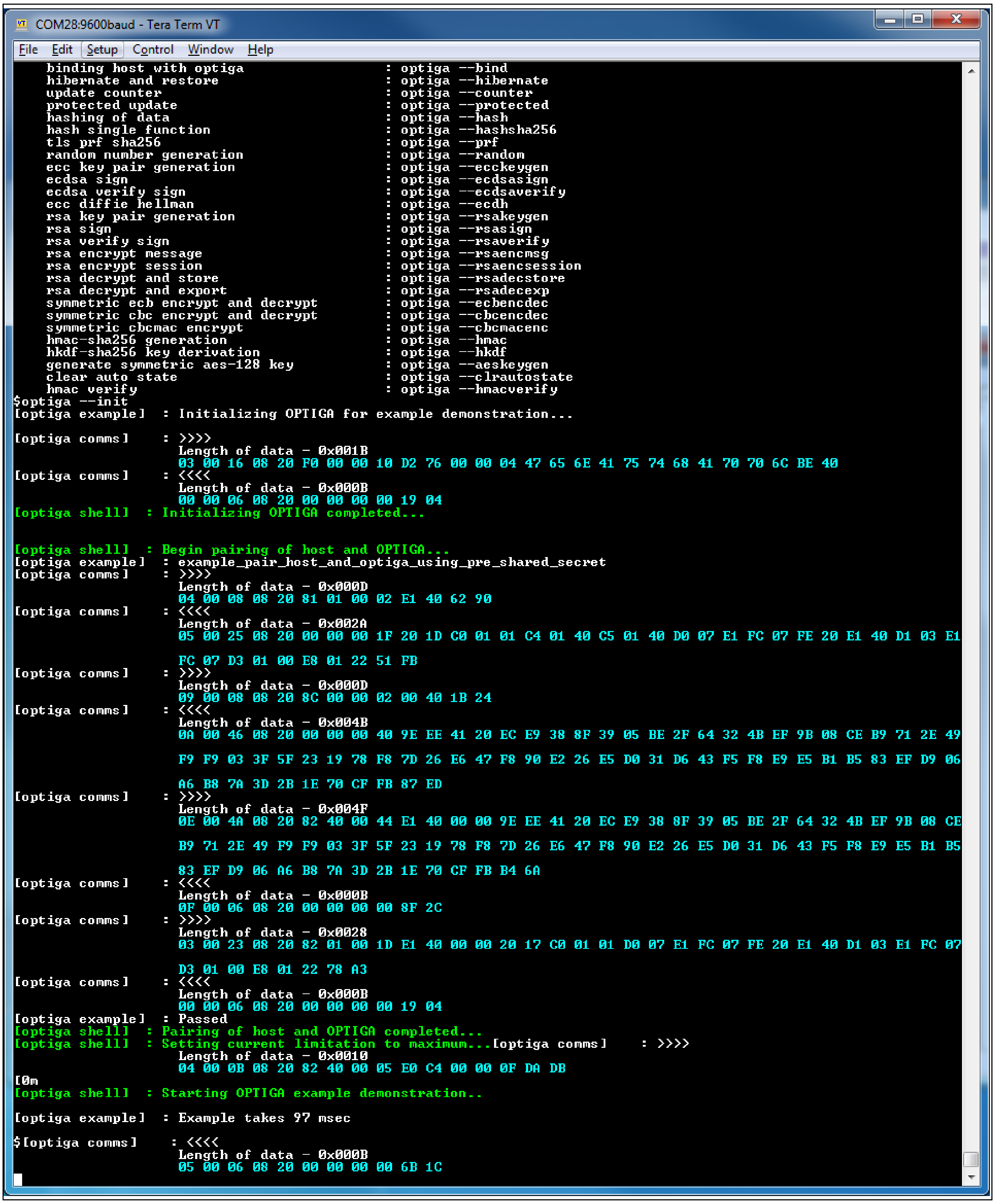
The macro OPTIGA_LIB_ENABLE_LOGGING provides complete control to enable/disable logging at host code. In addition, logging at UTIL, CRYPT, CMD and COMMS layer can be controlled using the following macros,

- OPTIGA_LIB_ENABLE_UTIL_LOGGING
- OPTIGA_LIB_ENABLE_CRYPT_LOGGING
- OPTIGA_LIB_ENABLE_CMD_LOGGING
- OPTIGA_LIB_ENABLE_COMMS_LOGGING

For Example,

1. To enable logging for only COMMS layer, enable OPTIGA_LIB_ENABLE_COMMS_LOGGING and disable rest all layer macros.
2. Build and run the project as described in section 4.2.

Using OPTIGA™ Trust M


COM28:9600baud - Tera Term VT
File Edit Setup Control Window Help

```
    binding host with optiga                : optiga --bind
    hibernate and restore                   : optiga --hibernate
    update counter                          : optiga --counter
    protected update                        : optiga --protected
    hashing of data                         : optiga --hash
    hash single function                    : optiga --hashsha256
    tls prf sha256                          : optiga --prf
    random number generation                : optiga --random
    ecc key pair generation                 : optiga --ecckeygen
    ecdsa sign                              : optiga --ecdsasign
    ecdsa verify sign                       : optiga --ecdsaverify
    ecc diffie hellman                      : optiga --ecdh
    rsa key pair generation                 : optiga --rsakeygen
    rsa sign                                : optiga --rsasign
    rsa verify sign                         : optiga --rsaverify
    rsa encrypt message                     : optiga --rsaencmsg
    rsa encrypt session                     : optiga --rsaencsession
    rsa decrypt and store                   : optiga --rsadecstore
    rsa decrypt and export                  : optiga --rsadecexp
    symmetric ecb encrypt and decrypt       : optiga --ecbencdec
    symmetric cbc encrypt and decrypt       : optiga --cbcencdec
    symmetric cbcmac encrypt                : optiga --cbcmacenc
    hmac-sha256 generation                  : optiga --hmac
    hkdf-sha256 key derivation              : optiga --hkdf
    generate symmetric aes-128 key          : optiga --aeskeygen
    clear auto state                        : optiga --clrautostate
    hmac verify                             : optiga --hmacverify
$optiga --init
[optiga example]  : Initializing OPTIGA for example demonstration...

[optiga comms]    : >>>>
                    Length of data - 0x001B
                    03 00 16 08 20 F0 00 00 10 D2 76 00 00 04 47 65 6E 41 75 74 68 41 70 70 6C BE 40
[optiga comms]    : <<<<
                    Length of data - 0x000B
                    00 00 06 08 20 00 00 00 00 19 04
[optiga shell]  : Initializing OPTIGA completed...


[optiga shell]  : Begin pairing of host and OPTIGA...
[optiga example]  : example_pair_host_and_optiga_using_pre_shared_secret
[optiga comms]    : >>>>
                    Length of data - 0x000D
                    04 00 08 08 20 81 01 00 02 E1 40 62 90
[optiga comms]    : <<<<
                    Length of data - 0x002A
                    05 00 25 08 20 00 00 00 1F 20 1D C0 01 01 C4 01 40 C5 01 40 D0 07 E1 FC 07 FE 20 E1 40 D1 03 E1

                    FC 07 D3 01 00 E8 01 22 51 FB
[optiga comms]    : >>>>
                    Length of data - 0x000D
                    09 00 08 08 20 8C 00 00 02 00 40 1B 24
[optiga comms]    : <<<<
                    Length of data - 0x004B
                    0A 00 46 08 20 00 00 00 40 9E EE 41 20 EC E9 38 8F 39 05 BE 2F 64 32 4B EF 9B 08 CE B9 71 2E 49

                    F9 F9 03 3F 5F 23 19 78 F8 7D 26 E6 47 F8 90 E2 26 E5 D0 31 D6 43 F5 F8 E9 E5 B1 B5 83 EF D9 06

                    A6 B8 7A 3D 2B 1E 70 CF FB 87 ED
[optiga comms]    : >>>>
                    Length of data - 0x004F
                    0E 00 4A 08 20 82 40 00 44 E1 40 00 00 9E EE 41 20 EC E9 38 8F 39 05 BE 2F 64 32 4B EF 9B 08 CE

                    B9 71 2E 49 F9 F9 03 3F 5F 23 19 78 F8 7D 26 E6 47 F8 90 E2 26 E5 D0 31 D6 43 F5 F8 E9 E5 B1 B5

                    83 EF D9 06 A6 B8 7A 3D 2B 1E 70 CF FB B4 6A
[optiga comms]    : <<<<
                    Length of data - 0x000B
                    0F 00 06 08 20 00 00 00 00 8F 2C
[optiga comms]    : >>>>
                    Length of data - 0x0028
                    03 00 23 08 20 82 01 00 1D E1 40 00 00 20 17 C0 01 01 D0 07 E1 FC 07 FE 20 E1 40 D1 03 E1 FC 07

                    D3 01 00 E8 01 22 78 A3
[optiga comms]    : <<<<
                    Length of data - 0x000B
                    00 00 06 08 20 00 00 00 00 19 04
[optiga example]  : Passed
[optiga shell]  : Pairing of host and OPTIGA completed...
[optiga shell]  : Setting current limitation to maximum...[optiga comms]    : >>>>
                    Length of data - 0x0010
                    04 00 0B 08 20 82 40 00 05 E0 C4 00 00 0F DA DB
[0m
[optiga shell]  : Starting OPTIGA example demonstration..

[optiga example]  : Example takes 97 msec

$[optiga comms]    : <<<<
                    Length of data - 0x000B
                    05 00 06 08 20 00 00 00 00 6B 1C
```

Figure 17 **Logging data with only COMMS layer enabled**

Note: *Execution time of example increase if more logging information is printed.*

4.2 Advanced Setup

This section explains the steps to build and run OPTIGA™ Trust M example application.

4.2.1 Setting up DAVE™ IDE on PC

1. Refer to the installation guide in <DAVE_PATH> to install DAVE™ on your PC.
2. Start DAVE™ from <DAVE_ PATH>\eclipse\DAVE.exe. The following splash screen will appear:

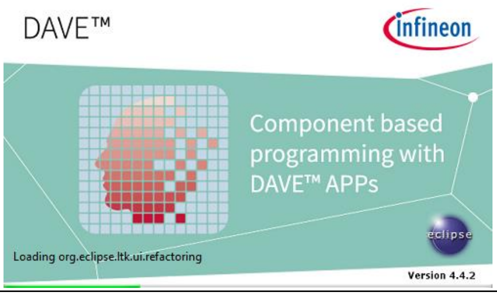
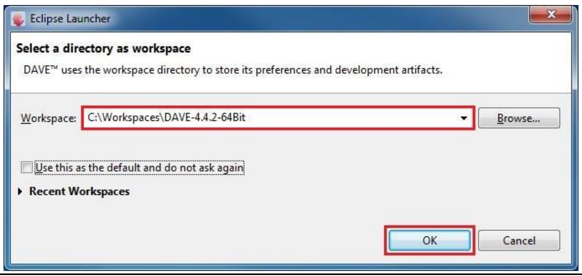

DAVE™
infineon
Component based programming with DAVE™ APPs
eclipse
Loading org.eclipse.ltk.ui.refactoring
Version 4.4.2

Figure 18 Opening DAVE™

3. Eclipse Launcher will pop-up. Select the workspace for DAVE™.



Eclipse Launcher

Select a directory as workspace

DAVE™ uses the workspace directory to store its preferences and development artifacts.

Workspace: C:\Workspaces\DAVE-4.4.2-64Bit

Figure 19 Select workspace

4. DAVE IDE Perspective window.

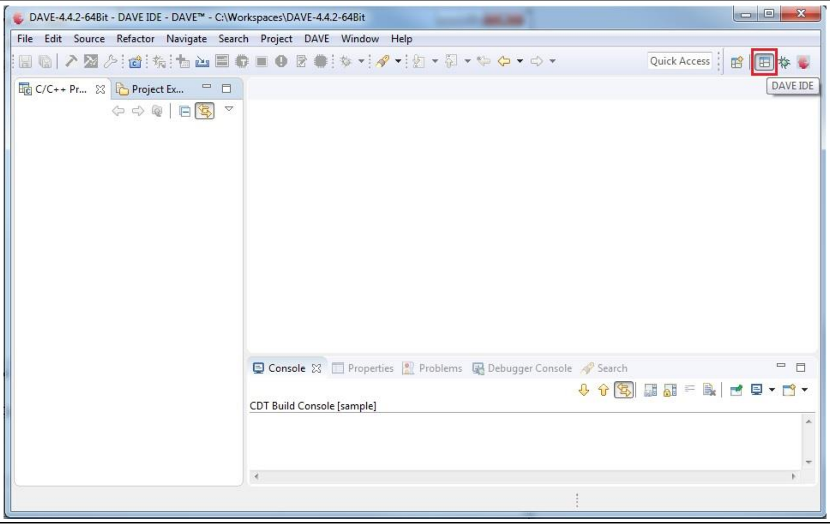

DAVE-4.4.2-64Bit - DAVE IDE - DAVE™ - C:\Workspaces\DAVE-4.4.2-64Bit
File Edit Source Refactor Navigate Search Project DAVE Window Help
Quick Access
DAVE IDE
C/C++ Pr...
Project Ex...
Console
Properties
Problems
Debugger Console
Search
CDT Build Console [sample]

Figure 20 DAVE IDE Perspective window

4.2.2 Running OPTIGA™ Trust M Example Application Project with DAVE™

1. Make the connections between XMC4800 IoT Connectivity Kit, My IoT Adapter and OPTIGA™ Shield2Go. Refer Figure 6.
2. Power up the kit by connecting Micro USB cable between PC and Debugger micro USB. For placement of Debugger micro USB refer Figure 3.
3. Import example application project into DAVE IDE, by navigating through **File -> Import.** In Import pop-up, select Existing Projects into Workspace under General and then click Next.

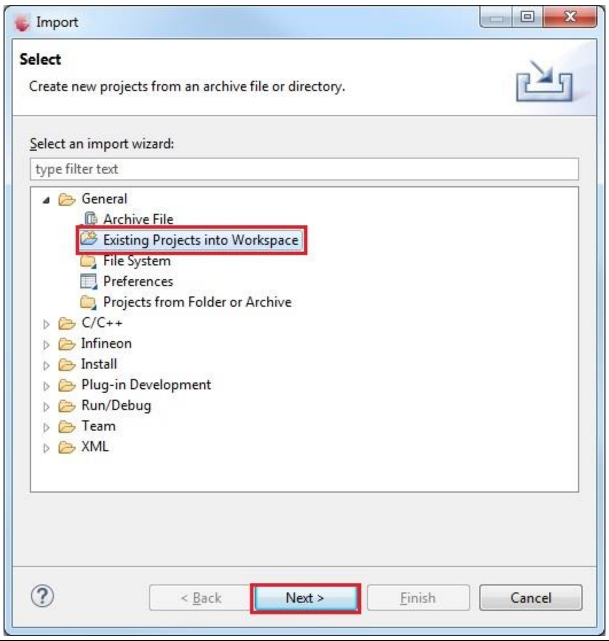


Figure 21 Import DAVE project window

4. Browse to **<INSTALLDIR>\projects\xmc4800_iot_kit\dave4** for Select root directory, select xmc4800_optiga_example and then click Finish.

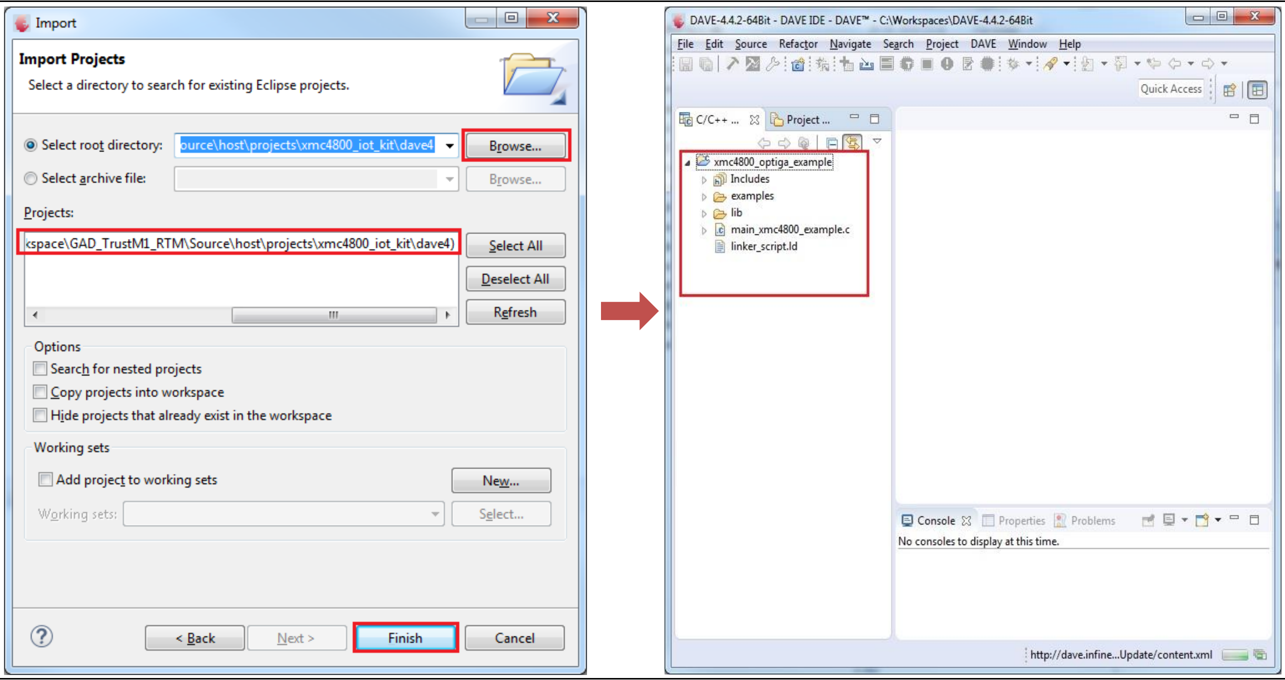

Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory: ource\host\projects\xmc4800_iot_kit\dave4
Browse...
Select archive file:
Browse...
Projects:
kspace\GAD_TrustM1_RTM\Source\host\projects\xmc4800_iot_kit\dave4)
Select All
Deselect All
Refresh
Options
Search for nested projects
Copy projects into workspace
Hide projects that already exist in the workspace
Working sets
Add project to working sets
New...
Working sets:
Select...
< Back
Next >
Finish
Cancel
DAVE-4.4.2-64Bit - DAVE IDE - DAVE™ - C:\Workspaces\DAVE-4.4.2-64Bit
File Edit Source Refactor Navigate Search Project DAVE Window Help
Quick Access
C/C++ ...
Project ...
xmc4800_optiga_example
Includes
examples
lib
main_xmc4800_example.c
linker_script.ld
Console
Properties
Problems
No consoles to display at this time.
http://dave.infine...Update/content.xml

Figure 22 Import a example project

5. Set example project as an active project by right-click on project and select **Set Active Project**.

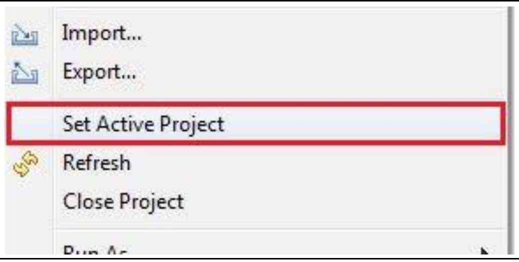


Figure 23 Example project set as active project

6. Select the build configuration by right-click on example project and then select **Build Configurations -> Set Active -> Debug.** To select Release build configuration, select **Build Configurations -> Set Active -> Release.**

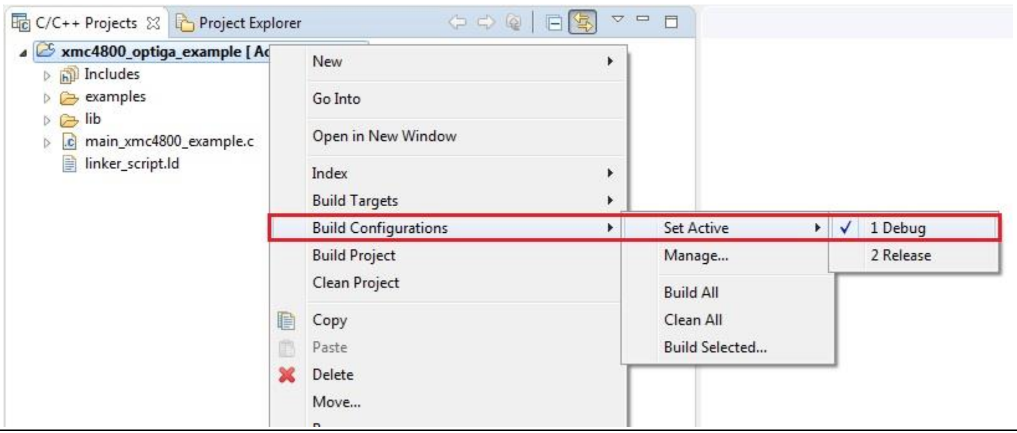


Figure 24 Build configuration selection

7. Build the project in Debug/Release configuration. It should be error free.

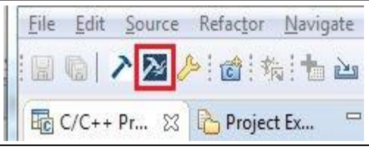
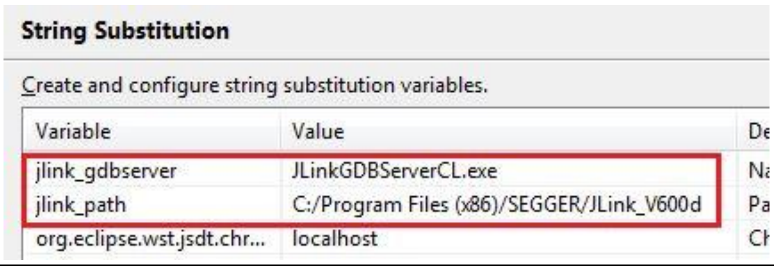


Figure 25 Example build in debug

8. Before launching the debugger, ensure that values are properly updated for variables like jlink_gdbserver and jlink_path. Navigate through **Window -> Preferences -> Run/Debug -> String Substitution** and update values as shown in the figure below:



String Substitution

Create and configure string substitution variables.

Variable	Value	De
jlink_gdbserver	JLinkGDBServerCL.exe	Na
jlink_path	C:/Program Files (x86)/SEGGER/JLink_V600d	Pa
org.eclipse.wst.jsdt.chr...	localhost	Cł

Figure 26 J-Link variable mapping

9. Launch debugger for debug of example application by clicking on bug symbol.

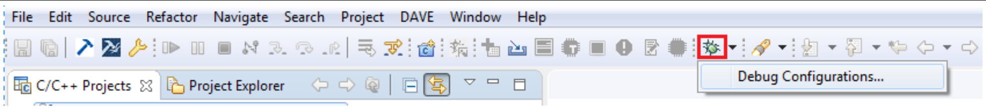
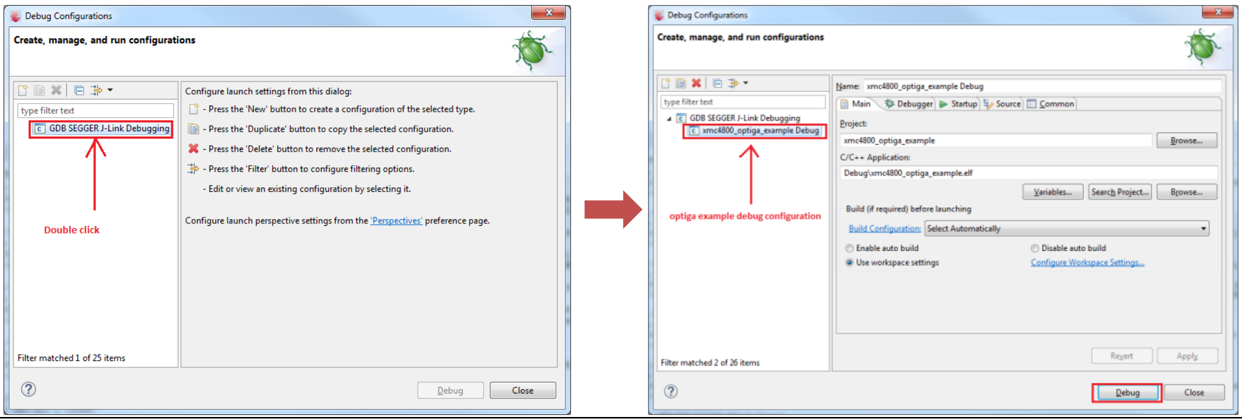


Figure 27 **Debugger launch**

10. Create a debug configuration and then click on Debug.


Debug Configurations
Create, manage, and run configurations
type filter text
GDB SEGGER J-Link Debugging
Double click
Filter matched 1 of 25 items
Debug
Close
optiga example debug configuration
Filter matched 2 of 26 items
Main
Debugger
Startup
Source
Common
Browse...
Variables...
Search Project...
Revert
Apply

Configure launch settings from this dialog:
- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

Name: xmc4800_optiga_example Debug

Project: xmc4800_optiga_example

C/C++ Application: Debug\xmc4800_optiga_example.elf

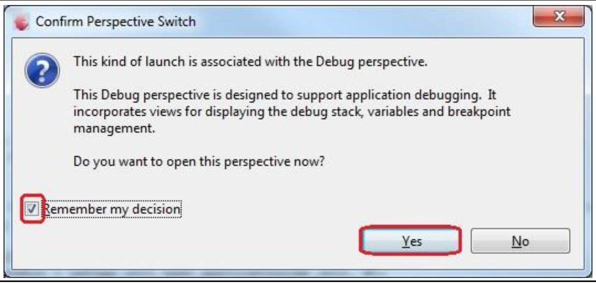
Build (if required) before launching

Build Configuration: Select Automatically

( ) Enable auto build ( ) Disable auto build
(•) Use workspace settings Configure Workspace Settings...

Figure 28 **Creating a example debug configuration**

11. If a window prompts to confirm the perspective switch, check the Remember my decision, and click yes.



This kind of launch is associated with the Debug perspective.

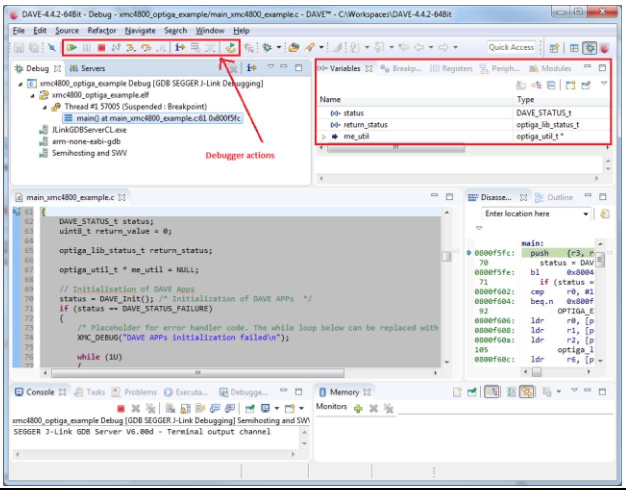
This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

[✓] Remember my decision

Figure 29 **Confirm perspective switch**

12. Start the debugger.



xmc4800_optiga_example Debug [GDB SEGGER J-Link Debugging]
xmc4800_optiga_example.elf
Thread #1 57005 (Suspended : Breakpoint)
main() at main_xmc4800_example.c:61 0x800f5fc
JLinkGDBServerCL.exe
arm-none-eabi-gdb
Semihosting and SWV

Name	Type
status	DAVE_STATUS_t
return_status	optiga_lib_status_t
me_util	optiga_util_t *

```
{
    DAVE_STATUS_t status;
    uint8_t return_value = 0;

    optiga_lib_status_t return_status;

    optiga_util_t * me_util = NULL;

    // Initialisation of DAVE Apps
    status = DAVE_Init(); /* Initialization of DAVE APPs  */
    if (status == DAVE_STATUS_FAILURE)
    {
        /* Placeholder for error handler code. The while loop below can be replaced with
        XMC_DEBUG("DAVE APPs initialization failed\n");

        while (1U)
```

```
           main:
0800f5fc:   push    {r3, r
 70           status = DAV
0800f5fe:   bl      0x8004
 71           if (status =
0800f602:   cmp     r0, #1
0800f604:   beq.n   0x800f
 92               OPTIGA_E
0800f606:   ldr     r0, [p
0800f608:   ldr     r1, [p
0800f60a:   ldr     r2, [p
105               optiga_l
0800f60c:   ldr     r6, [p
```

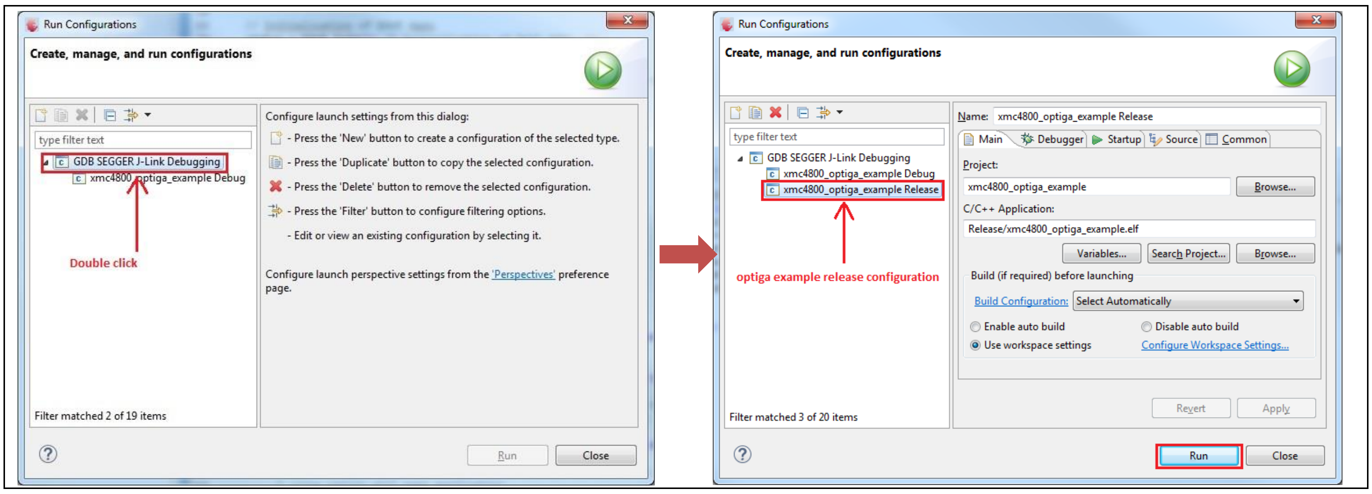
xmc4800_optiga_example Debug [GDB SEGGER J-Link Debugging] Semihosting and SWV

```
SEGGER J-Link GDB Server V6.00d - Terminal output channel
```

Figure 30 **Starting a debug session**

13. Refer section 4.1.3.1 to setup USBD_VCOM for logger.
14. Example logs can be seen on terminal as shown in Figure 16.
15. To build project in release configuration, select the build configuration as **Release** as shown in Figure 24 and build the project again.
16. Create a new configuration by right-click on example project and **Run As -> Run Configurations.** Double-click on GDB SEGGER J-Link Debugging and select Release configuration then click on Run. The logs of the executing example can be seen on the terminal as shown in Figure 16.

Note: Connect the USB cable to Debugger micro USB before running Release configuration. And use Micro USB for logger.


Run Configurations
Create, manage, and run configurations
type filter text
GDB SEGGER J-Link Debugging
xmc4800_optiga_example Debug
Double click
Filter matched 2 of 19 items
Run
Close
Run Configurations
Create, manage, and run configurations
type filter text
GDB SEGGER J-Link Debugging
xmc4800_optiga_example Debug
xmc4800_optiga_example Release
optiga example release configuration
Filter matched 3 of 20 items
Main
Debugger
Startup
Source
Common
Browse...
Variables...
Search Project...
Browse...
Build Configuration:
Revert
Apply
Run
Close

Configure launch settings from this dialog:
- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

Name: xmc4800_optiga_example Release

Project: xmc4800_optiga_example

C/C++ Application: Release/xmc4800_optiga_example.elf

Build (if required) before launching

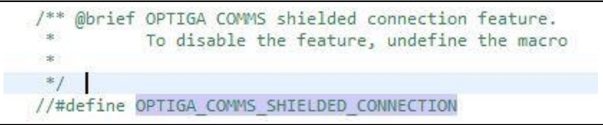
Build Configuration: Select Automatically

Enable auto build / Disable auto build

[✓ Use workspace settings] Configure Workspace Settings...

Figure 31 Run example with release configuration

17. To execute the example without shielded connection, disable the macro OPTIGA_COMMS_SHIELDED_CONNECTION in file optiga_lib_config.h.



```
/** @brief OPTIGA COMMS shielded connection feature.
 *        To disable the feature, undefine the macro
 *
 */
//#define OPTIGA_COMMS_SHIELDED_CONNECTION
```

Figure 32 OPTIGA_COMMS_SHIELDED_CONNECTION disable

18. To run example application with logging for different layers refer to section 4.1.3.2.

5 Troubleshooting

Table 6 Troubleshooting

No	Problem	Reason	Solution
1	The Green LED light is "Not on" on XMC4800 IoT Connectivity kit	No power supply	Verify that power supply is connected to XMC4800 IoT Connectivity kit.
2	CDC port not detected	SW not correctly installed	In device manager, click on the malfunctioning CDC port and select to manually install the driver. Provide directory as C:\ for path to install the driver.
3	Problem occurred during debugger launch	Debug session is not terminated	Go to Debug perspective and remove all terminated launches.
		Jlink path not updated	Refer [8] for correctly updating the path

Revision History

Table 7

Document version	Date of release	Description of changes
3.10	2020-09-21	Release to production
3.00	2020-06-01	ES Release
0.50	2020-05-27	Initial Version

Edition 2020-09-21

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: CSSCustomerService@infineon.com

Document reference