Getting started with TRAVEO™ T2G Body High Lite Kit

About this document

Scope and purpose

The TRAVEO™ T2G Body High Lite Evaluation Kit (KIT_T2G-B-H_LITE) is a low-cost development platform designed to help users evaluate the capabilities of the TRAVEO™ T2G CYT4BF device from the TRAVEO™ Body High family of microcontrollers. This document provides a step-by-step guide to getting started with the kit, including setting up the hardware and running the associated example software.

Intended audience

This document is intended for all users who are interested in evaluating the capabilities of the CYT4BF device, such as testing new software or exploring different hardware designs.

Keypoints

- Set up KIT_T2G-B-H_LITE and ModusToolbox™; create, build, program, and debug projects in Eclipse IDE
- Add retarget-io via Library Manager to use *printf ()* over UART (CM0+), while CM7_0 handles GPIO/LED and button interrupts
- Use HAL APIs for portable GPIO/interrupts and Device Configurator to remap LEDs/buttons without changing code
- Know when to run Clean vs. incremental builds; use IDE debug views and built-in API docs

About this product family

Product family

The TRAVEO™ T2G Body High family (CYT4BF series) is Infineon's automotive-grade MCU line for advanced body electronics and zonal architectures. These devices pair high-performance Arm® Cortex®-M7 processing with a companion Cortex®-M0+ for real-time control and integrate rich on-chip resources: safety features (ECC memories, diagnostics, watchdogs) to support ISO 26262, hardware security (cryptography, true random number generation, secure boot), and extensive connectivity including multiple CAN FD, LIN, Ethernet, SPI/I2C/UART. They also offer advanced timers/PWM for lighting and actuator control, SAR ADCs, capacitive sensing on selected parts, and external memory interfaces (QSPI/SMIF), all backed by the ModusToolbox ecosystem with BSPs, HAL/PDL middleware, and configurators for fast development.

Target applications

- Automotive body electronics & power distribution
- In-vehicle infotainment & HMI

Table of contents

About this document **1**
About this product family **1**
Table of contents **2**
1 Overview of ModusToolbox™ **3**
1.1 Software structure 3
1.2 Application interface 4
1.3 ModusToolbox™ Eclipse IDE overview 5
2 TRAVEO™ T2G Evaluation Kit (KIT_T2G-B-H_LITE) setup **7**
2.1 Board details 7
2.1.1 CYT4BF series 8
2.2 Hardware and software setup 9
2.2.1 Software setup 9
2.2.2 Hardware setup 9
2.3 “Hello World” application (single-core) 10
2.3.1 Create “Hello World” application 10
2.3.2 Build “Hello World” 12
2.3.3 Flash (Program and Test) “Hello World” 13
2.4 Create a basic multi-core application 14
2.5 Create “MyTRAVEO” application 15
2.6 Add “retarget-io” library 15
2.7 Coding 15
2.8 Build 19
2.9 Flash (Program and Test) 20
3 GPIO Handling – LED Blinking **21**
3.1 Overview 21
3.2 LED Blinking 21
3.3 LED/User button configuration 25
4 Related resources **29**
5 Appendix **30**
5.1 Appendix-A: Software installation 30
5.1.1 Infineon Developer Center Launcher installation 30
5.2 Appendix B – Final Code 35
5.2.1 MyTRAVEO.proj_cm0p Final Code 35
5.2.2 MyTRAVEO.proj_cm7_0 Final Code 37
5.3 Appendix C – Debugging 41
5.4 Appendix D– Documentation 42
6 References **43**
Revision history **44**
Disclaimer **45**

1 Overview of ModusToolbox™

This section is a high-level overview of ModusToolbox™ software and its features. ModusToolbox™ is a modern and extensible development ecosystem supporting a wide range of Infineon microcontroller devices, including TRAVEO™ T2G family MCUs. The tool supports embedded application development within Eclipse IDE, Microsoft Visual Studio Code, IAR Embedded Workbench, and Arm MDK with µVision IDE.

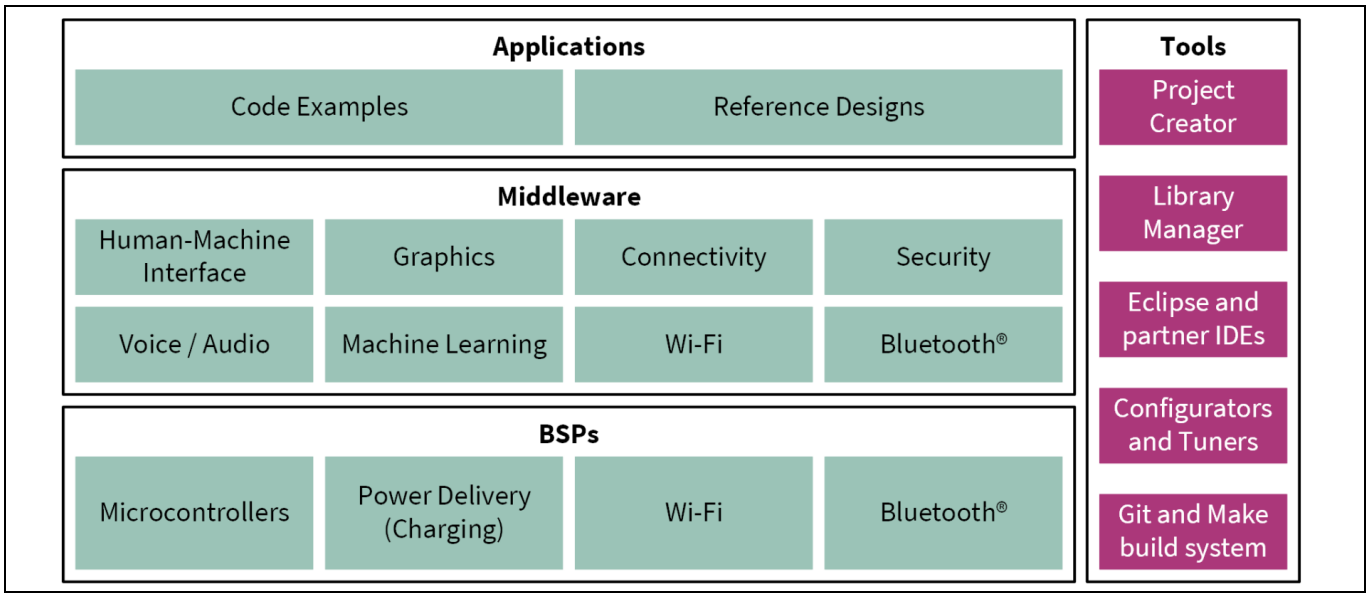


Figure 1 ModusToolbox™ software

Peripheral drivers are available to empower the developer to leverage portable application code and available middleware, while being able to access the full capabilities of the target device, as needed. The API documentation provides detailed information on the available functions and use cases available for the device-specific low-level drivers. Middleware libraries are provided as online GitHub repositories from Infineon and partners, and they can be easily integrated into the user's application using the provided ModusToolbox™ library manager. Available middleware includes software and libraries spanning HMI, graphics, connectivity, security, voice and audio, machine learning, Wi-Fi, Bluetooth®, and more.

ModusToolbox™ simplifies the edit, compile, and debug cycle by offering flexibility in the choice of the compiler, editor, debugger, and revision control system. Its setup program offers an option to install an Eclipse-based IDE. The Eclipse IDE for ModusToolbox™ includes additional features to enable easy access to the full suite of ModusToolbox™ tools and configurators, along with documentation, debug launch configurations, and command line terminal access for direct access to the ModusToolbox™ build system.

1.1 Software structure

The Tools provided in the ModusToolbox Software including Project Creator, Library Manager, and Configurators. This help empower you to create designs and adapt easily.

Some of the key uses of these tools include:

- Creation of new embedded applications
- Managing software components
- Configuring peripherals
- Developing and debugging applicationstools

Overview of ModusToolbox™

The ModusToolbox™ software also provides several applications that serve as a starting point for application development, including sample code examples and reference designs. These applications can be used to get started with the BH-LITE kit and then built upon to create custom applications.

Libraries and Middleware

The tool provides multiple libraries and middleware containing the middleware for all the code examples and other applications to run smoothly. Some of the key libraries and middlewares include:

- **Board support package (BSP):** Supports the target device and board with multiple options available based on user's design
- **Hardware abstraction layer (HAL):** Provides low-level drivers (LLD) of simple functions and high-level interface which are part of the HAL Drivers. These are very commonly used for easy migration
- **Peripheral Driver Library (PDL):** Provides LLD to support all hardware features of Infineon's MCUs, including CYT4BF. The PDL provided by Infineon, includes high-level APIs and offer better functionality

HAL is a software layer that serves as an interface between higher-level software and the hardware of a computing system. In the context of general-purpose input/output (GPIO) handling, the HAL is used to abstract the low-level hardware details of interacting with GPIO pins, making it easier to use for tasks like reading sensor data, controlling LEDs, or interfacing with external devices. The HAL provides a uniform way for the software to interact with these pins regardless of the specific hardware platform

Note: If ModusToolBox™ IDE is not already installed, refer to Appendix.

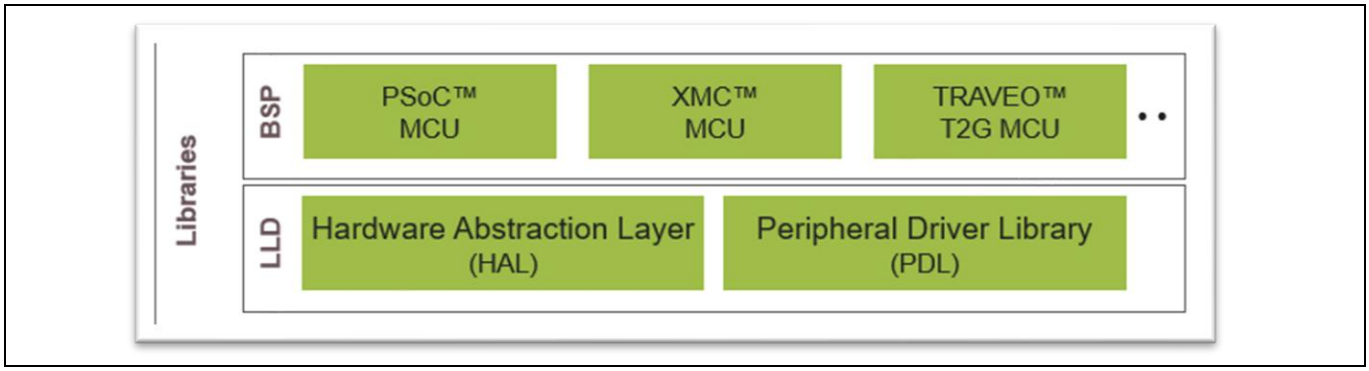


Figure 2 ModusToolbox™ libraries

The Documentation is provided in the ModusToolbox™ software and has all the required details for the APIs used, Low Level drivers information and even training manuals. These are hosted in GitHub repository.

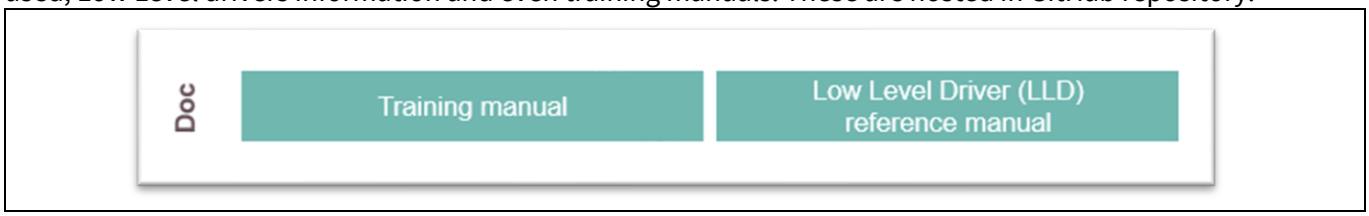


Figure 3 ModusToolbox™ documentation

1.2 Application interface

ModusToolbox™ provides a flexible application Interface that allows any application to access the hardware components in four different ways:

- Using HAL, which interacts with the PDL through structures created by the HAL

- Using PDL through structures created by configurator tools
- Using PDL through structures created manually
- Using direct register read and writes

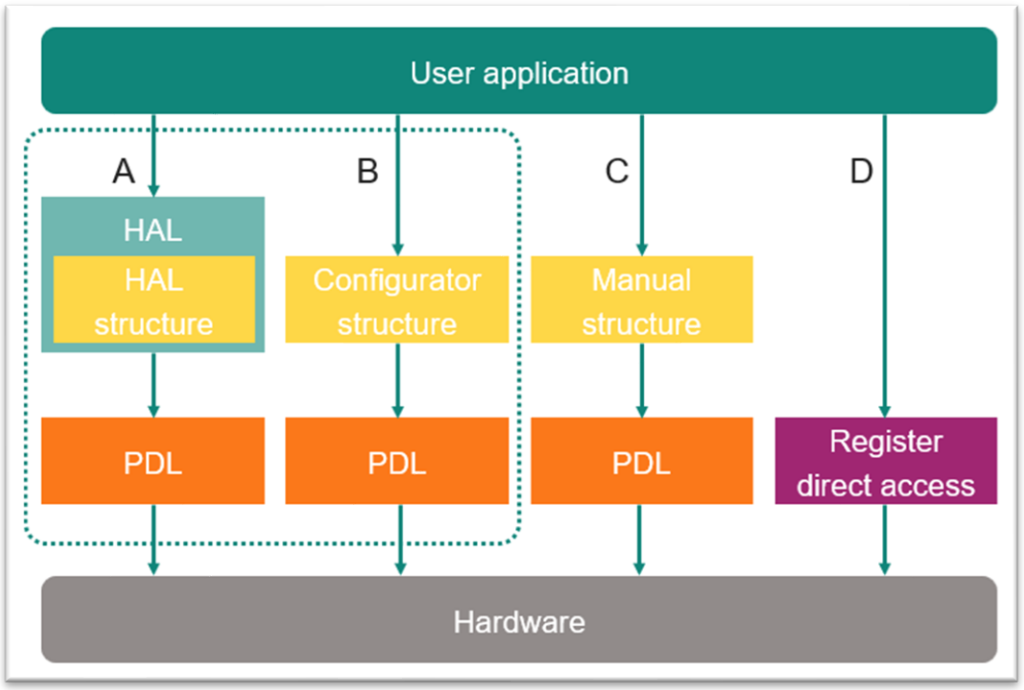


Figure 4 Primary ways for an application to interact with the hardware

1.3 ModusToolbox™ Eclipse IDE overview

Figure 5 shows the overview of the Eclipse IDE for ModusToolbox™. The following are the sections available in the Eclipse IDE:

- **Menus and Toolbars:** This section contains the main navigation menus and toolbars granting the access to a wide range of functions and features within ModusToolbox™
- **Project Explorer:** This area showcases the hierarchical structure of the current project, allowing easy navigation and management of files, folders, and project settings
- **Quick Panel:** The Quick Panel provides quick access to frequently used tools, commands, and settings, streamlining the development process
- **Code Editor:** This space serves as the central workspace for writing, editing, and managing code files, offering essential features for coding and software development
- **Console:** The console window provides real-time feedback by displaying system messages, build output and debugging information, enhancing the development experience

Overview of ModusToolbox™

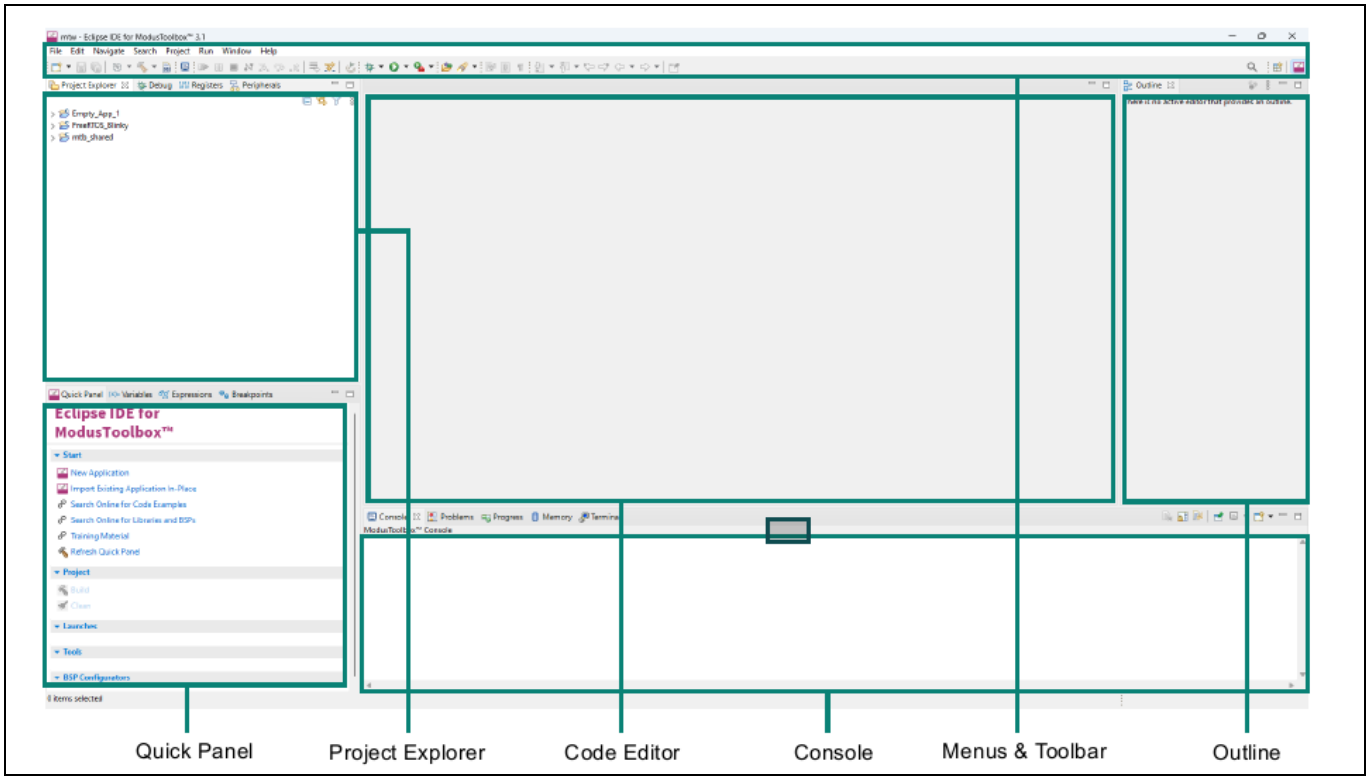
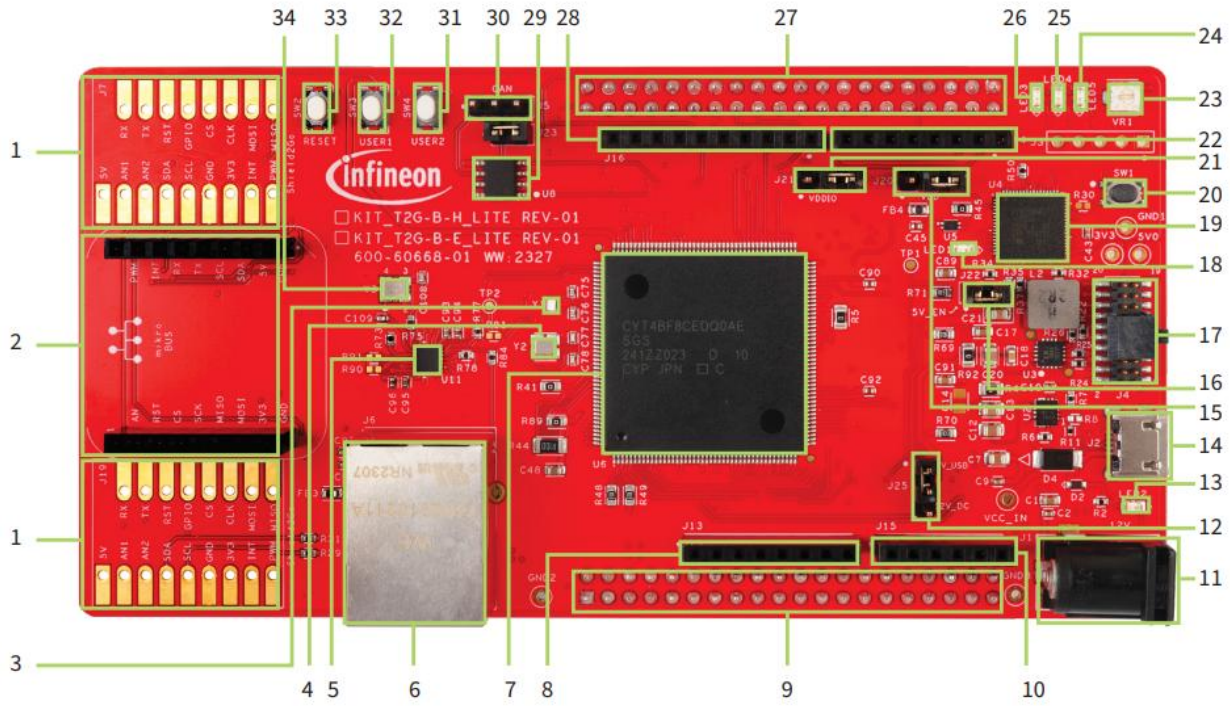


Figure 5 Eclipse IDE for ModusToolbox™

For more information about Eclipse IDE for ModusToolbox™, see [1] and [2] in the references section.

2 TRAVEO™ T2G Evaluation Kit (KIT_T2G-B-H_LITE) setup

2.1 Board details



1	Shield2Go connectors (not mounted) (J7, J19)	18	KitProg3 status LED (LED1)
2	mikroBUS connectors (J11, J12)	19	PSoC™ 5LP based KitProg3 (U4)
3	32.768-kHz oscillator for WCO (Y1)	20	Mode switch (SW1)
4	16-MHz oscillator for ECO (Y2)	21	VDDIO current measurement jumper (J21)
5	Ethernet physical layer (PHY) transceiver (U11)	22	8-pin Arduino header (J14)
6	RJ45 Ethernet connector (J6)	23	Potentiometer (VR1)
7	TRAVEO™ T2G microcontroller (U6)	24	USER LED3 (LED5)
8	8-pin Arduino header (J13)	25	USER LED2 (LED4)
9	Expansion header 1 (X1)	26	USER LED1 (LED3)
10	6-pin Arduino header (J15)	27	Expansion header 2 (X2)
11	DC power jack (J1)	28	10-pin Arduino header (J16)
12	Input power supply selection jumper (J25) - default closed (1-2)	29	CAN FD transceiver (U8)
13	Power LED (LED2)	30	CAN FD connector (J5)
14	KitProg3 USB micro-B connector (J2)	31	USER2 button (SW4)
15	VDD current measurement jumper (J20)	32	USER1 button (SW3)
16	5V Supply Enable jumper (J22) - default closed	33	RESET button (SW2)
17	MIPI10/20 debug connector (J4)	34	25-MHz crystal for Ethernet transceiver (Y3)

Figure 6 Peripheral and interfaces in the TRAVEO™ KIT_T2G-B-H_LITE Evaluation Kit

For more information about the TRAVEO™ KIT_T2G-B-H_LITE, see [3], [4], and [5] in the references section.

2.1.1 CYT4BF series

The KIT_T2G-B-H_LITE kit enables you to evaluate and develop applications using the TRAVEO™ T2G Body High family CYT4BF device. The TRAVEO™ T2G B-H MCU is specifically designed for automotive applications and features up to two 350 MHz Arm® Cortex® -M7 processors as the primary application processors, a 100-MHz Arm® Cortex® -M0+ processor that supports low-power operations, up to 8 MB flash and 1 MB SRAM, and programmable analog and digital peripherals that allow for faster time-to-market.

The TRAVEO™ T2G B-H Lite kit is equipped with a TRAVEO™ T2G Body High family CYT4BF MCU, two expansion headers, two expansion headers, two Shield2Go connectors, headers that are compatible with Arduino shield and mikroBUS.

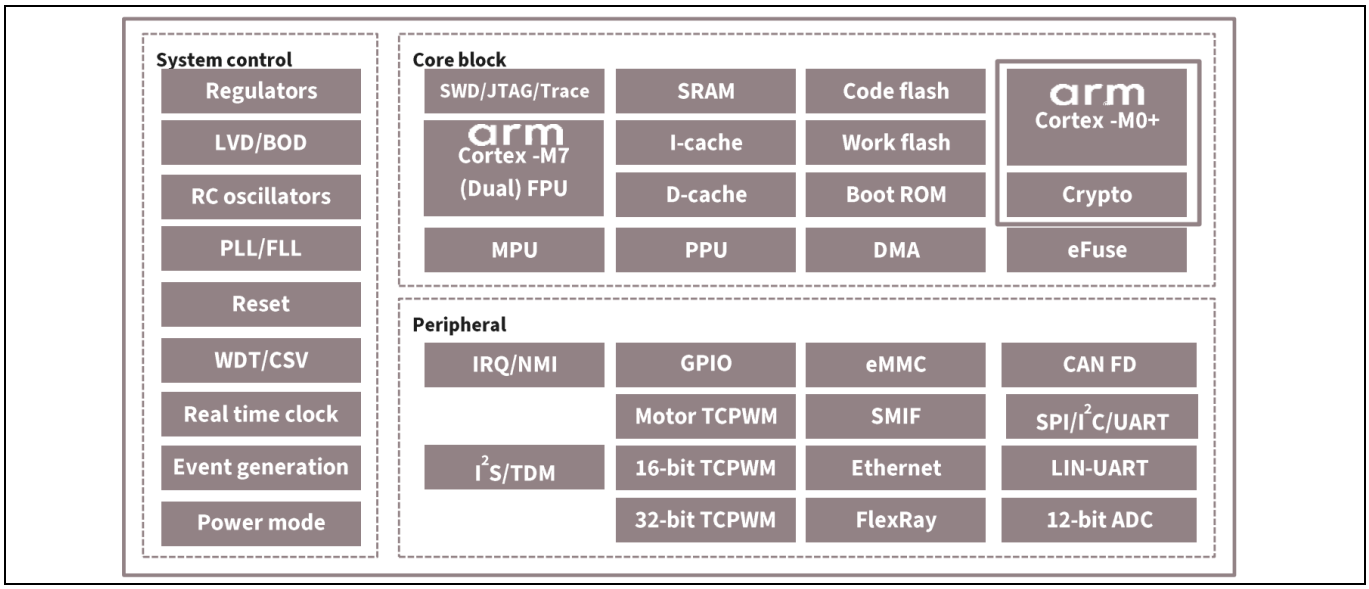


Figure 7 Block diagram of CYT4BF series

Additionally, the board features an onboard programmer/debugger (KitProg3), a 512-Mbit Dual QSPI NOR flash, a CAN FD transceiver, an Ethernet PHY transceiver with RJ45 connector interface, a Micro-B connector for an USB device interface, three user LEDs, one potentiometer, and two push buttons. The board supports operating voltages from 3.3 V to 5.0 V for the TRAVEO™ T2G B-H MCU.

To develop and debug projects on the TRAVEO™ T2G B-H Lite kit, you can use the ModusToolbox™ software, which is a set of tools that enable you to integrate these devices into your existing development methodology.

All the features of the CYT4BF MCU are listed below:

- 32-bit MCU core systems with 350-MHz Arm® Cortex®-M7 Dual, I/D-Cache
- 8MB flash, 256KB work flash, 1024KB SRAM, and Cortex®-M0+ for crypto
- Supply voltages ranging from 2.7V to 5.5V
- Multiple communication interfaces includes:
  - Up to 10-ch CAN FD, up to 11-ch SCB, and 20-ch LIN-UART
  - eMMC, SMIF (QSPI/HS-SPI), up to 2-ch 10/100/1000-Mbit ethernet and FlexRay
  - I^2S/TDM with TX 3-ch and RX 3-ch
- Analog to Digital Convertor (ADC), which has up to 96-ch, 12-bit with 3x successive approximation ADC (SAR ADC) units

- The CYT4BF has Up to 15-ch motor control, 87-ch 16-bit timer/counter/pulse-width modulation (TCPWM), and 16-ch 32-bit TCPWM and an event generation timer
- The different packages available are 176-pin TEQFP, 272-ball BGA, and 320-ball BGA

2.2 Hardware and software setup

2.2.1 Software setup

ModusToolbox™ 3.5 or above. After installing the software, see the ModusToolbox™ user guide to get an overview of the software.

2.2.2 Hardware setup

Before you start, ensure that you have the following:

- Ensure your board has:
  - Silk screen: KIT_T2G-B-H_LITE
  - MCU: CYT4BF8xx

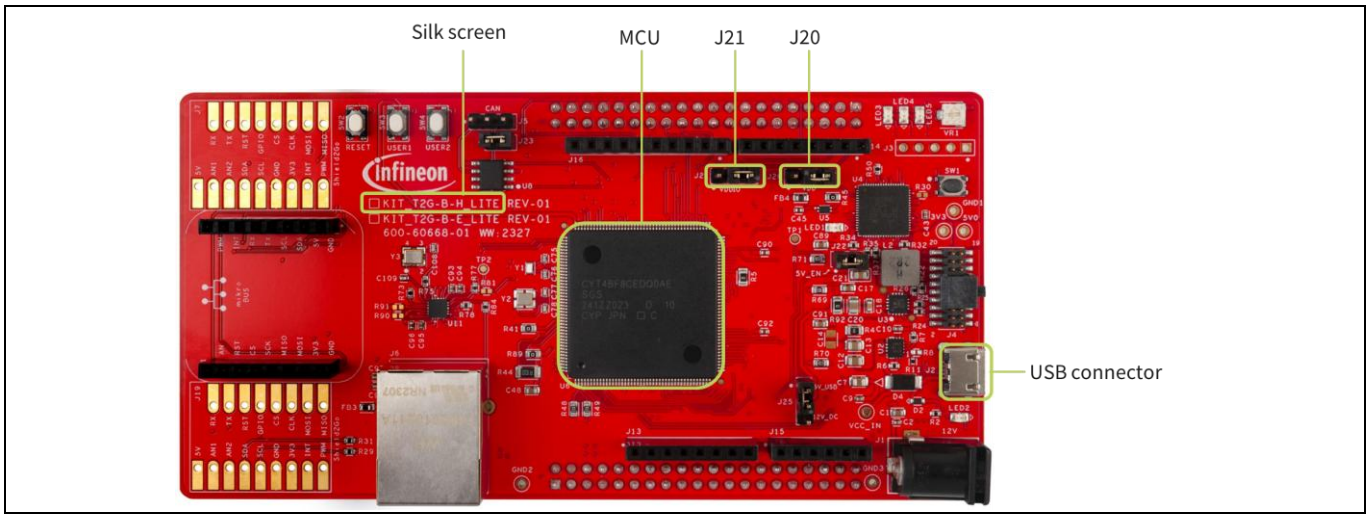


Figure 8 Powering-up the kit through USB Micro-B

- Ensure you have a USB Type-A to Micro-B cable to connect to the USB connector
- Ensure default jumpers are inserted on the evaluation kit:
  - J20: VDD power jumper
  - J21: VDDIO power jumper

These jumpers are essential for ensuring that the board receives the correct power supply for its operation.

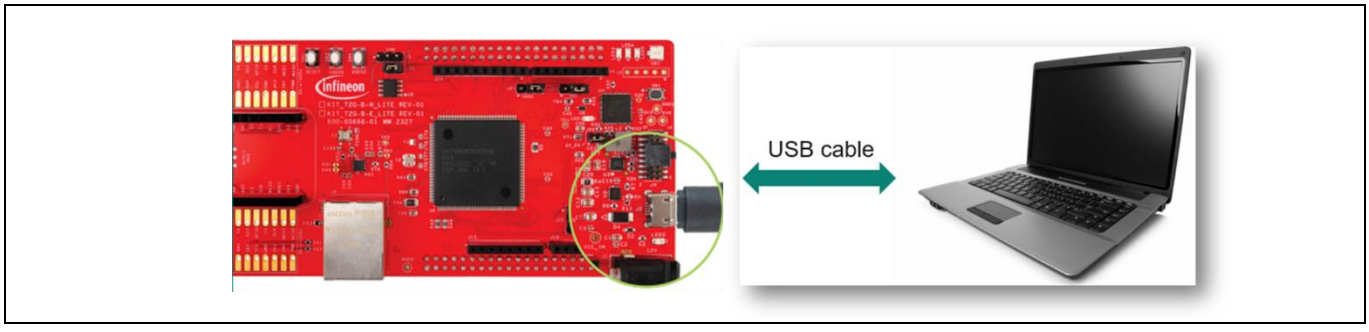


Figure 9 Power-up the kit

2.3 “Hello World” application (single-core)

In this section, you will test your software and hardware setup with the “Hello World” application.

2.3.1 Create “Hello World” application

To create a "Hello World" application, follow these steps:

1. **Launch Eclipse IDE**

   On Windows, select Eclipse for ModusToolbox™ item from the **Start** menu. For other operating systems, run the "modustoolbox" executable file. You can also launch Eclipse from the dashboard for all operating systems. For more information refer to the dashboard user guide

2. **Open Project Creator tool**

   Click the **New Application** in the Eclipse **Quick Panel**. You can also select **File** > **New** > **ModusToolbox™ Application**

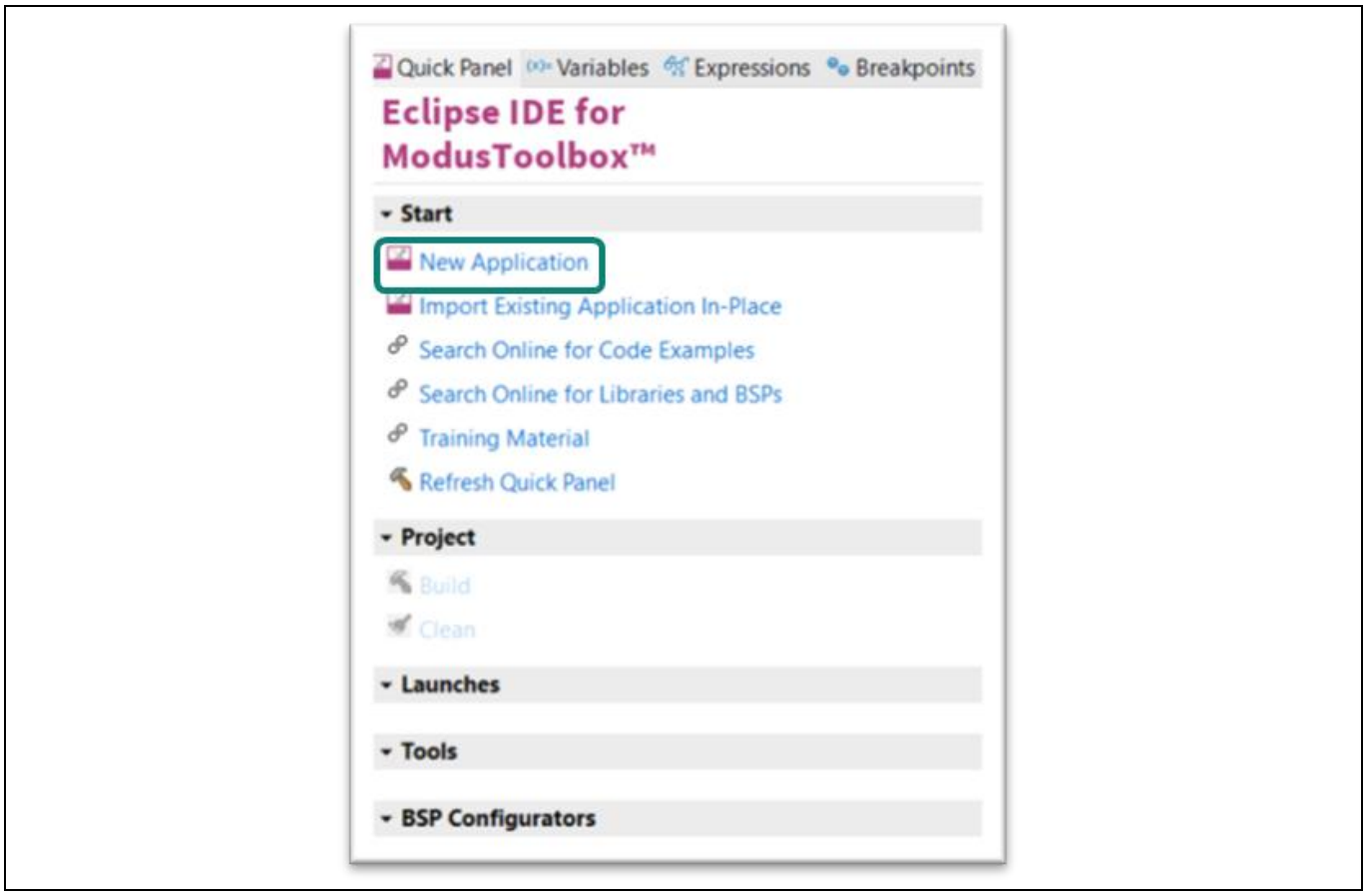


Figure 10 Open a New Application

3. **Choose board support package (BSP)**

   In the Project Creator, select **TRAVEO™ BSPs**, then **KIT_T2G-B-H_LITE**

TRAVEO™ T2G Evaluation Kit (KIT_T2G-B-H_LITE) setup

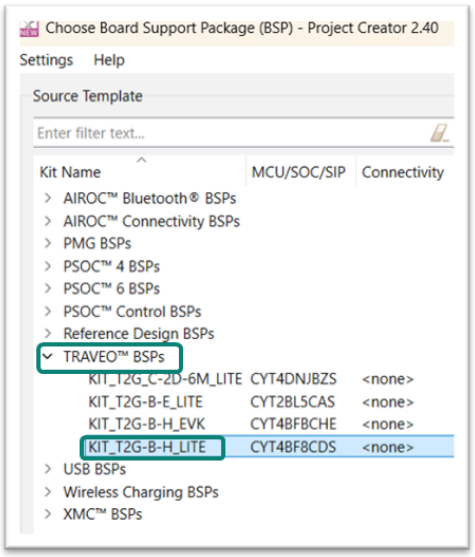


Figure 11 Choose the BSP

4. **Create application**
   a) Click **Next** > to open the **Select Application** page
   b) Select the check box next to the **Hello World** application
   c) Click **Create** begin the application creation process

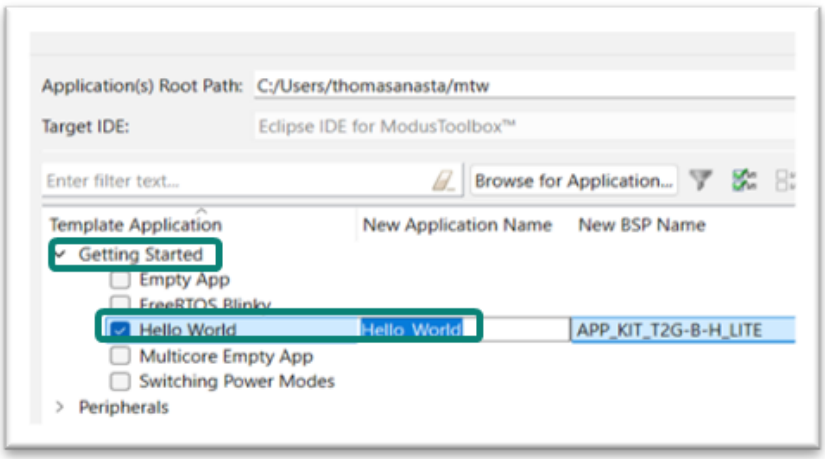


Figure 12 Creating a "Hello World" application

Note: The application creation process performs a git clone operation and downloads the selected application from the GitHub website. Depending on the selected application, this process can take several minutes.

5. When the cloning operation is complete, the Project Creator tool closes automatically if there are no errors or warnings. If there are warnings only, click **Close** and the application will open in Eclipse IDE

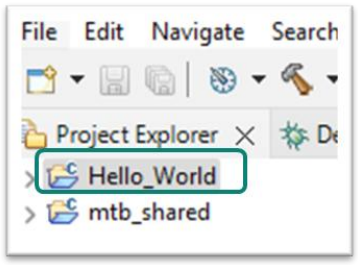


Figure 13 “Hello World” in Eclipse IDE

2.3.2 Build “Hello World”

From the previous section, the “Hello World” application should still be open.

1. Navigate to the **Project Explorer** window and select **Hello World** application

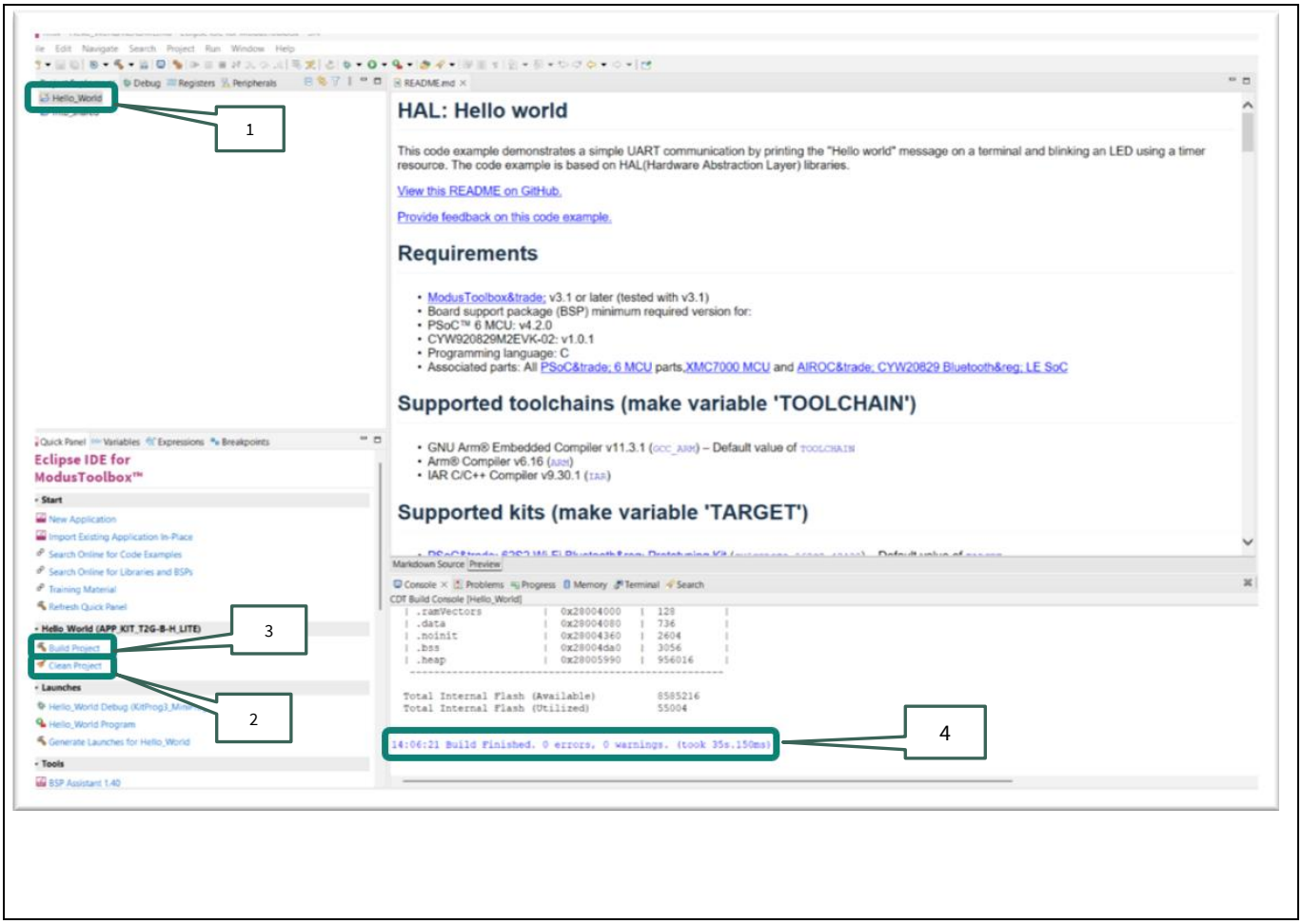


Figure 14 Build “Hello World” in Eclipse IDE

2. In the **Quick Panel**, under **Hello_World (APP_KIT_T2G-B-H_LITE)**, select **Clean Project**
3. When operation is complete, select **Build Project** in the same place
4. In the Console window, confirm successful build with no errors

2.3.3 Flash (Program and Test) “Hello World”

From the previous sections, the “**Hello World”** application and Tera Term should both still be open.

1. Navigate to the **Project Explorer** and select “**Hello World”** application

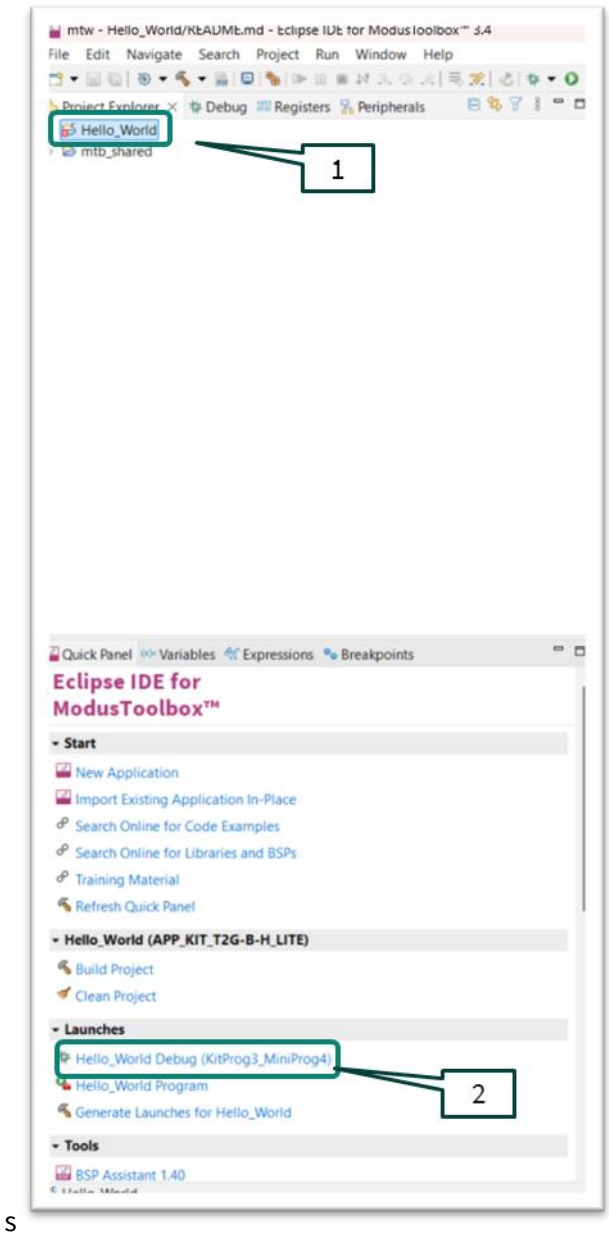


Figure 15 Selecting “Hello World” application

2. In the **Quick Panel**, under **“Launches”,** select **“Hello_World Program (KitProg3_MiniProg4)”**
3. Observe the message on the Tera Term, and verify that LED3 is blinking on the board

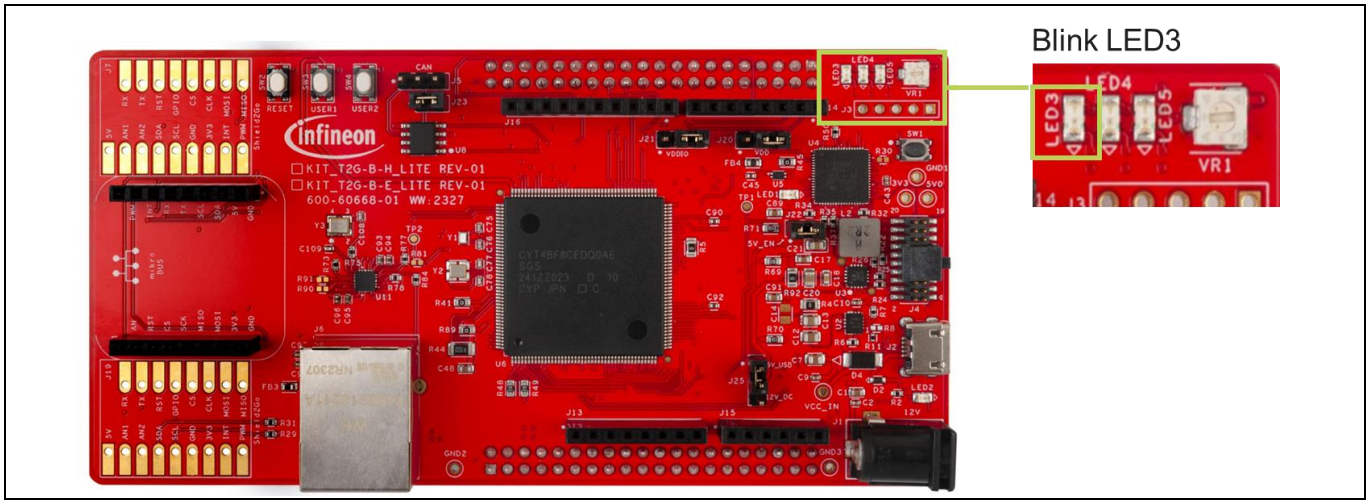


Figure 16 Running the "Hello_World Program (KitProg3_MiniProg4)"

4. Inside the Tera Term window, use the **Enter** key on your keyboard, and observe the message and the blinking LED change

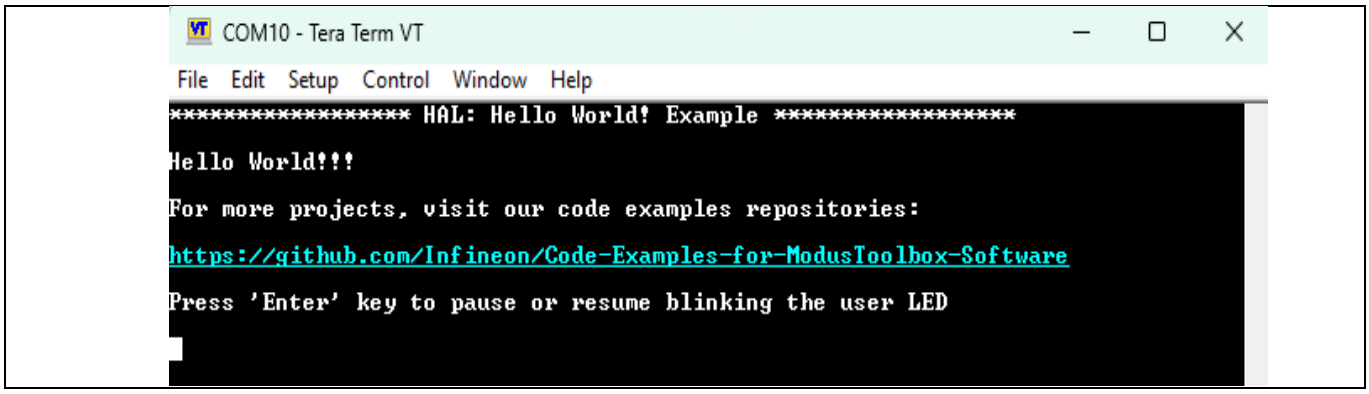


Figure 17 Printed UART message

2.4 Create a basic multi-core application

In this section, you will learn how to create an application to display a message on a UART terminal. Using the Library Manager, you will add the "retarget-io" library, which redirects standard I/O (STDIO) to a UART port, allowing you to print directly using `printf()`. You will create a "MyTRAVEO" application from the "Multicore" application and use UART to transmit data from MCU to PC. In this exercise, the UART will be controlled by the CM0+ core.

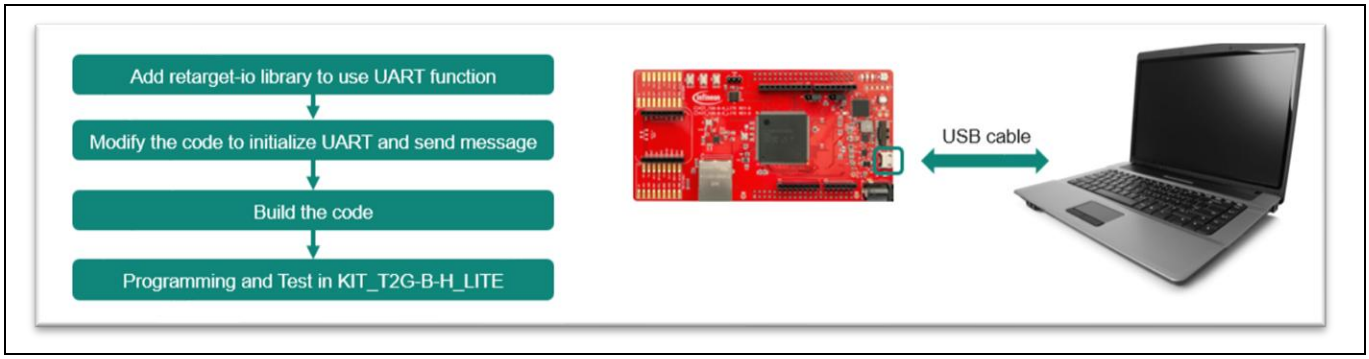


Figure 18 UART `printf()` workflow

2.5 Create “MyTRAVEO” application

Open a new project from the Project Creator in the same way you created the Hello World application in the previous section:

1. Ensure the Eclipse IDE is open. If not, launch it
2. In the **Quick Panel**, under **Start**, select **New Application** to open the Project Creator
3. In the Project Creator, select TRAVEO™ BSPs -> KIT_T2G-B-H_LITE and click Next
4. Select Getting Started, then Multicore Empty App
5. Change the name to **MyTRAVEO** and select **Create**
6. Once the application is created, return to the Eclipse IDE window, where the application will be displayed in the Project Explorer

2.6 Add “retarget-io” library

You will add the “retarget-io” library to the MyTRAVEO application. You can specify the TX pin, RX pin, and the baud rate through the cy_retarget_io_init() function. In this session, retarget-io library adds to proj_cm0p libraries.

1. In the **Project Explorer**, select **MyTRAVEO** to ensure the CM0+ project is active
2. In the Quick Panel, under Tools, select Library Manager 2.40
3. Select proj_cm0p Libraries, click on Add Library
4. Under **Peripherals**, select and enable **retarget-io**, and click **OK**
5. Ensure **retarget-io** has been added, then select **Update**. After the update, close the **Library Manager**

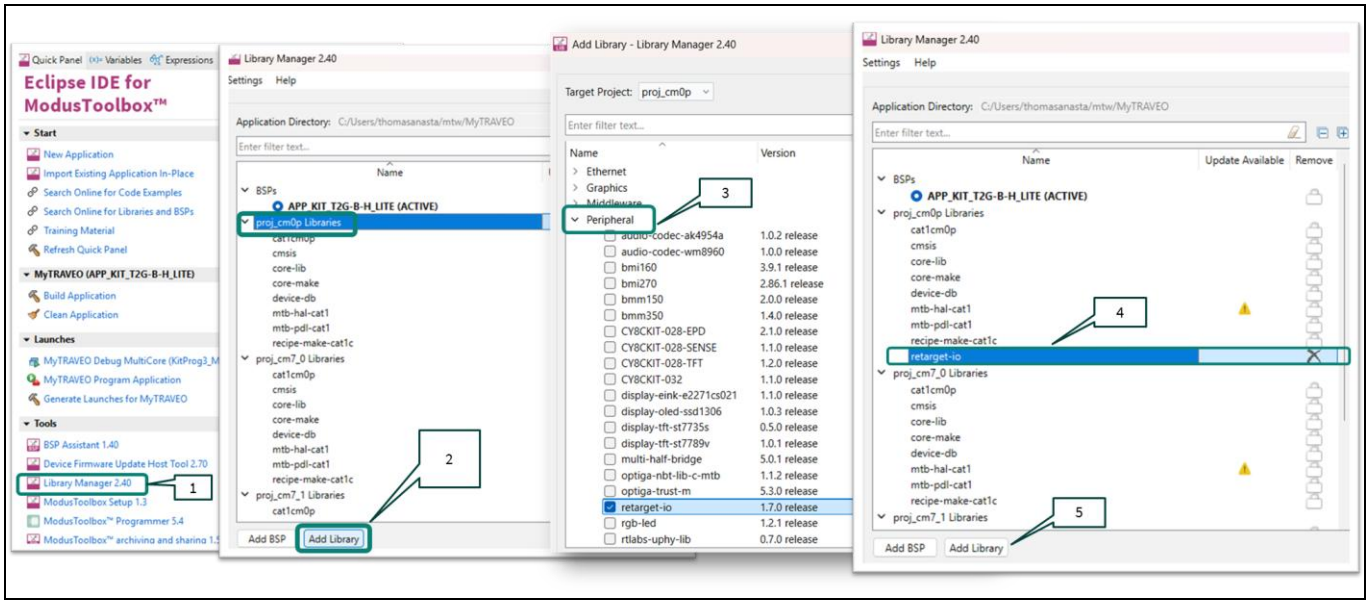


Figure 19 Add the retarget-io library

2.7 Coding

- Alter the main code to change the output to the terminal:
  - Expand **MyTRAVEO** and **MyTRAVEO.proj_cm0p**, then double-click on ***main.c*** to open the edit window (see Figure 20)
- Include the header file for retarget-io:
  - The header file provides APIs for transmitting messages to/from the terminal via standard “*printf()*” / “*scanf()*” functions

TRAVEO™ T2G Evaluation Kit (KIT_T2G-B-H_LITE) setup

- In the code listings in this document, black characters are already in the *main.c* file
- Copy and paste the highlighted text from the code listing into the appropriate place in the *main.c* file
- The following exercises in this document follow this same pattern:

```
#include "cy_pdl.h"
#include "cyhal.h"
#include "cybsp.h"
#include "cy_retarget_io.h"
```

The role of the added code in Code Listing 3:

1. Add a function to initialize retarget-io to use the UART port

- Initialization functions for redirecting low level IO commands to allow sending messages over a UART interface
- cy_retarget_io_init(tx, rx, baudrate)
  - Tx: UART TX pin
  - Rx: UART RX pin
  - Baudrate: UART baudrate
- You can use a different baud rate if you prefer

2. Add code to check the initialization result

- Check whether initilaztion is successful
- This is the recommended manner to create an application on ModusToolbox™

3. Add functions to display on the terminal

- Specify the character to display on the terminal
  1. Comment out the code to enter DeepSleep power mode
  2. Hints:
- **CYBSP_DEBUG_UART_TX** and **CYBSP_DEBUG_RX** pins are already defined by following header file in the BSP:
  - C:/your work space/MyTRAVEO/bsps/TARGET_APP_KIT_T2G-B-H_LITE/config/GeneratedSource/cycfg_pins.h
- **CY_RETARGET_IO_BAUDRATE** is already set to 115200 by following header file:
  - C:/your work space/mtb_shared/retarget-io/release-v1.4.0/ cy_retarget_io.h

Note: *For the final code, see Appendix B – Final Code.*

TRAVEO™ T2G Evaluation Kit (KIT_T2G-B-H_LITE) setup

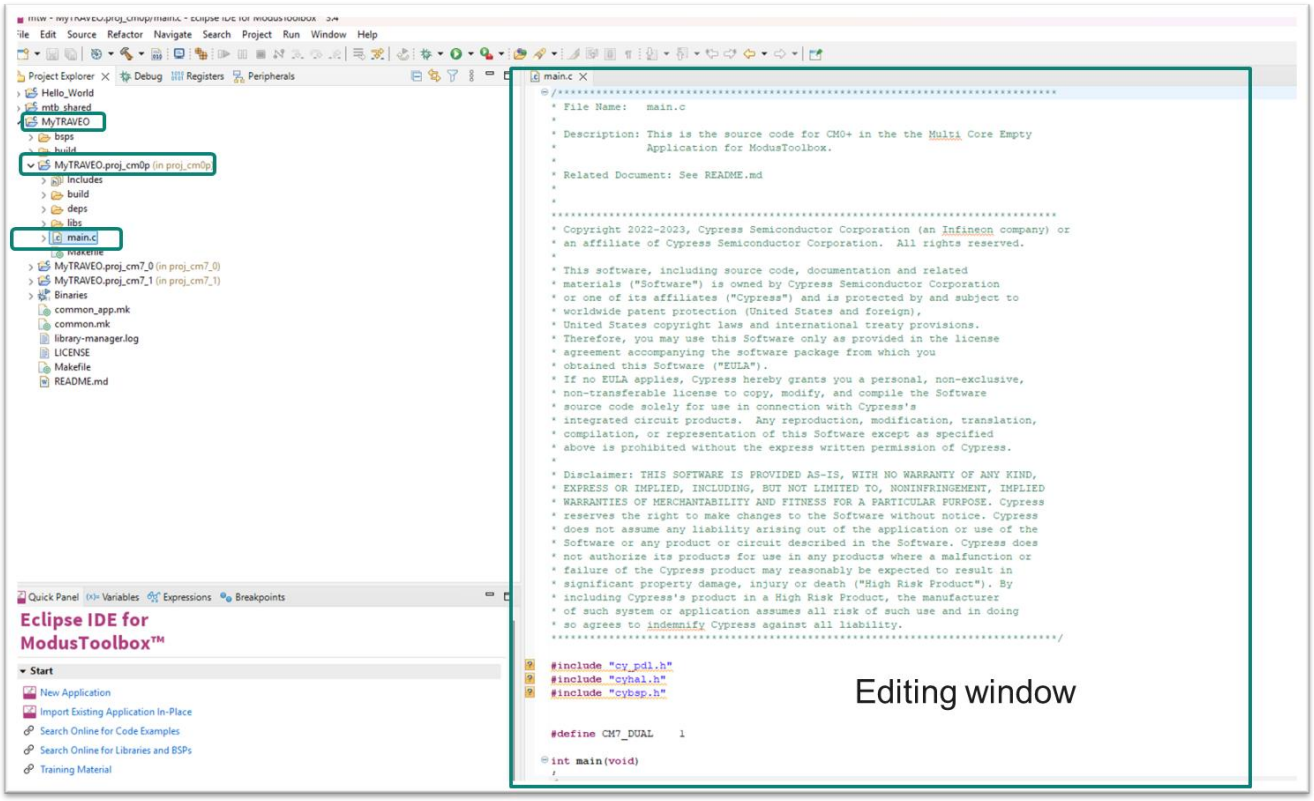


Figure 20 Update Code in main file

Code Listing 1

```
__enable_irq();

result = cy_retarget_io_init(CYBSP_DEBUG_UART_TX, CYBSP_DEBUG_UART_RX,
                             CY_RETARGET_IO_BAUDRATE);

if (result != CY_RSLT_SUCCESS)
{
    CY_ASSERT(0);
}

/* Enable CM7_0/1. CY_CORTEX_M7_APPL_ADDR is calculated in linker script,
check it in case of problems. */

Cy_SysEnableCM7(CORE_CM7_0, CY_CORTEX_M7_0_APPL_ADDR);

#if CM7_DUAL
    Cy_SysEnableCM7(CORE_CM7_1, CY_CORTEX_M7_1_APPL_ADDR);
```

TRAVEO™ T2G Evaluation Kit (KIT_T2G-B-H_LITE) setup

Code Listing 1

```
#endif /* CM7_DUAL */

printf("\x1b[2J\x1b[;H");

printf("Hello World!!!\r\n\n");

printf("Welcome to TRAVEO Training!!!\r\n\n");

for(;;)
{
    /* Cy_SysPm_CpuEnterDeepSleep(CY_SYSPM_WAIT_FOR_INTERRUPT); */
}
```

2.8 Build

Clean and build the project from the Quick Panel, as you did for the “Hello World” application.

*Note: After editing, select **File** -> **Save**, or select the **save** button in the toolbar.*

1. In the **Project Explorer**, select **MyTRAVEO**
2. In the **Quick Panel**, under **MyTRAVEO (APP_KIT_T2G-B-H_LITE**, select **Clean Project**
3. Once the operation is complete, select **Build Project** in the same place
4. Verify that the build is successful and there are no errors in the console window

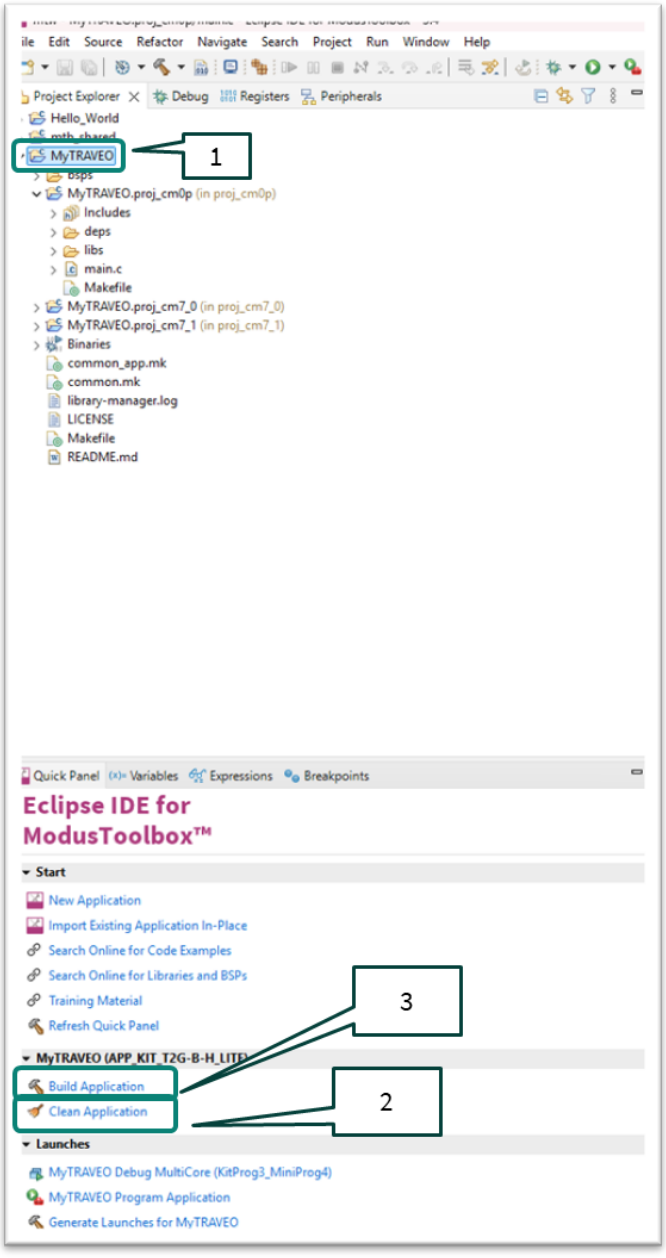


Figure 21 Build the application

2.9 Flash (Program and Test)

1. In the **Project Explorer**, select **MyTRAVEO** application
2. In the Quick Panel, under Launches, select MyTRAVEO Program Application (KitProg3_MiniProg4)
3. Observe for the new message on the Tera Term

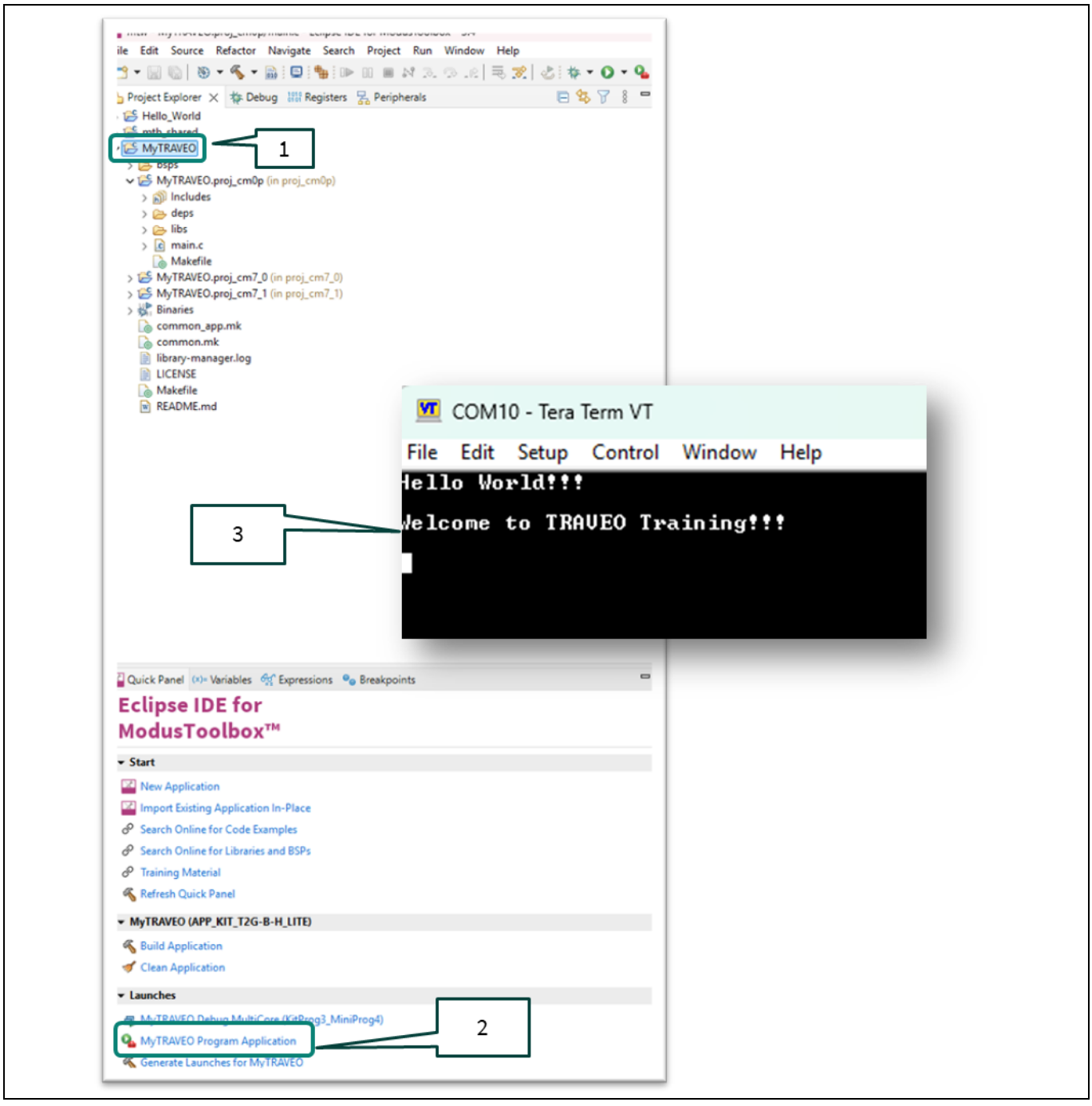


Figure 22 Programming and application and console results

For more information, or to see more coding examples, see [6] in the references section.

3 GPIO Handling – LED Blinking

3.1 Overview

In this section, you will learn how to create an application using Hardware Abstraction Layer (HAL). HAL APIs functions are prefixed with "***cyhal_xxxx***". You will use the HAL function to configure GPIO and blink LED3. To begin, the LED blinks in a one-second cycle; when the USER1 button is pressed, a GPIO interrupt is generated that toggles the LED flashes in a cycle of 250 milliseconds. Each time the USER1 button is pressed, the flashing cycle of the LED changes between 1 second and 250 milliseconds. Here, LED3 is controlled by CM7_0 core.

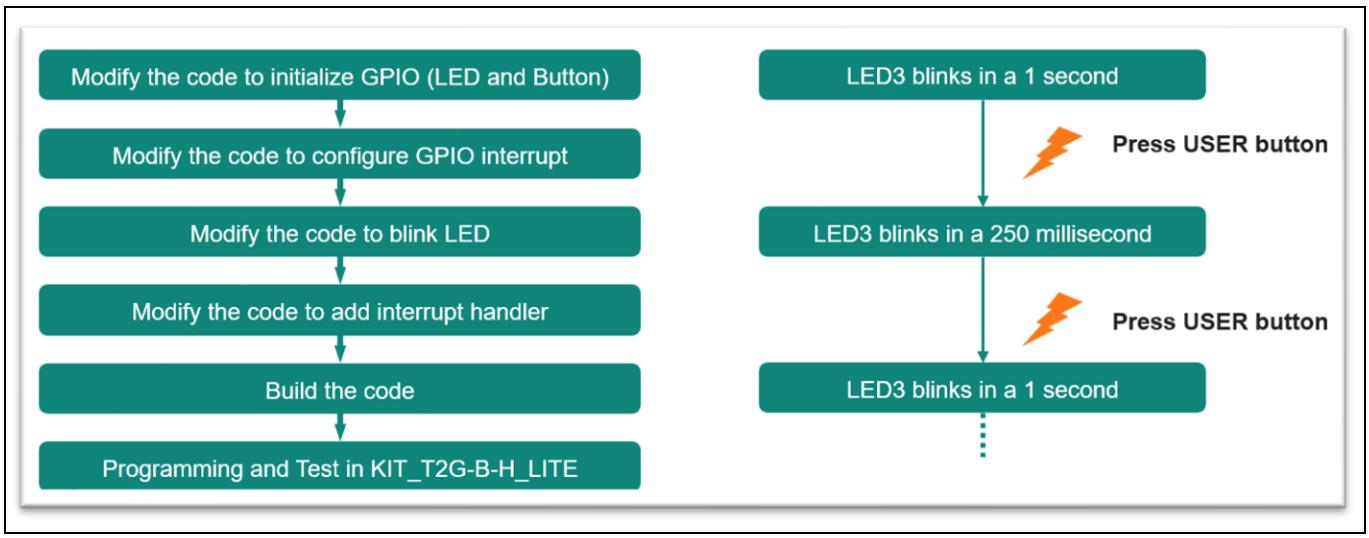


Figure 23 GPIO/LED3 workflow and behavior

3.2 LED Blinking

1. Double-click the "*main.c*" of MyTRAVEO.proj_cm7_0 in the MyTRAVEO project to open the edit window

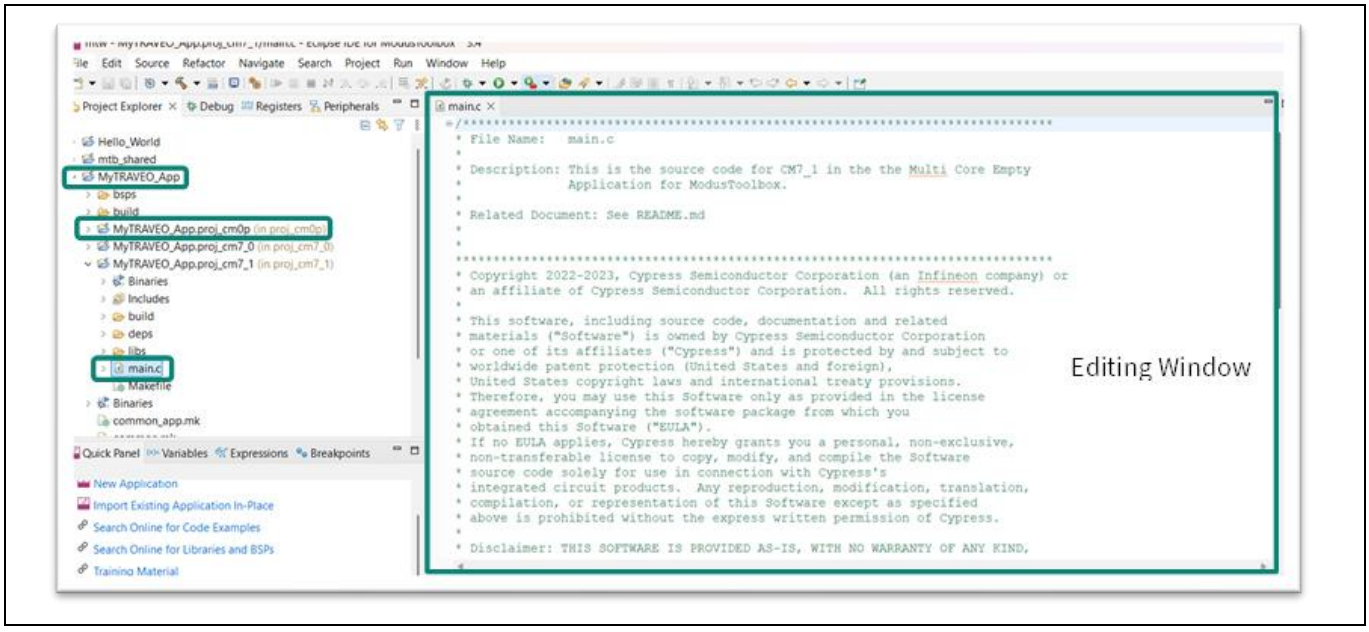


Figure 24 Open main.c in the editor

GPIO Handling – LED Blinking

2. Add the **green text** from Code Listing 2, which performs the following functions:
   - Add definitions
     a. **DELAY_MS**: defines wait time for normal blinking
     b. **DELAY_MS_FAST**: defines wait time for fast blinking
        i. The number specifies the wait time in milliseconds
     c. **GPIO_INTERRUPT_PRIORITY**: defines USER button interrupt priority
   - Add following variables
     d. **gpio_intr_flag**: indicate interrupt detection
     e. **gpio_btn_callback_dat**: store interrupt handler address
   - Add interrupt handler declarations
   - Add local variable and initialization in ***main()*** function
     f. Set local variable and initialize to DELAY_MS
   - Add function to initialize GPIO port to use LED output and button input, and add code to check initialization result
     g. Initialize the GPIO pin
     h. ***cyhal_gpio_init(pin, direction, drive_mode, init_val)***
        i. pin: GPIO pin to initialize
        ii. direction: pin direction
        iii. drive_mode: pin drive mode
        iv. init_val: initial value on the pin
     i. Check is initialization is successful
   - Add function to register interrupt handler
     j. Register/clear an interrupt handler for pin events
     k. ***cyhal_gpio_register_callback(pin, callback_data)***
        i. pin: GPIO pin
        ii. callback_data: the callback data to register. Use NULL to unregister.
   - Add function to enable USER button interrupt
     l. Enable or Disable specified GPIO event
     m. ***cyhal_gpio_enable_event(pin, event, intr_priority, enable)***
        i. Pin: GPIO pin
        ii. Event: GPIO event
        iii. Intr_priority: priority for NVIC interrupt events
        iv. Enable: true (Enable interrupts)/false (Disable interrupts)
   - Add LED control code
     n. Change the wait time, USER button interrupt is detected each time
     o. Check if interrupt is detected
     p. Interrupt flag set to false when interrupt detected
     q. Switch the wait time
   - Add function to write the GPIO port and wait LED blink cycle
     r. Set the output value for the pin
     s. ***cyhal_gpio_write(pin, value)***

i. Pin: GPIO pin
ii. Value: the value to be set: high (true)/low (false)

t. Requests that the current operation delay for at least the specified length of time
u. ***cyhal_system_delay_ms(milliseconds)***
i. Milliseconds: the number of milliseconds to delay for

- Add interrupt handler under *main()* function
v. Set the "gpio_intr_flag" to true, when USER button interrupt occur

- Hints
  - **CYHAL_GPIO_xxx** is already defined in the HAL header file
  - **CYBSP_xxx** is already definted in the BSP header file
  - For the final code, see Appendix B – Final Code

Code Listing 2

```
#include "cy_pdl.h"
#include "cyhal.h"
#include "cybsp.h"

#define DELAY_MS           (500)
#define DELAY_MS_FAST      (125)
#define GPIO_INTERRUPT_PRIORITY (7u)

volatile bool gpio_intr_flag = false;
cyhal_gpio_callback_data_t gpio_btn_callback_data;

static void gpio_interrupt_handler(void *handler_arg, cyhal_gpio_event_t
event);

int main(void)
{
cy_rslt_t result;
uint32_t delay_led_blink = DELAY_MS;

__enable_irq();

result = cyhal_gpio_init(CYBSP_USER_LED, CYHAL_GPIO_DIR_OUTPUT,
                  CYHAL_GPIO_DRIVE_STRONG, CYBSP_LED_STATE_OFF);

if (result != CY_RSLT_SUCCESS)
{
    CY_ASSERT(0);
```

GPIO Handling – LED Blinking

Code Listing 2

```
}

result = cyhal_gpio_init(CYBSP_USER_BTN, CYHAL_GPIO_DIR_INPUT,
                  CYBSP_USER_BTN_DRIVE, CYBSP_BTN_OFF);

if (result != CY_RSLT_SUCCESS)
{
     CY_ASSERT(0);
}

gpio_btn_callback_data.callback = gpio_interrupt_handler;

cyhal_gpio_register_callback(CYBSP_USER_BTN,
                             &gpio_btn_callback_data);

cyhal_gpio_enable_event(CYBSP_USER_BTN, CYHAL_GPIO_IRQ_FALL,
                              GPIO_INTERRUPT_PRIORITY, true);

for (;;)
{
if (true == gpio_intr_flag)
     {
gpio_intr_flag = false;

           /* Update LED toggle delay */
           if (DELAY_MS == delay_led_blink)
           {
delay_led_blink = DELAY_MS_FAST;
           }

           else
           {
             delay_led_blink = DELAY_MS;
           }
     }

  cyhal_gpio_write(CYBSP_USER_LED, CYBSP_LED_STATE_ON);
     cyhal_system_delay_ms(delay_led_blink);
```

GPIO Handling – LED Blinking

Code Listing 2

```
        cyhal_gpio_write(CYBSP_USER_LED, CYBSP_LED_STATE_OFF);
        cyhal_system_delay_ms(delay_led_blink);
    }
}

static void gpio_interrupt_handler(void *handler_arg, cyhal_gpio_event_t
event)
{
gpio_intr_flag = true;
}
```

3. After editing, go to **File** -> **Save**, or click the **save** button to save the changes
4. In the **Project Explorer**, select **MyTRAVEO**, then in the **Quick Panel**, select **Build Application**. Check the build window for successful completion without error
5. Next, select **MyTRAVEO Program Application (KitProg3_MiniProg4)** to flash the program to the board. After flashing, the program will run automatically
6. Observe the blinking LED (LED3). Also, you can verify that the LED blinking speed changes by pressing the USER1 button each time

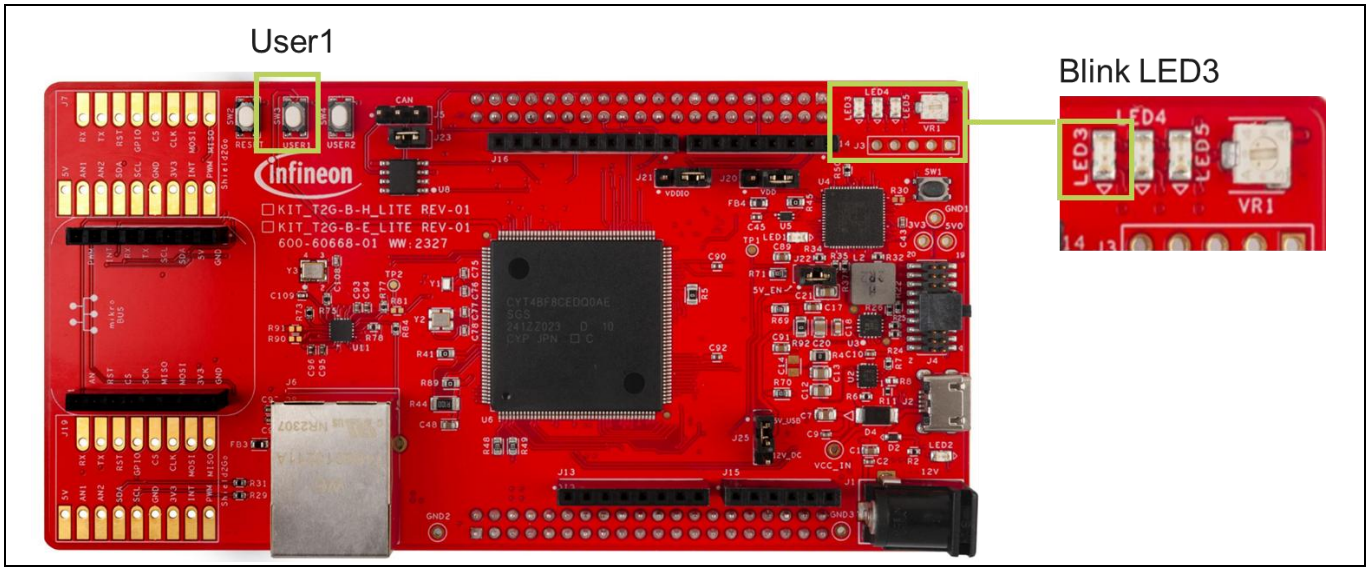


Figure 25 LED3 and USER1

3.3 LED/User button configuration

- HAL structure is used by a HAL function, and defined by the BSP
- In BSP, some functions are defined by the device or board, such as LEDs
- These definitions can be used directly in HAL functions
- If you use a different board, just assign the defined names to the appropriate port so that your application will work without any source code changes

GPIO Handling – LED Blinking

- In this exercise, you will learn how to change these assignments using the **Device Configurator**
- Default setup: The correspondence between the board LEDs, the CYT4BF8xx device pins, and the port pins are predefined and shown in Table 1

Table 1 User LEDs

USER LED	Ref. designator	CYT4BF8xx	
		Pin	Pin
USER LED 1	LED3	29	P5.0
USER LED 2	LED4	30	P5.1
USER LED 3	LED5	31	P5.2

You can change the LED/User Button using the Device Configurator:

1. In the Project Explorer, select MyTRAVEO
2. In the **Quick Panel**, under **BSP Configurators (APP_KIT_T2G-B-H_LITE)** , select **Device Configurator 5.30** (or latest version). You may have to scroll through the Quick Panel to find the Device Configurator

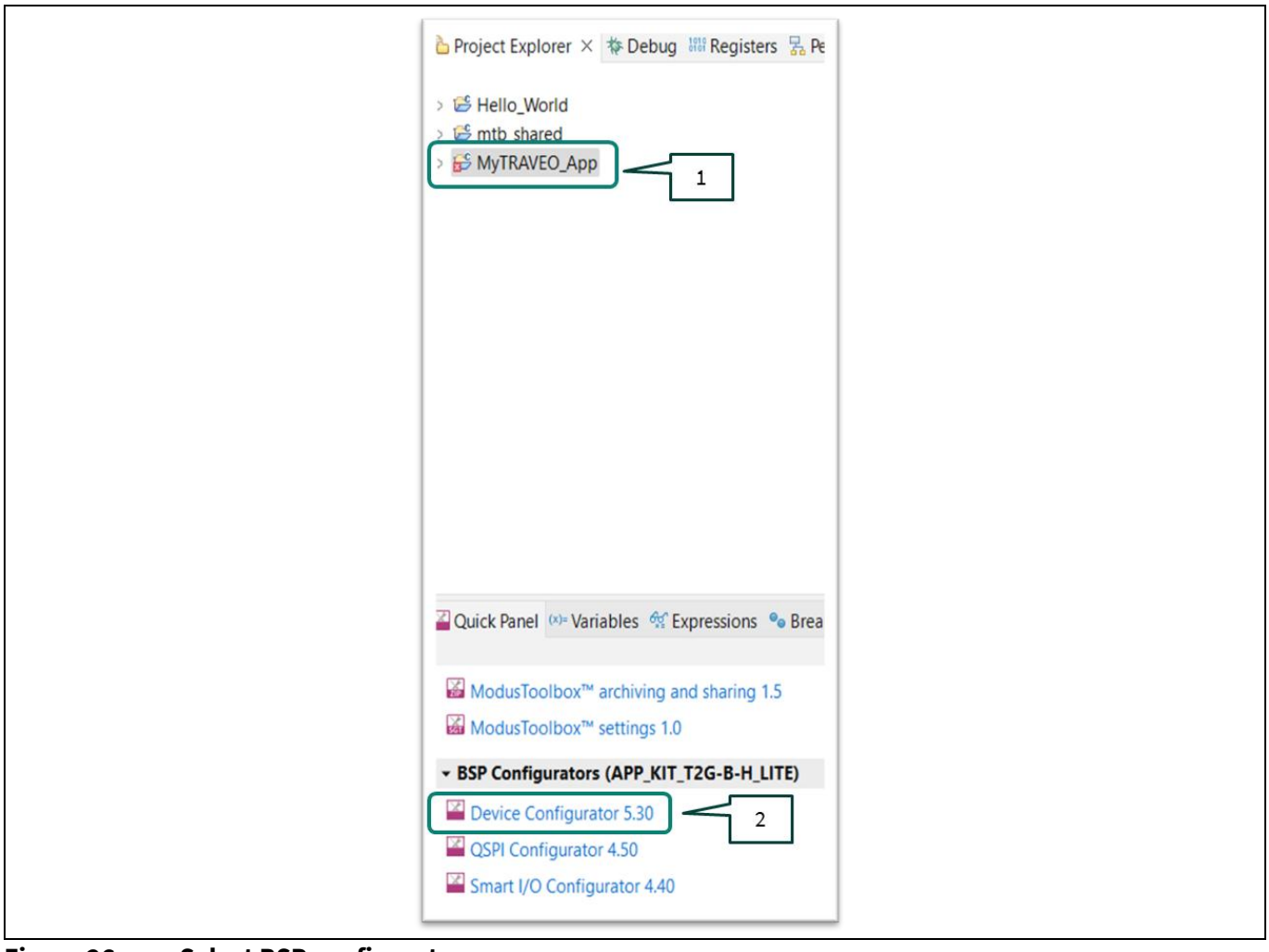


Figure 26 Select BSP configurators

3. Once the Device Configurator opens, navigate to the **Pins** tab

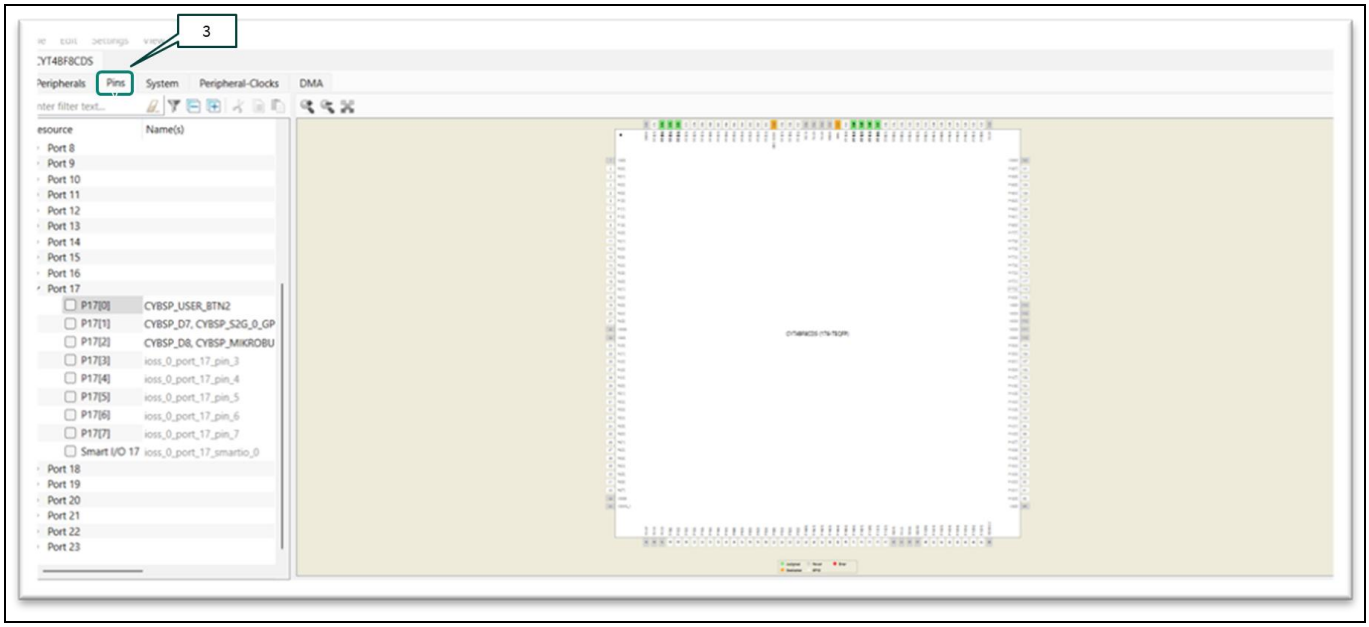


Figure 27 Pins tab

4. You can change blinking LED from LED3 to LED5:
   a) Cut (ctrl + X) “CYBSP_USER_LED” from **P5[0]**
   b) Paste (ctrl + V) it into **P5[2]**
5. You can change the User Button from USER1 to USER2:
   a) Cut “CYBSP_USER_BTN” from **P5[3]**
   b) Paste it into **P17[0]**

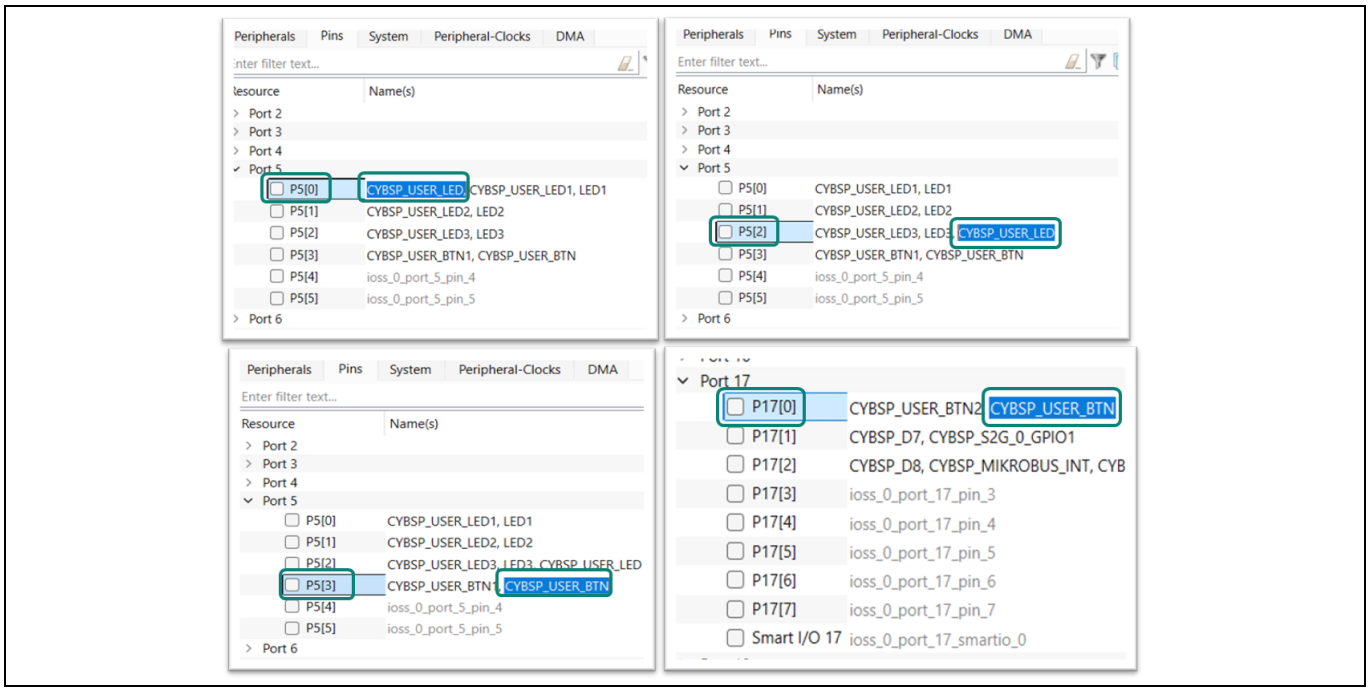


Figure 28 Reassigning pins in Device Configurator

GPIO Handling – LED Blinking

6. Save the configuration (**File** -> **Save**, or click Save button)
7. After saving, the header file (cycfg_pibs.h) will be updated based on the Device Configuration
   a) Before:
   ```
   #define CYBSP_USER_LED(P5_0)
   ```
   b) After:
   ```
   #define CYBSP_USER_LED(P5_2)
   ```
8. In the **Project Explorer**, select **MyTRAVEO**, then in the **Quick Panel**, select **Build Application**. Confirm the build completes with no errors
9. In the **Project Explorer**, select **MyTRAVEO**, then in the **Quick Panel**, select **MyTRAVEO Program Application (KitProg3_MiniProg4)** to flash the program to the board
10. Verify that LED5 now blinks instead of LED3, and that the LED blinking speed toggles when you press USER2 button instead of the USER1 button

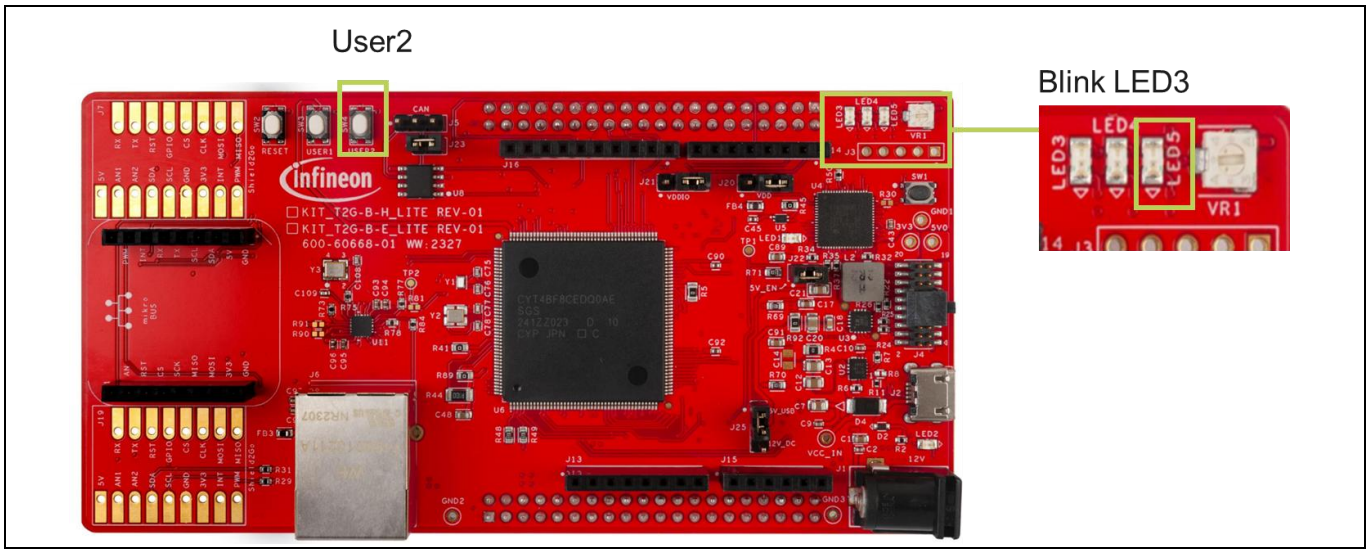


Figure 29 Toggling LED blinking

4 Related resources

- 32-bit TRAVEO™ T2G Arm® Cortex® microcontroller
- TRAVEO™ T2G Documentation
- TRAVEO™ T2G Developer community
- KIT_T2G-B-H_LITE TRAVEO™ T2G Body High Lite Kit user guide
- TRAVEO™ T2G Body High Lite Kit Quick start guide
- Getting started with TRAVEO™ T2G family MCUs in ModusToolbox™

5 Appendix

5.1 Appendix-A: Software installation

5.1.1 Infineon Developer Center Launcher installation

Note: Administration rights are required to install these programs.

Step 1: Download and installation

1. Visit Infineon Developer Center Launcher, select your **operating system** and click **download** [7]

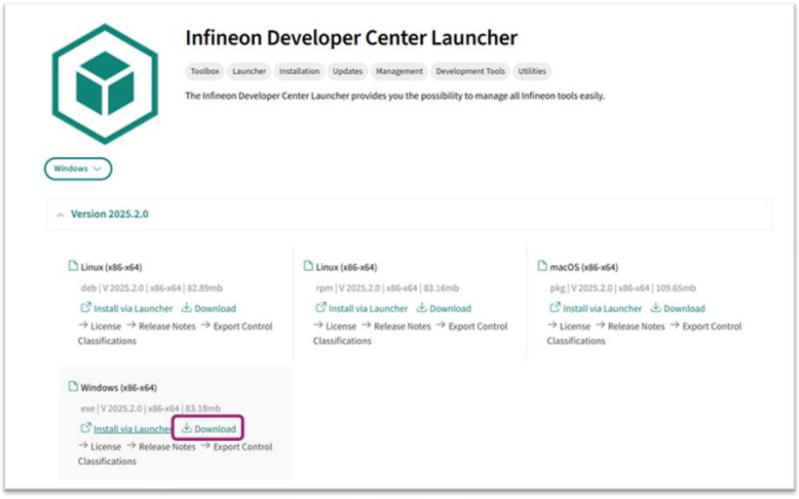


Figure 30 Download Infineon Developer Center Launcher

2. When the download completes, run *launcher2_2025.2.0_Windows_x86-x64.exe* from your download folder

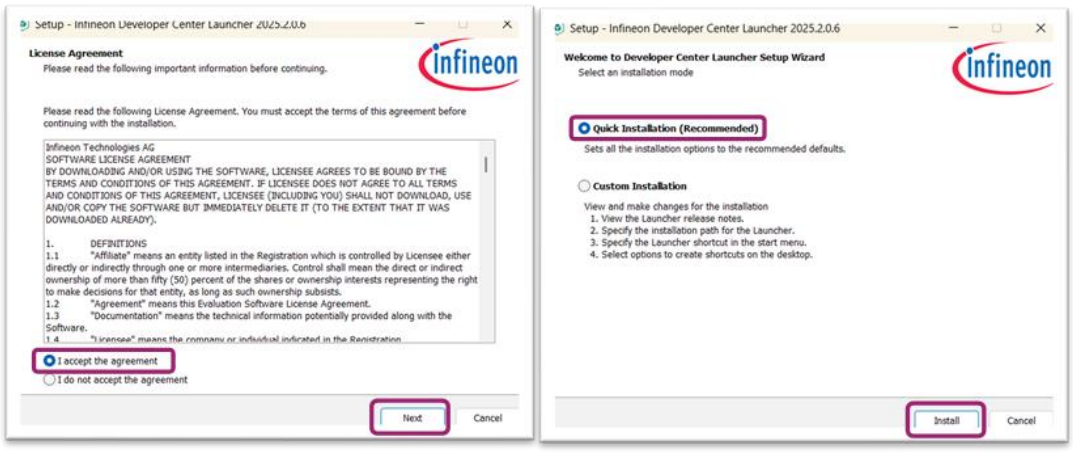


Figure 31 Install the launcher

3. Accept the **license agreement** and click **Next**
4. Choose **Quick Installation**, then click **Install**
5. When the installation is completed, select **Finish**

Appendix

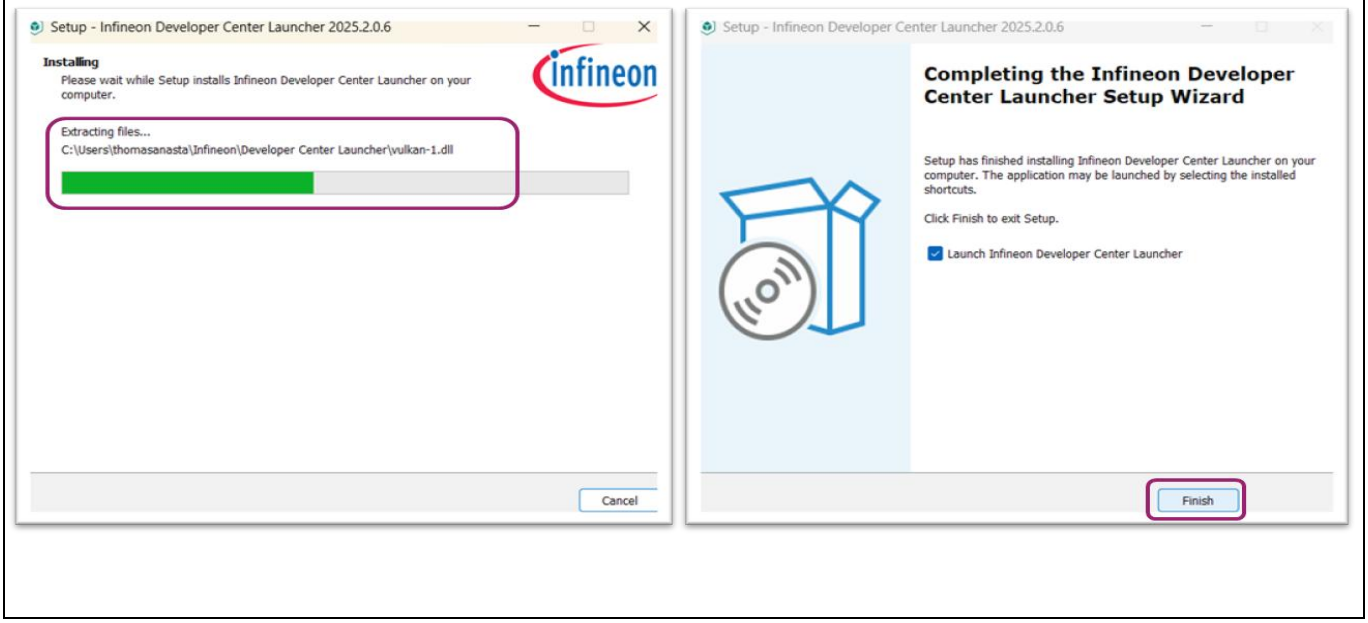


Figure 32 Completion of launcher setup wizard

Step 2: Install and launch ModusToolbox™

1. From the Start menu, open Infineon Developer Center Launcher

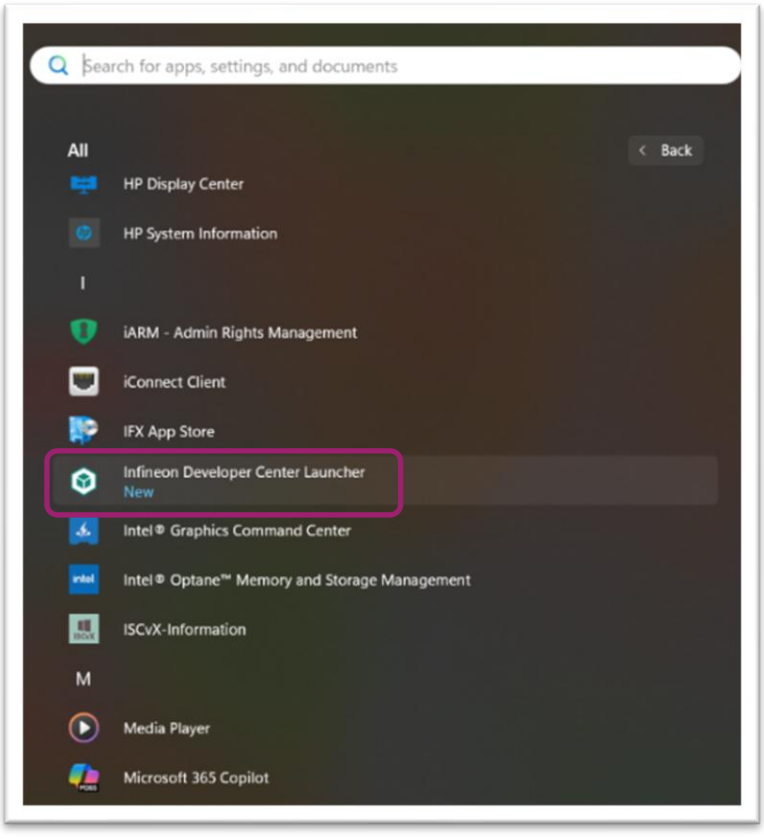


Figure 33 Launch Infineon Developer Center

Appendix

2. In the search bar, type **ModusToolbox**, select Version 1.3.0.1017 (or latest version), then click **Install**

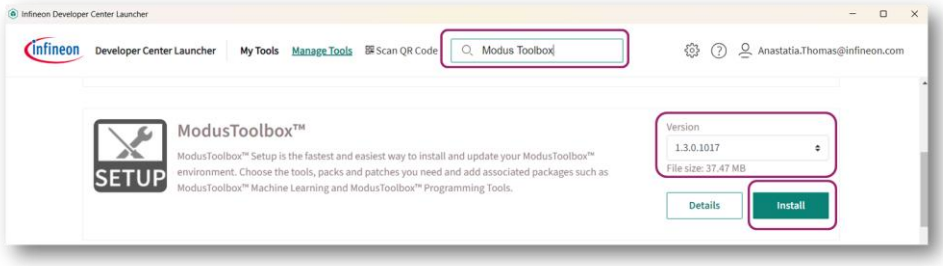


Figure 34 Select latest version of ModusToolbox™ and proceed to install

3. Accept the **licese aggrement** and click **Next**
4. Choose Quick Installation, then click **Install**

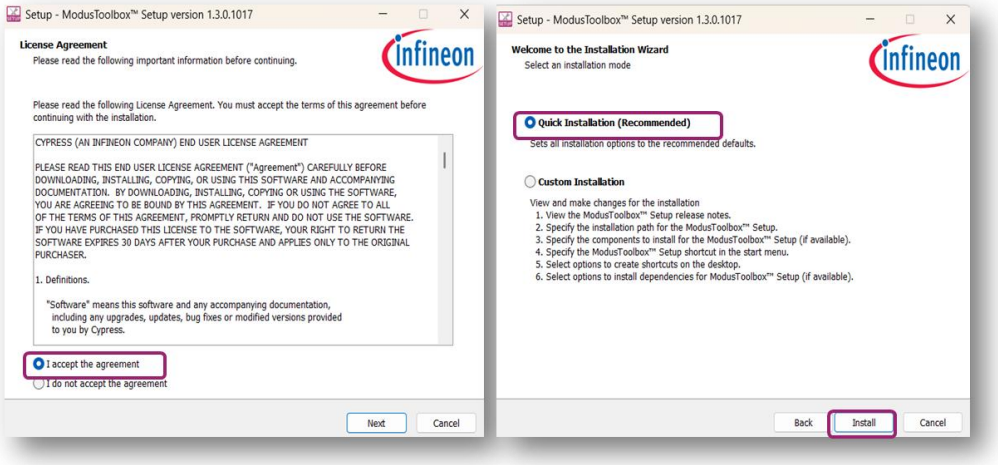


Figure 35 Accept the licese agreement

5. When installation is complete, select **Run ModusToolbox™ Setup**", then **Finish**

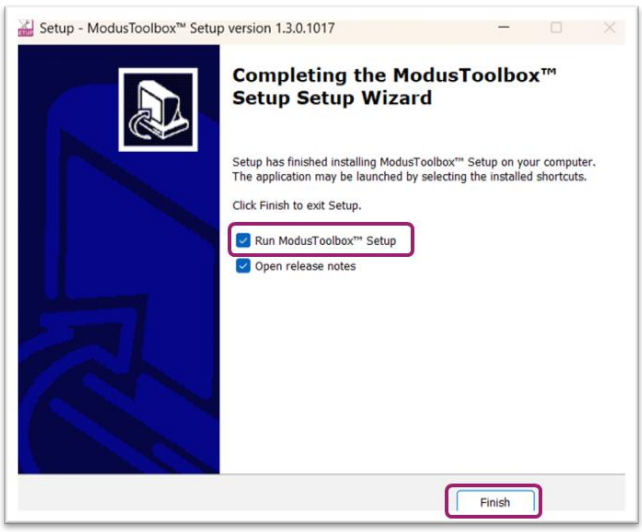


Figure 36 Complete the setup wizard

Appendix

Step 3: Install packages

1. When the ModusToolbox™ Setup opens, select the following packages to install:

- Arm® GNU Toolchain (GCC) – Version 11.3.1 (or latest version)
- ModusToolbox™ Tools Package – Version 3.5.0 (or latest version)
- Eclipse IDE for ModusToolbox™ – Version 3.4.0 (or latest version)
- ModusToolbox™ Edge Protect Security Suite – Version 1.4.0 (or latest version)
- ModusToolbox™ Programming Tools – Version 1.4.0 (or latest version)

2. Click **Next**, accept the **license aggrement**, then **Start**

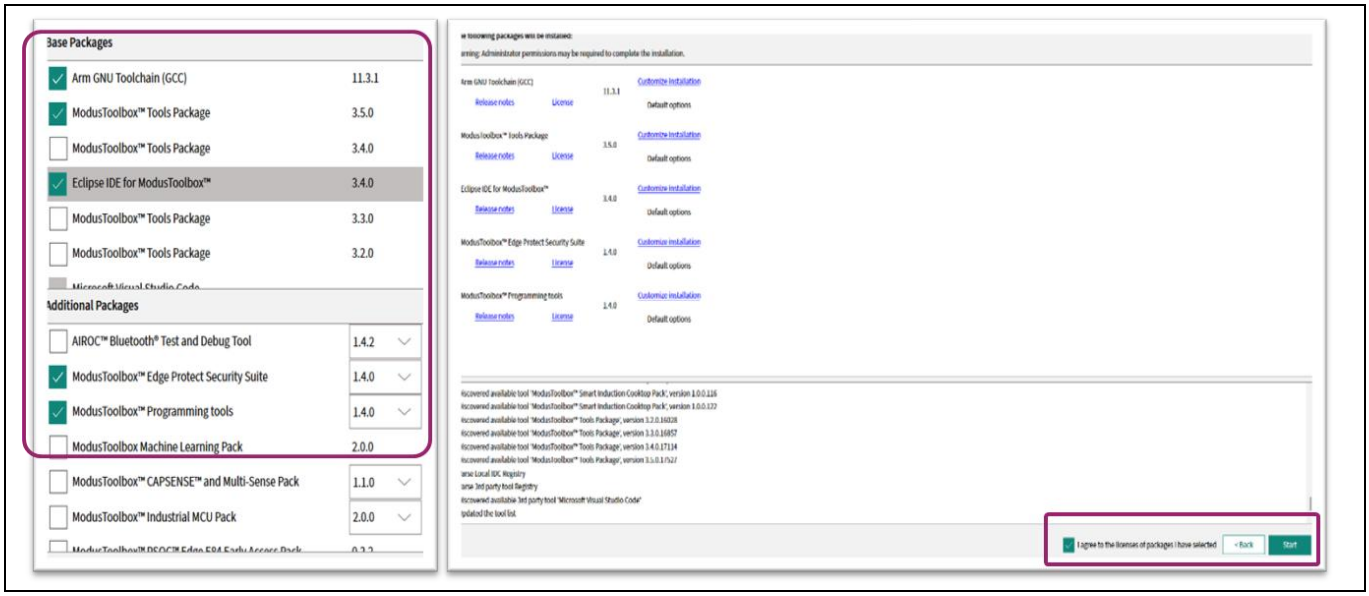


Figure 37 Install BSPs for ModusToolbox™

3. When installation completes, click **Run** next to **Eclipse IDE for ModusToolbox™ 1.4.0**

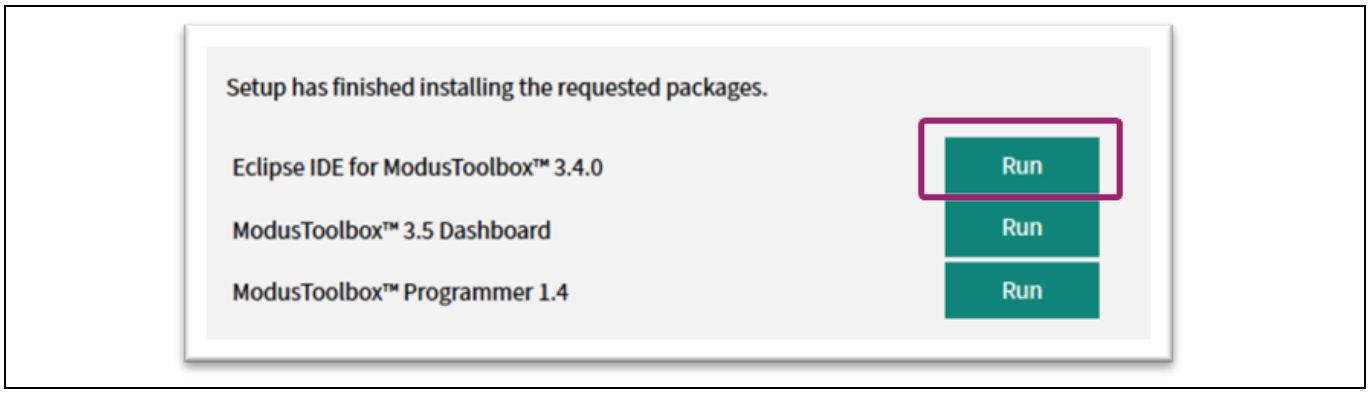


Figure 38 Run ModusToolbox™ 3.4.0

For more information about the Infineon Developer Center refer to [7].

Appendix

Step 4: Run Eclipse IDE

1. When prompted to select a workspace", use the default location and select **Launch**

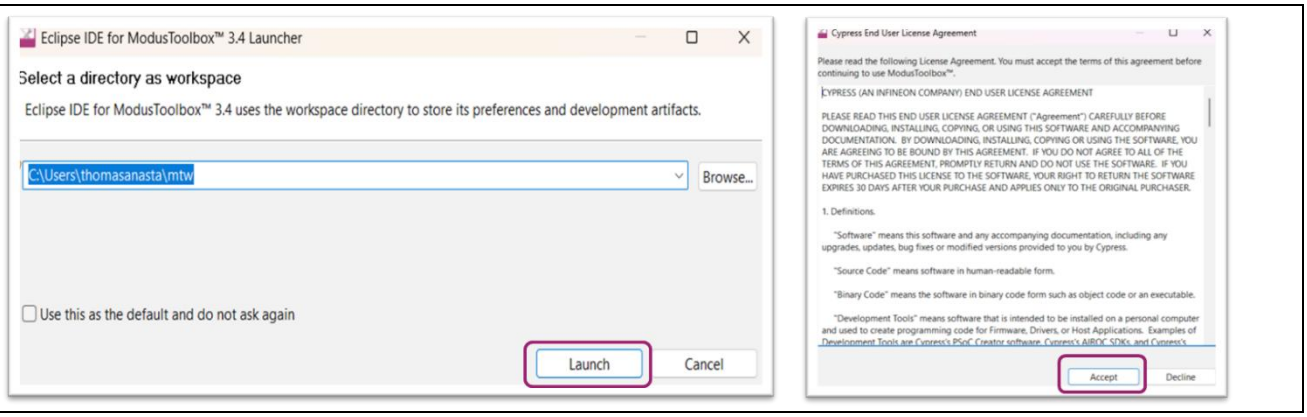


Figure 39 Launch Eclipse IDE

2. **Accept** the Cypress End User License Agreement
3. The Eclipse IDE opens

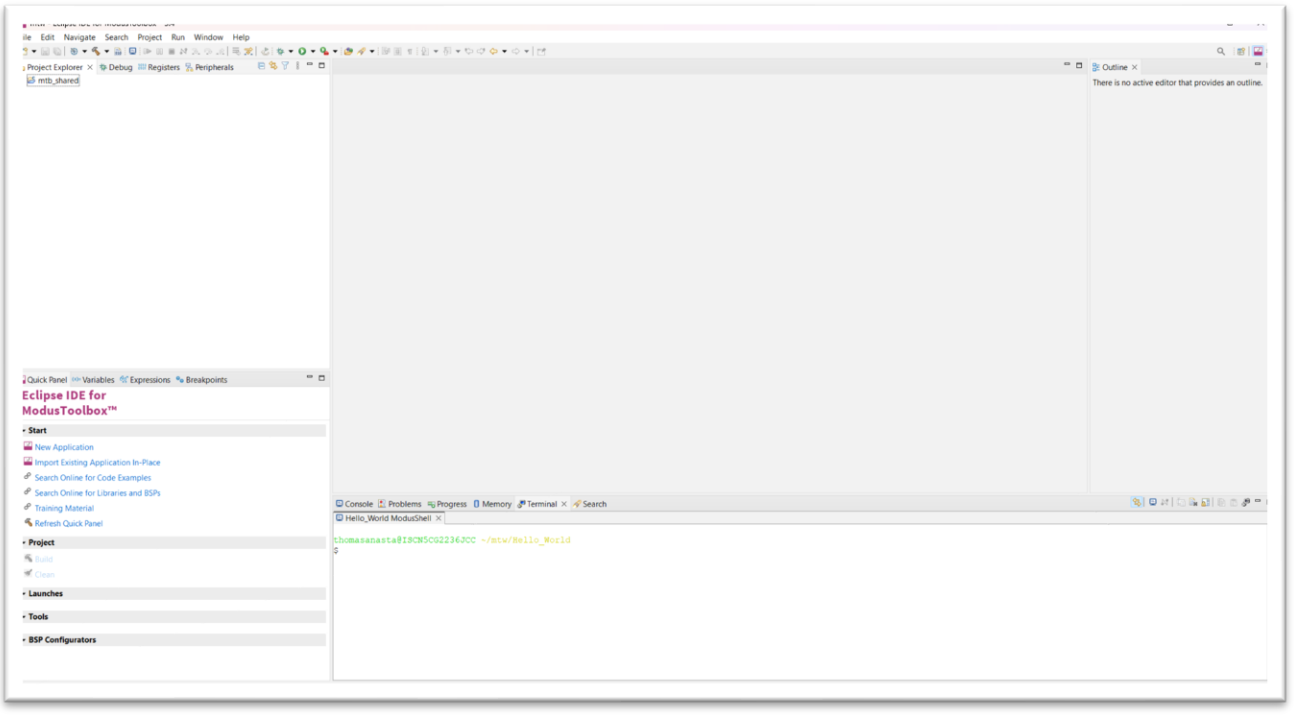


Figure 40 Eclipse IDE

Note: Do not close Eclipse IDE.

5.2 Appendix B – Final Code

5.2.1 MyTRAVEO.proj_cm0p Final Code

Code Listing 3

```
/*******************************************************************************
****
* File Name:   main.c
*
* Description: This is the source code for CM0+ in the the Multi Core Empty
*              Application for ModusToolbox.
*
* Related Document: See README.md
*
*
*******************************************************************************
****
* Copyright 2022-2023, Cypress Semiconductor Corporation (an Infineon
company) or
* an affiliate of Cypress Semiconductor Corporation.  All rights reserved.
*
* This software, including source code, documentation and related
* materials ("Software") is owned by Cypress Semiconductor Corporation
* or one of its affiliates ("Cypress") and is protected by and subject to
* worldwide patent protection (United States and foreign),
* United States copyright laws and international treaty provisions.
* Therefore, you may use this Software only as provided in the license
* agreement accompanying the software package from which you
* obtained this Software ("EULA").
* If no EULA applies, Cypress hereby grants you a personal, non-exclusive,
* non-transferable license to copy, modify, and compile the Software
* source code solely for use in connection with Cypress's
* integrated circuit products.  Any reproduction, modification,
translation,
* compilation, or representation of this Software except as specified
* above is prohibited without the express written permission of Cypress.
*
* Disclaimer: THIS SOFTWARE IS PROVIDED AS-IS, WITH NO WARRANTY OF ANY
KIND,
* EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, NONINFRINGEMENT,
IMPLIED
```

Appendix

Code Listing 3

```
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
Cypress
* reserves the right to make changes to the Software without notice.
Cypress
* does not assume any liability arising out of the application or use of
the
* Software or any product or circuit described in the Software. Cypress
does
* not authorize its products for use in any products where a malfunction or
* failure of the Cypress product may reasonably be expected to result in
* significant property damage, injury or death ("High Risk Product"). By
* including Cypress's product in a High Risk Product, the manufacturer
* of such system or application assumes all risk of such use and in doing
* so agrees to indemnify Cypress against all liability.
*******************************************************************************
****/

#include "cy_pdl.h"
#include "cyhal.h"
#include "cybsp.h"
#include "cy_retarget_io.h"

#define CM7_DUAL    1

int main(void)
{
    cy_rslt_t result;

    /* Initialize the device and board peripherals */
    result = cybsp_init() ;
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

    /* enable interrupts */
    __enable_irq();

    result = cy_retarget_io_init(CYBSP_DEBUG_UART_TX, CYBSP_DEBUG_UART_RX,
CY_RETARGET_IO_BAUDRATE);
```

Appendix

Code Listing 3

```
    if(result != CY_RSLT_SUCCESS){
       CY_ASSERT(0);
    }

    /* Enable CM7_0/1. CY_CORTEX_M7_APPL_ADDR is calculated in linker
script, check it in case of problems. */
    Cy_SysEnableCM7(CORE_CM7_0, CY_CORTEX_M7_0_APPL_ADDR);

#if CM7_DUAL
    Cy_SysEnableCM7(CORE_CM7_1, CY_CORTEX_M7_1_APPL_ADDR);
#endif /* CM7_DUAL */

    // \x1b[2J\x1b[;H - ANSI ESC sequence for clear screen
    printf("\x1b[2J\x1b[;H");
    printf("Hello World!!!\r\n\n");
    printf("Welcome to TRAVEO Training!!!\r\n\n");

    for(;;)
    {
       //Cy_SysPm_CpuEnterDeepSleep(CY_SYSPM_WAIT_FOR_INTERRUPT);
    }
}

/* [] END OF FILE */
```

5.2.2 MyTRAVEO.proj_cm7_0 Final Code

Code Listing 4

```
/*******************************************************************************
****
* File Name:   main.c
*
* Description: This is the source code for CM7_0 in the the Multi Core
Empty
*              Application for ModusToolbox.
*
* Related Document: See README.md
```

Appendix

Code Listing 4

```
*
*
********************************************************************************
* Copyright 2022-2023, Cypress Semiconductor Corporation (an Infineon company) or
* an affiliate of Cypress Semiconductor Corporation.  All rights reserved.
*
* This software, including source code, documentation and related
* materials ("Software") is owned by Cypress Semiconductor Corporation
* or one of its affiliates ("Cypress") and is protected by and subject to
* worldwide patent protection (United States and foreign),
* United States copyright laws and international treaty provisions.
* Therefore, you may use this Software only as provided in the license
* agreement accompanying the software package from which you
* obtained this Software ("EULA").
* If no EULA applies, Cypress hereby grants you a personal, non-exclusive,
* non-transferable license to copy, modify, and compile the Software
* source code solely for use in connection with Cypress's
* integrated circuit products.  Any reproduction, modification, translation,
* compilation, or representation of this Software except as specified
* above is prohibited without the express written permission of Cypress.
*
* Disclaimer: THIS SOFTWARE IS PROVIDED AS-IS, WITH NO WARRANTY OF ANY KIND,
* EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, NONINFRINGEMENT, IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress
* reserves the right to make changes to the Software without notice. Cypress
* does not assume any liability arising out of the application or use of the
* Software or any product or circuit described in the Software. Cypress does
* not authorize its products for use in any products where a malfunction or
* failure of the Cypress product may reasonably be expected to result in
* significant property damage, injury or death ("High Risk Product"). By
* including Cypress's product in a High Risk Product, the manufacturer
* of such system or application assumes all risk of such use and in doing
```

Appendix

Code Listing 4

```
* so agrees to indemnify Cypress against all liability.
*******************************************************************************
****/

#include "cy_pdl.h"
#include "cyhal.h"
#include "cybsp.h"

#define DELAY_MS    (500)
#define DELAY_MS_FAST     (125)
#define GPIO_INTERRUPT_PRIORITY  (7u)

volatile bool gpio_intr_flag = false;
cyhal_gpio_callback_data_t gpio_btn_callback_data;

static void gpio_interrupt_handler(void *handler_arg, cyhal_gpio_event_t
event);

int main(void)
{
    cy_rslt_t result;

    uint32_t delay_led_blink = DELAY_MS;

    /* Initialize the device and board peripherals */
    result = cybsp_init() ;
    if (result != CY_RSLT_SUCCESS)
    {
        CY_ASSERT(0);
    }

    /* Enable global interrupts */
    __enable_irq();

    // initialize the user LED
    result = cyhal_gpio_init(CYBSP_USER_LED, CYHAL_GPIO_DIR_OUTPUT,
CYHAL_GPIO_DRIVE_STRONG, CYBSP_LED_STATE_OFF);

    // init failed. Stop program execution
```

Appendix

Code Listing 4

```
    if(result != CY_RSLT_SUCCESS){
       CY_ASSERT(0);
    }

    // initialize the user button
    result = cyhal_gpio_init(CYBSP_USER_BTN, CYHAL_GPIO_DIR_INPUT,
CYBSP_USER_BTN_DRIVE, CYBSP_BTN_OFF);

    // init failed. Stop program execution
    if(result != CY_RSLT_SUCCESS){
       CY_ASSERT(0);
    }

    // configure GPIO interrupt
    gpio_btn_callback_data.callback = gpio_interrupt_handler;
    cyhal_gpio_register_callback(CYBSP_USER_BTN, &gpio_btn_callback_data);

    cyhal_gpio_enable_event(CYBSP_USER_BTN, CYHAL_GPIO_IRQ_FALL,
GPIO_INTERRUPT_PRIORITY, true);

    for (;;)
    {
       if(true == gpio_intr_flag){
              gpio_intr_flag = false;

              /* Update LED toggle delay */
              if(DELAY_MS == delay_led_blink){
                     delay_led_blink = DELAY_MS_FAST;
              }else{
                     delay_led_blink = DELAY_MS;
              }
       }

       cyhal_gpio_write(CYBSP_USER_LED, CYBSP_LED_STATE_ON);
       cyhal_system_delay_ms(delay_led_blink);
       cyhal_gpio_write(CYBSP_USER_LED, CYBSP_LED_STATE_OFF);
       cyhal_system_delay_ms(delay_led_blink);
    }
}
```

Appendix

Code Listing 4

```
static void gpio_interrupt_handler(void *handler_arg, cyhal_gpio_event_t
event){
        gpio_intr_flag = true;
}

/* [] END OF FILE */
```

5.3 Appendix C – Debugging

- **Start a debug session**
  - In the **Project Explorer window**, select **MyTRAVEO**
  - Click **MyTRAVEO Debug Multicore (KitProg3_MiniProg4)** in the **Quick Panel**. The debug window opens
- **Debug controls**
  - You can execute Resume, Suspend, Step Into, Step Over, Step Return, Terminate, and Restart
  - You can set hardware break point
  - Select Disassembly tab to debug with disassembled code

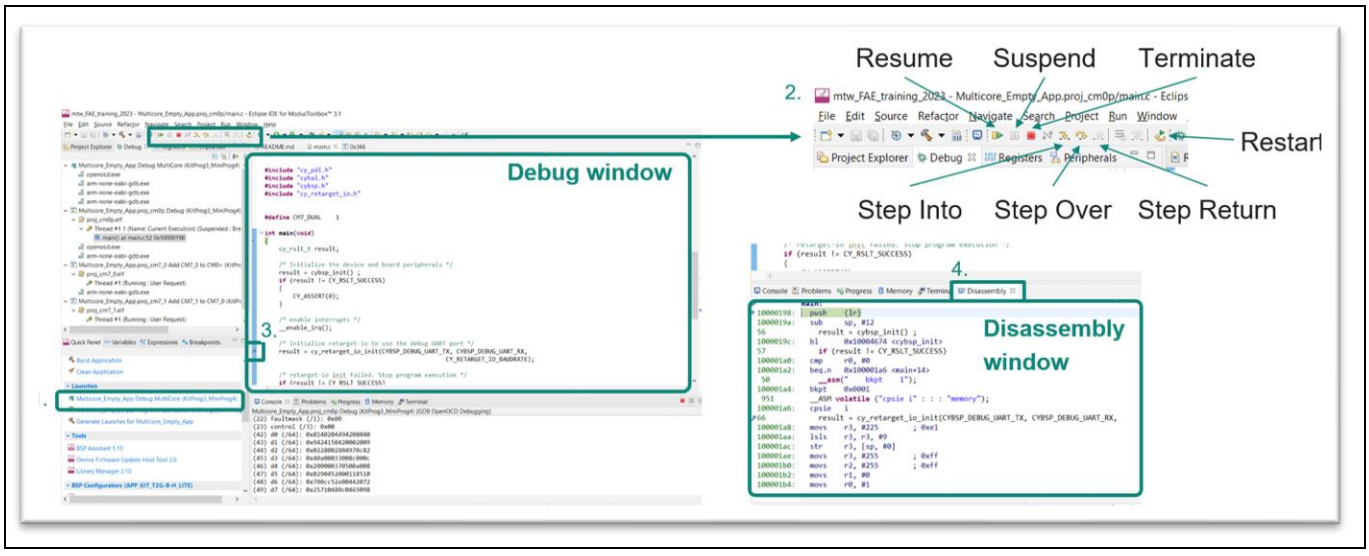


Figure 41 Eclipse IDE debug window

- **Register and peripherals view**
  - Select **Registers** tab to see the CPU specific register
  - Select **Peripherals** tab, select which peripherals you want to inspect, then open the peripheral register window

Appendix

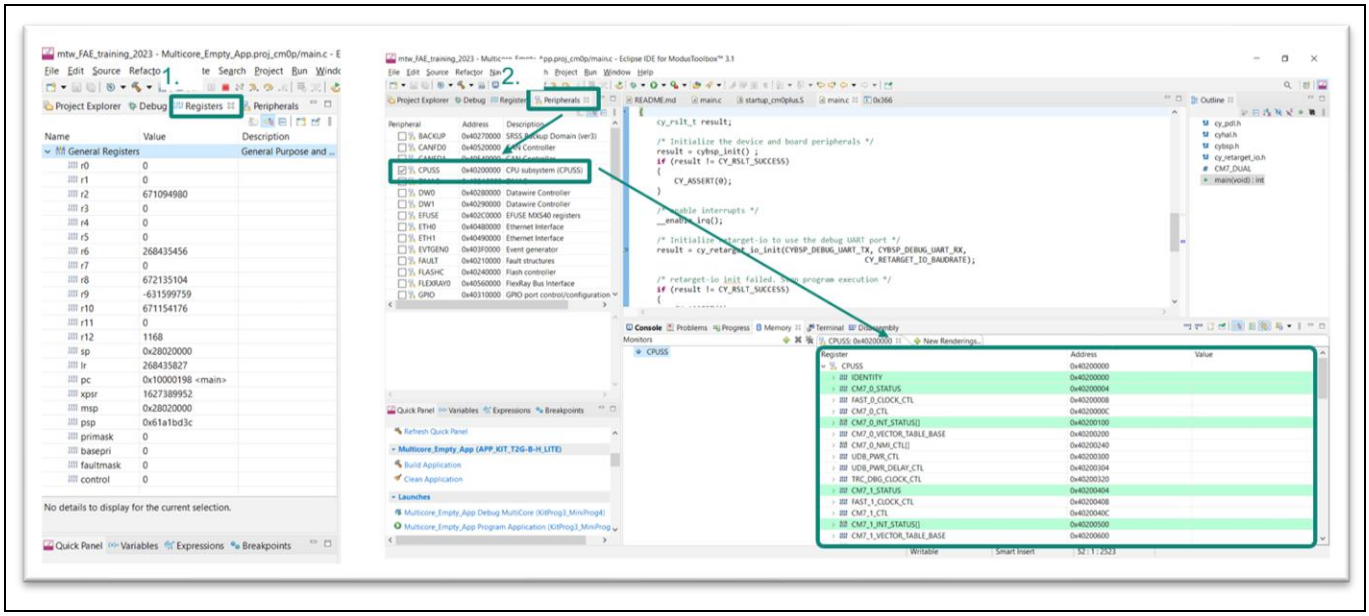


Figure 42 Debug view (Registers and peripherals windows)

5.4 Appendix D– Documentation

- Refer to the function details
  - You can see the function details from ModusToolbox™
  - If you want to check the "***cy_retarget_io_init()***":

  a) Click the "**Retarget IO (retarget-io)**" in Quick Panel

  b) Input "**cy_retarget_io_init**" in the search box, and click on the function you want to search
- You can check other function details, such as PDL/HAL details, in the same manner

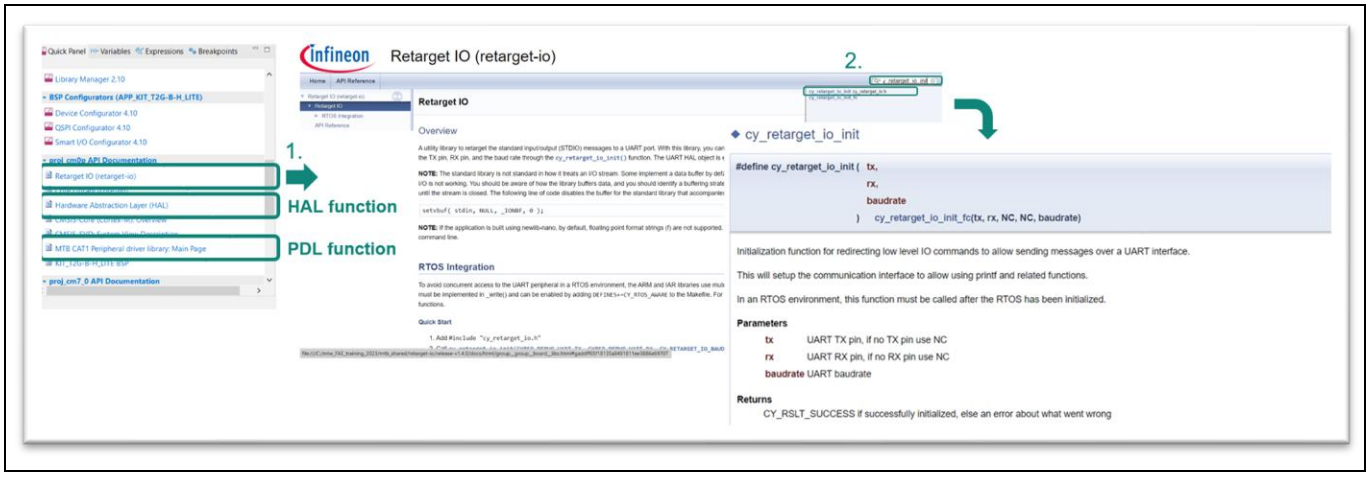


Figure 43 View API docs in ModusToolbox™

6 References

[1] Infineon Technologies AG: *ModusToolbox™ Software webpage* ; Available online

[2] Infineon Technologies AG: *ModusToolbox™ Tools Package Quick Start Guide application note*; Available online

[3] Infineon Technologies AG: *32-bit TRAVEO™ T2G Arm® Cortex® microcontroller webpage*; Available online

[4] Infineon Technologies AG: *KIT_T2G-B-H_LITE webpage*; Available online

[5] Infineon Technologies AG: KIT_T2G-B-H_LITE TRAVEO™ T2G Body High Lite Kit user guide; Available online

[6] Infineon Technologies AG: *TRAVEO™ T2G ModusToolbox™ code examples*; Available online

[7] Infineon Technologies AG: *Infineon Developer Center Launcher*; Available online

Revision history

Document revision	Date	Description of changes
V 1.0	2025-09-26	Initial release

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

Edition 2025-09-26

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2026 Infineon Technologies AG. All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

AN0046

Important Notice

Products which may also include samples and may be comprised of hardware or software or both ("Product(s)") are sold or provided and delivered by Infineon Technologies AG and its affiliates ("Infineon") subject to the terms and conditions of the frame supply contract or other written agreement(s) executed by a customer and Infineon or, in the absence of the foregoing, the applicable Sales Conditions of Infineon. General terms and conditions of a customer or deviations from applicable Sales Conditions of Infineon shall only be binding for Infineon if and to the extent Infineon has given its express written consent.

For the avoidance of doubt, Infineon disclaims all warranties of non-infringement of third-party rights and implied warranties such as warranties of fitness for a specific use/purpose or merchantability.

Infineon shall not be responsible for any information with respect to samples, the application or customer's specific use of any Product or for any examples or typical values given in this document.

The data contained in this document is exclusively intended for technically qualified and skilled customer representatives. It is the responsibility of the customer to evaluate the suitability of the Product for the intended application and the customer's specific use and to verify all relevant technical data contained in this document in the intended application and the customer's specific use. The customer is responsible for properly designing, programming, and testing the functionality and safety of the intended application, as well as complying with any legal requirements related to its use.

Unless otherwise explicitly approved by Infineon, Products may not be used in any application where a failure of the Products or any consequences of the use thereof can reasonably be expected to result in personal injury. However, the foregoing shall not prevent the customer from using any Product in such fields of use that Infineon has explicitly designed and sold it for, provided that the overall responsibility for the application lies with the customer.

Infineon expressly reserves the right to use its content for commercial text and data mining (TDM) according to applicable laws, e.g. Section 44b of the German Copyright Act (UrhG).

If the Product includes security features:

Because no computing device can be absolutely secure, and despite security measures implemented in the Product, Infineon does not guarantee that the Product will be free from intrusion, data theft or loss, or other breaches ("Security Breaches"), and Infineon shall have no liability arising out of any Security Breaches.

If this document includes or references software:

The software is owned by Infineon under the intellectual property laws and treaties of the United States, Germany, and other countries worldwide. All rights reserved. Therefore, you may use the software only as provided in the software license agreement accompanying the software.

If no software license agreement applies, Infineon hereby grants you a personal, non-exclusive, non-transferable license (without the right to sublicense) under its intellectual property rights in the software (a) for software provided in source code form, to modify and reproduce the software solely for use with Infineon hardware products, only internally within your organization, and (b) to distribute the software in binary code form externally to end users, solely for use on Infineon hardware products. Any other use, reproduction, modification, translation, or compilation of the software is prohibited. For further information on the Product, technology, delivery terms and conditions, and prices, please contact your nearest Infineon office or visit https://www.infineon.com