I2C_Slave_Using_Callbacks for KIT_T2G-B-H_LITE

Customer training workshop

Q3 2024

Scope of work

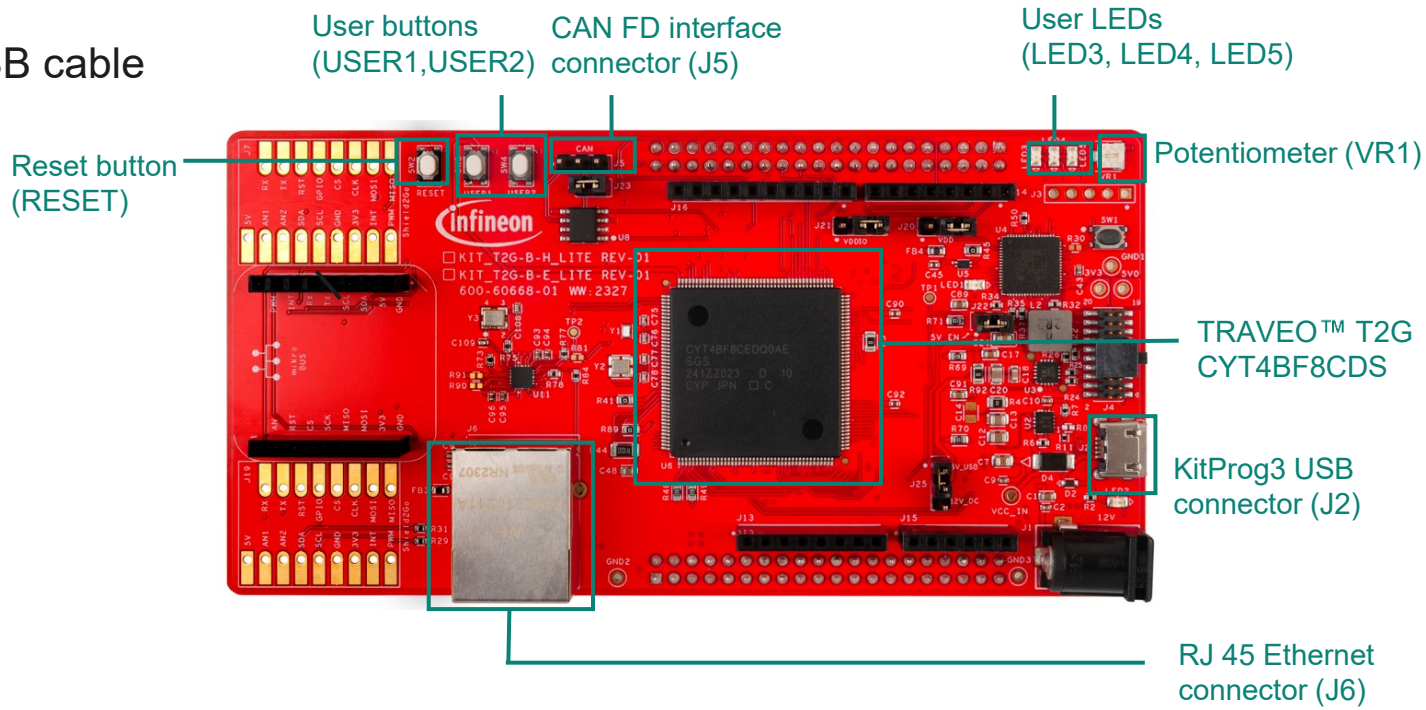
- This code example demonstrates the operation of the I2C (HAL) resource in slave mode using callbacks.
- **Device**
  - The TRAVEO™ T2G CYT4BF8CDS device is used in this code example.
- **Board**
  - The TRAVEO™ T2G KIT_T2G-B-H_LITE board is used for testing.

Introduction

- **I2C has the following features:**
  - Master, slave, and master/slave mode
  - Standard-mode (100 kbps), fast-mode (400 kbps), and fast-mode plus (1000 kbps) data-rates
  - 7-bit slave addressing
  - Clock stretching
  - Collision detection
  - Programmable oversampling of the I2C clock signal (SCL)
  - Auto ACK when RX FIFO is not full, including address
  - General address detection
  - FIFO mode
  - EZ and CMD_RESP modes
  - Interrupts or polling CPU interface
  - Analog glitch filter
  - Local loop-back control

Hardware setup

- This code example has been developed for the KIT_T2G-B-H_LITE board
- Connect the PC to the board using the provided USB cable through the KitProg3 USB connector (J2)



Implementation

- This code example demonstrates the operation of the I2C (HAL) resource in slave mode using callbacks. It initializes PWM to control the User LED1 (LED3) and initializes I2C to slave mode. If the initialization of I2C/PWM fails, the system will be in an infinite loop. After initializing, the slave receives packets from the master and configures the PWM to drive the User LED1 (LED3). The slave responds with the acknowledgment packet.

- **Follow these steps to configure this code example:**
  - PWM initialization with User LED1 (LED3)
  - I2C slave initialization
  - I2C slave receives packets from the master and control User LED1 (LED3)
  - I2C slave configure read buffer for the next read

- **PWM initialization with User LED1 (LED3)**
  - ***led_pwm_init()*** creates and configures a PWM object to drive PWM on User LED1 (LED3)
    - The ***cyhal_pwm_init_adv()*** function initializes the PWM
    - The ***cyhal_pwm_set_period()*** function sets the PWM period and initial pulse width
    - The ***cyhal_pwm_start()*** function starts the PWM

Implementation (contd.)

- **I2C slave initialization**
  - Creates and configure an I2C slave object to communicate with the I2C master by ***i2c_slave_init()***
    - The ***cyhal_i2c_init()*** function initializes the I2C peripheral
    - The ***cyhal_i2c_configure()*** function configures the I2C resource as a slave
    - The ***cyhal_i2c_slave_config_write_buffer()*** function configures the I2C slave write buffer such that the I2C master can write into it
    - The ***cyhal_i2c_slave_config_read_buffer()*** function configures the I2C slave read buffer such that the I2C master can read from it
    - Register ***handle_i2c_slave_events()*** are call-backed when an I2C slave event occurs using the ***cyhal_i2c_register_callback()*** function
    - The ***cyhal_i2c_enable_event()*** function configures and enables the I2C interrupt event

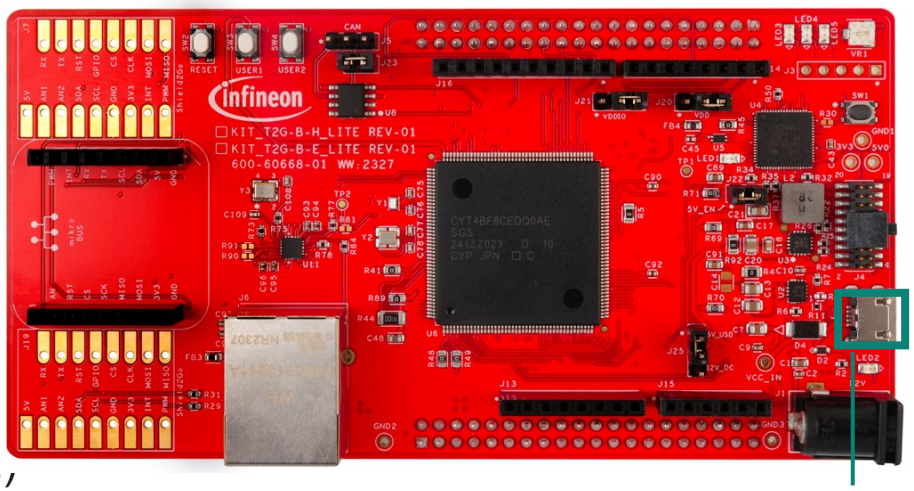
Implementation (contd.)

- **I2C slave receives packets from the master and control LED (LED3)**
  - After the slave is received, ***CYHAL_I2C_SLAVE_WR_CMPLT_EVENT*** calls back ***handle_i2c_slave_events()*** by HAL and sets the pulse width to control the brightness of the LED using the ***cyhal_pwm_set_period()*** function
  - Configure write buffer for the next write using the ***cyhal_i2c_slave_config_write_buffer()*** function

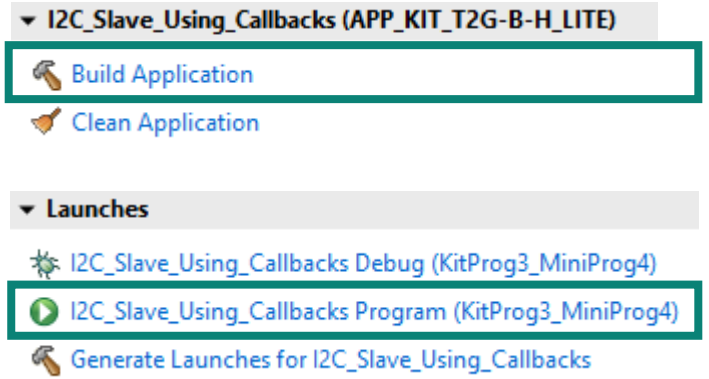
- **I2C slave configure read buffer for the next read**
  - After the read event of the slave ***CYHAL_I2C_SLAVE_RD_CMPLT_EVENT*** is notified, configure the read buffer for the next read by ***cyhal_i2c_slave_config_read_buffer()*** function.

Compiling and programming

1. Connect to power and USB cable
2. Use Eclipse IDE for ModusToolbox™ software for compiling and programming
3. For compilation:
   a. Select the target application project in the Project Explorer
   b. In the Quick Panel, scroll down, and click "Build Application" in I2C_Slave_Using_Callbacks (KIT_T2G-B-H_LITE,
4. For programming:
   a. Select the target application project in the Project Explorer
   b. In the Quick Panel, scroll down, and click "I2C_Slave_Using_Callbacks Program (KitProg3_MiniProg4)" in the Launches



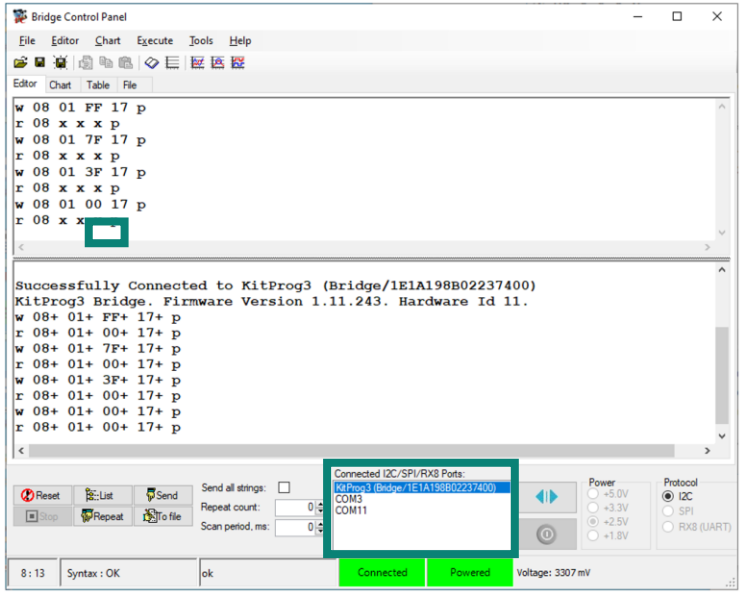
KitProg3 USB connector

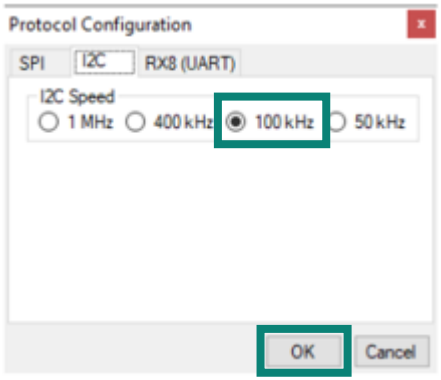


Compiling and programming (contd.)

5. Open the Bridge Control Panel (BCP) software for transmitting and receiving data over I2C.
   The BCP software is installed automatically as part of the PSoC™ Programmer installation.
   a. Open Bridge Control Panel.
   b. Select **KitProg3** under **Connected I2C/SPI/RX8 Ports**.
   c. Select **Tools** > **Protocol Configuration**, navigate to the **I2C** tab, and set the **I2C speed** to **100 kHz.** Click **OK**.

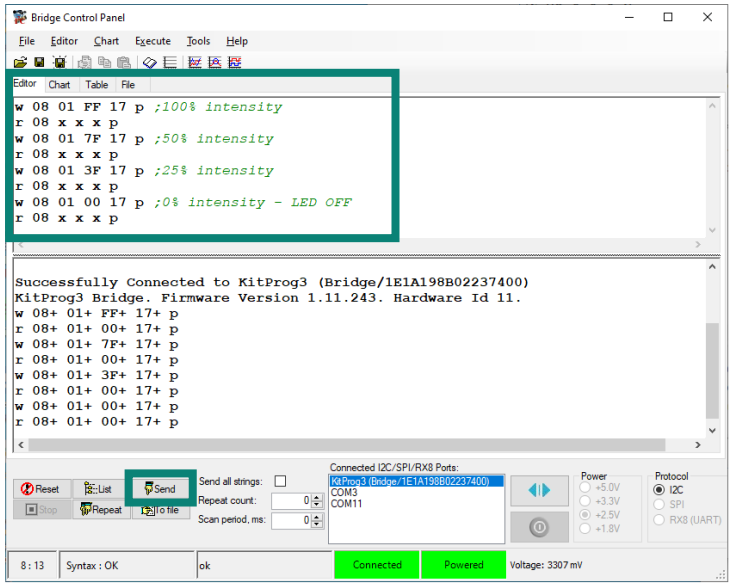
   BCP is now ready for transmitting and receiving data.





Run and test

1. After successful programming, the application starts automatically.
2. Open BCP[1] to send the command to control the User LED1 (LED3).
3. In the **Editor** tab of BCP, type the command to send the User LED1 (LED3) intensity data and then click **Send**. Observe that the User LED1 (LED3) turns ON with the specified intensity.
4. The command format that is used to write the data to the slave. In this example, BCP is used as the I2C master.
   - Sending the "**w 08 01 00 17 p**" command turns ON the User LED1 (LED3)
     with full intensity; sending the "**w 08 01 FF 17 p**" command turns OFF the User LED1 (LED3).



w/r	Slave address	SoP (Start of packet)	LED TCPWM compare value	EoP (End of packet)	Stop
w	0x08	0x1	0xFF	0x17	P
w	0x08	0x1	0x00	0x17	P

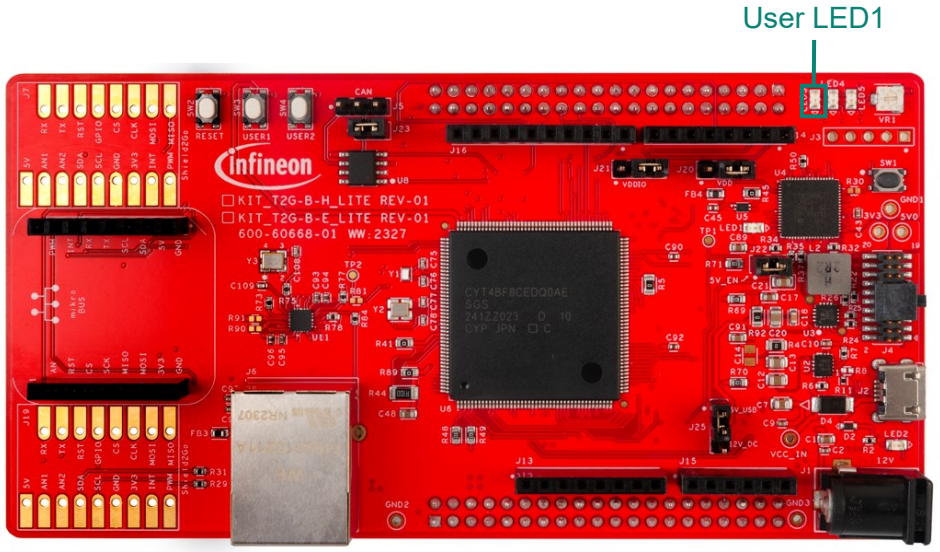
[1]: This code example cannot be debug via "KitProg3_MiniProg4" during using BCP, because both of them use the same KitProg3 port.

Run and test

5. Type the “**r 08 x x x p**” command to read the status of the write operation.
   - The following is the command format to read the status from the slave’s read buffer. The symbol ‘**x**’ denotes one byte to read from the slave’s read buffer. In this example, three bytes are read from the slave.
   - After each command is sent, the status packet must be read from the read buffer of the slave by sending the “**r 08 x x x p**” command.
   - If the packet read is in the format “**r 08 01 00 17 p**”, the status is set as '**success**'; if the packet read is “**r 08 01 FF 17 p**”, the status is set as '**fail**' for the command sent by the master.

w/r	Slave address	SoP (Start of packet)	LED TCPWM compare value	EoP (End of packet)	Stop
r	0x08	x	x	x	P

```
Successfully Connected to KitProg3 (Bridge/1E1A198B02237400)
KitProg3 Bridge. Firmware Version 1.11.243. Hardware Id 11.
w 08+ 01+ FF+ 17+ p
r 08+ 01+ 00+ 17+ p
w 08+ 01+ 7F+ 17+ p
r 08+ 01+ 00+ 17+ p
w 08+ 01+ 3F+ 17+ p
r 08+ 01+ 00+ 17+ p
w 08+ 01+ 00+ 17+ p
r 08+ 01+ 00+ 17+ p
```



6. Observe that the User LED1 (LED3) turns on/off with the specified intensity.

References

- **Datasheet**
  - CYT4BF TRAVEO™ T2G 32-bit Automotive MCU based on Arm® Cortex®- M7 dual
- **Architecture reference manual**
  - TRAVEO™ T2G Automotive MCU body controller high architecture reference manual
- **Registers reference manual**
  - TRAVEO™ T2G Automotive MCU: TVII-B-H-8M body controller high registers reference manual
- **PDL/HAL**
  - Peripheral driver library (PDL)
  - Hardware abstraction layer (HAL)
- **Training**
  - TRAVEO™ T2G training

Revision History

Revision	ECN	Submission Date	Description of Change
**	7782536	2022/07/06	Initial release
*A	8080838	2024/10/15	Replaced development board from KIT_T2G-B-H_EVK to KIT_T2G-B-H_LITE