PSoC 6 101 Training

Introduction to PSoC 6 and PSoC Creator

Dual Core features of PSoC 6 using cortex-m0+ and cortex-m4 cpu.

Hello world program in PSoC creator for blinking LED in 3 modes

1. Without CPU

2. From Firmware in M0 core

3. from M4 core .

Introduction to PDL libraries.

The M0 and M4 cores in PSoC 6 use the same registers and buses allowing for resource sharing between the 2 core. This enable application developers to partition their code to the core that best suits the needs.

RTOS

Free RTOS usage for handling multiple peripherals and communication blocks.

RTOS feature was added by selecting the checkbox for Free RTOS in the Peripheral driver library under the Build settings for the project.

UART  
UART Block code with Free RTOS for sending and receiving commands via tera term.

SCB 5 block was used for UART setup.  
Free RTSO Task was created with semaphores to run UART task on key press. And print a message.

Semaphores are enabled and use for ease of resource sharing between tasks.

Event groups are used to notify observation tasks like ezI2c and BLE of the change in the motor PWM duty cycle which can then be communicated to user.

Message queues are used to communicate the pwm duty cycle percentage between tasks, mainly to communicate the duty cycle to be set from capsense task to the motor control task.

Retarget IO

The peripheral driver library “Retarget io” was enabled in the projects build settings to re route the messages from printf() to UART on SCB5 for displaying the messages in terminal application.

Ez I2C

Basic ez I2C program was developed and same was used to send and retrieve data from the PSoC 6 using the bridge control panel.

Chart feature of Bridge control panel was used to observe the changes in the values retrieved from PSoC using the ezI2C.

The ezI2C feature was integrated into the main controller project to observe the motor speed as a graph in the charts tab of the bridge control panel.

**Logical Kill Switch**

Implemented a kill switch circuit using top schematic of PSoC creator by making use of digital multiplexer, not gates and a t-flipflop to stop PWM’s controlling motor operation by enabling and connecting the t-flipflop output to the kill pin of each of the PWM’s.

The kill switch makes use of PWM to blink a red led when enabled and solid green led when disabled by clicking on the user switch sw2 of the pioneer BLE kit.   
The project source can be found under [BasicKillSwitch\_UDB.cydsn](https://github.com/Denzerek/BLEControlledArm/tree/main/BLE_ARM_Workspace/BasicKillSwitch_UDB.cydsn) in the github repository.

**Capsense**

The capsense block was used to develop a basic program to vary the PWM duty cycle of the signal fed to an RGB led.

The capsense linear slider and 2 buttons were used.

Additional gesture controls were enabled, specifically the flick gesture on the slider which allows user to switch the color on the RGB led between blue ,green or red by flicking on the slider to the left or right.  
After selecting the required led color by flicking, the slider can be used to adjust the led brightness.

Clicking on the capsense buttons can start or stop the the PWM that controls the RGB pins.

When PWM is disabled, the Blue RGB is on.   
Here additionally, UART is enabled with retarget IO to print out the different gestures that are detected by the capsense module for learning purposes.  
The project source can be found under [CapSensing\_basic.cydsn](https://github.com/Denzerek/BLEControlledArm/tree/main/BLE_ARM_Workspace/CapSensing_basic.cydsn) in the github repository.

**Bluetooth 5.0 Peripheral**  
A basic program to control the brightness of an RGB LED over BLE was developed.

The BLE 5.0 component was added in the top schematic and configured as a peripheral device by check marking the ‘peripheral’ checkbox in the general tab.

An LED service that implements a characteristic called GREEN which is given read and write permissions was created in the ‘Gatt Settings’ tab.  
The device was given the name PSoC6 to identify the advertisement packet in the general section of the ‘Gap Settings’ tab.

The discovery mode was set to general with no timeout so its easy to find and connect to the device in the ‘Advertisement Settings’ of ‘Gap Settings’ tab without the advertisement being disabled after the timeout period.

This project can be used in combination with the BLECentralBasic project.

The advertisement packet was configured in the ‘Gap Settings’ tab to contain the device name and the LED service UUID.

API Description:

1. genericEventHandler():  
   To handle generic Bluetooth events like device connect, disconnect and write events.  
   When the device is disconnected, the ‘Blink’ PWM component is started which will blink the RED RGB led and the BLE advertisement is started  
   When the deivce is connected to, the ‘Blink’ PWM component is stopped and the ‘PWM\_DIM’ component is started which causes the green RGB led to glow.  
   A write operation to the ‘GREEN’ characteristic from the connected device, in this case CySmart BLE application, with an appropriate hex value changes the brightness of the RGB LED. (allowed write value range from 0x00-0x64)
2. bleInterruptNotify():  
   This routine is registered as the BLE application host callback for releasing the RTOS semaphore for processing BLE events when any BLE events occur.
3. bleTask();  
   This task is executed by the xTaskCreate() and vTaskStartScheduler() in the main routine to start the BLE stack and subsequent connect, disconnect and write event processes.

The project source can be found under [CustomBLEBasic.cydsn](https://github.com/Denzerek/BLEControlledArm/tree/main/BLE_ARM_Workspace/CustomBLEBasic.cydsn) in the github repository.  
 **Bluetooth 5.0 Central**

A basic program that uses the BLE component as central device to connect with a BLE peripheral device was developed.  
It uses the same LED characteristic that was used in the [CustomBLEBasic.cydsn](https://github.com/Denzerek/BLEControlledArm/tree/main/BLE_ARM_Workspace/CustomBLEBasic.cydsn) project. (By saving this service and then loading the same in this BLE central project so that the service with the same UUID is available in both projects.). This makes it possible for the firmware to access the UUID of the LED service via ‘cy\_ble\_customCServ[CY\_BLE\_CUSTOMC\_LED\_SERVICE\_INDEX].uuid’ and then compare this UUID with the one that was obtained after scanning the surrounding BLE devices advertisement packets. If the comparison returns true, then we have found the device that we have to connect to and can successfully accomplish the same.  
  
API Description:

1. genericEventHandler():  
   The stack on and device disconnect events will start the BLE GAP scan operation and at the same time turn on the red led ,to indicate device disconnection, and turns off the green led.  
   The scan progress result event is triggered when a device is detected in the scan operation and a compare operation on the UUID of the LED service is performed with the UUID of the service that was detected. If the UUID’s match, then a connect operation is performed and the scan operation is stopped.  
   Additional BLE events for unsuccessful/successful write, discovery status and connect/disconnect events are also handled.
2. bleTask()  
   It registers the genericEventHandler() and processes the BLE events in an infinite loop.  
   It also monitors the UART SCB5 for incoming characters ‘+’ and ‘-‘ to increase and decrease the LED brightness.
3. writeLED():  
   This routine takes the brightness value and sends the same to the GREEN characteristic in LED service .

The project source can be found under [BLECentralBasic.cydsn](https://github.com/Denzerek/BLEControlledArm/tree/main/BLE_ARM_Workspace/BLECentralBasic.cydsn)in the github repository.