

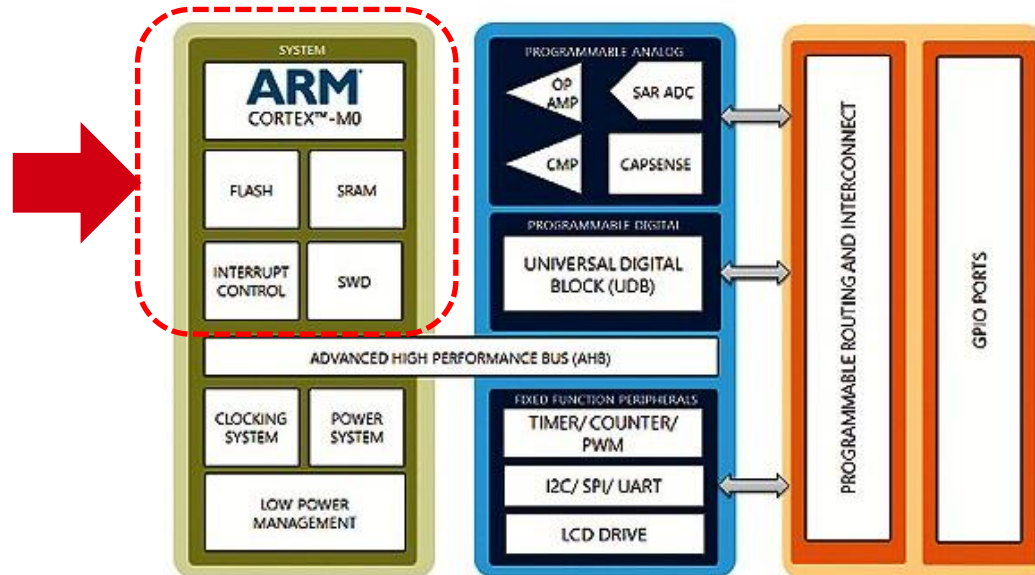
Chapter-5

Embedded Computer

Overview

Components

PSoC® 4



Details

 Non-Volatile
 Volatile

Processor

ALU

Registers

Executes Program Instruction

- **Clock Frequency (Max/ Min)**
- **Clock Cycles per Instruction**
- **Clock Freq vs. Power Consumption**
- **Supported Power Modes**
- **No of Cores, 8/32/64 bits**

Memory

FLASH

Stores Program Instruction

RAM

Stores Runtime Variables

ROM

Stores Silicon IDs and Non-Programmable SoC Component Settings

EEPROM

For Backing Up Data During Run-Time. Non-Volatile

E.g.,

```
#include "project.h"

const uint8_t NUM_SAMPLES = 3; // Number of samples to average

uint8_t adc_samples[NUM_SAMPLES]; // Array to store ADC samples
uint8_t sample_index = 0; // Index to store new ADC samples
uint16_t filtered_sample = 0; // Filtered sample value

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    ADC_SAR_Start();
    ADC_SAR_StartConvert();
    UART_Start();

    for(;;)
    {
        while(!ADC_SAR_IsEndConversion(ADC_SAR_WAIT_FOR_RESULT)); // Wait for ADC conversion to complete

        adc_samples[sample_index] = ADC_SAR_GetResult8();
        sample_index = (sample_index + 1) % NUM_SAMPLES;

        // Calculate moving average of ADC samples
        filtered_sample = (adc_samples[0] + adc_samples[1] + adc_samples[2]) / NUM_SAMPLES;

        // Send filtered sample via UART-Tx in byte format
        uint8_t tx_data[1];
        tx_data[0] = filtered_sample;
        UART_PutArray(tx_data, 1);

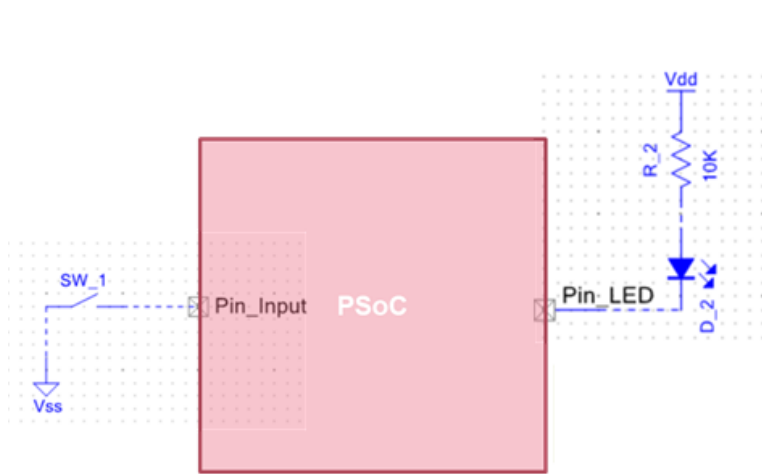
        CyDelay(1000); // Sample ADC every 1 second
    }
}
```

Question: Identify the variables which are stored in Flash, and which are in RAM

A program to read ADC, then perform a 2nd order moving average and transfer the result via UART.

Interrupts

Interrupts – The Need

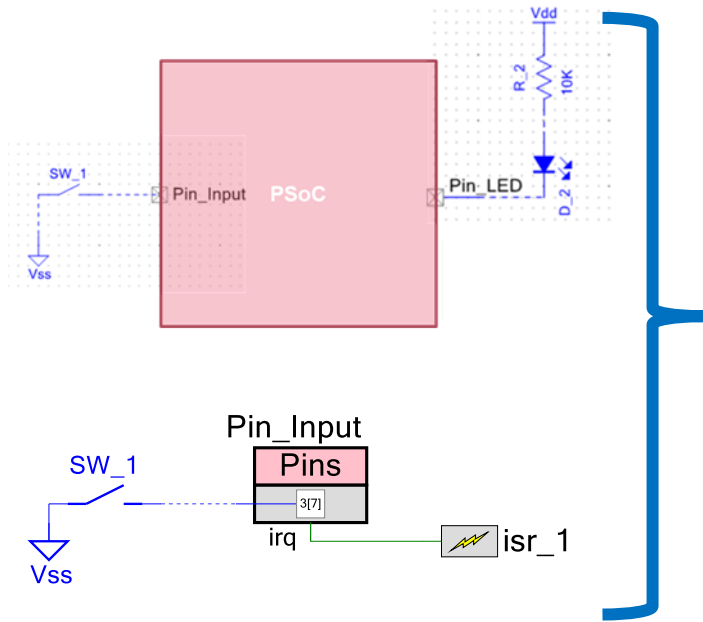


```
#include "project.h"

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    for(;;)
    {
        /* Check if switch is pressed */
        if (Pin_Input_Read() == 0) {
            /* Toggle LED */
            Pin_LED_Write(!Pin_LED_Read());
        }
    }
}
```

Using GPIO Interrupt



```
#include "project.h"

CY_ISR ( my_isr_code )
{
    // toggling the LED
    Pin_LED_Write(~Pin_LED_Read());

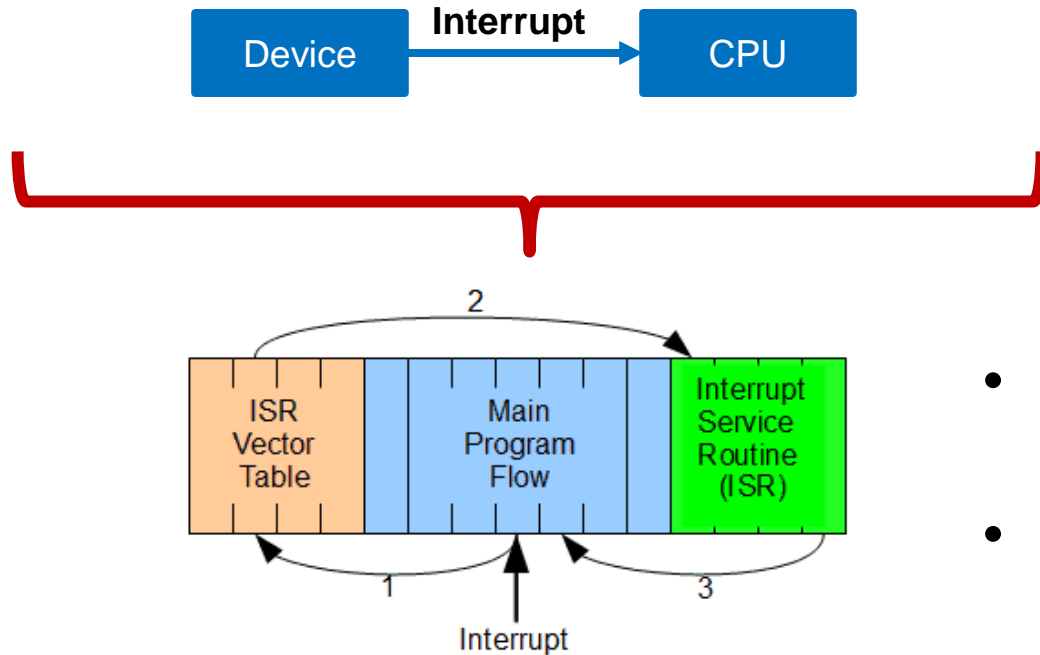
    // clear the pin interrupt
    Pin_Input_ClearInterrupt();
}

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    /* Place your initialization/startup code here (e.g. MyInst_Start()) */
    isr_1_StartEx(my_isr_code);

    for(;;)
    {
        /* Place your application code here. */
    }
}
```


Interrupt Handling



- **Interrupt Vector Table ?**
- **Stack Overflow ?**

Image Credit: <https://embedds.com/basic-understanding-of-microcontroller-interrupts/>

Interrupt Controller

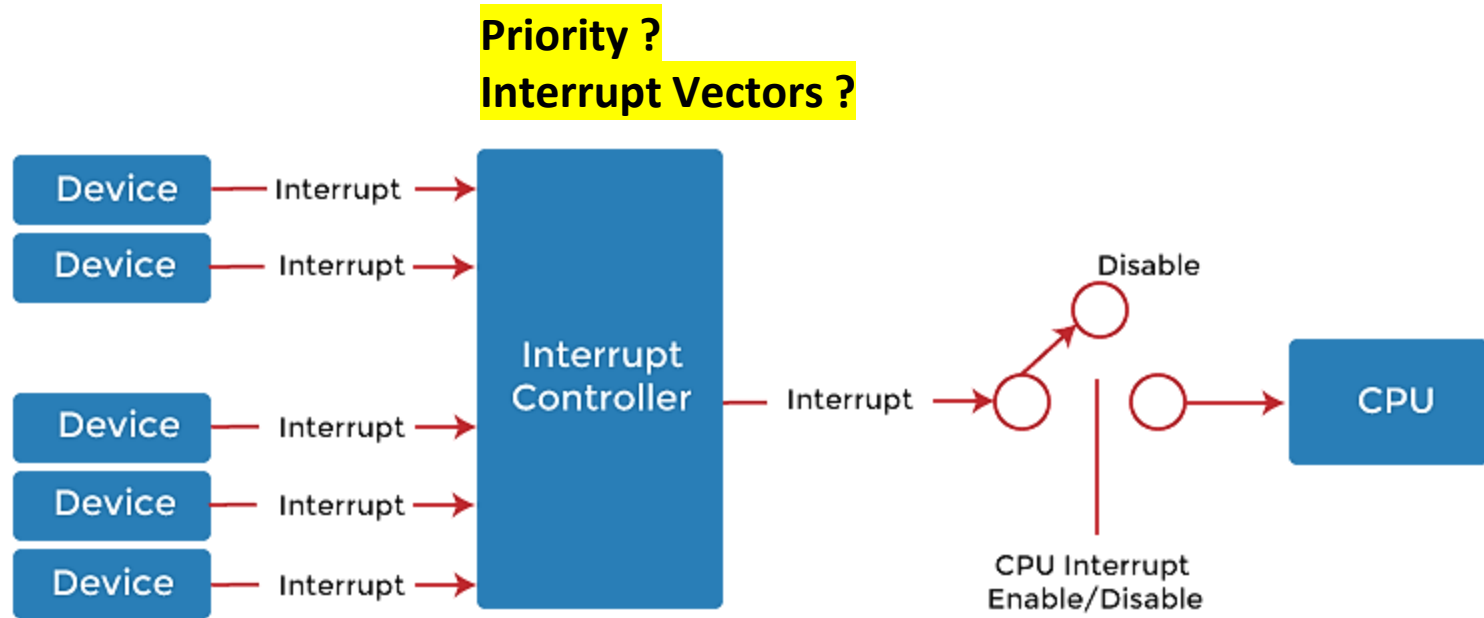
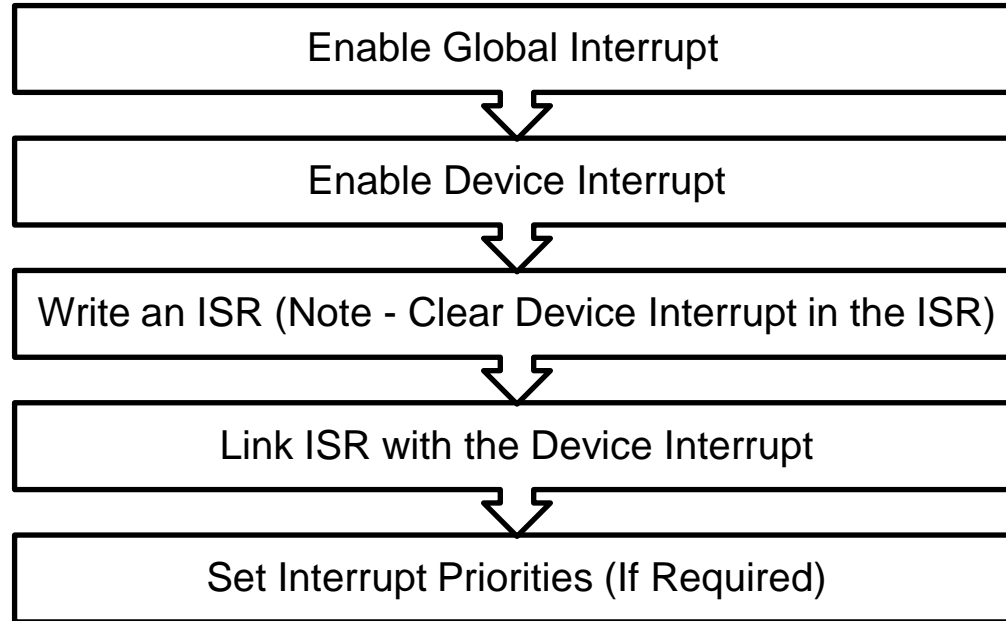


Image Credit: <https://www.javatpoint.com/what-is-interrupt-in-os>

Interrupt Workflow



The Need for RTOS

Task Scheduling

- Toggle an LED every 1 second.
- Toggle another LED every 2 second.
- Toggle another LED every 1.5 second.
- Toggle couple of other LED's 3 seconds, 4.1 seconds, and 4.5 seconds
- Add breathing effect to an LED. (i.e., increase/decrease the intensity of LED over time)
- Add ...



RTOS – How To ?

Define Tasks → Create Tasks → Call the Scheduler

```
#include "project.h"
#include "FreeRTOS.h"
#include "task.h"

void vLED1Task(void *pvParameters)
{
    for (;;)
    {
        /* Toggle LED1 */
        Cy_GPIO_Write(LED1_PORT, LED1_NUM, ~Cy_GPIO_Read(LED1_PORT, LED1_NUM));
        vTaskDelay(1000 / portTICK_PERIOD_MS); /* Delay for 1 second */
    }
}

void vLED2Task(void *pvParameters)
{
    for (;;)
    {
        /* Toggle LED2 */
        Cy_GPIO_Write(LED2_PORT, LED2_NUM, ~Cy_GPIO_Read(LED2_PORT, LED2_NUM));
        vTaskDelay(2000 / portTICK_PERIOD_MS); /* Delay for 2 seconds */
    }
}

void vLED3Task( void *pvParameters )
{
    for(;;)
    {
        /* Toggle LED3 */
        Cy_GPIO_Write(LED3_PORT, LED3_NUM, ~Cy_GPIO_Read(LED3_PORT, LED3_NUM));
        vTaskDelay(3200 / portTICK_PERIOD_MS); /* Delay for 3.2 seconds */
    }
}
```

```
int main(void)
{
    /* Enable global interrupts. */
    CyGlobalIntEnable;

    /* Create tasks */
    xTaskCreate(vLED1Task, "LED1 Task", configMINIMAL_STACK_SIZE, NULL, 1, NULL);
    xTaskCreate(vLED2Task, "LED2 Task", configMINIMAL_STACK_SIZE, NULL, 1, NULL);
    xTaskCreate(vLED3Task, "LED3 Task", configMINIMAL_STACK_SIZE, NULL, 1, NULL);

    /* Start the FreeRTOS scheduler */
    vTaskStartScheduler();

    /* The scheduler should never return */
    for (;;)
    {
    }
}
```

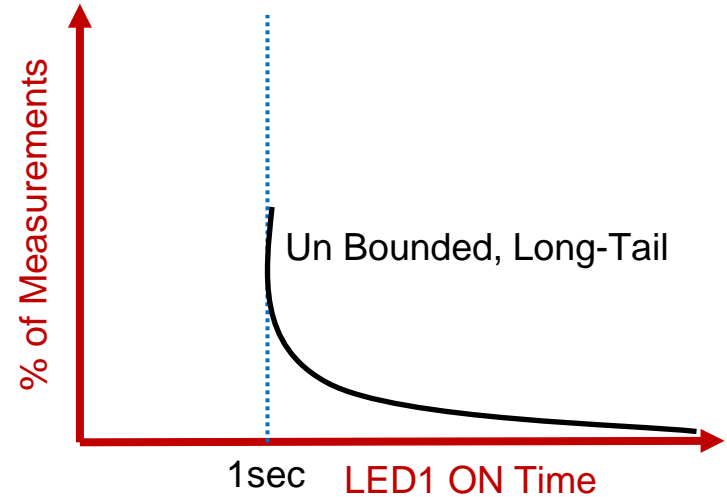
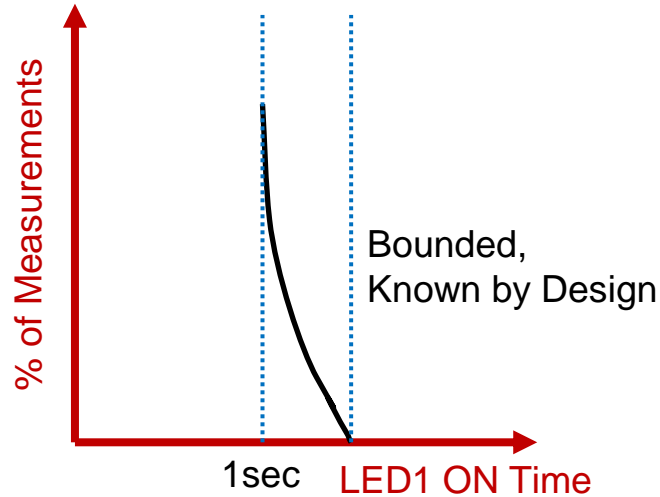
Note: Code is Generated by ChatGPT

Question: RTOS vs OS. What's the difference ?

Task-1, Task-2, and Task-3 toggles LED-1,2, and 3 every 1, 2 and 3.2 seconds respectively.

RTOS vs. Generic OS

```
void vLED1Task(void *pvParameters)
{
    for (;;)
    {
        /* Toggle LED1 */
        Cy_GPIO_Write(LED1_PORT, LED1_NUM, ~Cy_GPIO_Read(LED1_PORT, LED1_NUM));
        vTaskDelay(1000 / portTICK_PERIOD_MS); /* Delay for 1 second */
    }
}
```



Semaphores –The Need

```
// Shared resource
int shared_array[ARRAY_SIZE];

void task1(void *parameter)
{
    while (1)
    {
        // Generate random numbers and update the first
        // five elements of the array
        for (int i = 0; i < 5; i++)
        {
            shared_array[i] = rand() % 100;
        }
        // Sleep for some time
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}

void task2(void *parameter)
{
    while (1)
    {
        // Calculate checksum of the first five elements of the array
        int checksum = 0;
        for (int i = 0; i < 5; i++)
        {
            checksum += shared_array[i];
        }

        // Store checksum in the sixth element of the array
        shared_array[5] = checksum;

        // Sleep for some time
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

- Task-1 updates first five indices of **shared_array**.
- Task-2 computes checksum of the first five indices and store it in the sixth index of **shared_array**.

Question – Any Issues ?

Note: Code is Generated by ChatGPT

Using Semaphores

```
int shared_array[ARRAY_SIZE];

// Semaphore to protect the shared resource
SemaphoreHandle_t semaphore;

void task1(void *parameter)
{
    while (1)
    {
        // Wait for semaphore to be available
        xSemaphoreTake(semaphore, portMAX_DELAY);

        // Generate random numbers and update the
        // first five elements of the array
        for (int i = 0; i < 5; i++)
        {
            shared_array[i] = rand() % 100;
        }

        // Release the semaphore
        xSemaphoreGive(semaphore);

        // Sleep for some time
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

```
void task2(void *parameter)
{
    while (1)
    {
        // Wait for semaphore to be available
        xSemaphoreTake(semaphore, portMAX_DELAY);

        // Calculate checksum of the first five elements of the array
        int checksum = 0;
        for (int i = 0; i < 5; i++)
        {
            checksum += shared_array[i];
        }

        // Store checksum in the sixth element of the array
        shared_array[5] = checksum;

        // Release the semaphore
        xSemaphoreGive(semaphore);

        // Sleep for some time
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```

Note: Code is Generated by ChatGPT

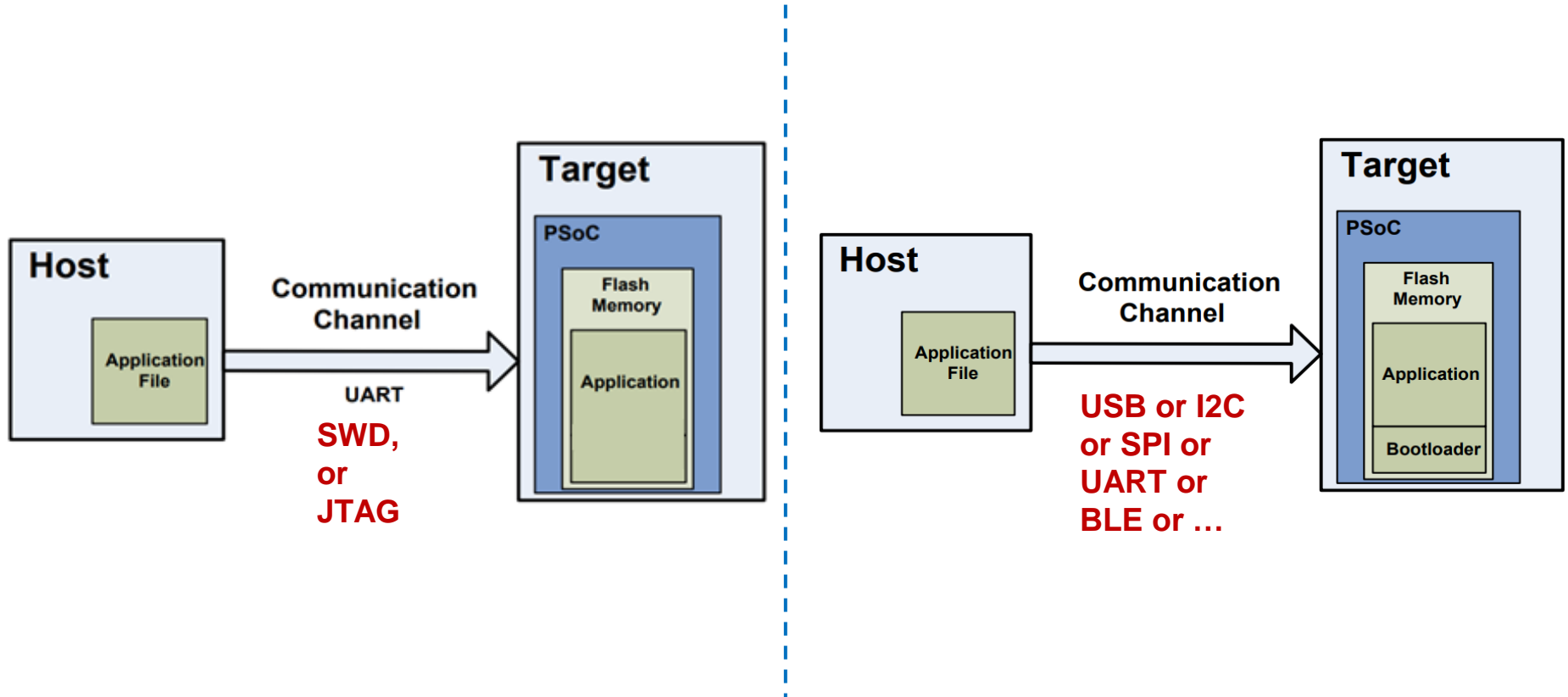
What Next ?

Only for interested students

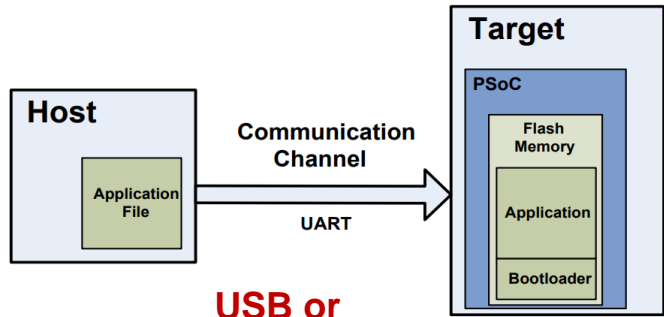
- Learn running FreeRTOS on PSoC-4
- Understand the importance of the arguments in xTaskCreate
- Learn – What is stack for a task in RTOS
- Learn how to delete a task that is already created (to free up resources)
- Task Priority – how to ?
- Sharing data between tasks. How to ? (Usage of Semaphores and Mutexes)

Programming & Bootloaders

The Difference



Bootloader E.g.,



USB or
I2C or SPI
or UART
or BLE or
...

Ref: AN68272 (Infineon)

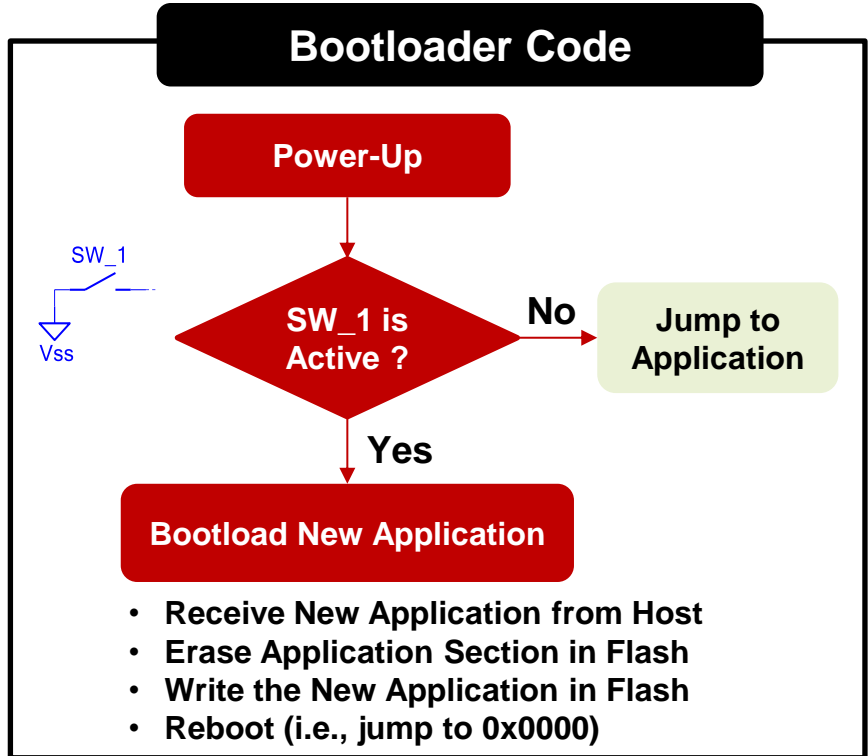
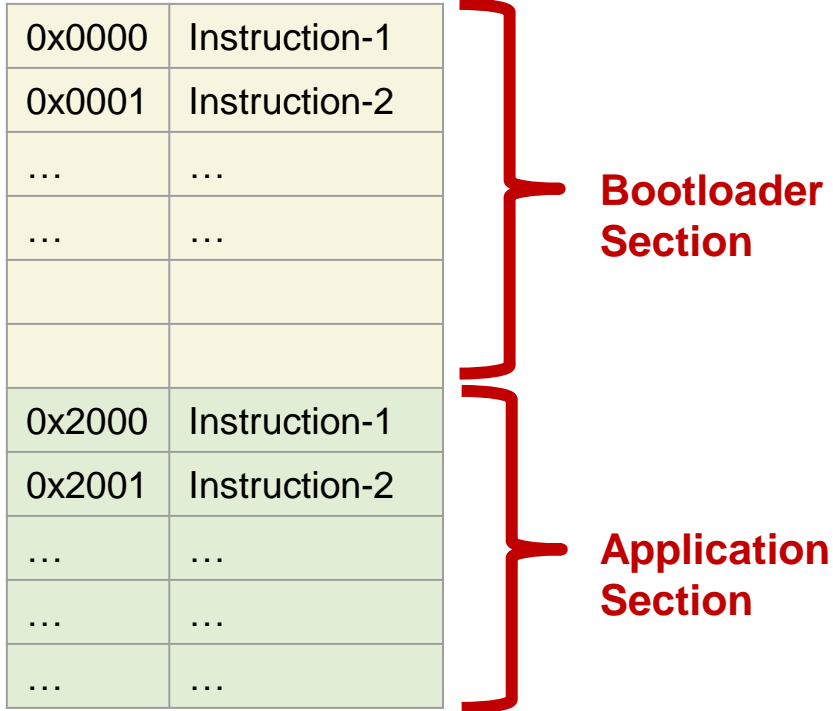
Target Flash

0x0000		}	Bootloader Code
0x0001			
...			
...			
0x2000		}	Application
0x2001			
...			
...			
...			
...			

Note: For simplicity, ISR vector table is not shown

Bootloader E.g.,

Target Flash



Note: Host should know the start address of the application section