API For Tiva C

Important Files to Include

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"
#include "driverlib/systick.h"
#include "inc/hw_types.h"
#include "driverlib/interrupt.h"
#include "inc/hw_nvic.h"
#include "inc/hw_ints.h"
#include "driverlib/timer.h"
#include "driverlib/uart.h"
#include "driverlib/pin_map.h"
```

Clock Set

```
// setting clock to use the external main oscillator at 16 mhz and using the PLL
to increase the frequency and using SYSCTL_SYSDIV_2_5 to get 80 mhz

SysCtlClockSet(SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ | SYSCTL_USE_PLL |
SYSCTL_SYSDIV_2_5);
```

DELAY

```
void SysCtlDelay(ui32Count);
```

GPIO

```
//This function enables the peripheral we are going to use
void SysCtlPeripheralEnable(ui32Peripheral);

// To set pins as an input
void GPIOPinTypeGPIOInput(ui32Port,ui8Pins);

//To set pins as an output
void GPIOPinTypeGPIOOutput(ui32Port,ui8Pins);
```

```
//To write a value on a pin
void GPIOPinWrite(ui32Port,ui8Pins,ui8Val);

//To read a value from a pin
int32_t GPIOPinRead(ui32Port,ui8Pins);

//To configure the max current to be drawn and set the pin type
(pullup/pulldown/open drain/analog pin)
void GPIOPadConfigSet(ui32Port, ui8Pins, ui32Strength,ui32PadType);

//Another way to set the direction of the pin
void GPIODirModeSet(ui32Port,ui8Pins, ui32PinIO);
```

Peripherals

```
SYSCTL_PERIPH_GPIOA --> GPIO A

SYSCTL_PERIPH_GPIOB --> GPIO B

SYSCTL_PERIPH_GPIOC --> GPIO C

SYSCTL_PERIPH_GPIOD --> GPIO D

SYSCTL_PERIPH_GPIOE --> GPIO E

SYSCTL_PERIPH_GPIOF --> GPIO F
```

PORTS

```
GPIO_PORTA_BASE --> PORTA

GPIO_PORTB_BASE --> PORTB

GPIO_PORTC_BASE --> PORTC

GPIO_PORTD_BASE --> PORTD

GPIO_PORTE_BASE --> PORTE

GPIO_PORTF_BASE --> PORTF
```

PINS

```
GPIO_PIN_0

GPIO_PIN_1

GPIO_PIN_2

GPIO_PIN_3

GPIO_PIN_4

GPIO_PIN_5

GPIO_PIN_6

GPIO_PIN_7
```

STRENGTH

```
// defines the max current for the pin
GPIO_STRENGTH_2MA // for 2 milli ampere

GPIO_STRENGTH_4MA // for 4 milli ampere

GPIO_STRENGTH_6MA // for 6 milli ampere

GPIO_STRENGTH_8MA // for 8 milli ampere
```

PadType

```
GPIO_PIN_TYPE_STD   // Push-pull

GPIO_PIN_TYPE_STD_WPU // Push-pull with pull-up

GPIO_PIN_TYPE_STD_WPD  // Push-pull with pull-down

GPIO_PIN_TYPE_OD      // open drain
```

PinIO

```
GPIO_DIR_MODE_IN    // Pin is a GPIO input

GPIO_DIR_MODE_OUT   // Pin is a GPIO output
```

Code Example

```
// Set pin 3 port f (PF3) as an output pin and set it high

// enable Port F
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

// set pin 3 in Port F as an output
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,GPIO_PIN_3);

// set pin 3 in Port F HIGH
GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_3,GPIO_PIN_3)

//set pin 0 port F as an input pullup and put the pin state in a variable

// enable Port F
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);

// set pin 0 in Port F as an input
GPIOPinTypeGPIOInput(GPIO_PORTF_BASE,GPIO_PIN_0);

//pull up pin 0
GPIOPadConfigSet(GPIO_PORTF_BASE,GPIO_PIN_0,GPIO_STRENGTH_8MA,GPIO_PIN_TYPE_STD_WP
U);

pin0_state=GPIOPinRead(GPIO_PORTF_BASE,GPIO_PIN_0);
```

UART

```
//To enable UART
SysCtlPeripheralEnable(SYSCTL_PERIPH_UART1); // enable UART1 needs to enable PORT
B
SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0); // enable UART0 needs to enable PORT
A

//To configure the pins for the alternate functions
// configuring UART1 pins (B0,B1)
GPIOPinConfigure(GPIO_PB0_U1RX);
GPIOPinConfigure(GPIO_PB1_U1TX);

// configuring UART0 pins (A0,A1)
GPIOPinConfigure(GPIO_PA0_U0RX);
GPIOPinConfigure(GPIO_PA1_U0TX);

//To configure the UART pins

// enabling A0,A1 for UART0
GPIOPinTypeUART(GPIO_PORTA_BASE,GPIO_PIN_0 | GPIO_PIN_1 );

// enabling B0,B1 for UART1
```

```
GPIOPinTypeUART(GPIO_PORTB_BASE,GPIO_PIN_0 | GPIO_PIN_1 );
-To configure the UART

//UARTX,CLOCK,Baud_Rate,(8 bits data to be send | no parity bit | one stop bit)

//configuring UART1
UARTConfigSetExpClk(UART1_BASE, SysCtlClockGet(), 9600, (UART_CONFIG_WLEN_8 |
UART_CONFIG_PAR_NONE | UART_CONFIG_STOP_ONE));

//configuring UART0
UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(), 9600, (UART_CONFIG_WLEN_8 |
UART_CONFIG_PAR_NONE | UART_CONFIG_STOP_ONE));

// To receive a character

// from UART 1
UARTCharGet(UART1_BASE);

// from UART 0
UARTCharGet(UART0_BASE);

//To send a character

// send a character on UART1
UARTCharPut(UART1_BASE,c);

// send a character on UART0
UARTCharPut(UART0_BASE,c);
```

INTERRUPTS

```
// to disable GPIO interrupts for pin F4
GPIOIntDisable(GPIO_PORTF_BASE, GPIO_PIN_4);

// to clear interrupt flag for pin F4
// you must clear the flag in the beginning of the interrupt handler
GPIOIntClear(GPIO_PORTF_BASE, GPIO_PIN_4);

// to set the interrupt handler for PORTF
GPIOIntRegister(GPIO_PORTF_BASE,INTERRUPT_HANDLER);

// to set the type od detection
GPIOIntTypeSet(GPIO_PORTF_BASE, GPIO_PIN_4,GPIO_FALLING_EDGE); // pin F4 interrupt
on falling edge

GPIO_FALLING_EDGE   // Interrupt on falling edge
GPIO_RISING_EDGE    // Interrupt on rising edge
GPIO_BOTH_EDGES     // Interrupt on both edges
GPIO_LOW_LEVEL      // Interrupt on low level
GPIO_HIGH_LEVEL     //Interrupt on high level
```

```
// to enable the interrupt for pin F4
GPIOIntEnable(GPIO_PORTF_BASE, GPIO_PIN_4);
```

TIMERS

```
// to enable the timer
//SYSCTL_PERIPH_TIMER0 -->timer 0
//SYSCTL_PERIPH_TIMER1 -->timer 1
//SYSCTL_PERIPH_WTIMER0 -->wide timer 0

SysCtlPeripheralEnable(SYSCTL_PERIPH_WTIMER0); // enabling wide timer 0

// to configure the timer
// configuring wide timer 0 as  split pair (using timer A ) periodic mode
TimerConfigure(WTIMER0_BASE, TIMER_CFG_SPLIT_PAIR | TIMER_CFG_A_PERIODIC);

// to set the value the timer counts to to timeout
TimerLoadSet(WTIMER0_BASE, TIMER_A, vlaue);

// to set the prescale for wide timer 0
TimerPrescaleSet(WTIMER0_BASE, TIMER_A, value);

// to assign the interrupt handler for wide timer 0 ,timer A
TimerIntRegister(WTIMER0_BASE, TIMER_A, INTERRUPT_HANDLER);

// to enable the timer interrupt for wide timer 0 timer A
TimerIntEnable(WTIMER0_BASE, TIMER_A);

// enable wide timer 0 ,timer A to start
TimerEnable(WTIMER0_BASE, TIMER_A);

// enable master enable
IntMasterEnable();

//to clear timeout flag for wide timer 0,timer A
// clear the flag ine the beginning of the INTERRUPT HANDLER
TimerIntClear(WTIMER0_BASE, TIMER_TIMA_TIMEOUT);
```

PWM

```
// configure GPIO pin F3 for the alternate function for timer 1
GPIOPinConfigure(GPIO_PF3_T1CCP1);

// configure PIN F3 for the timer pin
GPIOPinTypeTimer(GPIO_PORTF_BASE, GPIO_PIN_3);
```

```
//configure Timer 1 mode as split pair using timer B in PWM mode
TimerConfigure(TIMER1_BASE, TIMER_CFG_SPLIT_PAIR | TIMER_CFG_B_PWM);

// to set the frequency for the PWM    //put 50000
TimerLoadSet(TIMER1_BASE, TIMER_B, period);

// enabling timer B
TimerEnable(TIMER1_BASE, TIMER_B);

// to change the PWM duty cycle for timer 1 timer B
TimerMatchSet(TIMER1_BASE, TIMER_B, duty_cycle); // the max value for the PWM can
not excede the period value
```