Task 01: Change PWM duty cycle from 10% to 90% to control the brightness of the LED at PF1.

Below is the code used to perform the task

...

ROM\_GPIOPinTypePWM(GPIO\_PORTF\_BASE, GPIO\_PIN\_1); ROM\_GPIOPinConfigure(GPIO\_PF1\_M1PWM5);

//No more buttons being used.

//HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_LOCK) = GPIO\_LOCK\_KEY;

//HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_CR) |= 0x01;

//HWREG(GPIO\_PORTF\_BASE + GPIO\_O\_LOCK) = 0;

//ROM\_GPIODirModeSet(GPIO\_PORTF\_BASE, //GPIO\_PIN\_4|GPIO\_PIN\_0,GPIO\_DIR\_MODE\_IN);

//ROM\_GPIOPadConfigSet(GPIO\_PORTF\_BASE, //GPIO\_PIN\_4|GPIO\_PIN\_0,GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU);

//to activate the led as a pwm pin

ROM\_GPIOPinTypePWM(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

ui32PWMClock = SysCtlClockGet() / 64;

ui32Load = (ui32PWMClock / PWM\_FREQUENCY) - 1;

ROM\_PWMGenConfigure(PWM1\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_UP\_DOWN |PWM\_GEN\_MODE\_NO\_SYNC);

ROM\_PWMGenPeriodSet(PWM1\_BASE, PWM\_GEN\_2, ui32Load);

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui32Load \* .5);

ROM\_PWMOutputState(PWM1\_BASE, PWM\_OUT\_5\_BIT, true);

ROM\_PWMGenEnable(PWM1\_BASE, PWM\_GEN\_2);

while(1)

{

//the control of the pulse from 10% to 90%

int i;

for(i=((int) ui32Load \* .10);i<ui32Load;i=i+2){

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, i);

SysCtlDelay(2000);

}

for(i=((int) ui32Load \* .90);i>=0;i=i-2){

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, i);

SysCtlDelay(2000);

}

}

}

Task 02: Change PWM duty cycle from 10% to 90% to control the brightness of the all three

LED at PF1, PF2, and PF3 using three nested “for loops”.

Below is the code, using the for loops to control the leds from 10 to 90% of a PWM duty cycle.

…

int hundinc = 100;

...

ROM\_GPIOPinTypePWM(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2 | GPIO\_PIN\_3);

ROM\_GPIOPinConfigure(GPIO\_PF1\_M1PWM5); //map PF1 as PWM

ROM\_GPIOPinConfigure(GPIO\_PF2\_M1PWM6); //map PF2 as PWM

ROM\_GPIOPinConfigure(GPIO\_PF3\_M1PWM7); //map PF3 as PWM

ROM\_GPIOPinTypePWM(GPIO\_PORTF\_BASE, GPIO\_PIN\_1);

ui32PWMClock = SysCtlClockGet() / 64;

ui32Load = (ui32PWMClock / PWM\_FREQUENCY) - 1;

ROM\_PWMGenConfigure(PWM1\_BASE, PWM\_GEN\_2, PWM\_GEN\_MODE\_UP\_DOWN | PWM\_GEN\_MODE\_NO\_SYNC);

ROM\_PWMGenPeriodSet(PWM1\_BASE, PWM\_GEN\_2, ui32Load);

ROM\_PWMGenConfigure(PWM1\_BASE, PWM\_GEN\_3, PWM\_GEN\_MODE\_UP\_DOWN | PWM\_GEN\_MODE\_NO\_SYNC);

ROM\_PWMGenPeriodSet(PWM1\_BASE, PWM\_GEN\_3, ui32Load);

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, ui32Load \* .1);

// initial duty cycle for red led

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_6, ui32Load \* .1);

// initial duty cycle for green led

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_7, ui32Load \* .1);

// initial duty cycle for blue

ROM\_PWMOutputState(PWM1\_BASE, PWM\_OUT\_5\_BIT | PWM\_OUT\_6\_BIT | PWM\_OUT\_7\_BIT, true);

// enable output

ROM\_PWMGenEnable(PWM1\_BASE, PWM\_GEN\_2);

ROM\_PWMGenEnable(PWM1\_BASE, PWM\_GEN\_3);

while(1)

{

//the control of the pulse from 10% to 90%

int i;

for(i=((int) ui32Load \* .10); i < ui32Load; i = i + hundinc){

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, i);

int j;

for(j=((int) ui32Load \* .10); j < ui32Load; j = j + hundinc){

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_6, j);

int k;

for(k=((int) ui32Load \* .10); k < ui32Load; k = k + hundinc){

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_7, k);

SysCtlDelay(200000);

// to visibly see the change on camera

}

SysCtlDelay(200000);

}

SysCtlDelay(200000);

}

//the control of the pulse from 90% to 10%

for(i=((int) ui32Load \* .90); i >= ((int) ui32Load \* .10); i = i - hundinc){

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_5, i);

int j;

for(j=((int) ui32Load \* .90); j >= ((int) ui32Load \* .10); j = j - hundinc){

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_6, j);

int k;

for(k=((int) ui32Load \* .90); k >= ((int) ui32Load \* .10); k = k - hundinc){

ROM\_PWMPulseWidthSet(PWM1\_BASE, PWM\_OUT\_7, k);

SysCtlDelay(200000);

}

SysCtlDelay(200000);

}

SysCtlDelay(200000);

}

}

}