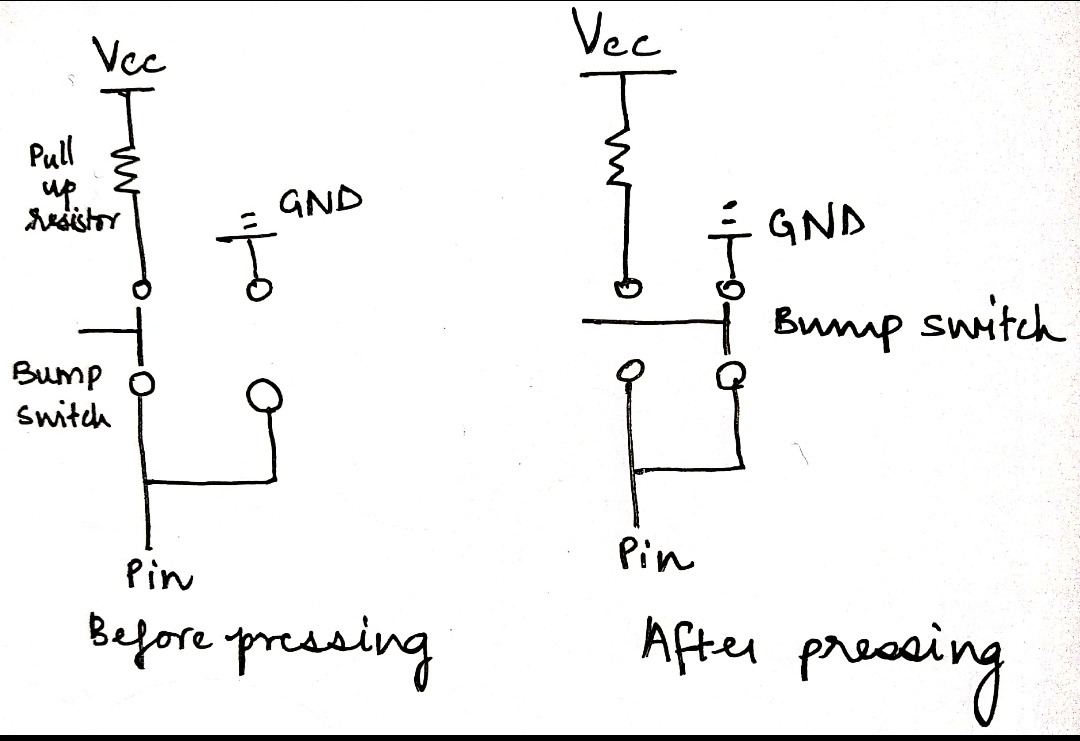
CE2107 Lab3 Assignment Sheet (to be submitted to NTULearn before next lab)

Name: Unnikrishnan Malavika Lab Group: SEP4 Date: 25/09/2020

1. Section 6. Other than the procedures outlined in the introduction of Exception Handling, what other registers need to be noted when using the Exception Handling System in ARM Cortex M4F processor? Think globally…

**ANS. BASEPRI – to selectively disable interrupts – based on base priority level**

**PRIMASK – for disabling all interrupts globally irrespective of priority.**

1. Section 6.2. The bump switch used in the lab is shown below. Pin 1 and 3 of the bump switch are connected to the MSP432. Draw the internal circuit of the bump switch and describe how the MSP432 GPIO can be used to detect that the switch is closed?  
    

**ANS. The t-shaped element is the bump-switch. When the switch is pressed i.e. closed, the pin goes to GND and hence we can read a LOW. Otherwise the pin will read a HIGH.**

1. Section 6.3. Write down the GPIO configuration used for pins connected to pin 3 of the Bump switch.

**ANS.**

P4->SEL0 &= ~0xED;

P4->SEL1 &= ~0xED; // Initialize Bump sensors

P4->DIR &= ~0xED; //Set pins as input

P4->REN |= 0xED; //enable pull-up/pull-down

P4->OUT |= 0xED; //activated interface pull-up

This is what we did to enable all the necessary pins, but to setup only pin 3, we must replace ED with 08

P4->SEL0 &= ~0x08;

P4->SEL1 &= ~0x08; // Initialize Bump sensors

P4->DIR &= ~0x08; //Set pin 4.3 as input

P4->REN |= 0x08; //enable pull-up/pull-down

P4->OUT |= 0x08; //activated interface pull-up

1. Section 6.3. Illustrate with detail working and APIs used how the systick timer is configured to interrupt the system at 1000Hz frequency.

**ANS. cycle waiting time = 10000 microseconds = 1000 Hz frequency.**

Systick timer interrupt happens whenever its reload register counts down to 0. Here, this reload count is thought to be the period argument passed into the function SystickInit(period, priority) when we first configure Systick timer.

Number of times the systick handler is called in 1 second = frequency/ amount of time that the period in the above API SystickInit() is counted to 0.

**∴ No. Of times the handler is called per second = 48MHz / 48000 = 1000 times.**

1. Section 6.4. In the Simple motor project, the api used to move the motor forward is Motor\_ForwardSimple(uint16\_t duty, uint32\_t time), where time is number of 10ms units, i.e. if time=2, motor will run for 2ms. Show and explain the code in the function that enable this 10ms unit timing.

**ANS. The function used to do this is SysTick\_Wait1us(duty);**

void Motor\_RightSimple(uint16\_t duty, uint32\_t time){

// Drives just the right motor forward at duty (100 to 9900)

// Left motor is stopped (sleeping)

// Runs for time duration (units=10ms), and then stops

// Stop the motor and return if any bumper switch is active

// Returns after time\*10ms or if a bumper switch is hit

uint32\_t counter;

counter = time;

// Forward direction

P1->OUT &= ~0xC0;

while(counter>0 && Bump\_Read()==0x3F){

//while(counter>0){

// Enable R motor

P3->OUT |= 0x40;

// PWM HIGH

P2->OUT |= 0x40;

SysTick\_Wait1us(duty);

//PWM LOW

P2->OUT &= ~0x40;

SysTick\_Wait1us(10000-duty);

counter--;

}

Motor\_StopSimple();

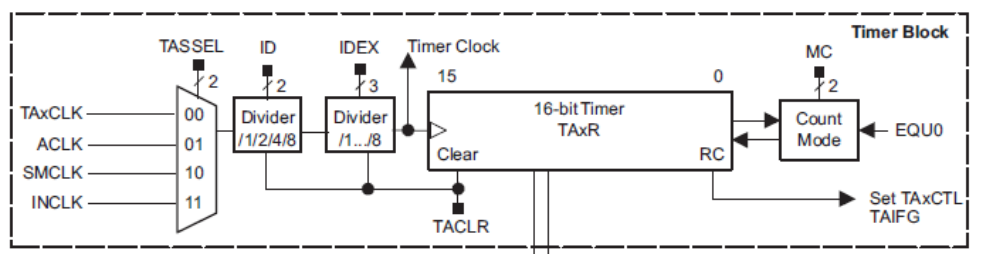
}

**Total time is passed into the counter variable. The variable duty is passed into the function SysTick\_Wait1us(). In the while loop, the motor is turned ON and is allowed to move for a time = duty using the SysTick\_Wait1us(duty) function. Then the motor is switched off for the remaining time (10ms-duty). Following this, the counter is decremented by one.**

**Thus every 10ms, the motor is switched on for duty µs and then switched off for the remainder of the time.**

1. Section 6.5. Reference to PWM\_Init1() in PWM.c, what is the base clock used to increment the counters in Timer\_A0? Show the details of how this base clock of Timer\_A0 is derived.

**ANS. The base clock that is used to increment the counters in Timer\_A0 is derived from SMCLK whose frequency is 12 MHz. Base clock is controlled by ID and IDEX bits which are in the TAxCTL and TAxEX0 registers.**



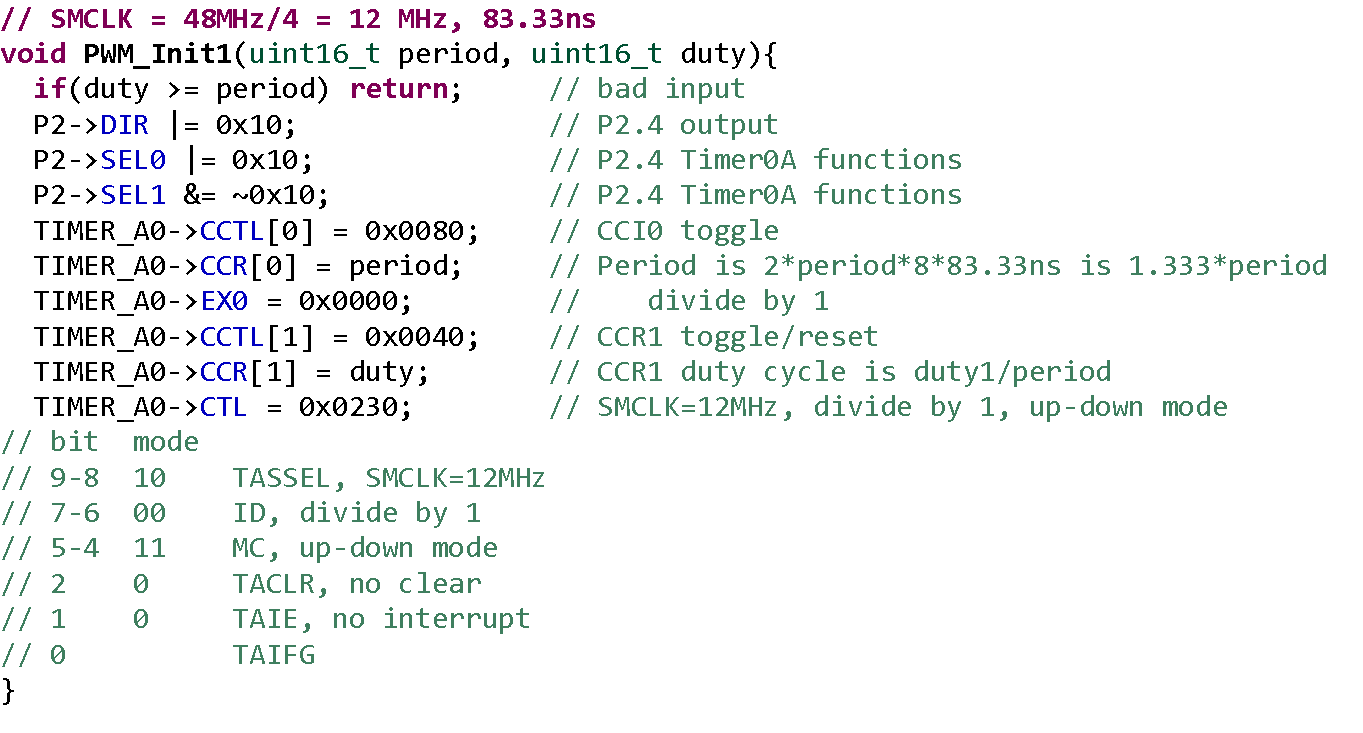
**Base clock = SMCLK x division ratio of ID x division ratio of IDEX**

So, from the code below,

division ratio of ID = 1 (from TIMER\_A0->CTL)

division ratio of IDEX = 1 (from TIMER\_A0->EX0)

**∴ Base clock = 12MHz\*1\*1 =12 MHz**



1. Section 6.5. What is the PWM frequency generated to the motor? illustrate with detail working.

**ANS. Here, we’re referencing the code for PWM\_Init1() from the previous question. Frequency of the base clock for PWM generation is 12MHz. So, every clock period = 1/12MHz = 83.33 ns, the timer counter value will increment by 1.** If we start counting from 0 to *period* where CCR[0] = *period* and then start to count down from there, we would need 2\*max\_count\*clock period for the PWM cycle to repeat.

As given in the comment, Period = 2 \* *period* \* 83.33 ns. = 166.67\**period*

**Frequency = 1 /(166.67\**period* ) x109 Hz**  where *period* is the value stored in TimerA0 -> CCR[0]

1. Section 6.5. Is the interrupt mechanism used in the PWM generation via Timers?

**ANS. No it is disabled because in TA0CTL bit 0 and 1 are both 0.**

1. Section 6.5. What is the IRQ number corresponding to the interrupt used by Timer\_A1 in Lab3\_TimerCompare\_Motor project use? What is the corresponding Exception number?

**ANS. IRQ number 10 is used. Since IRQ0 has exception no. 16, IRQ 10 has exception no. 26.**

NVIC->ISER[0] = 0x00000400