SC2107 Lab3 Assignment Sheet (to be submitted to NTULearn before next lab)

Name: Saravanakaleeswaran Arun Karthick Lab Group:SCEC Date: 25/09/2024

1. Section 6. Exception handling in ARM processor is handle at three levels (Global, NVIC, and Peripheral). Which are the two registers that configure the exception handling at a global level? Explain whether we need to configure these registers in our lab exercise.

**Answer Q1 Section 6:** Registers such as PRIMASK (Priority mask register) and BASEPRI (Base priority mask register) are some of the registers that control the registers at the global level. Fortunately, both these registers’ default value upon reset allows interrupts to be triggered, hence they are not initialized for our labs

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?

**Answer Q2 Section 6.2:**

|  |  |
| --- | --- |
|  |  |

The internal pull up register is enabled since there is no external resistor. The switch will report a logic of 1 or 0, where logic 1 means that no obstacles are encountered and logic 0 means some obstacles are encountered which is triggered when it hits an obstacle and closes the switch. This is how bump switches allow us to know if the robot encounters any obstacles.

1. Section 6.3. Write down the GPIO configuration used for pins connected to the Bump switches IF they are connected to Pin6.0 to P6.5.

**Answer Q3 Section 6.3:**

P6->SEL0 &= ~0x3F; // Clear the bits for P6.0 to P6.5 (set to GPIO mode)

P6->SEL1 &= ~0x3F; // Clear the bits for P6.0 to P6.5 (set to GPIO mode)

P6->DIR &= ~0x3F; // Set P6.0 to P6.5 as inputs

P6->REN |= 0x3F;

P6->OUT |= 0x3F; // Enable pull up resistor

The first 2 lines clear the bits for P6.0 to P6.5 and set to GPIO mode. The 3rd line sets P6.0 to P6.5 as inputs and the last 2 lines are to enable the pullup resistors on those pins P6.0 to P6.5.

1. Section 6.3. What is the frequency of the clock source of systick timer? Explain how systick timer is configured to interrupt the system at 1000Hz frequency. Illustrate with detail calculations and APIs used.

**Answer Q4 Section 6.3:**

The SysTick timer uses the MCLK (master clock) at 48Mhz frequency.

MCLK = 48Mz = 48 000 000Hz

Desired interrupt frequency = 1000Hz

The reload value is set to **N - 1** because the timer counts down from the reload value to 0, making it trigger after exactly **N** ticks.

Reload value =

= – 1

= 48 000 – 1

= 47 999

Reload value for SysTick is set to be 47999 for a 1000 Hz interrupt rate.

void SysTick\_Init(void) {

// SysTick Reload Value Register with 47,999

SysTick->LOAD = 47999; // (48 MHz / 1000) - 1 = 47999

// Clear the current SysTick Value Register

SysTick->VAL = 0;

// Select the clock source (system clock), enable SysTick and its interrupt

SysTick->CTRL = SysTick\_CTRL\_CLKSOURCE\_Msk | // Use system clock (48 MHz) SysTick\_CTRL\_TICKINT\_Msk | // Enable SysTick interrupt SysTick\_CTRL\_ENABLE\_Msk; // Enable SysTick timer

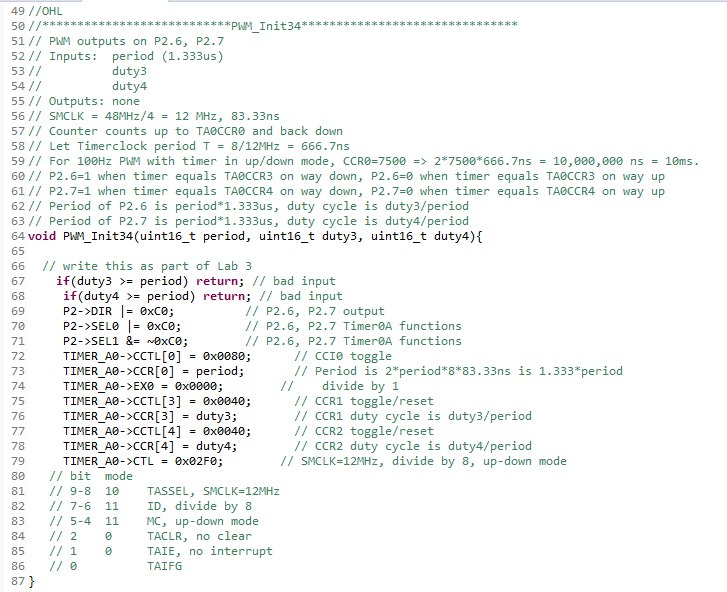
}

1. Section 6.3. What is the advantage the method of reading Reflectance sensor (in Lab3 section 6.3) has compared to the method used in Lab2?

**Answer Q5 Section 6.3:** In lab 2, we used a spin loop to implement the time delay tb before the sampling the capacitor discharge status. Spin loops are very inefficient as it hogs the CPU. In Lab3 we implement this delay using the systick interrupt based operation. In this way, CPU is using its resources more efficiently.

1. Section 6.4. Reference to PWM\_Init34() in PWM.c, what is the timer base clock used to increment the counters in Timer\_A0? Show the details of how this base clock of Timer\_A0 is derived, starting from processor clock. Note that SMCLK=12Mhz.

**Answer Q6 Section 6.4:**

* 

Here the SMCLK is configured to be 48Mhz/4 = 12Mhz. This clock will be used by the timer\_A0 to generate pulse width modulation (PWM) signals. The timer\_A0 is in up-down mode, so the counter counts up to TA0CCR0 and back down. Bits 7&6 corresponds to the clock division and the binary value 11 corresponds to a divide-by-8 operation. So, the timer clock frequency will be divided by 8. Thus, the timer base clock used to increment the counters in Timer\_A0 is:

Timer base clock =

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

**Answer Q7 Section 6.5:**

Timer clock frequency =

Timer count duration = ≈ 666.67ns

PWM frequency =

=

= 100Hz

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

**Answer Q8 Section 6.5:** No, interrupt mechanism isn’t explicitly used in the PWM generation.

1. Section 6.5. What is the IRQ number you need to reference to if Timer\_A2 instead of Timer A1 is used in Lab3\_TimerCompare\_Motor project? What is the corresponding Exception number?

**Answer Q9 Section 6.5:** The first 16 exception numbers (0-15) are reserved for system exceptions. The external exceptions start from 16th onwards. Thus, based on the interrupt vector table, the IQR ber for Timer\_A2 is 12 for CCR0 interrupt