CE2007 Lab3 Assignment Sheet (to be submitted to NTULearn before next lab)

Name: Do Anh Tu Lab Group:SEP1 Date: 19/10/2019

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…

We need to take note of these two registers: PRIMASK and BASEPRI

PRIMASK – Priority Mask Register is used to enable/disable all interrupts and exceptions globally (excepts NMI and hard fault). This disable interrupt indiscriminately, regardless of their priority level.

BASEPRI – Base Priority Mask Register is used to selectively disable the interrupts based on a base priority level. This BASEPRI has 8 bit value that support 255 priorities and will disable any interrupts with priority level equal or less than the set value.

Note that in Cortex M4F, the bigger priority number, the lower the priority is. E.g. Priority 0 > priority 1. By default, these two registers value are 0x0 (no interrupts are disabled and no effect from BASEPRI). For a more robust prioritising of interrupts, better to use BASEPRI.

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?

A close up of text on a black background

Description automatically generated

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

Pin 3 of the corresponding 6 bump switches are connected to pin 4.1, 4.2, 4.3, 4.5, 4.6, 4.7 respectively.

P4->SEL0 &= ~0xED;

P4->SEL1 &= ~0xED; //init GPIO

P4->DIR &= ~0xED; //init the 6 above pins as input

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

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

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

Refer to main code Lab3\_sensormain.c

Systick time derives it’s source from the CPU clock, which is kept at 48MHz in this code by the API call ClockInit48MHz(). Details of clock init is found in clock.c

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

Number of times the systick handler is called in a seconds is the frequency/ amount of time that the period in the above API SystickInit is counted to 0. Thus that would be 48MHz / 48000 = 1000 times.

So, it is 1000 hz for the interrupt.

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.

The definition of the above function:



Time (in 10 ms) is parsed in to become a counter variable.

The duty is a micro-second unit value is also parsed into the function.

In the main while loop, the motors will be enabled to go forward for a period = duty microseconds. This duration is achieved by setting the motors to on and wait for a period = duty using the SysTick\_Wait1us(duty).

Then they will be turned off for the remaining amount of time of 10 ms (10 ms – duty microsecs)

After that, the counter is decreased and the same enable/disable of motors happens again in an interval of 10 ms (hence the restriction of time duration to be 10 ms).

In short, in every 10 ms interval, the motors are on for duty microsecs and off for the remaining time. This is then repeated by the counter ‘time’ to replicate the duration of time x 10 ms of the motor working at the duty cycle given.

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.

Base clock = SMCLK \* div ratio ID \* div ratio IDEX.

A screenshot of a video game

Description automatically generated

From the code:



SMCLK = 48M/4 = 12MHz (in comment)

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

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

Hence base clock is 12 MHZ

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

Refer to PWM\_Init1() from the above section. We are not discussing about PWM generated to both motor, only to 1 motor and using the above function which has:.

Frequency of base clock for PWM generation is 12MHz. This means that every clock period = 1/12MHz = 83.33 ns, the timer counter value will count by 1.

Hence if we start counting from 0 to XXX where CCR[0] = XXX and then start to count down from then, we would need 2\*max\_count\*clock period for a cycle of PWM to repeat.

Or 2 \* XXX \* 83.33 ns.

Then frequency = 1 /( 2\*XXX\*83.33ns) where XXX is the value stored in TimerA0 -> CCR[0]

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

No Interrupts used in Timer A0

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?

From NVIC Table (p 118 MSP432 Datasheet):

TimerA1 interrupt is interrupt input 10 (since we are dealing only with TA1CCTL0.CCIFG).

Hence IRQ number 10 is used and need configuration via ISER0 register, which corresponds to exception number 16 + 10 = 26. (as External interrupts start with offset 16)