Barak Barclay

Jeffery Heckey

ECE3400-004

HW7

1.)

AREA |.text|, CODE, READONLY, ALIGN=2

THUMB

INCLUDE "msp432p401r.s"

Stack\_Size EQU 0x00000200

AREA STACK, NOINIT, READWRITE, ALIGN=3

Stack\_Mem SPACE Stack\_Size

\_\_initial\_sp

; <h> Heap Configuration

; <o> Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>

; </h>

Heap\_Size EQU 0x00000000

AREA HEAP, NOINIT, READWRITE, ALIGN=3

\_\_heap\_base

Heap\_Mem SPACE Heap\_Size

\_\_heap\_limit

PRESERVE8

THUMB

; Vector Table Mapped to Address 0 at Reset

AREA RESET, DATA, READONLY

EXPORT \_\_Vectors

EXPORT \_\_Vectors\_End

EXPORT \_\_Vectors\_Size

\_\_Vectors DCD \_\_initial\_sp ; Top of Stack

DCD main ; Run main

DCD NMI\_Handler ; NMI Handler

DCD HardFault\_Handler ; Hard Fault Handler

DCD MemManage\_Handler ; MPU Fault Handler

DCD BusFault\_Handler ; Bus Fault Handler

DCD UsageFault\_Handler ; Usage Fault Handler

DCD 0 ; Reserved

DCD 0 ; Reserved

DCD 0 ; Reserved

DCD 0 ; Reserved

DCD SVC\_Handler ; SVCall Handler

DCD DebugMon\_Handler ; Debug Monitor Handler

DCD 0 ; Reserved

DCD PendSV\_Handler ; PendSV Handler

DCD SysTick\_Handler ; SysTick Handler

; External Interrupts

DCD TA1FG\_Handler

DCD TA0CCR1\_Handler

DCD PCM\_IRQHandler ; 2: PCM Interrupt

DCD WDT\_A\_IRQHandler ; 3: WDT\_A Interrupt

DCD FPU\_IRQHandler ; 4: FPU Interrupt

DCD FLCTL\_IRQHandler ; 5: FLCTL Interrupt

DCD COMP\_E0\_IRQHandler ; 6: COMP\_E0 Interrupt

DCD COMP\_E1\_IRQHandler ; 7: COMP\_E1 Interrupt

DCD TA0\_0\_IRQHandler ; 8: TA0\_0 Interrupt

DCD TA0\_N\_IRQHandler ; 9: TA0\_N Interrupt

DCD TA1\_0\_IRQHandler ; 10: TA1\_0 Interrupt

DCD TA1\_N\_IRQHandler ; 11: TA1\_N Interrupt

DCD TA2\_0\_IRQHandler ; 12: TA2\_0 Interrupt

DCD TA2\_N\_IRQHandler ; 13: TA2\_N Interrupt

DCD TA3\_0\_IRQHandler ; 14: TA3\_0 Interrupt

DCD TA3\_N\_IRQHandler ; 15: TA3\_N Interrupt

DCD EUSCIA0\_IRQHandler ; 16: EUSCIA0 Interrupt

DCD EUSCIA1\_IRQHandler ; 17: EUSCIA1 Interrupt

DCD EUSCIA2\_IRQHandler ; 18: EUSCIA2 Interrupt

DCD EUSCIA3\_IRQHandler ; 19: EUSCIA3 Interrupt

DCD EUSCIB0\_IRQHandler ; 20: EUSCIB0 Interrupt

DCD EUSCIB1\_IRQHandler ; 21: EUSCIB1 Interrupt

DCD EUSCIB2\_IRQHandler ; 22: EUSCIB2 Interrupt

DCD EUSCIB3\_IRQHandler ; 23: EUSCIB3 Interrupt

DCD ADC14\_IRQHandler ; 24: ADC14 Interrupt

DCD T32\_INT1\_IRQHandler ; 25: T32\_INT1 Interrupt

DCD T32\_INT2\_IRQHandler ; 26: T32\_INT2 Interrupt

DCD T32\_INTC\_IRQHandler ; 27: T32\_INTC Interrupt

DCD AES256\_IRQHandler ; 28: AES256 Interrupt

DCD RTC\_C\_IRQHandler ; 29: RTC\_C Interrupt

DCD DMA\_ERR\_IRQHandler ; 30: DMA\_ERR Interrupt

DCD DMA\_INT3\_IRQHandler ; 31: DMA\_INT3 Interrupt

DCD DMA\_INT2\_IRQHandler ; 32: DMA\_INT2 Interrupt

DCD DMA\_INT1\_IRQHandler ; 33: DMA\_INT1 Interrupt

DCD DMA\_INT0\_IRQHandler ; 34: DMA\_INT0 Interrupt

DCD PORT1\_IRQHandler ; 35: PORT1 Interrupt

DCD PORT2\_IRQHandler ; 36: PORT2 Interrupt

DCD PORT3\_IRQHandler ; 37: PORT3 Interrupt

DCD PORT4\_IRQHandler ; 38: PORT4 Interrupt

DCD PORT5\_IRQHandler ; 39: PORT5 Interrupt

DCD PORT6\_IRQHandler ; 40: PORT6 Interrupt

DCD 0 ; 41: Reserved

DCD 0 ; 42: Reserved

DCD 0 ; 43: Reserved

DCD 0 ; 44: Reserved

DCD 0 ; 45: Reserved

DCD 0 ; 46: Reserved

DCD 0 ; 47: Reserved

DCD 0 ; 48: Reserved

DCD 0 ; 49: Reserved

DCD 0 ; 50: Reserved

DCD 0 ; 51: Reserved

DCD 0 ; 52: Reserved

DCD 0 ; 53: Reserved

DCD 0 ; 54: Reserved

DCD 0 ; 55: Reserved

DCD 0 ; 56: Reserved

DCD 0 ; 57: Reserved

DCD 0 ; 58: Reserved

DCD 0 ; 59: Reserved

DCD 0 ; 60: Reserved

DCD 0 ; 61: Reserved

DCD 0 ; 62: Reserved

DCD 0 ; 63: Reserved

DCD 0 ; 64: Reserved

\_\_Vectors\_End

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;The vector table above would need to be changed to include the handlers that are needed for this ;program. Two have been included above.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

\_\_Vectors\_Size EQU \_\_Vectors\_End - \_\_Vectors

AREA |.text|, CODE, READONLY

; Dummy Exception Handlers (infinite loops which can be modified)

NMI\_Handler PROC

EXPORT NMI\_Handler [WEAK]

B .

ENDP

HardFault\_Handler\

PROC

EXPORT HardFault\_Handler [WEAK]

B .

ENDP

MemManage\_Handler\

PROC

EXPORT MemManage\_Handler [WEAK]

B .

ENDP

BusFault\_Handler\

PROC

EXPORT BusFault\_Handler [WEAK]

B .

ENDP

UsageFault\_Handler\

PROC

EXPORT UsageFault\_Handler [WEAK]

B .

ENDP

SVC\_Handler PROC

EXPORT SVC\_Handler [WEAK]

B .

ENDP

DebugMon\_Handler\

PROC

EXPORT DebugMon\_Handler [WEAK]

B .

ENDP

PendSV\_Handler PROC

EXPORT PendSV\_Handler [WEAK]

B .

ENDP

SysTick\_Handler PROC

EXPORT SysTick\_Handler [WEAK]

B .

ENDP

Default\_Handler PROC

EXPORT PSS\_IRQHandler [WEAK]

EXPORT CS\_IRQHandler [WEAK]

EXPORT PCM\_IRQHandler [WEAK]

EXPORT WDT\_A\_IRQHandler [WEAK]

EXPORT FPU\_IRQHandler [WEAK]

EXPORT FLCTL\_IRQHandler [WEAK]

EXPORT COMP\_E0\_IRQHandler [WEAK]

EXPORT COMP\_E1\_IRQHandler [WEAK]

EXPORT TA0\_0\_IRQHandler [WEAK]

EXPORT TA0\_N\_IRQHandler [WEAK]

EXPORT TA1\_0\_IRQHandler [WEAK]

EXPORT TA1\_N\_IRQHandler [WEAK]

EXPORT TA2\_0\_IRQHandler [WEAK]

EXPORT TA2\_N\_IRQHandler [WEAK]

EXPORT TA3\_0\_IRQHandler [WEAK]

EXPORT TA3\_N\_IRQHandler [WEAK]

EXPORT EUSCIA0\_IRQHandler [WEAK]

EXPORT EUSCIA1\_IRQHandler [WEAK]

EXPORT EUSCIA2\_IRQHandler [WEAK]

EXPORT EUSCIA3\_IRQHandler [WEAK]

EXPORT EUSCIB0\_IRQHandler [WEAK]

EXPORT EUSCIB1\_IRQHandler [WEAK]

EXPORT EUSCIB2\_IRQHandler [WEAK]

EXPORT EUSCIB3\_IRQHandler [WEAK]

EXPORT ADC14\_IRQHandler [WEAK]

EXPORT T32\_INT1\_IRQHandler [WEAK]

EXPORT T32\_INT2\_IRQHandler [WEAK]

EXPORT T32\_INTC\_IRQHandler [WEAK]

EXPORT AES256\_IRQHandler [WEAK]

EXPORT RTC\_C\_IRQHandler [WEAK]

EXPORT DMA\_ERR\_IRQHandler [WEAK]

EXPORT DMA\_INT3\_IRQHandler [WEAK]

EXPORT DMA\_INT2\_IRQHandler [WEAK]

EXPORT DMA\_INT1\_IRQHandler [WEAK]

EXPORT DMA\_INT0\_IRQHandler [WEAK]

EXPORT PORT1\_IRQHandler [WEAK]

EXPORT PORT2\_IRQHandler [WEAK]

EXPORT PORT3\_IRQHandler [WEAK]

EXPORT PORT4\_IRQHandler [WEAK]

EXPORT PORT5\_IRQHandler [WEAK]

EXPORT PORT6\_IRQHandler [WEAK]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Code to set P2.4 as an input and P5.6 as an output would need to set PxIN, PxOUT, PxDIR, and PxSEL to ;configure them. However, I don’t have enough time to look them up in the data sheet to figure out ;what bits I need to set.

;example of changes in bit 5 to do an output

;PxDIR = xx1x\_xxxx

;PxSEL0 = xx1x\_xxxx

;PxSEL1 = xx0x\_xxxx

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;INPUT CAPTURE PM\_TA0.1;P2.4 Timer0\_A1

;each time a edge is detected, perform an isr to say a copy of TA0CCRx; check switch values in isr; ;calculate values. switches P2.0 & P2.1

LDRH R0, =TA0CTL

LDRH R1, [R0]

ORR R1, 0x20 ; to set timer in continuous mode

STRH R1, [R0]

LDRH R0, =TA0CCTL1

LDRH R1, [R0]

ORR R1, 0xC100 ; to set timer to capture on both rising and falling edge

STRH R1, [R0]

;LDR R0, =TACCR1

;LDRH R1, [R0]

;LDR R0, =PrevCap

;LDRH R2, [R0]

;SUB R3,R1,R2

;STRH R1, [R0]

;LDR R0, =period

;STRH R3, [R0]

;LDR R0, =TA0CCTL1

;...

;BX LR

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;OUTPUT COMPARE TA2.1;P5.6 Timer2\_A1

;use two TA2CCRx registers to control freq and duty cycle

LDRH R0, =TA2CTL

LDRH R1, [R0]

ORR R1, 0x10 ; to set timer in up mode

STRH R1, [R0]

LDRH R0, =TA0CCTL1

LDRH R1, [R0]

ORR R1, 0x0000 ; to set timer to compare and change output mode

STRH R1, [R0]

;LDR R0, =P2DIR

;LDRB R1, [R0]

;ORR R1, #0x8

;STRB R1, [R0]

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

TA1FG\_Handler LDR R0, =TA1CCTL

LDR R1, =RollCount

LDR R2, [R1]

ADD R2, #0x10000

STR R2, [R1]

LDRH R2, [R0]

BIC R2, #TA1FG

BX LR

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

TA0CCR1\_Handler LDR R0, =TA0CCR1

LDR R1, =RollCount

LDRH R2, [R0]; CURRENT TIME 0x0005

LDR R3, [R1]; ROUNTCOUNT 0x3

;LSL R3, [R1]

ADD R2, R3; CURRENT TIME 0x3\_0005

LDR R1, =PrevTime

LDR R3, [R1];0x0000\_FFF0

SUB R2, R3;CURRENT MINUS PREVIOUS

2.)

a. When the input frequency becomes too high, the time between consecutive interrupts becomes too short for the CPU to keep up. Even if the ISR was instantaneous, eventually the two edges would capture the same TAR value since TAR is not counting forward quickly enough. A longer ISR will further limit the highest frequency that can be measured. Shorter ISRs will allow the system to measure faster frequencies.

b. In this situation, the 16-bit TAR register will overflow more than once. Since we are not accounting for multiple timer (TAR) overflow events, the actual frequency would be measured higher than what it really is. The limited precision of the TAR register and the capture registers puts a floor on the slowest frequency we can measure.

c. The limit is due to the limited resolution of the 16-bit TAR timer. At some point, the calculated period becomes 0.

d. In this situation, the calculated values of Ht and/or Lt would exceed 16-bit precision (requiring a wider register). Since we are bound to 16-bit limitations, this puts a floor on the lowest frequency we can generate.

e. This puts an inherent limit on the maximum frequency that can be measured via input capture without encountering capture overflow (a second capture to TA0CCR0 before the previous capture was copied).