Shuaib Shameem
ECE 367 - Spring 2012
March 16, 2012

Assume standard program setup (like that given to us for experiment 1) for all code below.

1)
a)

```
L0: LDAA #$01 ; Load AcmA with 00000001
STAA $240     ; Set PortT0 high
JSR Fivemsec_Delay ; Delay 5mSec
LDAA #$00 ; Load AcmA with 00000000
STAA $240 ; Set PortT0 low
JSR Fivemsec_Delay ; Delay 5mSec
BRA L0 ; Loop
Fivemsec_Delay:
LDAA #5       ; Outer Loop - 1 Clock Cycle
A1: LDY #3000 ; Inner Loop - 2 Clock Cycles
A0: LBRN A0   ; 3 Clock Cycles
DEY                 ; 1 Clock Cycle
LBNE A0       ; 4 Clock Cycles
DECA                ; 1 Clock Cycle
BNE A1        ; 3 Clock Cycles
RTS; [(8*3000 + 2 + 1 + 3) * 5]/24000000 = ~5mSec
```

b)

```
L0: LDAA #$01  ;Load AcmA with 00000001
STAA $240            ;Set PortT0 High
JSR OnefiftyMicroSec_Delay   ; Delay .15mSec
LDAA #$00            ; Load AcmA with 00000000
STAA $240            ; Set PortT0 Low
JSR OnefiftyMicroSec_Delay   ; Delay .15mSec
BRA L0        ; Loop
OnefiftyMicroSec_Delay:
LDY #450      ; Loop Counter - 2 Cycles
A0: LBRN A0; 3 Clock Cycles
DEY           ; 1 Clock Cycles
LBNE A0; 4 Clock Cycles
RTS;(8*450+2)/24000000 = ~.15mSec
```

c)

```
L0: BSET $240, $01    ;Set PortT0 high
LDAA #1               ;Set Outer loop counter
JSR QuartersevenmSec_Delay   ;Delay 1.5mSec
BCLR $240,$01         ; Set PortT0 low
LDAA #3               ; Set Outer loop counter
JSR QuartersevenmSec_Delay   ;Delay 5.5mSec
BRA L0                ; Loop
QuartersevenmSec_Delay:
A1: LDY #5250         ; Inner Loop - 2 Cycles
A0: LBRN A0           ; 3 Clock Cycles
DEY                         ; 1 Clock Cycle
LBNE A0               ; 4 Clock Cycles
DECA                        ; 1 Clock Cycle
BNE A1                ; 3 Clock Cycles
RTS;(8*5250 +2+1+3) /24000000 = ~1.75mSec * value in AcmA
```

d)

```
L0: BSET $240,$01           ;Set PortT0 high
LDAA #3               ; Set outer loop counter
JSR QuarterthirtymSec_Delay  ; Delay 22.5mSec
BCLR $240,$01         ; Set PortT0 low
LDAA #1               ; Set outer loop counter
JSR QuarterthirtymSec_Delay  ; delay 7.5mSec
BRA L0                ;loop
QuarterthirtymSec_Delay:
```

```
A1: LDY #22500        ; inner loop - 2 cycles
A0: LBRN A0           ; 3 cycles
DEY                         ; 1 cycle
LBNE A0               ; 4 cycles
DECA                        ; 1 cycle
BNE A1                ; 3 cycles
RTS;(8*22500 +2+1+3)/24000000 = ~7.5mSec * value in AcmA

2)
a)
MOVB #$06, $4D ; set prescale to 64
MOVB #$01,$40 ; enable OC0 for output compare
MOVB #$90, $46 ; enable TCNT and fast flags clear
HERE: BSET $240, $01 ; set PT0 high
JSR delayby5ms
BCLR $240, $01; set PT0 low
JSR delayby5ms
BRA HERE
delayby5ms:
LDD $44 ; get current TCNT
ADDD #1875 ;add 1875
STD $50; store new tc0
MOVB #$01, $4E ; clear C0F flag
BRCLR $4E,$01,*
LDD TC0; clear flag
RTS

b)
MOVB #$06, $4D ; set prescale to 64
MOVB #$01,$40 ; enable OC0 for output compare
MOVB #$90, $46 ; enable TCNT and fast flags clear
HERE: BSET $240, $01 ; set PT0 high
JSR delayby150us
BCLR $240, $01; set PT0 low
JSR delayby150us
BRA HERE
delayby150us:
LDD $44 ; get current TCNT
ADDD #56 ;add 56
STD $50; store new tc0
MOVB #$01, $4E ; clear C0F flag
BRCLR $4E,$01,*
LDD TC0; clear flag
RTS

c)
MOVB #$06, $4D ; set prescale to 64
MOVB #$01,$40 ; enable OC0 for output compare
MOVB #$90, $46 ; enable TCNT and fast flags clear
HERE: BSET $240, $01 ; set PT0 high
JSR delaybyquarter7ms
BCLR $240, $01; set PT0 low
JSR delayby3quarter7ms
BRA HERE
delaybyquarter7ms:
LDD $44 ; get current TCNT
ADDD #656 ;add 656
STD $50; store new tc0
MOVB #$01, $4E ; clear C0F flag
BRCLR $4E,$01,*
LDD TC0; clear flag
RTS
delayby3quarter7ms:
LDD $44 ; get current TCNT
ADDD #1969 ;add 1969
STD $50; store new tc0
```

```
MOVB #$01, $4E ; clear C0F flag
BRCLR $4E,$01,*
LDD TC0; clear flag
RTS

d)
MOVB #$06, $4D ; set prescale to 64
MOVB #$01,$40 ; enable OC0 for output compare
MOVB #$90, $46 ; enable TCNT and fast flags clear
HERE: BSET $240, $01 ; set PT0 high
JSR delayby3quarter30ms
BCLR $240, $01; set PT0 low
JSR delaybyquarter30ms
BRA HERE
delaybyquarter30ms:
LDD $44 ; get current TCNT
ADDD #2812 ;add 2812
STD $50; store new tc0
MOVB #$01, $4E ; clear C0F flag
BRCLR $4E,$01,*
LDD TC0; clear flag
RTS
delayby3quarter30ms:
LDD $44 ; get current TCNT
ADDD #8438 ;add 8438
STD $50; store new tc0
MOVB #$01, $4E ; clear C0F flag
BRCLR $4E,$01,*
LDD TC0; clear flag
RTS

3)
All programs for this question have the proper vector (ORG$FFE0,FDB TC7_ISR)

a)
SEI ; Turn off interrupts
MOVB #$06, $4D ;set prescale to 64
MOVB #$80, $40 ;enable OC7 for Output compare
MOVB #$90, $46 ; enable TCNT & fast flags clear
BSET $4C, $80 ; Enable TC7 interrupt
LDD TCNT ;get current TCNT
ADDD #1875 ;increment TCNT count by 1875 and store in TC7
STD $5E;"successful compare" in 1875 clicks
MOVB #$80, $4E ; clear C0F
CLI ;Turn on interrupts
BSET $240, $01 ; Set PortT0 high
BRA * ; branch forever (only interrupt will execute)
TC7_ISR: LDD $5E      ; get last interrupt count (clear flag)
ADDD #1875     ; add 1875
STD $5E        ; store new interrupt count
LDAA $240              ; get PT bits
BITA #$01              ; test PT0
BEQ high               ; if PT0 is high, clear bit
BSET $240, $01 ; set PT0 high
RTI
high: BCLR $240, $01; set PT0 low
RTI

b)
SEI ; Turn off interrupts
MOVB #$06, $4D ;set prescale to 64
MOVB #$80, $40 ;enable OC7 for Output compare
MOVB #$90, $46 ; enable TCNT & fast flags clear
BSET $4C, $80 ; Enable TC7 interrupt
LDD TCNT ;get current TCNT
ADDD #56 ;increment TCNT count by 56 and store in TC7
```

```
STD $5E;"successful compare" in 56 clicks
MOVB #$80, $4E ; clear C0F
CLI ;Turn on interrupts
BSET $240, $01 ; Set PortT0 high
BRA * ; branch forever (only interrupt will execute)
TC7_ISR: LDD $5E      ; get last interrupt count (clear flag)
ADDD #56              ; add 56
STD $5E        ; store new interrupt count
LDAA $240             ; get PT bits
BITA #$01             ; test PT0
BEQ high              ; if PT0 is high, clear bit
BSET $240, $01 ; set PT0 high
RTI
high: BCLR $240, $01; set PT0 low
RTI

c)
SEI ; Turn off interrupts
MOVB #$06, $4D ;set prescale to 64
MOVB #$80, $40 ;enable OC7 for Output compare
MOVB #$90, $46 ; enable TCNT & fast flags clear
BSET $4C, $80 ; Enable TC7 interrupt
LDD TCNT ;get current TCNT
ADDD #656 ;increment TCNT count by 656 and store in TC7
STD $5E;"successful compare" in 656 clicks
MOVB #$80, $4E ; clear C0F
CLI ;Turn on interrupts
BSET $240, $01 ; Set PortT0 high
BRA * ; branch forever (only interrupt will execute)
TC7_ISR:
BITA #$01             ;test PT0
BEQ high              ;if PT0 is high, clear bit
LDD $5E        ; get last interrupt count
ADDD #656             ; add 656
STD $5E        ; store new interrupt count
BSET $240, $01 ;set PT0 high
RTI
high: LDD $5E  ;get last interrupt count
ADDD #1969     ; add 1969
STD $5E        ; store new interrupt count
BCLR $240, $01 ; set PT0 low
RTI

d)
SEI ; Turn off interrupts
MOVB #$06, $4D ;set prescale to 64
MOVB #$80, $40 ;enable OC7 for Output compare
MOVB #$90, $46 ; enable TCNT & fast flags clear
BSET $4C, $80 ; Enable TC7 interrupt
LDD TCNT ;get current TCNT
ADDD #8438 ;increment TCNT count by 8438 and store in TC7
STD $5E;"successful compare" in 8438 clicks
MOVB #$80, $4E ; clear C0F
CLI ;Turn on interrupts
BSET $240, $01 ; Set PortT0 high
BRA * ; branch forever (only interrupt will execute)
TC7_ISR:
BITA #$01             ;test PT0
BEQ high              ;if PT0 is high, clear bit
LDD $5E        ; get last interrupt count
ADDD #8438     ; add 8438
STD $5E        ; store new interrupt count
BSET $240, $01 ;set PT0 high
RTI
high: LDD $5E  ;get last interrupt count
ADDD #2812     ; add 2812
```

```
STD $5E        ; store new interrupt count
BCLR $240, $01 ; set PT0 low
RTI

4)
SEI ; Turn off interrupts
MOVB #$06, $4D ;set prescale to 64
MOVB #$81, $40 ;enable OC7/0 for Output compare
MOVB #$90, $46 ; enable TCNT & fast flags clear
BSET $4C, $81 ; Enable TC7/TC0 interrupt
LDD TCNT ;get current TCNT
ADDD #13 ;increment TCNT count by 13 and store in TC7
STD $5E;"successful compare" in 13 clicks
ADDD #100; increment TCNT count by 113 and store in TC0
STD $50;"successful compare" in 113 clicks
MOVB #$81, $4E ; clear C0F
CLI ;Turn on interrupts
BSET $240, $81 ; Set PortT7/0 high
BRA * ; branch forever (only interrupt will execute)
TC0_ISR:
BITA #$01              ;test PT0
BEQ high               ;if PT0 is high, clear bit
LDD $50        ; get last interrupt count
ADDD #113              ; add 113
STD $50        ; store new interrupt count
BSET $240, $01 ;set PT0 high
RTI
high: LDD $50  ;get last interrupt count
ADDD #75               ; add 75
STD $50        ; store new interrupt count
BCLR $240, $01 ; set PT0 low
RTI
TC7_ISR:
BITA #$01              ;test PT0
BEQ high               ;if PT0 is high, clear bit
LDD $5E        ; get last interrupt count
ADDD #13               ; add 113
STD $5E        ; store new interrupt count
BSET $240, $01 ;set PT0 high
RTI
high: LDD $5E  ;get last interrupt count
ADDD #26               ; add 75
STD $5E        ; store new interrupt count
BCLR $240, $01 ; set PT0 low
RTI
ORG $FFE0
FDB TC7_ISR
ORG $FFEE
FDB TC0_ISR
```