```
; Author: Shuaib Shameem
; UIN: 673551999
; ECE 367 -Microprocessor-Based Design
; Experiment 12 - Clock
; 04/19/2012
;
; Purpose: Design a clock program
;           It should toggle between 12hr/24hr formats
;           It should have a speed toggle for debugging
;           It should be settable multiple times

;PAY ATTENTION TO THE ALIGNMENT BELOW
;Labels start in the first column  (left most column = colunm 1)
;OP CODES are at column 9
;COMMENTS follow a ";" symbol
;Blank lines are allowed (Makes the code more readable)

; Define symbolic constants
PORTT  EQU  $240        ;Define Register Locations
PORTM   EQU  $250
PortT   EQU  $240
PortM   EQU  $250
DDRT   EQU  $242
DDRM    EQU  $252
INITRG EQU  $11
INITRM EQU  $10
CLKSEL EQU  $39
PLLCTL EQU  $3A
CRGFLG EQU  $37
SYNR   EQU  $34
REFDV   EQU  $35
COPCTL EQU  $3C
TSCR1   EQU  $46
TSCR2   EQU  $4D
TIOS   EQU  $40
TCNT   EQU  $44
TC0     EQU  $50
TFLG1   EQU  $4E
TC5     EQU  $5A
TIE     EQU  $4C
RS    EQU    $01    ; Register Select (RS) at PT0 (0 = command, 1= Data)
ENABLE EQU   $02     ; LCD ENABLE at PT1
RCK   EQU    $08    ; RCK connect to PT2
SPCR1   EQU   $00D8
SPCR2  EQU   $00D9
SPIB    EQU   $00DA
SPSR    EQU   $00DB
SPDR    EQU   $00DD
INITEE EQU   $0012
;
; The ORG statment below would normally be followed by variable definitions
; There are no variables needed for this project.
;
        ORG  $3800    ; Beginning of RAM for Variables
Form    DS.B 1
Speed   DS.B 1
HourTen DS.B 1
HourOne DS.B 1
MinTen  DS.B 1
MinOne  DS.B 1
sCNT    DS.B 1
mCNT    DS.B 1
Tset    DS.B 1
Init    DS.B 1
COUNT   DS.B 1
PRINT   DS.B 1
ColonOn DS.B 1
EnterOne DS.B 1
EnterTwo DS.B 1
EnterThree DS.B 1
EnterFour DS.B 1
;
```

```
; The main code begins here. Note the START Label
;
       ORG   $4000         ; Beginning of Flash EEPROM
START  LDS   #$3FCE     ; Top of the Stack
       SEI              ; Turn Off Interrupts
        MOVB  #$00, INITRG; I/O and Control Registers Start at $0000
        MOVB  #$39, INITRM  ; RAM ends at $3FFF
;
; We Need To Set Up The PLL So that the E-Clock = 24MHz
;
       BCLR  CLKSEL,$80      ; disengage PLL from system
       BSET  PLLCTL,$40      ; turn on PLL
       MOVB  #$2,SYNR        ; set PLL multiplier
       MOVB  #$0,REFDV       ; set PLL divider
       NOP                    ; No OP
       NOP                  ; NO OP
plp    BRCLR CRGFLG,$08,plp  ; while (!(crg.crgflg.bit.lock==1))
       BSET  CLKSEL,$80      ; engage PLL
;
;


       MOVB #$07, TSCR2    ; Prescale: 128
       MOVB #$20, TIOS
       MOVB #$90, TSCR1
       MOVB #$20, TIE
       LDD TCNT
       ADDD #INCREMENT
       STD TC5
       MOVB #$20, TFLG1

       LDAA  #$22
       STAA  SPIB           ; SPI clocks a 1/24 of E-Clock
        MOVB  #$3F, DDRM      ; Setup PortM data direction
;
; Setup for Master, enable, high speed SPI, and Built-in Timer
;
      LDAA  #$50
        STAA  SPCR1
      LDAA  #$00
      STAA  SPCR2

;
       CLI              ; Turn ON Interrupts




;
; Initialize the LCD Display
;
        LDAA  #00
        BSET  PortM, RCK    ; Set RCK to Idle HIGH

        JSR   InitLCD     ; Initialize the LCD

      LDAA #$FF
      STAA Tset       ; Tset = 00 -> Set Time Enabled
                      ; Tset = FF -> Set Time Disabled
      LDAA #$00
      STAA Init       ; Init = 00 -> Start-up Initialization
                      ; Init = FF -> Regular Operation
      STAA sCNT       ; Holds count for a second (4 interrupts = 1 sec)
      STAA mCNT       ; Holds count for a minute (60 increments = 240 interrupts = 1 min)
      STAA MinOne     ; Holds minute (ones digit)
      STAA MinTen     ; Holds minute (tens digit)
      STAA Form       ; Form = 00 -> 12hr Format
                      ; Form = FF -> 24hr Format
      STAA HourOne    ; Holds hour, starts at 12a
      STAA HourTen
      STAA Speed      ; Speed = 00 -> Regular Intervals
                      ; Speed = FF -> 120x Intervals
```

```
       STAA EnterOne   ; Boolean for number entry
       STAA EnterTwo   ; Boolean for number entry
       STAA EnterThree ; Boolean for number entry
       STAA EnterFour  ; Boolean for number entry
       STAA ColonOn    ; Boolean for colon omission
       STAA PRINT

       LDAA  #$0F      ; Make PortT Outbound on the lower 4 pins
       STAA  DDRT

;
; Initial Reset Location
;


Loop
        JSR GetKey       ; Continuously get key
        BRA Loop



;
; Update Display
;
UpDisp: BRCLR Form, $FF, AMPM  ;Check format (12/24)
        JSR ClearDisp          ; if 24, clear display
Back    LDAA HourTen           ; load tens digit of hour
        CMPA #$00              ; if 0, omit
        BEQ Jump               ; jump to omit
        JSR AconvP             ; else display
        BRA Next               ; jump to next digit
Jump    LDX #NoColon           ; load space
        JSR PrintString        ; print space
Next    LDAA HourOne           ; load ones digit of hour
        JSR AconvP             ; display
        BRSET ColonOn, $FF, Space   ; if no display colon, jump
        LDX #Colon             ; else load colon
        JSR PrintString        ; print coon
        BRA Skip               ; jump to min
Space   LDX #NoColon           ; load space
        JSR PrintString        ; print space
Skip    LDAA MinTen            ; load tens digit of min
        JSR AconvP             ; display
        LDAA MinOne            ; load ones digit of min
        JSR AconvP             ; display
        RTS                    ; return

AMPM                           ; if 12 hour mode
        JSR ClearDisp          ; clear display
        LDAA HourTen           ; load tens digit of hour
        CMPA #$00              ; if 0
        BEQ HourZero           ; branch to related method
        CMPA #$01              ; if 1
        BEQ HourNext           ; branch to related method
        CMPA #$02              ; if 2
        LBEQ HourTwo           ; branch to related method

HourZero                       ; if 0
        LDAB HourOne           ; load ones digit of hour
        CMPB #$00              ; compare to 0
        BEQ Midnight           ; if 0, midnight
        LDX #NoColon           ; otherwise load space
        JSR PrintString        ; print space
        LDAA HourOne           ; load ones digit
        JSR AconvP             ; print digit
Back2   BRSET ColonOn, $FF, Space2   ; check colon
        LDX #Colon             ; load colon
        JSR PrintString        ; print colon
        BRA Skip2              ; jump
Space2  LDX #NoColon           ; load space
        JSR PrintString        ; print space
Skip2   LDAA MinTen            ; load tens of min
```

```
        JSR AconvP              ; print
        LDAA MinOne             ; load ones of min
        JSR AconvP              ; print
        LDX #NotPM              ; load "am"
        JSR PrintString         ; print
        RTS                     ; return

Midnight                        ; if midnight
        LDAA #$01               ; load 1
        JSR AconvP              ; print
        LDAA #$02               ; load 2
        JSR AconvP              ; print
        BRA Back2               ; print rest of time

BackUp                          ; if am, but >9
        LDAA HourTen            ; load hour
        JSR AconvP              ; print
        LDAA HourOne            ; load hour
        JSR AconvP              ; print
        BRA Back2               ; print rest of time


HourNext                        ; if 1
        LDAB HourOne            ; load ones digit of hour
        CMPB #$2                ; compare to 2
        BLO BackUp              ; if less than, jump
        BEQ Noon                ; if equal, noon
        LDX #NoColon            ; load space
        JSR PrintString         ; print space
        LDAA HourOne            ; load hour
        DECA                    ; decrement
        DECA                    ; by 2
        JSR AconvP              ; print
Back3   BRSET ColonOn, $FF, Space3  ; check colon
        LDX #Colon              ; load colon
        JSR PrintString         ; print colon
        BRA Skip3               ; jump
Space3  LDX #NoColon            ; load space
        JSR PrintString         ; print space
Skip3   LDAA MinTen             ; load min
        JSR AconvP              ; print
        LDAA MinOne             ; load min
        JSR AconvP              ; print
        LDX #NotAM              ; load "pm"
        JSR PrintString         ; print
        RTS                     ; return

Noon                            ; if noon
        LDAA #$01               ; load 1
        JSR AconvP              ; print
        LDAA #$02               ; load 2
        JSR AconvP              ; print
        BRA Back3               ; print rest of time

HourTwo                         ; if 2
        LDAB HourOne            ; load ones hour
        CMPB #$02               ; compare to 2
        BLO Evening             ; if less than 2, evening
        LDAA #$01               ; load 1
        JSR AconvP              ; print
        LDAA HourOne            ; load hour
        DECA                    ; decrement by 2
        DECA                    ;
        JSR AconvP              ; print
        BRA Back3               ; print rest of time

Evening                         ; if evening
        LDX #NoColon            ; load space
        JSR PrintString         ; print space
        LDAA HourOne            ; load hour
        ADDA #$08               ; inc by 8
        JSR AconvP              ; print
```

```
        BRA Back3              ; print rest of time



;
; Initialize the LCD
;
InitLCD  JSR   delay3
      LDAA  #$30        ; Could be $38 too.
      JSR    Command
      JSR    delay3   ; need extra delay at startup
      LDAA  #$30      ; see data sheet. This is way
      JSR    Command    ; too much delay
      JSR    delay3
      LDAA  #$30
      JSR    Command
      LDAA  #$38       ; Use 8 - words (command or data) and
      JSR    Command      ; and both lines of the LCD
      LDAA  #$0C       ; Turn on the display
      JSR    Command
      LDAA  #$01       ; clear the display and put the cursor
      JSR    Command      ; in home position (DD RAM address 00)
      JSR   delay     ; clear command needs more time
      JSR    delay      ; to execute
      JSR    delay
      RTS
;
; Convert a hex to Ascii and Print the number
;
AconvP  LDAB  #$30          ; Load $30 on Accl B
        ABA                 ; Add A and B
      JSR   Print         ; Print Accl A
      RTS
;
; Print or Command
;
Print   BSET  PRINT, $FF
        JMP   spi_a
Command  BCLR  PRINT, $FF
spi_a: BRCLR SPSR,  $20,spi_a    ; Wait for register empty flag (SPIEF)
;    LDAB  SPDR         ; Read the SPI data register. This clears the flag automatically
        STAA  SPDR          ; Output command  via SPI to SIPO
CKFLG1  BRCLR SPSR,  $80, CKFLG1  ; Wait for SPI Flag
        LDAA  SPDR
        NOP                       ; Wait
        BCLR  PortM, RCK          ; Pulse RCK
        NOP
      NOP
      NOP
        BSET  PortM, RCK          ; Command now available for LCD
        BRCLR PRINT, $FF, ComL
        BSET  PortM, RS
        JMP   F1
ComL  BCLR  PortM, RS         ; RS = 0 for commands
F1     NOP
      NOP                    ; Probably do not need to wait
      NOP                    ; but we will, just in case ...
      BSET  PortM, ENABLE       ; Fire ENABLE
      NOP                    ; Maybe we will wait here too ...
      NOP
      NOP
      NOP
      BCLR  PortM, ENABLE     ; ENABLE off
      JSR   delay
      RTS
;
;  Blink the Display 4 times
;
BlinkDisp
        MOVB  #$04, COUNT       ; Initialize a counter
A4    LDAA  #$08            ; Turn off display but keep memory values
      JSR    Command
```

```
       JSR   delay3
       LDAA  #$0C            ; Turn on display. So, we Blinked!
       JSR    Command
       JSR    delay3
       DEC    COUNT
       BNE    A4            ; Blink 4 times
       RTS
;
; Clear the Display
;
ClearDisp
        LDAA  #$01             ; Clear the display and send cursor home
       JSR    Command
       JSR    delay           ; Clear needs more time so 3 delays
       JSR    delay
       JSR    delay
         RTS
;
; Print the String at the address loaded at X
;
PrintString
Loop7    LDAA  0,X         ; Load a character into ACMA
      BEQ    Done7             ; quit when if last character is $00
      JSR   Print            ; and output the character
      INX                     ; let's go get the next character
      BRA    Loop7
Done7   RTS
;
; Shift the second line to the left
;
ShiftSecondLine
        LDAA  #$C0          ; Jump to line 2
        JSR    Command
        LDAA  #$0C             ; Shift the Line to the left
        JSR   Command
        JSR   delay2           ; Delay it by some
        RTS

;
; We use the CPU clock cycles to create a delay

; Subroutine to delay the controller

delay   LDY    #8000        ; Command Delay routine. Way to long. Overkill!
A2:      DEY               ; But we do need to wait for the LCD controller
        BNE    A2         ; to do it's thing.  How much time is this
        RTS                 ; anyway? 2.5 msec

delay2 LDY    #$F000        ; Long Delay routine.  Adjust as needed.
      PSHA           ; Save ACMA (do we need to?)
A3:      LDAA  #$4A          ; Makes the delay even longer! (Nested loop.)
AB:    DECA
       BNE    AB           ;
       DEY
       BNE    A3           ;
       PULA          ; Get ACMA back
       RTS

delay3  LDAA  #$0F
AA6:  LDY    #$FFFF       ; Blink Delay routine.
A6:      DEY               ;
        BNE    A6
        DECA
        BNE    AA6          ;
        RTS

;
; Delay of about 1 Sec with the switching control
;
SDELAY: PSHY
        LDY #65535   ; Loop counter = 15000 - 2 clock cycles
A0:     LBRN A0      ; 3 clock cycles \
```

```
        DEY          ; 1 clock cycles | 8 clock cycles in loop
        LBNE A0      ; 4 clock cycles / Time = 8*<Y>/(24*10**6) + 2 =
;                    ; [8X15000 + 2]/24000000 ~= 5msec
        PULY
        RTS
;
; End of counter code

;
; Get the Value of the Key pressed
;
GetKey: LDAA #$0F          ; Load Acc. A with $F0
        STAA PORTT         ; Output high on all rows
        BRSET PORTT, $80, *; Check Column 1 for pressed key
        NOP
        BRSET PORTT, $40, *; Check Column 2 for pressed key
        NOP
        BRSET PORTT, $20, *; Check Column 3 for pressed key
        NOP
        BRSET PORTT, $10, *; Check Column 4 for pressed key
GKEY:   LDAA #$08          ; Once all keys are released, load Acc. A with $80
        STAA PORTT         ; Output high on row 1
        JSR SDELAY
        BRSET PORTT, $80, KEY1    ;If high, key 1 was pressed
        NOP
        BRSET PORTT, $40, KEY2    ;If high, key 2 was pressed
        NOP
        BRSET PORTT, $20, KEY3    ;If high, key 3 was pressed
        NOP
        BRSET PORTT, $10, KEYA    ;If high, key A was pressed
        LDAA #$04          ; No key press yet, load Acc. A with $40
        STAA PORTT         ; Output high on row 2
        JSR SDELAY
        BRSET PORTT, $80, KEY4    ;If high, key 4 was pressed
        NOP
        BRSET PORTT, $40, KEY5    ;If high, key 5 was pressed
        NOP
        BRSET PORTT, $20, KEY6    ;If high, key 6 was pressed
        NOP
        BRSET PORTT, $10, KEYB    ;If high, key B was pressed
        LDAA #$02          ; No key press yet, load Acc. A with $20
        STAA PORTT         ; Output high on row 3
        JSR SDELAY
        BRSET PORTT, $80, KEY7    ;If high, key 7 was pressed
        NOP
        BRSET PORTT, $40, KEY8    ;If high, key 8 was pressed
        NOP
        BRSET PORTT, $20, KEY9    ;If high, key 9 was pressed
        NOP
        ;BRSET PORTT, $10, KEYC    ;If high, key C was pressed
        LDAA #$01          ; No key press yet, load Acc. A with $10
        STAA PORTT         ; Output high on row 4
        JSR SDELAY
        BRSET PORTT, $80, KEY0    ;If high, key 0 was pressed
        NOP
        BRSET PORTT, $40, KEYF
        JSR UpDisp
        LBRA GKEY         ; No key press, check again

;
; Set of labels to set KEY to the pressed key's value
;            OR to branch to a relevant routine
;
KEY1:   LDAA #$01
        JSR KeyPress
        RTS                ; Return to GETKEY's calling routine

KEY2:   LDAA #$02       ; Set KEY to 2
        JSR KeyPress
        RTS                ; Return to GETKEY's calling routine

KEY3:   LDAA #$03       ; Set KEY to 3
```

```
        JSR KeyPress
        RTS                ; Return to GETKEY's calling routine

KEYA:   LDAA #$0A       ; Set KEY to A
        JSR KeyPress
        RTS

KEY4:   LDAA #$04       ; Set KEY to 4
        JSR KeyPress
        RTS                ; Return to GETKEY's calling routine

KEY5:   LDAA #$05       ; Set KEY to 5
        JSR KeyPress
        RTS                ; Return to GETKEY's calling routine

KEY6:   LDAA #$06       ; Set KEY to 6
        JSR KeyPress
        RTS                ; Return to GETKEY's calling routine

KEYB:   LDAA #$0B       ; Set KEY to B
        JSR KeyPress
        RTS

KEY7:   LDAA #$07       ; Set KEY to 7
        JSR KeyPress
        RTS                ; Return to GETKEY's calling routine

KEY8:   LDAA #$08       ; Set KEY to 8
        JSR KeyPress
        RTS                ; Return to GETKEY's calling routine

KEY9:   LDAA #$09       ; Set KEY to 9
        JSR KeyPress
        RTS                ; Return to GETKEY's calling routine

KEYC:   LDAA #$0C       ; Set KEY to C
        JSR KeyPress
        RTS

KEY0:   LDAA #$00       ; Set KEY to 0
        JSR KeyPress
        RTS                ; Return to GETKEY's calling routine

KEYF:   LDAA #$0F
        JSR KeyPress
        RTS


Flash:  LDAA  #$08            ; Turn off display but keep memory values
      JSR    Command
      JSR   delay3
      LDAA  #$0C            ; Turn on display. So, we Blinked!
      JSR    Command
      JSR    delay3
      MOVB #$FF, Init
      RTI

helpDONE
        RTI

ISR_TC5:LDD TC5                   ; load counter
        BRSET Speed, $FF, Fast    ; check speed
        ADDD INCREMENT
        BRA Store                 ; inc counter
Fast:   ADDD FASTEMENT
Store:  STD TC5                   ; store counter
        BRCLR Tset, $FF, helpDONE ; check for entry mode
        BRCLR Init, $FF, Flash    ; check for startup
        LDAA sCNT
        CMPA #$00                 ; keep track of interrupts
        BEQ Swap                  ; 3 = 1sec
        BRA Pass
```

```
Swap    BRSET ColonOn, $FF, Swit  ; toggle colon
        MOVB #$FF, ColonOn
        BRA Pass
Swit    MOVB #$00, ColonOn
Pass    LDAA sCNT
        INCA
        STAA sCNT
        CMPA #$03
        BNE DONE

        MOVB #$00, sCNT
        LDAA mCNT                 ; 60 mCNT = 1min
        INCA
        STAA mCNT
        CMPA #$3C
        BNE DONE

        MOVB #$00, mCNT
        LDAA MinOne               ; 60 min = 1hr
        INCA
        STAA MinOne
        CMPA #$0A
        BNE DONE

        MOVB #$00, MinOne
        LDAA MinTen
        INCA
        STAA MinTen
        CMPA #$06
        BNE DONE

        MOVB #$00, MinTen
        LDAA HourOne
        INCA
        STAA HourOne
        CMPA #$04
        BHS CheckHr
        CMPA #$0A
        BNE DONE

        MOVB #$00, HourOne
        LDAA HourTen
        INCA
        STAA HourTen
        BRA DONE

CheckHr:LDAA HourTen
        CMPA #$02               ; if hour = 24, reset to 00
        BNE DONE

        MOVB #$00, HourTen
        MOVB #$00, HourOne

DONE:   RTI



Colon   FCC ":"
        DC.B $00

NoColon FCC " "
        DC.B $00

NotPM   FCC "am"
        DC.B $00

NotAM   FCC "pm"
        DC.B $00

KeyPress
        CMPA #$0A     ; A -> Format
        BEQ Format
```

```
        CMPA #$0B     ; B -> Speed
        BEQ ChangeSpd
        CMPA #$0F     ; F -> Enter Time
        BEQ StartEnterTime
        BRSET Tset, $FF, NoEntry  ; Test for entry access
        BRA EnterNumber
NoEntry RTS                    ; return

EnterNumber
        BRSET EnterFour, $FF, EntryDone   ; if 4 # entered, no more entry
        BRSET EnterThree, $FF, EntryFour  ; if 3 # entered, enter ones min
        BRSET EnterTwo, $FF, EntryThree   ; if 2 # entered, enter tens min
        BRSET EnterOne, $FF, EntryTwo     ; if 1 # entered, enter ones hour
EntryOne                                  ; else enter tens hour
        CMPA #$02                         ; make sure between 0-2
        BHI Foul
        STAA HourTen                      ; store
        MOVB #$FF, EnterOne               ; toggle
Foul    RTS

EntryTwo
        STAA HourOne                      ; store
        MOVB #$FF, EnterTwo               ; toggle
        RTS

EntryThree
        CMPA #$06                         ;make sure between 0-6
        BHI Foul2
        STAA MinTen                        ; store
        MOVB #$FF, EnterThree             ; toggle
Foul2   RTS

EntryFour
        STAA MinOne                     ; store
        MOVB #$FF, EnterFour            ; toggle
EntryDone
        RTS

Format
        BRSET Form, $FF, Twelve         ; toggle format (12/24)
        MOVB #$FF, Form
        RTS
Twelve  MOVB #$00, Form
        RTS

ChangeSpd
        BRSET Speed, $FF, Slow          ; toggle speed (1x/120x)
        MOVB #$FF, Speed
        RTS
Slow    MOVB #$00, Speed
        RTS

StartEnterTime                          ; toggle run mode (run/enter time)
        BRSET Tset, $FF, AllowTime
        MOVB #$FF, Tset
        RTS
AllowTime
        MOVB #$00, Tset
        MOVB #$FF, Init
        RTS


        ORG $5000
INCREMENT: DC.W $EDA1 ; 45625 -> .25s at 128 prescale
FASTEMENT: DC.W $01FB ; 380 -> .25s/120 at 128 prescale

; Define TC5 Interrupt Vector

        ORG $FFE4
        FDB ISR_TC5
```