```
;
;           Shuaib Shameem
;             67355 1999
;
;              ECE  367
;         Final Exam Project
;           Due: 5/04/2012
;
;  The Code below is the compilation of 5 previous
;  lab experiments. The challenge of this was to make
;  all 5 usable while sticking to the original
;  requirements and devices.
;
;             Pin Usage
; PortAD 0-6  ->  7-seg LED Latches Data    (a|b|c|d|e|f|g)
; PortM 2-3   ->  7-seg LED Latches Enable  (Data latch: Right | Left)
; PortM 3-5   ->  SPI to SIPO for LCD       (RCK|MOSI|SCK)
; PortM 0-1   ->  LCD Control Lines         (Enable | RS)
; PortT 0-7   ->  4x4 Keypad Matrix
;

; Define symbolic constants
PORTT  EQU  $240         ; PortT Data
PORTM   EQU  $250         ; PortM Data
PortT   EQU  $240         ; PortT Data
PortM   EQU  $250         ; PortM Data
PORTAD  EQU  $270         ; PortAD Data
PortAD  EQU  $270         ; PortAD Data
DDRAD   EQU  $272         ; PortAD Direction
DDRT   EQU  $242         ; PortT Direction
DDRM    EQU  $252         ; PortM Direction
INITRG EQU  $11          ; Chip Initialization
INITRM EQU  $10          ; Chip Initialization
CLKSEL EQU  $39          ; Chip Initialization
PLLCTL EQU  $3A          ; Chip Initialization
CRGFLG EQU  $37          ; Chip Initialization
SYNR   EQU  $34          ; Chip Initialization
REFDV    EQU  $35          ; Chip Initialization
COPCTL EQU  $3C          ; Chip Initialization
TSCR1    EQU  $46          ; Timer System Control 1
TSCR2    EQU  $4D          ; Timer System Control 2
TIOS   EQU  $40          ; Timer Input Output Select
TCNT   EQU  $44          ; Timer Count
TC0      EQU  $50          ; Timer Capture 0
TFLG1    EQU  $4E          ; Timer Flags 1
TC5     EQU  $5A          ; Timer Capture 5
TIE     EQU  $4C          ; Timer Interrupt Enable
RS    EQU    $01         ; Register Select (RS) (0 = command, 1= Data)
ENABLE EQU   $02         ; LCD ENABLE
RCK   EQU    $08         ; RCK
SPCR1   EQU   $00D8       ; Serial Control 1
SPCR2  EQU   $00D9       ; Serial Control 2
SPIB    EQU   $00DA       ; Serial Baud Rate
SPSR    EQU   $00DB       ; Serial Status
SPDR    EQU   $00DD       ; Serial Data


        ORG  $3800      ; Beginning of RAM for Variables
;Program 3 Variables
ONES4   DS.W  1           ; Ones Counter
TENS4   DS.W  1           ; Tens Counter

;Program 4 Variables
COUNTP4 DS.W 1            ; Time Counter
FLAG1   DS.B 1            ; Run/Stop Flag
TIMETEN DS.W 1            ; Tens Counter
TIMEONE DS.W 1            ; Ones Counter

;Program 7 Variables
KEYP7   DS.B 1            ; Hold Key Press
PIN     DS.B 4            ; Hold Lock Pin
PINSET  DS.B 1            ; Boolean check of pin being set
```

```
INPUT   DS.B 4            ; Hold last 4 key presses
MODE    DS.W 1            ; Hold Current State of lock
OUT     DS.B 1            ; Hold arbitrary image state
SHOW    DS.W 1            ; Hold bottom LED image

;Program 9 Variables
COUNT DS.B 1             ; Delay Counter
PRINT DS.B 1             ; Print Variable
KEY     DS.B 1           ; - Upper 8-bits of the Variable \
KEY_1   DS.B 1           ; - Lower 8-bits of the Variable / 16-bits
GEN     DS.B 1           ; Bad Input Flag
NEGCH   DS.B 1           ; Negate Answer Flag
PORTMSK DS.B 1           ; Set/Clear Mask
DIGIT1  DS.B 1           ; Ones holder
DIGIT2  DS.B 1           ; Tens holder
SOLU    DS.B 1           ; - Upper 8-bits of the Variable \
SOLU_L  DS.B 1           ; - Lower 8-bits of the Variable / 16-bits
FANS    DS.B 1           ; Check Answer

;Program 12 Variables
Form    DS.B 1           ; 12/24hr flag
Speed   DS.B 1           ; 1x/120x flag
HourTen DS.B 1           ; Hour - Tens Digit
HourOne DS.B 1           ; Hour - Ones Digit
MinTen  DS.B 1           ; Minute - Tens Digit
MinOne  DS.B 1           ; Minute - Ones Digit
sCNT    DS.B 1           ; Count up to 1 sec
mCNT    DS.B 1           ; Count up to 1 min
Tset    DS.B 1           ; Time entry flag
Init    DS.B 1           ; Start-Up Initialization flag
ColonOn DS.B 1           ; Colon flag
EnterOne DS.B 1          ; One digit entered flag
EnterTwo DS.B 1          ; Two digits entered flag
EnterThree DS.B 1        ; Three digits entered flag
EnterFour DS.B 1         ; Four digits entered flag


; General Variables
Exp3    DS.B 1           ; Exp 3 running flag
Exp4    DS.B 1           ; Exp 4 running flag
Exp7    DS.B 1           ; Exp 7 running flag
Exp9    DS.B 1           ; Exp 9 running flag
Exp12   DS.B 1           ; Exp 12 running flag

;
; 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      ; Set Prescale to 128
       MOVB #$21, TIOS       ; Set TC5/TC0 to Output Compare
       MOVB #$90, TSCR1      ; Turn on Timer with FFlags
       MOVB #$00, TIE        ; Turn off interrupts
       MOVB #$21, TFLG1      ; Clear flags
       BCLR Exp3, $FF        ; Exp 3 not running
        BCLR Exp4, $FF        ; Exp 4 not running
```

```
        BCLR Exp7, $FF        ; Exp 7 not running
        BCLR Exp9, $FF        ; Exp 9 not running
        BCLR Exp12, $FF       ; Exp 12 not running




;
;
      CLI

RESTART:MOVB #$00, TIE        ; Reset Interrupts to off
        BCLR Exp3, $FF        ; Reset Exp 3 not running
        BCLR Exp4, $FF        ; Reset Exp 4 not running
        BCLR Exp7, $FF        ; Reset Exp 7 not running
        BCLR Exp9, $FF        ; Reset Exp 9 not running
        BCLR Exp12, $FF       ; Reset Exp 12 not running
        LDAA #$0F
        STAA DDRT             ; Set PortT to outbound on lower 4 pins (0-3)
        LDAA #$FF
        STAA DDRAD            ; Set PortAD to outbound on all pins (0-7)
        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            ; Enable SPI Master
        STAA  SPCR1
      LDAA  #$00            ; Keep extra features off
      STAA  SPCR2
      LDAA  #$90            ; Reinforce Time System Enable FFlags
      STAA  TSCR1

        LDAA  #00
        BSET  PortM, RCK      ; Set RCK to Idle HIGH
        JSR InitLCD           ; Initialize LCD

      LDX   #Intro1       ; Load base address of String1
        JSR   PrintString     ; Print String

      LDAA  #$C0            ; First line is done jump to line 2
        JSR    Command         ; Run Command

      LDX   #Intro2         ; Load base address of String2
        JSR   PrintString     ; Print String

        JSR    delay2         ; Let's display the message a while

        JSR   BlinkDisp       ; Blink the display 4 times

        JSR   ClearDisp       ; Clear the display

Redo  LDX   #Intro3         ; Load base address of String3
        JSR   PrintString     ; Print String

      LDAA  #$C0          ; First line is done jump to line 2
        JSR    Command         ; Run Command

      LDX   #Intro4         ; Load base address of String4
        JSR   PrintString     ; Print String

        JSR    delay2         ; Let's display the message a while

        JSR   ShiftSecondLine ; Shift the second line to left

        LDX   #Intro5         ; Load base address of String5
        JSR   PrintString     ; Print String

        JSR    delay2         ; Let's display the message a while

        JSR   ShiftSecondLine ; Shift cursor to second line
```

```
        LDX   #Intro6       ; Load base address of String6
        JSR   PrintString     ; Print String

         JSR    delay2        ; Let's display the message a while


        JSR   ShiftSecondLine ; Shift cursor to second line

        LDX   #Intro7       ; Load base address of String7
        JSR   PrintString     ; Print String

         JSR    delay2        ; Let's display the message a while


        JSR   ShiftSecondLine ; Shift cursor to second line

        LDX   #Intro8       ; Load base address of String8
        JSR   PrintString     ; Print String

       JSR GetKey            ; Check for key press
        LDAA KEY_1            ; Load key press data
        CMPA #$01             ; If 1
        LBEQ Prgm3            ; Load Exp 3
        CMPA #$02             ; If 2
        LBEQ Prgm4            ; Load Exp 4
        CMPA #$03             ; If 3
        LBEQ Prgm7            ; Load Exp 7
        CMPA #$04             ; If 4
        LBEQ Prgm9            ; Load Exp 9
        CMPA #$05             ; If 5
        LBEQ Prgm12           ; Load Exp 12

        JSR ClearDisp         ; Clear the display
        LDX #Error1           ; Load Error string 1
        JSR PrintString       ; Print the string

        LDAA #$C0             ; load enter command
        JSR Command           ; send command

        LDX #Error2           ; load error string 2
        JSR PrintString       ; print the string

        JSR delay2            ; delay

        BRA Redo              ; jump back to menu




;
;
;       Program 3 Code
; Runs a Count Up counter
; User input for Increment, Reset
;
;
Prgm3:  BSET Exp3, $FF        ; Exp 3 is running
        LDAA #$FF           ; Make PortAD Outbound
        STAA DDRAD           ; on all pins
        LDAA  #$0C           ; Make PortM outbound
        STAA  DDRM           ; on pins 2 and 3
        LDX  #$0000          ; Clear
        STX  ONES4           ; -Ones count
        STX  TENS4           ; -Tens count
        LDAA #$10            ; Turn off
        STAA SPCR1           ; -SPI
        LDAA #$00            ; Clear
        STAA PORTM           ; -PortM
        STAA PORTAD          ; -PortAD

; Resets Counter to 00
HERE3   LDAA TABLE           ; Load 0 (7-seg code)
```

```
        STAA  PORTAD          ; Store on PortAD
        BSET  PORTM, $0C      ; Enable both latches
        NOP                   ; wait
        NOP                   ; wait
        BCLR  PORTM, $0C      ; Disable both latches
        LDAA  #$00            ; Restore 00 in Acm.A

; Analyzes Input
INCRMNT3: JSR GetKey          ; Get key press
        LDAA KEY_1            ; load key press data
        CMPA #$0A             ; if A
        BEQ DOIT3             ; Increment counter
        CMPA #$0B             ; if B
        BEQ HERE3             ; Reset Counter
        CMPA #$0E             ; if E
        LBEQ RESTART          ; Jump back to menu
        BRA INCRMNT3          ; loop

; Increments Counter
DOIT3   LDX ONES4             ; Load ones counter
        INX                   ; increment
        STX ONES4             ; store ones counter
        LDX #$000A            ; if ones counter
        CPX ONES4             ; is not equal to A
        BNE UPDATE3           ; update display
        LDX #$00              ; else
        STX ONES4             ; reset ones to 0
        LDX TENS4             ; load tens counter
        INX                   ; increment
        STX TENS4             ; store tens counter
        LDX #$000A            ; if tens counter
        CPX TENS4             ; is not equal to A
        BNE UPDATE3           ; update display
        LDX #$0000            ; else
        STX TENS4             ; reset tens to 0
        BRA UPDATE3           ; update display

; Updates LED display
UPDATE3: BCLR PORTM, $FF      ; clear PortM
        LDX ONES4             ; load ones counter
        LDAA TABLE, X         ; load 7-seg code
        STAA PORTAD           ; store on PortAD
        BSET PORTM, $04       ; Enable Ones latch
        NOP                   ; wait
        NOP                   ; wait
        BCLR PORTM, $04       ; Disable Ones latch
        NOP                   ; wait
        NOP                   ; wait
        LDX TENS4             ; load tens counter
        LDAA TABLE, X         ; load 7-seg code
        STAA PORTAD           ; store on PortAD
        BSET PORTM, $08       ; Enable Tens latch
        NOP                   ; wait
        NOP                   ; wait
        BCLR PORTM, $08       ; Disable Tens latch
        NOP                   ; wait
        NOP                   ; wait
        JSR DELAY             ; delay
        JSR DELAY             ; delay
        LBRA INCRMNT3         ; get another key press
;
;
;       End Program 3 Code
;
;       Program 4 Code
; Simulates an NBA shot clock
; User input for start/stop, reset
;
;
Prgm4   MOVB #$00, FLAG1      ; set pause flag to STOP
      MOVW #$0003, TIMETEN  ; reset tens to 0
      MOVW #$0006, TIMEONE  ; reset ones to 0
```

```
       LDAA  #$FF            ; Make PortAD Outbound
       STAA  DDRAD           ; on all pins
       LDAA #$10             ; Turn off
        STAA SPCR1            ; -SPI
        LDAA #$00             ; Clear
        STAA PORTM            ; -PortM
        STAA PORTAD           ; -PortAD
        LDAA  #$0C            ; Make PortM pins 2 and 3 Outbound
       STAA  DDRM
       BSET Exp4, $FF        ; Exp 4 is running
       LDD TCNT              ; Load current timer count
        ADDD INCREMENT4       ; increment
        STD TC5               ; store on TC5
        MOVB #$20, TIE        ; enable interrupts
        MOVB #$20, TFLG1      ; clear flags
        LDD MAXCOUNT          ; load count value
        STD COUNTP4           ; store initial count value
;
; Analyze User Input
;

        JSR UPDATE4           ; Initialize Display
Rpt4    JSR GetKey            ; get key press
        LDAA KEY_1            ; load key press data
        CMPA #$0A             ; if A
        BEQ StartStop         ; toggle pause flag
        CMPA #$0B             ; if B
        BEQ SetTo24           ; reset to 24
        CMPA #$0E             ; if E
        LBEQ RESTART          ; jump back to menu
        BRA Rpt4              ; otherwise, repeat

; Toggles Pause Flag
StartStop
        BRCLR FLAG1, $FF, abc ; Complement FLAG1
        BCLR FLAG1, $FF
        BRA Rpt4
abc     BSET FLAG1, $FF
        BRA Rpt4              ; return

; Resets clock to 24
SetTo24
        MOVW #$0003, TIMETEN  ; Set Tens to 2
        MOVW #$0006, TIMEONE  ; Set Ones to 5
        JSR UPDATE4           ; Update display
        BRA Rpt4              ; return

; Checks current count
; if 0, it decrements the count and checks for 00
Check4: BRCLR FLAG1, $01, HERE4a ; If Paused, return
        LDD COUNTP4           ; Load count
        BNE HERE4a            ; If not 0, return
        JSR UPDATE4           ; If 0, updates display
        JSR DONEYET           ; Checks for 00
        LDD MAXCOUNT          ; Loads Initial Count value
        STD COUNTP4           ; Resets count
HERE4a  RTS                   ; Returns

; Decrements the count
; Updates the display
UPDATE4:LDY TIMEONE           ; Load Ones counter
        DEY            ; If 0, reset to A
        BEQ RESET1                   ; decrement
BACK0   STY TIMEONE           ; Store ones counter
        JSR ONES              ; Update display (Ones)
        JSR TEN               ; Update display (Tens)
        RTS                   ; return

; Sets IY to A and decrements Tens
RESET1: LDY TIMETEN           ; Load tens counter
        DEY                   ; decrement
        STY TIMETEN           ; store tens counter
```

```
        LDY #$000A          ; load IY with A
        BRA BACK0           ; go back to updating

; Checks current count for 00
DONEYET:LDY TIMETEN         ; load tens counter
        DEY
        BEQ DONEONE         ; if 0, check ones
BACK1:  RTS                 ; else return
DONEONE:LDY TIMEONE         ; load ones counter
        DEY
        BNE BACK1           ; if not 0, return
        JSR FLASH           ; else, flash

; Updates the Ones display
ONES:   LDY TIMEONE         ; Load Ones counter
        LDAA TABLEa, Y      ; Load 7-seg code
        STAA PORTAD         ; Store on PortAD
        BSET PORTM, $04     ; Enable the Ones latch
        NOP                 ; wait
        NOP                 ; wait
        BCLR PORTM, $04     ; Disable the Ones latch
        RTS                 ; return

; Updates the Tens display
TEN:    LDY TIMETEN         ; Load tens counter
        LDAA TABLEa, Y      ; load 7-seg code
        STAA PORTAD         ; Store on portAD
        BSET PORTM, $08     ; Enable Tens latch
        NOP                 ; wait
        NOP                 ; wait
        BCLR PORTM, $08     ; Disable Tens latch
        RTS                 ; return

; Flashes the display 3 times
FLASH   BCLR  PortAD, $FF   ; Clears out PortT
        JSR   OPENB         ; Opens both M1 and M2 for Latch Enable
        JSR   DELAY         ; Delays the program for one second

        BSET  PortAD, $3F   ; Sets the value $7E in portT
        JSR   OPENB         ; Opens both M1 and M2 for Latch Enable
        JSR   DELAY         ; Delays the program for one second

        BRA   FLASH         ; Branch to "flashy" for infinite loop

; Opens both ports
OPENB   BSET  PortM, $0C    ; Set Bits of PortM to 11
        NOP                 ; wait
        NOP                 ; wait
        BCLR  PortM, $FF    ; Clear all the bits of PortM
        NOP                 ; wait
        RTS                 ; return

;
;
;       End Program 4 Code
;
;       Program 7 Code
; Simulates a digital combination lock
; Can open, lock, change pin
;
;
Prgm7   BSET Exp7, $FF      ;_exp 7 is running
        LDD #$00            ; \
        STD PIN             ; Set PIN to 0000
        STD PIN+2           ;_/
        STD INPUT           ; Set INPUT to 0000
        STD INPUT+2         ;_/
        STAA PINSET         ; Set PINSET to FALSE
        STAA KEY            ; Set KEY to 00
        STAA OUT            ; Set OUT to 00
        STAA SHOW           ; Set SHOW to 00
        STAA SPIB           ; Set SPI Baud Rate to Eclock/2
```

```
        STAA MODE             ; Set MODE to LOCKED
        LDAA #$00             ;  /
        STAA MODE+1           ;_/
        LDAA #$12             ; Set SHOW to off
        STAA SHOW+1           ;_/
        LDAA #$0F             ; Set PortT direction to outbound
        STAA DDRT             ; on the bottom 4 pins
        LDAA #$FF             ; Set PortAD outbound
        STAA DDRAD            ; on all pins
        LDAA #$0C             ; Set PortM outbound
        STAA DDRM             ; on pins 2 and 3
        LDAA #$10             ; Disable the
        STAA SPCR1            ; SPI system
        LDAA #$00             ; Clear
        STAA PORTM            ; -PortM
        STAA PORTAD           ; -PortAD
        JSR DIGIT             ; Update Displays
        JSR STATUS            ;_/

;Analyzes User Input
L0:     JSR GetKey            ; Get Key press
        LDAA KEY_1            ; Load key press data
        CMPA #$0D             ; if D
        LBEQ PROGRAM          ; program lock
        BRCLR PINSET, $01, L0 ; Wait for Program to occur
        LDAA KEY_1            ; Load key
        CMPA #$09             ; If 0A-FF
        BHI BRNCH             ; Get another key (Related routine already ran)
        LDAA INPUT+1          ; Shift Input
        STAA INPUT            ;_/
        LDAA INPUT+2          ;_/
        STAA INPUT+1          ;_/
        LDAA INPUT+3          ;_/
        STAA INPUT+2          ;_/
        LDAA KEY_1            ; Add Key press to end of Input
        STAA INPUT+3          ;_/
        JSR ALT               ; Toggle Arbitrary Image
        BRA L0                ; Loop
BRNCH   LDAA KEY_1            ; Load key
        CMPA #$0A             ; if A
        LBEQ ENTER            ; Check input
        CMPA #$0B             ; if B
        LBEQ LOCK             ; lock
        CMPA #$0C             ; if C
        LBEQ CLEAR            ; clear input
        CMPA #$0E             ; if E
        LBEQ RESTART          ; jump to menu

        BRA L0                ; Loop

; Check the input
ENTER:  BRCLR PINSET, $01, RET0     ; If PIN hasn't been set, rts
        LDD PIN                     ; Load PIN0/1
        CPD INPUT                   ; Check against INPUT0/1
        BNE RET0                    ; If unequal, rts
        LDD PIN+2                   ; Load PIN2/3
        CPD INPUT+2                 ; Check against INPUT2/3
        BNE RET0                    ; If unequal, rts
        LDAA #$01                   ; Set MODE to OPEN
        STAA MODE+1                 ;_/
        LDAA #$00                   ;_/
        STAA MODE                   ;_/
        JSR STATUS                  ; Update STATUS Display
RET0:   LDD #$0000                  ; Clear INPUT
        STD INPUT                   ;_/
        STD INPUT+2                 ;_/
        BRA L0                      ; Return to calling routine

; Locks the lock
LOCK:   LDAA #$00                   ; Set MODE to LOCKED
        STAA MODE                   ;_/
        STAA MODE+1                 ;_/
```

```
        JSR STATUS              ;_Update STATUS Display
        LDD #$0000              ; \
        STD INPUT               ; Clear INPUT
        STD INPUT+2             ;_/
        LBRA L0                 ; Return to calling routine

; Checks pin and programs new code
PROGRAM:LDY #0002               ; Set MODE to PROGRAM
        STY MODE                ;_/
        JSR STATUS              ; Update STATUS Display
        JSR GetKey              ; Get a Key Press
        LDAA KEY_1              ;_/
        STAA INPUT              ; Store in INPUT
        STAA SHOW+1             ; Store in SHOW
        JSR DIGIT               ; Update Bottom Display
        JSR GetKey              ; Get a Key Press
        LDAA KEY_1              ;_/
        STAA INPUT+1            ; Store in INPUT
        STAA SHOW+1             ; Store in SHOW
        JSR DIGIT               ; Update Bottom Display
        JSR GetKey              ; Get a Key Press
        LDAA KEY_1              ;_/
        STAA INPUT+2            ; Store in INPUT
        STAA SHOW+1             ; Store in SHOW
        JSR DIGIT               ; Update Bottom Display
        JSR GetKey              ; Get a Key Press
        LDAA KEY_1              ;_/
        STAA INPUT+3            ; Store in INPUT
        STAA SHOW+1             ; Store in SHOW
        JSR DIGIT               ; Update Bottom Display
        LDD PCode               ; Load program code
        CPD INPUT               ; Check INPUT against secret PCode
        BNE RET1                ; if not same, return to loop
        LDD PCode+2             ; load program code
        CPD INPUT+2             ; check INPUT against secret PCode
        BNE RET1                ;_If unequal, return
        LDAA #$0E               ; \
        STAA SHOW+1             ; Store 'E' in SHOW (Enter)
        JSR DIGIT               ; Update Bottom Display
        JSR GetKey              ; Get Input
        LDAA KEY_1              ;_/
        STAA PIN                ; Store in PIN
        STAA SHOW+1             ; \
        JSR DIGIT               ; Store in SHOW and update Bottom Display
        JSR GetKey              ; Get a Numerical Input
        LDAA KEY_1              ;_/
        STAA PIN+1              ; Store in PIN
        STAA SHOW+1             ; \
        JSR DIGIT               ; Store in SHOW and update display
        JSR GetKey              ;_/
        LDAA KEY_1              ; Get a Numerical Input
        STAA PIN+2              ; Store in PIN
        STAA SHOW+1             ; \
        JSR DIGIT               ; Store in SHOW and update display
        JSR GetKey              ; Get a Numerical Input
        LDAA KEY_1              ; \
        STAA PIN+3              ; Store in PIN
        STAA SHOW+1             ; STore in SHOW and update display
        JSR DIGIT               ;_/
        COM PINSET              ; Complement PINSET boolean
RET1:   LDD #$0000              ; \
        STD INPUT               ; Clear INPUT
        STD INPUT+2             ;_/
        STD MODE                ; Set MODE to LOCKED
        JSR STATUS              ; Update display
        LBRA L0                 ; Return to calling routine

; clears the input
CLEAR:  LDD #$0000              ; Clear INPUT
        STD INPUT               ;  /
        STD INPUT+2             ;_/
        LBRA L0                 ; Return to calling routine
```

```
; alternates between 2 arbitrary images for privacy
ALT:    LDAA OUT                    ; Check OUT against one of two values
        CMPA #$10                   ; 10 or 11
        BNE ALT1                    ; If not one, then set it to that one
        LDAA #$11                   ; Otherwise set it to the other
        STAA OUT                    ;
        STAA SHOW+1                 ;  _
        LDAA #$00                   ; |_|.
        STAA SHOW                   ; Update display
        JSR DIGIT                   ;_/
        LBRA L0                     ; return

ALT1:   LDAA #$10                   ;  _
        STAA OUT                    ; |_|
        STAA SHOW+1                 ;    .
        LDAA #$00                   ; \
        STAA SHOW                   ; Update display
        JSR DIGIT                   ;_/
        LBRA L0                     ; return

;
; Updates LED dispaly for the DIGIT display
;
DIGIT:  LDY SHOW                ; Load Reg. Y with ONE
        LDAA TABLE, Y           ; Load Acc. A with offset TABLE value
        STAA PORTAD             ; store in portAD
        BSET PORTM, $08         ; Enable Latch (Latch)
        NOP                     ; wait
        NOP                     ; wait
        BCLR PORTM, $08         ; Disable Latch
        RTS                     ; Return to calling routine
;
; Updates LED display for STATUS display
;
STATUS: LDY MODE                ; Load Reg. Y with TEN
        LDAA TABLE2, Y          ; Load Acc. A with offset TABLE value
        STAA PORTAD             ; Store on portAD
        BSET PORTM, $04         ; Enable Latch (Latch)
        NOP                     ; wait
        NOP                     ; wait
        BCLR PORTM, $04         ; Disable Latch
        RTS                     ; Return to calling routine
;
;
;       End Program 7 Code
;
;       Program 9 Code
; Displays a Math Flash Card System
; It tests the user over
;  - Addition
;  - Subtraction
;  - Multiplication
;  - Division
; It allows for decimal and negative answers/inputs
;
;
Prgm9
      LDAA  #$0F            ; Make PortT outbound
      STAA  DDRT             ; on pins 0-3
      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             ; Enables SPI
        STAA  SPCR1           ;_/
      LDAA  #$00             ; Disables extra features
      STAA  SPCR2           ;_/
      LDAA  #$80             ; Enables Timer without FFlags
      STAA  TSCR1           ;_/
```

```
;
; Initialize Variables to $00
;
        BCLR  PRINT, $FF
        BCLR  KEY,   $FF
        BCLR  KEY_1, $FF
       BCLR  GEN,   $FF
       BCLR  FANS,  $FF
       BCLR  SOLU,  $FF
       BCLR  SOLU_L,$FF
;
; Initialize the LCD Display
;
        LDAA  #00           ; \
        BSET  PortM, RCK    ; Set RCK to Idle HIGH

        JSR   InitLCD     ; Initialize the LCD

;
; User Interface
;
Loop0    LDX   #String1        ; Load base address of String1
        JSR   PrintString

       LDAA  #$C0            ; First line is done jump to line 2
         JSR    Command

       LDX   #String2          ; Load base address of String2
         JSR   PrintString

         JSR    delay2        ; Let's display the message a while

         JSR   BlinkDisp         ; Blink the display 4 times

Begin   JSR   ClearDisp         ; Clear the display

       LDX   #String3          ; Load base address of String3
         JSR   PrintString

       LDAA  #$C0          ; First line is done jump to line 2
         JSR    Command

       LDX   #String4          ; Load base address of String4
         JSR   PrintString

         JSR    delay2        ; Let's display the message a while

         JSR   ShiftSecondLine   ; Shift the second line to left

         LDX   #String5        ; Load base address of String5
         JSR   PrintString

         JSR    delay2        ; Let's display the message a while

         JSR   ShiftSecondLine

         LDX   #String6         ; Load base address of String6
         JSR   PrintString

         JSR    delay2        ; Let's display the message a while


         JSR   ShiftSecondLine

         LDX   #String7         ; Load base address of String7
         JSR   PrintString

         JSR    delay2        ; Let's display the message a while


         JSR   ShiftSecondLine
```

```
        LDX   #String8         ; Load base address of String8
        JSR   PrintString

       JSR    delay2          ; Let's display the message a while


Select   BCLR  GEN, $FF          ; Clear the GEN variable
        JSR   GetKey            ; Get the Keypad input

        JSR   ClearDisp         ; Clear the Display

        LDX   #Strin13          ; Load base address of String13
        JSR   PrintString

         LDAA  #$C0          ; Jump to line 2
        JSR    Command

        JSR   Method            ; Determine what arithmetic to use
        nop
        nop
        BRSET GEN, $FF, Select  ; If a wrong key is pressed, try again

        LDAA  #$3D              ; Print the "=" sign
       JSR   Print

       JSR   CheckAns          ; Chech if the Answer is correct or not

        JMP   Begin             ; Start Over
;
; ================================================================================
;
; SubRoutines
;
; Print Digit1
;
PDig1   LDAA  DIGIT1            ; Load Digit1 on Accl A
        BMI   PrintNeg1         ; Branch if the MSB is set, hence the number is negative
        JSR   AconvP            ; Convert the value of Accl A to Ascii and Print
        JMP   con1
PrintNeg1
        LDAA  #$2D
        JSR   Print             ; Print the "-" sign if negative
        LDAA  DIGIT1            ; Load Digit1 on Accl A
        COMA                    ; Get Two's compliment
        INCA
        JSR   AconvP            ; Convert the value of Accl A to Ascii and Print
con1    RTS
;
; Print Digit2
;
PDig2   LDAA  DIGIT2            ; Load Digit2 on Accl A
       BMI   PrintNeg2         ; Branch if the MSB is set, hence the number is negative
        JSR   AconvP            ; Convert the value of Accl A to Ascii and Print
        JMP   con2
PrintNeg2
        LDAA  #$2D              ; Print the "-" sign if negative
        JSR   Print             ; Load Digit2 on Accl A
        LDAA  DIGIT2            ; Get Two's compliment
        COMA
        INCA
        JSR   AconvP            ; Convert the value of Accl A to Ascii and Print
con2    RTS
;
; Choose and do the Arithmetic Method
;
Method: JSR   RanNum            ; Generate Random Numbers
        LDAA  KEY_1             ; Load KEY_1 on Accl A
        CMPA  #$01              ; Compare Accl A to hex $01
        BEQ   Add               ; Branch to Add if Key 1 is pressed
        CMPA  #$02
        BEQ   Sub               ; Branch to Sub if Key 2 is pressed
        CMPA  #$03
```

```
        BEQ   Mult              ; Branch to Mult if Key 3 is pressed
        CMPA  #$04
        BEQ   Divi              ; Branch to Divi if Key 4 is pressed
        CMPA  #$0E
        LBEQ   RESTART

        BSET  GEN, $FF          ; Set GEN if some other key has been pressed
        JSR   ShiftSecondLine   ; Shift line to to the left
        LDX   #Strin12          ; Load base address of Strin12
        JSR   PrintString       ; Print the String
        RTS
;
; Add the Numbers
;
Add:    LDAA  DIGIT1            ; Load Digit1 on Accl A
        ADDA  DIGIT2            ; Add Accl A and Digit2
        STAA  SOLU_L            ; Store Accl A to Low bits of SOL
        JSR   PDig1             ; Print Digit1
         LDAA  #$2B             ; Print the "+" sign
      JSR   Print
        JSR   PDig2            ; Print Digit2
        RTS
;
; Subtract the Numbers
;
Sub:    LDAA  DIGIT1            ; Load Digit1 on Accl A
        SUBA  DIGIT2            ; Sub Digit2 from Accl A
        STAA  SOLU_L            ; Store Accl A to Low bits of SOL
        JSR   PDig1            ; Print Digit1
      LDAA  #$2D              ; Print the "-" sign
      JSR   Print
      JSR   PDig2             ; Print Digit2
        RTS
;
; Multiply the Numbers
;
Mult:   LDAA  DIGIT1            ; Load Digit1 on Accl A
        LDAB  DIGIT2            ; Load Digit2 on Accl B
        MUL                     ; Multiply A and B
        STD   SOLU              ; Store D to SOLU
        JSR   PDig1             ; Print Digit1
        LDAA  #$2A              ; Print the "*" sign
      JSR   Print
      JSR   PDig2             ; Print Digit2
        RTS
;
; Divide the Numbers
;
Divi:   LDAA  DIGIT1            ; Load Digit1 on Accl A
        LDAB  DIGIT2            ; Load Digit2 on Accl B
        IDIV                    ; Divide
         STD   SOLU             ; Store D to SOLU
        JSR   PDig2             ; Print Digit1
         LDAA  #$2F             ; Print the "/" sign
      JSR   Print
      JSR   PDig2             ; Print Digit2
        RTS
;
; Check if the Answer is Correct of not
;
CheckAns:
        MOVB  #$00, FANS        ; Clear FANS
        MOVB  #$00, NEGCH       ; Clear NEGCH
V0      JSR   GetKey            ; Get the value of Key pressed

        LDAA  KEY_1             ; Load KEY_1 on Accl A
        CMPA  #$0C              ; Branch to check Answer if A = $0C
        BEQ   V1
        CMPA  #$0B              ; Branch to negative input if A = $0B
        BEQ   V2
        CMPA  #$0A              ; If Any other key than Numbers, E and F,
        BGE   V0                ; try again
```

```
        LDAA  FANS              ; Load FANS to Accl A
        LDAB  #10               ; Load #10 on Accl B
        MUL                     ; Multiply A and B
        STAB  FANS              ; Store B back to FANS to get the tenth place
                                ; of the original value in FANS

        LDAA  KEY_1             ; Load KEY_1 on Accl A
        ADDA  FANS              ; Add Accl A and FANS
        STAA  FANS              ; Store A back to FANS

        LDAA  KEY_1             ; Load KEY_1 on Accl A
        JSR   AconvP            ; Convert the value to Ascii and Print
        JMP   V0                ; Go back to V0

V2      MOVB  #$FF, NEGCH       ; Set NEGCH
        LDAA  #$2D              ; Print the "-" Sign
        JSR   Print
        JMP   V0                ; Go Back to V0

V1      LDAA  FANS              ; Load FANS on Accl A
        BRCLR NEGCH, $FF, PosSol; Branch if NEGCH is Clear, the solution is Positive,
        COMA                    ; Two's Compliment A
        INCA
PosSol  CMPA  SOLU_L            ; Compare A to the Low bits of SOLU
        BNE   IncAns            ; If not equal, the answer is incorrect

         LDAA  #$C0          ; Jump to line 2
        JSR    Command
      LDX   #Strin10          ; Load base address of Strin10
        JSR   PrintString      ; Print the String
        JSR   delay2
        JSR   BlinkDisp         ; Blink the Display
        RTS

IncAns  LDAA  #$C0           ; Jump to line 2
        JSR    Command
       LDX   #Strin11        ; Load base address of Strin11
        JSR   PrintString       ; Print the String
        JSR   delay2
        JSR   BlinkDisp         ; Blink the Display
        RTS
;
; Random Number Generator
;
RanNum  LDD TCNT                ; load current timer count
Loop1   CMPB #10                ; compare to 10 (lower half)
        BLO  Save1              ; if lower, save
        SUBB #10                ; else subtract
        BRA  Loop1              ; and check again
Save1   STAB DIGIT1             ; save digit A
Loop2   CMPA #10                ; compare to 10 (upper half)
        BLO  Save2              ; if lower, save
        SUBA #10                ; else subtract
        BRA  Loop2              ; and check again
Save2   STAA DIGIT2             ; save digit B
;
; 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
;
;
;       End Program 9 Code
;                              ------------------------------------------------
;       Program 12 Code
; 24 hour Clock
; Can display in 12 or 24 hr formats
; Can run at 120x speed
; Can be reset at any time
;
;
Prgm12
        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           ; enable SPI
       LDAA  #$00
       STAA  SPCR2           ; disable features

        BSET Exp12, $FF       ; exp 12 is running
        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

       LDD TCNT        ; initialize timer interrupt
       ADDD INCREMENT
       STD TC0
       LDAA $01
       STAA TIE
       MOVB $01, TFLG1



;
; Initial Reset Location
;


Loop
        JSR GetKey       ; Continuously get key
        LDAA KEY_1       ; load key data
        JSR KeyPress     ; analyze key press
        BRA Loop         ; 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

; Analyze User Input/Reset Time
KeyPress
        CMPA #$0A     ; A -> Format
        BEQ Format
        CMPA #$0B     ; B -> Speed
        BEQ ChangeSpd
```

```
        CMPA #$0F     ; F -> Enter Time
        BEQ StartEnterTime
        CMPA #$0E     ; E -> Jump to menu
        LBEQ RESTART
        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
        LDAA #$00
        STAA EnterOne
        STAA EnterTwo
        STAA EnterThree
        STAA EnterFour
        RTS

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

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

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

;
; Get the Value of the Key pressed
;
Check4a:JSR Check4
        BRA Pcheck
Check12:JSR UpDisp
        BRA Pcheck
```

```
GetKey: BCLR KEY, $FF         ; Clear variable KEY contents
        BCLR KEY_1, $FF
        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:   BRSET Exp4, $FF, Check4a
        BRSET Exp12, $FF, Check12
Pcheck  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
        BRSET PORTT, $40, KEYF    ;If high, key F was pressed
        BRSET PORTT, $20, KEYE    ;If high, key E was pressed
        BRSET PORTT, $10, KEYD    ;If high, key D was pressed
        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:   BSET KEY_1, $01       ; Set KEY to 1
        RTS                ; Return to GETKEY's calling routine

KEY2:   BSET KEY_1, $02       ; Set KEY to 2
        RTS                ; Return to GETKEY's calling routine

KEY3:   BSET KEY_1, $03       ; Set KEY to 3
        RTS                ; Return to GETKEY's calling routine

KEYA:   BSET KEY_1, $0A       ; Set KEY to A
        RTS

KEY4:   BSET KEY_1, $04       ; Set KEY to 4
        RTS                ; Return to GETKEY's calling routine
```

```
KEY5:   BSET KEY_1, $05       ; Set KEY to 5
        RTS                 ; Return to GETKEY's calling routine

KEY6:   BSET KEY_1, $06       ; Set KEY to 6
        RTS                 ; Return to GETKEY's calling routine

KEYB:   BSET KEY_1, $0B       ; Set KEY to B
        RTS

KEY7:   BSET KEY_1, $07       ; Set KEY to 7
        RTS                 ; Return to GETKEY's calling routine

KEY8:   BSET KEY_1, $08       ; Set KEY to 8
        RTS                 ; Return to GETKEY's calling routine

KEY9:   BSET KEY_1, $09       ; Set KEY to 9
        RTS                 ; Return to GETKEY's calling routine

KEYC:   BSET KEY_1, $0C       ; Set KEY to C
        RTS

KEY0:   BSET KEY_1, $00       ; Set KEY to 0
        RTS                 ; Return to GETKEY's calling routine

KEYD:   BSET KEY_1, $0D       ; Set KEY to D
        RTS

KEYE:   BSET KEY_1, $0E       ; Set KEY to E
        RTS                 ; Return to GETKEY's calling routine

KEYF:   BSET KEY_1, $0F       ; Set KEY to F
        RTS                 ; Return to GETKEY's calling routine
;
;




;
; Delay functions
;
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

sdelay: PSHY
        LDY #15000   ; 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
```

```
DELAY  PSHA     ; Save accumulator A on the stack
        LDY  #01    ; We will repeat this subroutine 10 times
        MOVB #$90,TSCR1 ; enable TCNT & fast flags clear
        MOVB #$07,TSCR2   ; configure prescale factor to 64
        MOVB #$01,TIOS  ; enable OC0
        LDD  TCNT   ; Get current TCNT value
AGAIN    ADDD #18750   ; start an output compare operation
        STD  TC0    ; with 100 ms time delay
WAIT   BRCLR  TFLG1,$01,WAIT  ; Wait for TCNT to catch up
        LDD  TC0    ; Get the value in TC0
        DBNE Y,AGAIN    ; 1 X 100ms = 100 ms
        PULA     ; Pull A
        RTS

SHORT_DELAY:
        LDAA #5   ; Outer Loop Counter ? 1 clock cycle
B3:     LDY #5000 ; Inside Loop Counter 2 clock cycles
B2:     LBRN B2    ; 3 clock cycles  \
        DEY        ; 1 clock cycle    | 8 clock cycles in loop
        LBNE B2    ; 4 clock cycles  /
        DECA       ; 1 clock cycle
        BNE B3     ; 3 clock cycles
        RTS        ; return from delay - 5 clock cycles

; Exp 4 Timer Interrupt Service Routine
ISR_TC5:LDD TC5
        ADDD INCREMENT4
        STD TC5
        BRCLR FLAG1, $10, DONE4
        LDD COUNTP4
        BEQ DONE4
        SUBD #$0001
        STD COUNTP4
DONE4:  RTI



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
      RTS

helpDONE
        RTI

; Exp 12 Timer Interrupt Service Routine
ISR_TC0:LDD TC0                   ; load counter
        BRSET Speed, $FF, Fast    ; check speed
        ADDD INCREMENT
        BRA Store                 ; inc counter
Fast:   ADDD FASTEMENT
Store:  STD TC0                   ; 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                  ; 4 = 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 #$04
        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




; Declare Constants
        ORG $5000
TABLE:  DC.B $3F, $06, $5B, $4F, $66, $6D, $7D, $07, $7F, $6F, $77, $7F, $39, $3F, $79, $71, $63, $5C, $00
;Order:       0    1    2    3    4    5    6    7    8    9    A    B    C    D    E    F    o^   o.   off
TABLEa:  DC.B $00, $3F, $06, $5B, $4F, $66, $6D, $7D, $07, $7F, $67
;
;    Order  :off, 0, 1, 2, 3, 4, 5 ,6 ,7 ,8 ,9
MAXCOUNT:DC.W 50  ; Exp4 Count Clicks
INCREMENT4: DC.W 3750   ; Exp4 Timer Increment
PCode   DC.B  $01, $02, $03, $05 ; Press 1235
TABLE2: DC.B $38, $3F, $73, $00
;Order:       L    O    P    off
String1  FCC    "Welcome to Math "    ; Exp9 Strings |      Intro
        DC.B  $00                                  ;|
String2  FCC    "Flash Cards     "                  ;|      Intro
        DC.B  $00                                  ;|
String3 FCC   "Options:        "                   ;|      Menu
        DC.B  $00                                  ;|
String4 FCC   "Press the Key   "                   ;|      Menu
        DC.B  $00                                  ;|
String5 FCC   "1.Addition      "                   ;|      Menu
```

```
        DC.B  $00                               ;|
String6 FCC   "2.Subtraction   "                ;|      Menu
        DC.B  $00                               ;|
String7 FCC   "3.Multiplication"                ;|      Menu
        DC.B  $00                               ;|
String8 FCC   "4.Division      "                ;|      Menu
        DC.B  $00                               ;|
Strin10 FCC   "That is correct!"                ;|     Answer Check
        DC.B  $00                               ;|
Strin11 FCC   "Incorrect :/    "                ;|     Answer Check
        DC.B  $00                               ;|
Strin12 FCC   "Invalid Option. "                ;|     Input Check
        DC.B  $00                               ;|
Strin13 FCC   "Question:       "                ;V     Question Output
        DC.B  $00                              ;___^
Colon   FCC ":"                  ; Exp 12 Colon String
        DC.B $00

NoColon FCC " "                 ; Exp 12 No Colon String/Space String
        DC.B $00

NotPM   FCC "am"                 ; Exp 12 AM string
        DC.B $00

NotAM   FCC "pm"                 ; Exp 12 PM string
        DC.B $00
INCREMENT: DC.W $EDA1 ; 45625 -> .25s at 128 prescale
FASTEMENT: DC.W $0177 ; 380 -> .25s/120 at 128 prescale

Intro1  FCC   "Welcome To My   "    ; Menu Intro String
        DC.B $00

Intro2  FCC   "Integrated Lab  "    ; Menu Intro String
        DC.B $00

Intro3  FCC   "Choose Program  "    ; Menu Instruction String
        DC.B $00

Intro4  FCC   "1: Now Serving  "    ; Menu Option String
        DC.B $00

Intro5  FCC   "2: Shot Clock   "    ; Menu Option String
        DC.B $00

Intro6  FCC   "3: Combo Lock   "    ; Menu Option String
        DC.B $00

Intro7  FCC   "4: Flash Cards  "    ; Menu Option String
        DC.B $00

Intro8  FCC   "5: Clock        "    ; Menu Option String
        DC.B $00

Error1  FCC   "Bad Input       "    ; Input Error String
        DC.B $00

Error2  FCC   "Try Again       "    ; Input Error String
        DC.B $00

; Define TC0 Interrupt Vector

        ORG $FFEE
        FDB ISR_TC0


; Define TC5 Interrupt Vector

        ORG $FFE4
        FDB ISR_TC5


; Define Power-On Reset Interrupt Vector
```