```
*******************************************************
;* This stationery serves as the framework for a              *
;* user application. For a more comprehensive program that    *
;* demonstrates the more advanced functionality of this       *
;* processor, please see the demonstration applications       *
;* located in the examples subdirectory of the                *
;* Freescale CodeWarrior for the HC12 Program directory       *
;**************************************************************
;**************************************************************
;* This stationery serves as the framework for a              *
;* user application. For a more comprehensive program that    *
;* demonstrates the more advanced functionality of this       *
;* processor, please see the demonstration applications       *
;* located in the examples subdirectory of the                *
;* Freescale CodeWarrior for the HC12 Program directory       *
;**************************************************************
;; Include derivative-specific definitions
            INCLUDE 'derivative.inc'

; export symbols
            XDEF Entry, _Startup, main, RTI_ISR, IRQ_ISR
            ; we use export 'Entry' as symbol. This allows us to
            ; reference 'Entry' either in the linker .prm file
            ; or from C/C++ later on

            XREF __SEG_END_SSTACK, init_LCD, display_string, read_pot, pot_value, SendsChr,
PlayTone      ; symbol defined by the linker for the end of the stack




; variable/data section
MY_EXTENDED_RAM: SECTION
; Insert here your data definition.



;booleans: these are to be set to 1 if they are true, and 0 if they are false
Bool_SongPlaying        ds.b 1    ;a boolean to determine if a song is currently being played
Bool_BatteryCharging    ds.b 1
Bool_HexInputAsk        ds.b 1
Bool_WelcomeUser             ds.b   1
Bool_GeneralPauseTF   ds.b   1
Bool_LoginActive             ds.b   1      ;0 at start, 1 when login is active, 2 when login
complete
Bool_UserLogLEDUorD   ds.b   1   ;controls the LED direction when Login is active
Bool_SongMenu           ds.b  1     ;this is 1 when the song menu is on and 0 when it is off
Bool_BadPassword        ds.b  1
Bool_BadUser            ds.b  1
Bool_SongMenuFast            ds.b   1
Bool_LogoutRequest           ds.b   1
Bool_NewUserLoggedIn  ds.b   1
Bool_WrongUserLogin     ds.b    1
Bool_SongPaused                     ds.b   1
Bool_NotePlaying             ds.b   1
Bool_NoDisplay          ds.b  1
Bool_PastState          ds.b  1
```

```
Bool_IRQFlip                 ds.b   1
Bool_SongGoing               ds.b   1
Bool_ShortRest               ds.b   1
Bool_UserPick                ds.b   1
Bool_Die                            ds.b   1

        ;(pauses are a class of booleans that are evaluated in the RTI section)
Pause_WelcomeDisplay    ds.b        1


;timers:  These are for the tracking over various timed items
Timer_SongOneSec        ds.w        1    ;this counter is for counting up to 1 second for the
song duration/stepper motor, resets after 977 is reached
Timer_SongSeconds       ds.b        1
Timer_SongMinutes       ds.b        1
Timer_DCTwentySeconds   ds.w        1
Timer_MsVariable             ds.b   1
Timer_GeneralPause           ds.w    1
Timer_LEDVariable       ds.b    1     ;this timer is for use with the LED lighting subroutines
Timer_UserPick               ds.b   1



;counters:  These are for counting up things
Counter_BatteryControl  ds.b        1
Counter_StepperControl  ds.b        1
Counter_DCMotoSpeed     ds.b        1
Counter_DCMotoFifteen   ds.b        1
Counter_WelUserArray  ds.b   1
Counter_WelUserLED           ds.b   1
Counter_LEDArray          ds.b      1
Counter_GeneralCounter  ds.b        1
Counter_GeneralCounter1 ds.b        1
Counter_GeneralCounter2 ds.b        1
Counter_SM_Hexpad            ds.b   1
Counter_HexpadPass           ds.b   1
Counter_IntCounter1          ds.b   1
Counter_IntCounter2          ds.b   1
Counter_IntCounter3          ds.w   1
Counter_IntCounter4          ds.b   1

;variable:  These are for storing and passing info
Variable_HexpadInput    ds.b        1
Variable_HexpadMask     ds.b        1
Variable_HexpadCounter  ds.b        1
Variable_HexTemp        ds.b    1
Variable_XPause                     ds.w   1
Variable_PauseDelay          ds.b   1
Variable_SongCurrent  ds.b   1
Variable_YHold               ds.w   1
Variable_BHold               ds.b   1
Variable_General             ds.b   1
Variable_Note                ds.b   1
Variable_LEDNote             ds.b   1
Variable_WhichSong      ds.b  1
Variable_HoldB               ds.b   1
Variable_HoldA               ds.b   1
```

```
;arrays

Array_UsersLogPass      ds.b        36    ;this is the user login array, it has 4 spaces for
each of the 9 possible users
                                          ;with the first of the 4 numbers being the user
number and the next 3 being their unique password
LCD_SongMenuFinal            ds.b          36
LCD_SongTimer                ds.b          36




Constant: Section

;arrays:
Array_StepperControl    dc.b  $0A, $12, $14, $0C   ;this is the array of stepper control
numbers that need to be incremented through to rotate the stepper
Array_HexPullCombo      dc.b  $88, $84, $82, $81, $48, $44, $42, $41, $28, $24, $22, $21, $18,
$14, $12, $11
Array_HexPullKey        dc.b  $1, $2, $3, $C, $4, $5, $6, $d, $7, $8, $9, $E, $A, $0, $B, $F
Array_WelcomeUserLED  dc.b   $81, $42, $24, $18, $24, $42, $81, $42, $24, $18, $24, $42, $81
Array_UserLoginLED    dc.b   $01, $02, $04, $08, $10, $20, $40, $80
Array_SongMenuLED       dc.b  $18, $24, $42, $81
Array_PreDefUsers       dc.b  1, 1, 4, 7, 2, 2, 5, 8, 3, 3, 6, 9
Array_SandstormLED           dc.b   0,0,0,0,0,0,0,0,0,0,0,0,$02,$04,$08,$10,$20,$40,$80

SANDSTORM                              dc.b
16,2,16,2,16,2,16,2,16,2,200,8,200,12,13,4,16,2,16,2,16,2,16,2,16,4,200,8,200,16
Sandstorm2                             dc.b
16,2,16,2,16,2,16,2,16,4,200,4,16,2,16,2,16,2,16,2,16,4,200,4,16,2,16,2,16,2,16,2,16,4,200,4,1
6,2,16,2,16,1,16,1,16,4,200,4,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2
,16,2,16,2,16,2
Sandstorm3                             dc.b
16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2
Sandstorm4                             dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,12,2,12,2,12,2,12,2,12,4
,13,2,13,2,13,2,13,2,13,2,13,2,13,4,18,2,18,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,
2,16,2,16,4,12,2,12,2
Sandstorm5                             dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4
,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,12,2,12,2,12,2,12,2,12,4,13,2,13,2,13,2,13,2,13,
2,13,2,13,4,18,2,18,2
Sandstorm6                             dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4
,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,
2,16,2,16,4,12,2,12,2
Sandstorm7                             dc.b
12,2,12,2,12,2,12,2,12,4,13,2,13,2,13,2,13,2,13,2,13,2,13,4,18,2,18,2,16,2,16,2,16,2,16,2,16,4
,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,
2,16,2,16,4,12,2,12,2
Sandstorm8                             dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,12,2,12,2,12,2,12,2,12,4
,13,2,13,2,13,2,13,2,13,2,13,2,13,4,18,2,18,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,
2,16,2,16,4,12,2,12,2
Sandstorm9                             dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4
```

```
,16,2,16,2,16,2,16,2,16,2,16,2,16,4,13,4,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,
2,13,4,13,4
Sandstorm10                        dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,12,2,12,2,12,4,12,4,13,2
,13,2,13,2,13,2,13,4,13,4,18,2,18,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,
4,13,4
Sandstorm11                        dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,13,4,13,4,16,2,16,2,16,2,16,2,16,4,16,2
,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,12,2,12,2,12,4,12,4,13,2,13,2,13,2,13,2,13,4,13,4,18,
2,18,2
Sandstorm12                        dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,13,4,13,4,16,2,16,2,16,2,16,2,16,4,16,2
,16,2,16,2,16,2,16,2,16,2,13,4,13,4,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,13,
4,13,4
Sandstorm13                        dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,13,4,13,4,16,2,16,2,16,4,13,4,13,4,16,2
,16,2,16,4,13,4,13,4,16,2,16,2,16,4,13,4,13,4,16,2,16,2,16,4,13,4,13,4
Sandstorm14                        dc.b
16,2,16,2,13,4,16,2,16,2,13,4,16,2,16,2,13,4,16,2,16,2,13,4,16,2,16,2,13,4,16,2,16,2,13,4,16,2
,16,2,13,4,16,2,16,2,13,4
Sandstorm15                        dc.b
200,16,200,16,16,2,16,2,16,2,16,2,16,2,200,8,200,12,13,4,16,2,16,2,16,2,16,2,16,4,200,8,200,16
,14,4,11,4,11,2,11,2,11,2,11,2,13,2,200,2,200,16
Sandstorm16            dc.b
16,2,16,2,16,2,16,2,16,4,200,4,16,2,16,2,16,2,16,2,16,4,200,4,16,2,16,2,16,2,16,2,16,4,200,4,1
6,2,16,2,16,1,16,1,16,4,200,4
Sandstorm17            dc.b
16,2,16,2,16,2,16,2,16,2,200,8,200,12,16,2,16,2,16,2,16,2,16,2,200,8,200,12,16,2,16,2,16,2,16,
2,16,2,200,8,200,12,13,4,16,2,16,2,16,2,16,2,16,4,200,8,200,16
Sandstorm18            dc.b
16,2,16,2,16,2,16,2,16,4,200,4,16,2,16,2,16,2,16,2,16,4,200,4,16,2,16,2,16,2,16,2,16,4,200,4,1
6,2,16,2,16,1,16,1,16,4,200,4,16,2,16,2,16,2,16,2,16,4,200,4,16,2,16,2,16,2,16,2,16,4,200,4,16
,2,16,2,16,2,16,2,16,4,200,4,16,2,16,2,16,1,16,1,16,4,200,4
Sandstorm19            dc.b
16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2
,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2
Sandstorm191           dc.b
16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2
,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2
Sandstorm20            dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,12,2,12,2,12,2,12,2,12,4
,13,2,13,2,13,2,13,2,13,2,13,2,13,4,18,2,18,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,
2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16
,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,12,2,12,2,12,2,12,2,12,4,1
3,2,13,2,13,2,13,2,13,2,13,2,13,4,18,2,18,2
Sandstorm21            dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4
,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,
2,16,2,16,4,12,2,12,2,12,2,12,2,12,2,12,2,12,4,13,2,13,2,13,2,13,2,13,2,13,2,13,4,18,2,18,2,16
,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4,1
6,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,
16,2,16,4,12,2,12,2,12,2,12,2,12,2,12,2,12,4,13,2,13,2,13,2,13,2,13,2,13,2,13,4,18,2,18,2
Sandstorm22            dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4
,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,
2,16,2,16,4
```

```
Sandstorm23             dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,13,4,13,4,16,2,16,2,16,2,16,2,16,4,16,2
,16,2,16,2,16,2,16,2,16,2,13,4,13,4,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,13,
4,13,4,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,13,4,13,4
Sandstorm24             dc.b
16,2,16,2,16,4,13,4,13,4,16,2,16,2,16,4,13,4,13,4,16,2,16,2,16,4,13,4,13,4,16,2,16,2,16,4,13,4
,16,2,16,2,13,4,16,2,16,2,13,4,16,2,16,2,13,4,16,2,16,2,13,4,16,2,16,2,13,4,16,2,16,2,13,4,16,
2,16,2,13,4,16,2,16,2,13,4
Sandstorm25             dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,12,2,12,2,12,2,12,2,12,4
,13,2,13,2,13,2,13,2,13,2,13,2,13,4,18,2,18,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,
2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2
Sandstorm251                 dc.b
16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2
,16,2,16,2,16,4,12,2,12,2,12,2,12,2,12,2,12,2,12,4,13,2,13,2,13,2,13,2,13,2,13,2,13,4,18,2,18,
2
Sandstorm26             dc.b
16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,12,2,12,2,12,2,12,2,12,4
,13,2,13,2,13,2,13,2,13,2,13,2,13,4,18,2,18,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,
2,16,2,16,4,12,2,12,2,16,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,16
,2,16,2,16,2,16,2,16,4,16,2,16,2,16,2,16,2,16,2,16,2,16,4,12,2,12,2,12,2,12,2,12,2,12,2,12,4,1
3,2,13,2,13,2,13,2,13,2,13,2,13,4,18,2,18,2
Sandstorm27             dc.b
16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2
,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,16,2,250

AFTERNOONDELIGHT             dc.b
22,2,22,2,18,4,15,4,22,2,15,2,22,2,13,4,15,4,18,4,18,2,18,2,27,4,22,2,22,4,27,4,22,2,22,4,27,2
,16,4,15,2,18,2,20,2,22,4,22,2,18,2,15,2,15,2,15,2,15,4,15,2,18,2,13,4,15,4,18,4,18,4,20,2,22,
2,22,2,22,2,22,2,22,2,22,2,22,2,16,4,15,4,15,16,200,16
AD2                           dc.b
15,8,16,2,18,2,16,4,15,4,200,8,200,4,15,4,15,2,14,4,14,2,12,4,200,16,200,16,22,8,200,4,20,4,15
,4,15,8,15,4,16,2,18,4,200,16,200,16,20,8,200,4,18,4,15,4,15,2,15,4,16,2,18,4,20,2,22,4,200,8,
200,16,200,8
AD3                           dc.b
15,4,15,4,15,2,15,2,15,2,15,2,13,4,18,2,18,2,18,2,18,2,22,2,22,2,22,2,22,2,22,2,22,2,22,2,22,2
,16,4,15,2,15,4,200,2,22,2,22,2,18,2,15,2,15,2,15,2,15,2,15,2,15,2,15,2,13,4,15,4,18,4,18,4,18
,2,22,2,22,2,22,2,22,2,22,2,22,2,200,2
AD4                           dc.b
22,2,16,4,15,4,15,4,200,2,15,8,16,2,18,2,16,4,15,4,200,8,200,4,15,4,15,2,12,4,12,2,12,4,200,16
,200,16,22,8,200,4,20,4,15,4,15,2,15,4,16,2,18,4,200,16,200,16,20,8,200,4,18,4,15,4,15,2,15,4,
16,2,18,4,20,2,22,4,200,4,200,2
AD5                           dc.b
18,16,16,2,18,2,16,2,18,2,16,2,18,2,16,2,18,2,16,4,15,4,15,16,200,8,200,4,18,16,16,2,18,2,16,2
,18,2,16,2,18,2,16,2,18,2,16,4,15,4,15,4,22,2,22,2,15,4,15,4,15,2,15,2,15,2,15,2,13,4,16,4,18,
2,18,2,18,2,18,2,22,2,22,2,22,2,22,2,22,2,22,2,22,2,22,2
AD6                           dc.b
16,4,15,2,15,4,200,2,22,2,22,2,18,2,15,2,15,2,15,2,15,2,15,2,15,2,15,2,13,4,15,4,18,4,18,4,22,
2,22,2,22,2,22,2,22,2,22,2,22,2,22,2,22,2,16,4,15,4,15,4,200,2,15,8,16,2,18,2,16,4,15,4,200,4,
200,8,15,4,15,2,12,4,12,2,12,4,200,16,200,16,22,8,200,4
AD7                           dc.b
20,4,15,4,15,2,15,4,16,2,18,4,200,16,200,16,20,8,200,4,18,4,15,4,15,2,15,4,16,2,18,4,20,2,22,4
,200,2,200,8,200,8,20,16,18,16,15,8,15,4,200,4,15,4,15,2,15,4,16,2,18,4,16,2,15,2,200,16,200,1
6,22,8,200,2,22,16,22,4,15,4,15,2,15,4,18,2,18,1,250

JINGLEBELLS                          dc.b
26,4,16,4,18,4,20,4,26,4,45,4,45,4,26,2,26,2,26,4,16,4,18,4,20,4,24,4,30,4,30,4,24,4,15,4,16,4
```

```
,18,4,22,4,26,4,26,4,200,4,13,4,13,4,15,4,18,4,16,4,45,4,45,4,200,4,26,4,16,4,18,4,20,4,26,4,4
5,4,45,4,26,4,26,4,16,4,18,4,20,4,26,4,45,4,45,4
JB2                           dc.b
26,4,26,4,16,4,18,4,20,4,24,4,30,4,30,4,200,4,24,4,15,4,16,4,18,4,15,4,15,4,15,4,15,4,12,4,13,
4,15,4,18,4,20,8,13,8,16,4,16,4,16,8,16,4,16,4,16,8,16,4,13,4,20,4,18,4,16,16,15,4,15,4,15,4,1
5,4,15,4,16,4,16,4,16,2,16,2,16,4,18,4,18,4,16,4,18,2,13,2,26,4,26,4,26,8,16,4,16,4,16,8,16,4,
13,4,20,4,18,4,26,16,24,4,24,4,24,4,24,4,15,4,26,4,26,4,26,2,26,2,16,4,16,4,15,4,18,4,20,8,45,
4,250

;LCD arrays
LCD_Welcome             dc.b 'Welcome!        Ishmael Inc     ',0
LCD_LoginUser    dc.b 'Enter User #    Press F for New ',0
LCD_PassEnter   dc.b  'Enter Password: ---             ',0 ;REPLACING * with -, start at +15
LCD_PassEnter1 dc.b   'Enter Password: *--             ',0
LCD_PassEnter2 dc.b   'Enter Password: **-             ',0
LCD_BadPassDisp       dc.b   'Pass Incorrect   Enter User #   ',0
LCD_BadUserDisp dc.b  'User Incorrect   Press F for new',0
LCD_MadeNewUser dc.b    'Choose User #   F to go back    ',0
LCD_NewPassword dc.b    'Enter a 3 digit password        ',0
LCD_UserNumUsed dc.b    'User # taken    Select another  ',0
LCD_NoNewSpace  dc.b    'No new space fornew user. Sorry.',0
LCD_PressToGo   dc.b    'Press any key   to continue     ',0
LCD_SongMenu1  dc.b   'Sandstorm   <---AfternoonDelight',0
LCD_SongMenu2  dc.b   'AfternoonDelightJingle Bells    ',0
LCD_SongMenu3  dc.b   'Jingle Bells<---Random          ',0
LCD_SongMenu4  dc.b   'Random      <---Logout          ',0
LCD_SongMenu5  dc.b   'Logout      <---                ',0
LCD_SongTimer1 dc.b   '                         :      ',0
LCD_NoDisplay     dc.b  '                                ',0


;definitions
TOFCount       equu      997  ;this is the amount of TOF's needed to equate 1 second
CRG_INT   equ   $38 ;the RTI interrupt timing control
RTI_CTL   equ   $3B ;the RTI interrupt enable control
Port_P    equ   $258
Port_P_DDR  equ $25A
Port_T    equ   $240
Port_T_DDR  equ $242
Port_U    equ   $268
Port_U_DDR  equ $26A
Port_U_PSR  equ $26D
Port_U_PDE  equ $26C
Port_S        equ    $248
Port_S_DDR     equ    $24A


; code section
MyCode:     SECTION
main:
_Startup:
Entry:
;systemwide initializations
          lds  #__SEG_END_SSTACK      ;initialize the stack pointer

          bset  CRG_INT, #$80        ;enables RTI interrupts
          ;bset       CRG_FLG, #80
          bset  RTICTL, #$10        ;sets the RTI timer to $40, so every 977 interrupts = 1ms
          movb  #%00011110, Port_P_DDR
          movb  #$8, Port_T_DDR
          bset  Port_U_DDR, #$F0
          bset  Port_U_PSR, #$0F
          bset  Port_U_PDE, #$0F
          movb #$FF, Port_S_DDR
          movb    #$C0, INTCR


;item specific initializations
;booleans

                     movb #0, Bool_SongPlaying
                     movb #0, Bool_BatteryCharging
                     movb #0, Bool_HexInputAsk
                     movb #1, Bool_WelcomeUser
                     movb #0, Bool_GeneralPauseTF
                     movb #0, Bool_LoginActive
                     movb #0, Bool_UserLogLEDUorD
                     movb #0, Bool_SongMenu
                     movb #0, Bool_BadPassword
                     movb #0, Bool_BadUser
                     movb #0, Bool_SongMenuFast
                     movb #0, Bool_LogoutRequest
                        movb #0, Bool_NewUserLoggedIn
                        movb #0, Bool_WrongUserLogin
                        movb #0, Bool_SongPaused
                        movb #0, Bool_NotePlaying
                        movb #0, Bool_NoDisplay
                        movb #0, Bool_PastState
                        movb #0, Bool_IRQFlip
                        movb #0, Bool_SongGoing
                        movb #0, Bool_ShortRest
                        movb #0, Bool_UserPick
                        movb #0, Bool_Die



;timers
                     movb #0, Timer_SongSeconds
                     movb #0, Timer_SongMinutes
                     movb #0, Timer_SongOneSec
                     movb #0, Timer_DCTwentySeconds
                     movb #0, Timer_MsVariable
                     movb #0, Timer_LEDVariable
                     movb #0, Timer_UserPick



;counters
                     movb #0, Counter_StepperControl
                     movb #$F, Counter_DCMotoSpeed
                     movb #0, Counter_DCMotoFifteen
                     movb #0, Counter_WelUserArray
                     movb #0, Counter_WelUserLED
                     movb #0, Counter_LEDArray
```

```
                    movb #0, Counter_SM_Hexpad
                    movb #8, Counter_IntCounter1
                    movb #8, Counter_IntCounter2
                    movw #0, Counter_IntCounter3
                    movb #1, Counter_IntCounter4


;arrays
                    ldx      #Array_UsersLogPass                ;this chunk initialized the
current user array for the predefined users
                    ldy      #Array_PreDefUsers                 ;
                    movb     #0, Counter_GeneralCounter         ;
Init_UserArray:     ldaa     1, Y+                              ;
                    staa     1, X+                              ;
                    inc      Counter_GeneralCounter             ;
                    ldab     Counter_GeneralCounter             ;
                    cmpb     #12                                ;
                    bne      Init_UserArray                     ;this ends the chunk

Init_OtherUsers:    ldaa     #0                                 ;this chunk sets the rest of
the array to 0
                    staa     1, X+                              ;this sets the array up to
be searched for an active user in the login screen
                    inc      Counter_GeneralCounter             ;
                    ldab     Counter_GeneralCounter             ;
                    cmpb     #36                                ;
                    bne      Init_OtherUsers                    ;this ends this chunk

                                   ldaa   #0
                                   ldx           #LCD_SongTimer1
                                   ldy           #LCD_SongTimer
Init_SongDisp:       ldab   1, X+
                                   stab   1, Y+
                                   inca
                                   cmpa   #35
                                   bne           Init_SongDisp




;BEGINNING OF PROGRAM, welcome the user and call the startup routine - all other code is to be
reached via subroutine calls
                                cli                          ;enable interrupt
TestLoop:                     jsr   init_LCD
                     jsr    LCD_WelcomeUser
Main_Pro_WelUserWait: brset  Bool_WelcomeUser, 2, Main_Pro_WelUserDone
                                           bra    Main_Pro_WelUserWait

Main_Pro_WelUserDone: jsr    LCD_UserLogin
                                           brset  Bool_NewUserLoggedIn, #1,
Main_Pro_WelUserDone
                                           movb   #0, Timer_LEDVariable

Main_Pro_UserLogDone:   movb #0, Bool_LogoutRequest
                                           jsr   Control_SongMenu
```

```
                                            brset  Bool_LogoutRequest, #1,
Main_Pro_WelUserDone
                                            bra    Main_Pro_UserLogDone


;this is the IRQ_ISR subroutine, it makes the LCD's go blank if pressed when a song is playing
IRQ_ISR:

                                     brclr  Variable_PauseDelay, $FF, IRQ_ISR_Continue
                                     bra           IRQ_ISR_Exit
IRQ_ISR_Continue:     movb #250, Variable_PauseDelay
                  ldaa  Bool_SongPlaying
                  beq   IRQ_ISR_Exit
                        ldaa  Bool_NoDisplay
                  bne   IRQ_ISR_On
                  ldab  #1
                  stab  Bool_NoDisplay
                  bra   IRQ_ISR_Exit
IRQ_ISR_On:       ldab  #0
                  stab  Bool_NoDisplay
IRQ_ISR_Exit:     movb  #$80, CRGFLG
                  rti



;RTI interrupt code, to be executed on RTI interrupts. These interrupts are to happen every
1ms. All timing counters are to be 1 byte length memory locations.
;accessing these will be easy. Simply push B to the stack, load B with the memory location,
increment by 1, and store it back, resetting and branching to necessary subroutine
;after B has been pulled back off the stack.

RTI_ISR:
                                            ldaa   Bool_Die
                                            beq           RTI_Go
                                            rti
RTI_Go:
                                            ldaa   Bool_SongPlaying             ;check if B_SP
is high or low
                                            lbeq   RTI_ISR_NoSong               ;if low, no
song playing, go to no song
                                            ldaa   Bool_NotePlaying             ;check B_NP
                                            beq           RTI_ISR_NoNote               ;if
low, no note is playing, go to No Note
                                            ldaa   Variable_Note                ;if note is
playing, load the Note #
                                            psha
;push A and play the note
                                            movb   #$28, Port_T_DDR
                                            jsr           SendsChr
                                            pula
                                            jsr           PlayTone
RTI_ISR_NoNote:                      ldd           Counter_IntCounter3          ;load D with
current note duration counter
                                            beq           RTI_ISR_NoteOver             ;if 0,
go to NoteOver
                                            decb
;if not, decrement B
```

```
                                            bne           RTI_ISR_BNotZero             ;if B
is not zero, go to BNotZero
                                            staa   Variable_General             ;if B is zero,
store A to check if it is zero
                                            beq           RTI_ISR_BNotZero             ;if so,
go to BNotZero
                                            deca
;if not, decrement A
                                            ldab   #$FF                         ;load B
with $FF
RTI_ISR_BNotZero:            std           Counter_IntCounter3          ;store D to check if
it is all 0
                                            lbne          RTI_ISR_NoSong               ;if
not, then go to NoSong
                                            movb   #0, Bool_NotePlaying    ;if it is 0, load 0
to NotePlaying, as the note should stop
                                            movb   #$8, Port_T_DDR                     ;turn
off the speaker
                                            movb   #100, Counter_IntCounter4      ;set the
counter
RTI_ISR_NoteOver:            dec           Counter_IntCounter4          ;decrement the
counter
                                            lbne   RTI_ISR_NoSong               ;if not 0, go
to NoSong
                                            ldy           Variable_YHold               ;if 0,
then
                                            ldaa   1, Y+
                                            pshb
                                            ldab  Variable_WhichSong
                                            cmpb  #1
                                            bne   RTI_ISR_NotSS
                                            cmpa   #16
                                            bne    RTI_ISR_NoNote16
                                            ldaa   #15
RTI_ISR_NoNote16:           cmpa  #18
                        bne   RTI_ISR_NoNote18
                        ldaa  #16
RTI_ISR_NoNote18:       bra   RTI_ISR_StoreA
RTI_ISR_NotSS:          cmpb  #2
                        bne   RTI_ISR_NotAD
                        bra   RTI_ISR_StoreA
RTI_ISR_NotAD:

RTI_ISR_StoreA:         staa Variable_Note
                        pulb
                                            movb   #1, Bool_NotePlaying
                                            sty           Variable_YHold
                                            cmpa   #250
                                            bne           RTI_ISR_NotDoneYet
                                            movb   #0, Bool_NotePlaying
                                            movb   #0, Bool_SongPlaying
                                            movb   #0, Bool_SongGoing
                                            bra           RTI_ISR_NoSong
RTI_ISR_NotDoneYet:          cmpa   #200
                                            bne           RTI_ISR_NotARest
                                            bra           RTI_ISR_IsARest
RTI_ISR_IsARest:             movb   #0, Bool_NotePlaying
```

```
                                        movb   #1, Bool_ShortRest
                                        bra           RTI_ISR_Continue
RTI_ISR_NotARest:             movb   #0, Bool_ShortRest
                                        movb   #1, Bool_NotePlaying
RTI_ISR_Continue:             ldy    Variable_YHold
                                        ldab   1, Y+
                                        staa   Variable_HoldA
                                        ldaa  Variable_WhichSong
                                        cmpa  #1
                                        bne   RTI_ISR_NotSST
                                        ldaa   #15
                                        mul
                                        ldaa   #25
                                        mul
                                        bra   RTI_ISR_StoreT
RTI_ISR_NotSST:         cmpa  #2
                        bne   RTI_ISR_NotADT
                        ldaa  #15
                        mul
                        stab Variable_HoldB
                        ldab Bool_ShortRest
                        bne         RTI_ISR_RestAlt
                        ldaa  #52
                        bra         RTI_ISR_RestRet
RTI_ISR_RestAlt:              ldaa  #25
RTI_ISR_RestRet:        ldab Variable_HoldB
                                        mul
                        bra   RTI_ISR_StoreT
RTI_ISR_NotADT:         ldaa  #15
                        mul
                        ldaa  #65
                        mul
                                        ldab   Variable_HoldB

RTI_ISR_StoreT:               std           Counter_IntCounter3
                        ldaa Variable_HoldA
                                        sty           Variable_YHold

RTI_ISR_NoSong:         bset  CRGFLG, #$80
                        dec         Counter_IntCounter2
                                        ldaa   Counter_IntCounter2
                                        beq           RTI_ISR_SecondSkip
                                        lbra   RTI_ISR_NoSongP
RTI_ISR_SecondSkip:           movb   #8, Counter_IntCounter1
                                        movb   #8, Counter_IntCounter2
                                        jsr   DCMoto_PluggedIn         ;check if "plugged
in" and if so, change boolean and update speed
                              jsr   DCMoto_BeenTwentyS       ;see if it has been 20s, to
control the DC Motor speed
                              jsr   DCMoto_TurnTheMoto       ;turn the DC motor


                              brset  Bool_WelcomeUser, $01, RTI_ISR_WelUserCurrent
;Controls the Welcome LED (DONE)
                              brset  Bool_WelcomeUser, $02, RTI_ISR_WelUserSkip         ;
                              jsr    State_WelcomeUser
;
```

```
RTI_ISR_WelUserCurrent:      inc     Timer_LEDVariable
       ;
                                            ldaa    Timer_LEDVariable
                      ;
                                            cmpa    #166
                      ;
                                            bne     RTI_ISR_WelUserSkip
       ;
                                            jsr     LED_WelcomeUser
                      ;
                                            inc     Counter_WelUserLED
              ;
                                            ldaa    Counter_WelUserLED
              ;
                                            cmpa    #12
                      ;
                                            bne     RTI_ISR_WelUserSkip
       ;
                                            movb    #0, Port_S
              ;
                                            movb    #2, Bool_WelcomeUser
       ;Ends the Welcome LED calls


RTI_ISR_WelUserSkip:  ldaa Bool_GeneralPauseTF
                                            cmpa #0
                                            beq RTI_ISR_GenPauseSkip            ;Controls the
Ms_Variable timer
                                            ldx Timer_GeneralPause
                                            dex
                                            cpx #0
                                            stx Timer_GeneralPause
;
                                            beq RTI_ISR_GenPauseDone            ;
                                            bra     RTI_ISR_GenPauseSkip
       ;
RTI_ISR_GenPauseDone: movb   #0, Bool_GeneralPauseTF
;Ends the Ms_Variable timer


RTI_ISR_GenPauseSkip: brclr  Bool_LoginActive, $FF, RTI_ISR_NoLoginLED
;Controls of the Login LED (DONE)
                                            brset  Bool_LoginActive, $2, RTI_ISR_NoLoginLED
              ;
                                            jsr     LED_UserLogin

                                            ;Ends the Login LED calls



RTI_ISR_NoLoginLED:     brclr Bool_SongMenu, $FF, RTI_ISR_NoPause                ;Controls
the Song Menu LED
                                            brset Bool_SongPlaying, 01, RTI_ISR_LEDPlay
                        jsr   LED_SongMenu                                        ;Ends the
Song Menu LED

RTI_ISR_LEDPlay:             brclr Bool_SongPlaying, #$FF, RTI_ISR_UserPick
```

```
                                            jsr           LED_SandStormPlay

RTI_ISR_UserPick:            brclr Bool_UserPick, #$FF, RTI_ISR_NoPause
                                            inc           Timer_UserPick


RTI_ISR_NoPause:             brclr Variable_PauseDelay, $FF, RTI_ISR_NoHexAsk
                                            dec    Variable_PauseDelay

RTI_ISR_NoSongLED:      brclr Bool_HexInputAsk, #$FF, RTI_ISR_NoHexAsk
;Controls the Hexpad Input Ask
                                            jsr      Hexpad_GetInput


RTI_ISR_NoHexAsk:       brclr Bool_SongPlaying, #$FF, RTI_ISR_NoSongP
                                            jsr   LCD_UpdateSongTimer
;update sec/min and move the stepper motor if needed

RTI_ISR_NoSongP:
                                            ldaa   Bool_Die
                                            beq           RTI_ISR_End
                                            sei
RTI_ISR_End:
                              rti

;this is the subroutine for playing the song stored in Y. It is out of place currently and
will be moved when done
Song_Play:

                              pshd
                              pshx
                              movb   #1, Bool_SongPlaying
                              movb   #0, Timer_SongSeconds
                              movb   #0, Timer_SongMinutes


                              movb   #1, Bool_SongGoing
                              ;movw  #50000,Timer_GeneralPause

Song_P_Loop:  brclr  Bool_SongGoing, #$FF, Song_P_Pass1         ;if Bool is clr, branch to
exit
                              brset  Bool_SongPaused, #1, Song_P_Paused1                ;if
not, if Bool is 1, branch
                              bra           Song_P_Skip
Song_P_Pass1:  lbra   Song_P_Pass
Song_P_Paused1:       lbra   Song_P_Paused
Song_P_Skip:  ldab   Port_T
;load B with Port T
                              andb   #%00000001
              ;compare bit 1 with 1
                              cmpb   #1
                              lbne   Song_P_NoPause
       ;if not equal, switch not flipped, go to end
                              ldab   #0
                              stx           Variable_XPause
                      ;if occured, then button pressed. Store X in variable
```

```
                              movb    #1, Bool_SongPaused
       ;move 1 to SongPaused to make the song as paused
                              movb    #0, Bool_SongPlaying
;move 0 to SongPlaying to stop the whole song
                              movb    #'P',LCD_SongTimer
                      movb    #'a',LCD_SongTimer+1
                      movb    #'u',LCD_SongTimer+2
                      movb    #'s',LCD_SongTimer+3
                      movb    #'e',LCD_SongTimer+4
                      movb    #'d',LCD_SongTimer+5
                      movb    #' ',LCD_SongTimer+6
                      movb    #' ',LCD_SongTimer+7
                              movb    #' ',LCD_SongTimer+8
                              movb    #' ',LCD_SongTimer+9
                              movb    #' ',LCD_SongTimer+10
                              movb    #' ',LCD_SongTimer+11
                              movb    #' ',LCD_SongTimer+12
                              movb    #' ',LCD_SongTimer+13
                              movb    #' ',LCD_SongTimer+14
                              movb    #' ',LCD_SongTimer+15
                              pshb
                              ldd             #LCD_SongTimer
                              jsr             display_string
                              pulb
                              lbra            Song_P_NoPause
              ;branch to end
Song_P_Paused: ldab    Port_T
;load B with Port T
                              andb    #%00000001
              ;compare bit 1 with 1
                              cmpb    #0
                              lbeq            Song_P_UnPaused

                              ldab    #0
                              lbra            Song_P_NoPause
Song_P_UnPaused:ldx           Variable_XPause
                              movb    #0, Bool_SongPaused
                              movb    #1, Bool_SongPlaying
                              ldab    Variable_SongCurrent
                              cmpb    #1
                              lbne            Song_P_NotSong1
                              movb #'S',LCD_SongTimer
                      movb #'a',LCD_SongTimer+1
                      movb #'n',LCD_SongTimer+2
                      movb #'d',LCD_SongTimer+3
                      movb #'s',LCD_SongTimer+4
                      movb #'t',LCD_SongTimer+5
                      movb #'o',LCD_SongTimer+6
                      movb #'r',LCD_SongTimer+7
                              movb #'m',LCD_SongTimer+8
                      movb #'o',LCD_SongTimer+6
                      movb #'r',LCD_SongTimer+7
                              movb #'m',LCD_SongTimer+8
                              movb #' ',LCD_SongTimer+9
                              movb #' ',LCD_SongTimer+10
                              movb #' ',LCD_SongTimer+11
                              movb #' ',LCD_SongTimer+12
```

```
                             movb #' ',LCD_SongTimer+13
                             movb #' ',LCD_SongTimer+14
                             movb #' ',LCD_SongTimer+15
                             lbra    Song_P_NoPause
Song_P_NotSong1:      cmpb   #2
                             lbne           Song_P_NotSong2
                             movb #'A',LCD_SongTimer
                      movb #'f',LCD_SongTimer+1
                      movb #'t',LCD_SongTimer+2
                      movb #'e',LCD_SongTimer+3
                      movb #'r',LCD_SongTimer+4
                      movb #'n',LCD_SongTimer+5
                      movb #'o',LCD_SongTimer+6
                      movb #'o',LCD_SongTimer+7
                             movb #'n',LCD_SongTimer+8
                             movb #'D',LCD_SongTimer+9
                             movb #'e',LCD_SongTimer+10
                             movb #'l',LCD_SongTimer+11
                             movb #'i',LCD_SongTimer+12
                             movb #'g',LCD_SongTimer+13
                             movb #'h',LCD_SongTimer+14
                             movb #'t',LCD_SongTimer+15
                             lbra    Song_P_NoPause
Song_P_NotSong2: movb #'J',LCD_SongTimer
                      movb #'i',LCD_SongTimer+1
                      movb #'n',LCD_SongTimer+2
                      movb #'g',LCD_SongTimer+3
                      movb #'l',LCD_SongTimer+4
                      movb #'e',LCD_SongTimer+5
                      movb #' ',LCD_SongTimer+6
                      movb #'B',LCD_SongTimer+7
                             movb #'e',LCD_SongTimer+8
                             movb #'l',LCD_SongTimer+9
                             movb #'l',LCD_SongTimer+10
                             movb #'s',LCD_SongTimer+11
                             movb #' ',LCD_SongTimer+12
                             movb #' ',LCD_SongTimer+13
                             movb #' ',LCD_SongTimer+14
                             movb #' ',LCD_SongTimer+15
Song_P_NoPause:       ldab   Port_P
                             andb    #%00100000
                             cmpb    #0
                             lbne    Song_P_Loop
                             movb    #0, Bool_SongGoing
                             movw    #0, Timer_GeneralPause
                             movb    #200, Variable_PauseDelay
Song_P_Temp:   brclr  Variable_PauseDelay, #$FF, Song_P_Passed
                             bra            Song_P_Temp
Song_P_Passed: lbra   Song_P_Loop


Song_P_Pass:
                             movb    #0, Bool_SongPlaying
                             movb    #0, Bool_SongPaused
                             pulx
                             puld
                             rts
```

```
Control_Random:
                                        pshd
                                        pshx
                                        ldab   Timer_UserPick
                                        andb   #$0F
                                        cmpb   #$A
                                        bhs    Control_R_Sandstorm
                                        bra    Control_R_Next1
Control_R_Sandstorm:  movb #1, Variable_WhichSong
                                        bra Control_R_Exit
Control_R_Next1:              cmpb   #5
                                        bls Control_R_AD
                                        bra    Control_R_Next2
Control_R_AD:                 movb #2, Variable_WhichSong
                                        bra    Control_R_Exit
Control_R_Next2:              movb #3, Variable_WhichSong
Control_R_Exit:                       pulx
                                        puld
                                        rts



;this subroutine is called when the song menu is entered. It controls the song menu LCD and
LED displays, as well as manage the commands
Control_SongMenu:
                        pshd
                        pshx
                        pshy
                        movb  #1, Bool_SongMenu


Control_SM_FastLoop:  movb   #1, Bool_SongMenuFast

                                        pshd
                                        pshx
Control_SM_OptionSel:

                                        jsr    read_pot
                                        ldd           pot_value

                                        cpd           #20
                                        bhi           Control_SM_Menu2Check
                                        bra           Control_SM_MenuOne
Control_SM_Menu2Check:       cpd           #40
                                        bhi           Control_SM_Menu3Check
                                        bra           Control_SM_MenuTwo
Control_SM_Menu3Check:       cpd           #60
                                        bhi           Control_SM_Menu4Check
                                        bra           Control_SM_MenuThree
Control_SM_Menu4Check:       cpd           #80
                                        bhi           Control_SM_Menu5Check
                                        bra           Control_SM_MenuFour
Control_SM_Menu5Check:       cpd           #100
                                        bhi           Control_SM_MenuFive
                                        bra            Control_SM_OptionSel
Control_SM_MenuOne:   ldd           #LCD_SongMenu1
                                        jsr            display_string
                                        jsr            Hexpad_GetInput
                                        ldab   Variable_HexpadInput
                                        cmpb   #$F
                                        beq            Control_SM_Sandstorm
                                        bra            Control_SM_OptionSel
Control_SM_MenuTwo:   ldd           #LCD_SongMenu2
                                        jsr            display_string
                                        jsr            Hexpad_GetInput
                                        ldab   Variable_HexpadInput
                                        cmpb   #$F
                                        lbeq   Control_SM_AfternoonDelight
                                        bra            Control_SM_OptionSel
Control_SM_MenuThree:         ldd           #LCD_SongMenu3
                                        jsr            display_string
                                        jsr            Hexpad_GetInput
                                        ldab   Variable_HexpadInput
                                        cmpb   #$F
                                        lbeq   Control_SM_JingleBells
                                        bra            Control_SM_OptionSel
Control_SM_MenuFour:  ldd           #LCD_SongMenu4
                                        jsr            display_string
                                        jsr            Hexpad_GetInput
                                        ldab   Variable_HexpadInput
                                        cmpb   #$F
                                        lbeq   Control_SM_Random
                                        bra            Control_SM_OptionSel

Control_SM_MenuFive:  ldd           #LCD_SongMenu5
                                        jsr            display_string
                                        jsr            Hexpad_GetInput
                                        ldab   Variable_HexpadInput
                                        cmpb   #$F
                                        lbeq   Control_SM_Logout
                                        lbra   Control_SM_OptionSel


Control_SM_Sandstorm:   ldy         #SANDSTORM
                                        sty            Variable_YHold
                                        ldaa  #1
                                        staa  Variable_WhichSong
                                        movb #'S',LCD_SongTimer
                                movb #'a',LCD_SongTimer+1
                                movb #'n',LCD_SongTimer+2
                                movb #'d',LCD_SongTimer+3
                                movb #'s',LCD_SongTimer+4
                                movb #'t',LCD_SongTimer+5
                                movb #'o',LCD_SongTimer+6
                                movb #'r',LCD_SongTimer+7
                                        movb #'m',LCD_SongTimer+8
                                movb #'o',LCD_SongTimer+6
                                movb #'r',LCD_SongTimer+7
                                        movb #'m',LCD_SongTimer+8
                                        movb #' ',LCD_SongTimer+9
```

```
                                          movb #' ',LCD_SongTimer+10
                                          movb #' ',LCD_SongTimer+11
                                          movb #' ',LCD_SongTimer+12
                                          movb #' ',LCD_SongTimer+13
                                          movb #' ',LCD_SongTimer+14
                                          movb #' ',LCD_SongTimer+15
                                   movb #'0',LCD_SongTimer+24
                                   movb #'0',LCD_SongTimer+26
                                          movb #'0',LCD_SongTimer+27
                                          movb #1, Variable_SongCurrent


                                          jsr           Song_Play
                                          bra           Control_SM_AfternoonDelight

Control_SM_AfternoonDelight:       ldy           #AFTERNOONDELIGHT
                                          sty           Variable_YHold
                                          ldaa  #2
                                          staa  Variable_WhichSong
                                          movb #'A',LCD_SongTimer
                                   movb #'f',LCD_SongTimer+1
                                   movb #'t',LCD_SongTimer+2
                                   movb #'e',LCD_SongTimer+3
                                   movb #'r',LCD_SongTimer+4
                                   movb #'n',LCD_SongTimer+5
                                   movb #'o',LCD_SongTimer+6
                                   movb #'o',LCD_SongTimer+7
                                          movb #'n',LCD_SongTimer+8
                                          movb #'D',LCD_SongTimer+9
                                          movb #'e',LCD_SongTimer+10
                                          movb #'l',LCD_SongTimer+11
                                          movb #'i',LCD_SongTimer+12
                                          movb #'g',LCD_SongTimer+13
                                          movb #'h',LCD_SongTimer+14
                                          movb #'t',LCD_SongTimer+15

                                   movb #'0',LCD_SongTimer+24
                                   movb #'0',LCD_SongTimer+26
                                          movb #'0',LCD_SongTimer+27
                                          movb #2, Variable_SongCurrent

                                          jsr           Song_Play
                                          bra           Control_SM_JingleBells

Control_SM_JingleBells:            ldy           #JINGLEBELLS
                                          sty           Variable_YHold
                                          ldaa  #3
                                          staa  Variable_WhichSong
                                          movb #'J',LCD_SongTimer
                                   movb #'i',LCD_SongTimer+1
                                   movb #'n',LCD_SongTimer+2
                                   movb #'g',LCD_SongTimer+3
                                   movb #'l',LCD_SongTimer+4
                                   movb #'e',LCD_SongTimer+5
                                   movb #' ',LCD_SongTimer+6
                                   movb #'B',LCD_SongTimer+7
                                          movb #'e',LCD_SongTimer+8
```

```
                                        movb #'l',LCD_SongTimer+9
                                        movb #'l',LCD_SongTimer+10
                                        movb #'s',LCD_SongTimer+11
                                        movb #' ',LCD_SongTimer+12
                                        movb #' ',LCD_SongTimer+13
                                        movb #' ',LCD_SongTimer+14
                                        movb #' ',LCD_SongTimer+15

                                movb #'0',LCD_SongTimer+24
                                movb #'0',LCD_SongTimer+26
                                        movb #'0',LCD_SongTimer+27
                                        movb #3, Variable_SongCurrent

                                        jsr            Song_Play
                                        lbra   Control_SM_Sandstorm

Control_SM_Random:      movb #0, Bool_UserPick
                                        jsr            Control_Random
                                        ldab   Variable_WhichSong
                                        cmpb #1
                                        bne            Control_SM_NextS1
                                        lbra           Control_SM_Sandstorm
Control_SM_NextS1:            cmpb #2
                                        bne            Control_SM_NextS2
                                        lbra           Control_SM_AfternoonDelight
Control_SM_NextS2:            lbra           Control_SM_JingleBells

Control_SM_Logout:      movb #1, Bool_LogoutRequest
                                        movb #0, Bool_SongMenuFast
                                        pulx
                                        puld
                                        puly
                                        pulx
                                        puld
                        rts

;is subroutine is called every 16ms, increases the DC Motor timer by 1, and if 20 seconds have
passed, calls the subroutine to lower the
;DC Motor speed and then the subroutine to actually decrease the speed of the motor. It takes
Timer_DCTwentySeconds, increments it,
;the sees if it is == to 2000 (==20 seconds). If so, then it calls the subroutine to decrease
the speed of the motor
DCMoto_BeenTwentyS:
                      pshx
                      ldx   Timer_DCTwentySeconds
                      inx
                      cpx   #16629
                      beq   DCMoto_BTS_BeenTime
                      stx   Timer_DCTwentySeconds
                      bra   DCMoto_BTS_NotYet
DCMoto_BTS_BeenTime:  movb  #0, Timer_DCTwentySeconds
                      pulx
                      jsr   DCMoto_SlowTheMoto
                      bra    DCMoto_BTS_Exit
DCMoto_BTS_NotYet:    pulx
DCMoto_BTS_Exit:        rts
```

```
;this subroutine is to determine if the music player is plugged in (that is if bit 7 of the
switches (Port T) is high). this is to run every 1ms and if bit 7
;is high, the Bool_BatteryCharging is set to 1, and the DCMoto_CurrentSpeed is set to $F
DCMoto_PluggedIn:
                                             psha
                                             ldaa   Port_T
                                             anda   #%10000000
                                             cmpa   #$80
                              beq            DCMoto_PI_SevenHigh
                              movb   #0, Bool_BatteryCharging
                              bra    DCMoto_PI_ExitSR
DCMoto_PI_SevenHigh:          movb   #$f, Counter_DCMotoSpeed
                              movb   #1, Bool_BatteryCharging
DCMoto_PI_ExitSR:             pula
                                             rts



;this subroutine controls the DC Motor speed variable, which is to simulate a battery. This
subroutine is called by the DCMoto_BeenTwenty, once every 20 seconds,
;First, the variable DCMoto_CurrentSpeed is pulled in Acc A. This has the number that
corresponds
;to which # of ms of voltage is to be provided to the motor, starting at F and decrementing
down to 1. All this subroutine does is lower that number by 1,
DCMoto_SlowTheMoto:
                       pshd
                       ldaa   Bool_Die
                       beq    DC_NotSone
                       jsr    ZZEND_THE_PROGRAM
DC_NotSone             ldaa  Counter_DCMotoSpeed
                       cmpa  #0
                       bne   DCMoto_SM_NotZero
                       movb   #1, Bool_Die
                       puld
                       jsr   ZZEND_THE_PROGRAM
DCMoto_SM_NotZero:     deca
                                             cmpa #2
                                             bls DCMoto_SM_LowB
                                             bra DCMoto_SM_HighB
DCMoto_SM_LowB:                      movb #'L',LCD_SongTimer+30
                                             movb #'B',LCD_SongTimer+31
                                             pshd
                                             pshx
                                             ldd #LCD_SongTimer
                                             jsr    display_string
                                             pulx
                                             puld
DCMoto_SM_HighB:       staa  Counter_DCMotoSpeed
                       puld
                       rts

;this subroutine actually turns the DCmotor. This is to be called every 0.016s, and it loads
the DCMoto_FifteenCounter and the DCMoto_CurrentSpeed. It increments
;FifteenCounter by 1, and compares FifteenCounter to 15. If == then reset it to zero. Then
compare it with CurrentSpeed, and if it is lower than the
```

```
;CurrentSpeed, it give the motor voltage for that 1ms, then store the Counter back, and exit
subroutine.
DCMoto_TurnTheMoto:
                      pshd
                      ldaa  Counter_DCMotoFifteen
                      inca
                      cmpa  #15
                      bne   DCMoto_TM_NotFifteen    ;branch if the counter is not yet 15
                      ldaa  #0
DCMoto_TM_NotFifteen: ldab  Counter_DCMotoSpeed
                      cba
                      bgt   DCMoto_TM_StopPower     ;if B>A, that is if the Speed>Counter
                      bset  Port_T, #$08            ;give the motor power, and branch to
subroutine end
                      bra   DCMoto_TM_EndOfTM
DCMoto_TM_StopPower:  bclr  Port_T, #$08            ;if reached, stop the power from going to
the motor
DCMoto_TM_EndOfTM:    staa  Counter_DCMotoFifteen
                      puld
                      rts

;this subroutine is the 1ms debouncing delay allowed for the hexkeypad input
Delay_Debounce:
                      pshx
                      ldx   #1000
Delay_DB_Cycle:       dex
                      bne   Delay_DB_Cycle
                      pulx
                      rts


;this subroutine is the main delay function. This is uses the TOI interrupt, and therefore is
to be used for delays that are in the fractions of a second as opposed to the fractions of a
millisecond. This function is to be called immediately after loading one of the preset
Delay_MainUseSet:



;this subroutine is called from the RTI anytime the Bool_HexInputAsk is high. That means the
player is asking the user to enter some information on
;the hexpad. This subroutine starts loading and scanning Port_U for a pull down request. Once
a button input has been detected, it triggers a 1ms debounce
;then enters the key the was pressed into Variable_HexpadInput, sets Bool_HexInputAsk to low,
and then exits the subroutine
Hexpad_GetInput:
                      pshd                                 ;push D and X to get them out of
the way
                      pshx
                      movb   #0, Counter_SM_Hexpad
                      movb  #0, Counter_HexpadPass
                      movb  #0,  Variable_HexpadInput
                      movb  #$88, Variable_HexpadMask      ;set A to mask of %10001000, after
AND mask, this will allow the Pull-Down to occur
                      ldaa  Variable_HexpadMask            ;for the upper nibble
Hexpad_GI_NotFound:   cmpa  #%00010000                     ;check to see if the mask is about
to be erased (with "rora" (rotate A) bit7 is loaded with C bit
                    clc                                 ;clear carry bit so it doesn't mess
up the bne evaluation
                    bne   Hexpad_GI_NoCSet              ;if either of the 1's in the mask
is in danger of being replaced with a 0 (should the C bit be 0)
                    sec                                 ;set the C bit to 1
Hexpad_GI_NoCSet:   rora                                ;rotate A with the C bit
                    anda  #$F0                          ;and A with $F0, ensuring only the
upper nibble is evaluated
                    staa  Port_U                        ;store A into Port_U. The upper
nibble has one 1 and three 0's, and the lower has four 0's
                    jsr     Delay_Debounce
                    ldab  Port_U                        ;load A with Port U, so any changes
can be checked and evaluated
                    cba
                    beq   Hexpad_GI_NotFound1
                      bra  Hexpad_GI_Found
Hexpad_GI_NotFound1:  brset  Bool_SongMenuFast, #1, Hexpad_GI_Except
                                     bra  Hexpad_GI_Found
Hexpad_GI_Except:      inc  Counter_HexpadPass
                                     ldab Counter_HexpadPass
                                     cmpb #4
                                     bne  Hexpad_GI_NotFound
                                     bra  Hexpad_GI_ExceptExit

Hexpad_GI_Found:    stab  Variable_HexTemp
                    ldx   #Array_HexPullCombo           ;load X with the HexPullCombo Array
                    movb  #0, Variable_HexpadCounter    ;set the HexpadCounter to 0, so we
can look through the entire HexpadComba Array
Hexpad_GI_ScanArray:  ldab  1, X+                       ;load B witht he first element of
the Array, moving X to the next
                    cmpb  Variable_HexTemp              ;compare B and with the input
element
                    beq   Hexpad_GI_FoundValue          ;if they are matching, go to
FoundValue
                    pshb                                ;if they do not match, push B, as
we need the 8 bit accumulator
                    ldab  Variable_HexpadCounter        ;load B with the HexpadCounter
                    incb                                ;increment the counter by 1
                    stab  Variable_HexpadCounter        ;store B into the HexpadCounter,
updating it
                    cmpb   #16                          ;compare B with 16, that is to see
if the entire X array has been checked
                    pulb                                ;pull B off the stack, Note: this
does not foul the beq command
                    beq   Hexpad_GI_NotFound            ;if the array has been checked,
then a button has not been pressed yet, so continue scanning
                    bra   Hexpad_GI_ScanArray           ;if reached, then the entire array
has not yet been checked, so load the next number and check it
Hexpad_GI_FoundValue: ldx   #Array_HexPullKey           ;reached only when a value is
found, load X with the HexpadKey
                    ldab  Variable_HexpadCounter        ;load B with the number of items
that were scanned
                    ldaa  B, X                          ;load A with the Bth element of X,
this is the button # that was pressed
                    staa  Variable_HexpadInput          ;store A into the output Variable
                    movb  #0, Bool_HexInputAsk          ;set the InputAsk Boolean to low
                    movb  #1, Bool_GeneralPauseTF
```

```
                      movw  #350, Timer_GeneralPause
Test:                 brset   Bool_GeneralPauseTF, 1, Test
Hexpad_GI_ExceptExit: pulx                                ;pull items off the stack, and exit
the subroutine
                      puld
                      rts


;this subroutine simply tells the user that they did not enter the correct password
LCD_BadPassword:
            pshd
            pshx
                      movb #1,Bool_BadPassword
                      ldd #LCD_BadPassDisp
                      jsr display_string
                      pulx
                      puld
            rts


;this subroutine simply tells the user that the user they requested doesn't exist
LCD_BadUser:
            pshd
            pshx
                      movb #1, Bool_BadUser
                      ldd #LCD_BadUserDisp
                      jsr display_string
                      pulx
                      puld
            rts


;this subroutine begins the login process. It asks the user to press a hexkeypad button
associated with their login, or press '0' to make new user
LCD_LoginRequest:
                             pshd
                             pshx
                        ldd #LCD_LoginUser
                        jsr display_string
                        pulx
                        puld
                             rts


;this subroutine is called from the Login_LoginUser subroutine, each time a password is
requested. It tells the user to enter their password on the first line and
;and on the second line displays 0-3 asterisks, tracking along with the non-zero entries in
the Array_PasswordVariable
LCD_PasswordRequest:
                        pshd
                        pshx
                        ldd #LCD_PassEnter
                        jsr display_string
                        pulx
                        puld
                        rts
```

```
;this subroutine is made to update the minutes part of the time played display of the LED what
shows whenever a song is currently playing.
;     It takes the Timer_SongMinutes and prints it's contents into the minutes part
;     It branches from the Interrupt_RTI SP area
LCD_UpdateMinutes:
                      pshb
                      ldab  Timer_SongMinutes
                      addd   #48
                      stab LCD_SongTimer+24
                      pulb
                      rts



;this subroutine is made to update the seconds part of the time played display of the LED that
shows whenever a song is currently being played.
;     It is to take Timer_SongSeconds and print it's contents into the seconds part
;     It is branched to from the Interrupt_RTI SP area.
LCD_UpdateSeconds:
                      pshd
                      pshx
                      ldab  Timer_SongSeconds
                      ldaa   #0
                      ldx    #10
                      idiv
                      addd   #48
                      stab   LCD_SongTimer+27
                      xgdx
                                      addd  #48
                      stab   LCD_SongTimer+26
                      ldd    #LCD_SongTimer
                      jsr    display_string
                      pulx
                      puld
                      rts



;the subroutine updates the timers for the seconds/minutes and calls the subroutines to have
them updated on the LCD if necessary
;this also calls the stepper motor when needed
LCD_UpdateSongTimer:
                      pshb                          ;push D so we can use the accumulator.
Even though the interrupt automatically pushes all registers, it is safe coding
                      pshx
                      ldab  Bool_SongPlaying        ;code to see if a song is playing, if so,
increment the appropriate counter so it can increase the songplayed timer
                      beq   LCD_UST_EndOfSP            ;if the song is not playing, skip the
next part
                      ldaa    Bool_NoDisplay        ;load A with NoDisp
                      beq     LCD_UST_Show          ;if 0, display is enabled, skip
                      ldaa    Bool_PastState        ;if 1, then display is disabled, load in
PastState
                      bne     LCD_UST_Continue      ;if 1, then display is already disabled,
skip
```

```
                      ldd     #LCD_NoDisplay        ;if 0, then display is not disabled, so
turn it off
                      jsr     display_string
                      ldaa    #1                    ;and set PastState to 1
                      staa    Bool_PastState
                      bra     LCD_UST_Continue      ;continue
LCD_UST_Show:         staa          Bool_PastState


LCD_UST_Continue:     ldx   Timer_SongOneSec
                      inx                          ;if it is, increment it, and see if it has
been 1 second
                      stx    Timer_SongOneSec
                      cpx   #TOFCount
                      bne   LCD_UST_NotASecond      ;if not 1 second yet, skip past SP
                      jsr   STEPMOTO_TurnTheMoto    ;calls the subroutine to move the stepper
motor forward 1 tick
                      ldx    #0                      ;if it has been, set the counter to 0,
store it back, and then have it printer
                      stx    Timer_SongOneSec        ;store B back to the
                      ldaa    Bool_NoDisplay
                      bne     LCD_UST_SkipSec
                      jsr   LCD_UpdateSeconds       ;call the subroutine to print seconds
LCD_UST_SkipSec:      inc    Timer_SongSeconds       ;load and increment the seconds counter
                      ldab   Timer_SongSeconds
                      cmpb  #60                     ;check to see if a minute has gone by
                      bne   LCD_UST_NotAMinute       ;if not, branch to NotAMinute
                      ldab  #0                      ;if so, set seconds to 0 and store
                      stab  Timer_SongSeconds
                      pulx
                      pulb                          ;pull B off the stack
                      inc   Timer_SongMinutes       ;increment the Minute timer
                      ldaa    Bool_NoDisplay
                      bne     LCD_UST_SkipMin
                      jsr   LCD_UpdateMinutes       ;call the subroutine to print the minute
timer
LCD_UST_SkipMin:      bra   LCD_UST_EndOfSP          ;branch to end of SP

LCD_UST_NotAMinute:   stab  Timer_SongSeconds       ;if reached, then it has been a second but
not a minute, store the seconds back and branch to end of SP
                      pulx
                      pulb
                      bra   LCD_UST_EndOfSP

LCD_UST_NotASecond:   stx  Timer_SongOneSec    ;if reached, then it has been an interrpt but
not a second, store the MS timer and branch to end of SP
                      pulx
                      pulb
LCD_UST_EndOfSP:      rts                     ;this is the end of the seconds/minutes song
update routine, and continues with the other interrupt tasks


;this subroutine handles the user login. It is called from the main program. It sets the login
LED going, and then asks the user to enter their user
;number and their password
LCD_UserLogin:
                        pshd
                        pshx
                                                   movb   #1, Bool_LoginActive
                                                   movb   #0, Timer_LEDVariable
                                                   movb  #0, Counter_LEDArray

                                                   jsr           LED_UserLogin
LCD_UL_LoginAgain:      jsr   LCD_LoginRequest              ;here is where the LCD instructs
the user to pick their user number and enter their password
LCD_UL_Restart:
                             jsr   Hexpad_GetInput         ; start running the hexpad input
subroutine
                        jsr   Delay_Debounce
                        jsr   Login_LoginUser
                        ldaa    Bool_WrongUserLogin
                        cmpa    #1
                        beq     LCD_UL_LoginAgain
                        ldaa  Bool_BadUser
                        cmpa  #0
                        beq   LCD_UL_GoodUser
                        jsr   LCD_BadUser
                        bra   LCD_UL_Restart
LCD_UL_GoodUser:        ldaa  Bool_BadPassword
                        cmpa  #0
                        beq   LCD_UL_GoodPassword
                        jsr   LCD_BadPassword
                        bra   LCD_UL_Restart
LCD_UL_GoodPassword:    movb #0, Bool_LoginActive
                                                   pulx
                        puld
                                                   rts


;this subroutine is called only when the player is first turned on. It displays a "Welcome" to
the LCD. To get this message to last for 2 seconds before
;being replaced with the login request, it also sets the Pause_WelcomeDisplay to high (Pause
is a class of Bool). This will create a 2
;second pause (not a delay loop) before the Login_StartupRoutine is called. During this pause,
the LED lights go back and forth.
LCD_WelcomeUser:
                      pshd
                      Ldd #LCD_Welcome
                      jsr display_string
                      movb  #1, Pause_WelcomeDisplay
                      movb  #0, Counter_LEDArray
                      jsr    LED_WelcomeUser
                      puld
                      rts

;this subroutine has the LEDs change in accordance with the musical notes
LED_SandStormPlay:
                             pshd
                             pshx
                             ;ldaa  Bool_NotePlaying
                             ;beq          LED_SP_NoNote
                             ldaa   Variable_Note
                             cmpa   Variable_LEDNote
                             beq           LED_SP_Same
```

```
                              staa   Variable_LEDNote
                              cmpa   #200
                              beq           LED_SP_NoNote
LED_SP_NoRest: ldx            #Array_SandstormLED
                              ldab   A, X
                              stab   Port_S
                              bra           LED_SP_Same
LED_SP_NoNote:  ldab  #$00
                              stab   Port_S
LED_SP_Same:   pulx
                              puld
                              rts

;this is the subroutine that controls the LED movement while the login screen is active. It is
called once when the LCD_Userlogin is called, and
;continued through the RTI as long as the Bool_LoginActive is 1
LED_UserLogin:
                              pshx
                              pshd
                              inc    Timer_LEDVariable
                              brset  Timer_LEDVariable, 166, LED_UL_Go
                              bra    LED_UL_No
LED_UL_Go:                    movb   #0, Timer_LEDVariable
                              ldx    #Array_UserLoginLED
                              ldab   Counter_LEDArray
                              ldaa   B, X
                              cmpb   #0
                              beq    LED_UL_GoUp
                              cmpb   #7
                              beq    LED_UL_GoDown
                              bra    LED_UL_Display1
LED_UL_GoUp:          movb   #0, Bool_UserLogLEDUorD
                              bra    LED_UL_Display1
LED_UL_GoDown:        movb   #1, Bool_UserLogLEDUorD
                              bra    LED_UL_Display1
LED_UL_Display1:              ldab   Bool_UserLogLEDUorD
                              cmpb   #0
                              beq    LED_UL_Display2
                              dec    Counter_LEDArray
                              bra    LED_UL_Display3

LED_UL_Display2:              inc    Counter_LEDArray
LED_UL_Display3:              staa   Port_S
LED_UL_No:                    puld
                              pulx
                              rts

;thu ssubroutine controls the LED while the song menu is active
LED_SongMenu:
                pshx
                pshd
                inc          Timer_LEDVariable
                brset Timer_LEDVariable, 166, LED_SM_Go
                              bra           LED_SM_No
LED_SM_Go:            movb   #0, Timer_LEDVariable
                              ldx           #Array_SongMenuLED
LED_SM_GoBack: ldab   Counter_LEDArray
```

```
                              inc     Counter_LEDArray
                              cmpb    #4
                              bhs             LED_SM_TooHigh
                              ldaa    B, X
                              cmpb    #3
                              bne             LED_SM_Display3
                              movb    #0, Counter_LEDArray
LED_SM_Display3:staa  Port_S
LED_SM_No:            puld
                              pulx
                              rts
LED_SM_TooHigh:       movb    #0, Counter_LEDArray
                              bra             LED_SM_GoBack

;this is the subroutine that controls the LED's when a user first turns on the player. It
cycles through the Welcome User Array by one step per call
LED_WelcomeUser:
                              pshx
                              pshd
                              ldx     #Array_WelcomeUserLED
                              ldab    Counter_WelUserArray
                              ldaa    B, X
                              incb
                              staa    Port_S
                              stab    Counter_WelUserArray
                              puld
                              pulx
                              rts

;this subroutine is called from LCD_UserLogin and occurs after the user has enter a number for
their user login
Login_LoginUser:
                         pshd
                         pshx
                         pshy
                         movb #0, Bool_NewUserLoggedIn
                         movb #0, Bool_BadPassword
                         movb #0, Bool_BadUser
                         ldaa  Variable_HexpadInput              ;load in the Hexpad input
                         cmpa  #$F                               ;compare to F to see in a new
user was requested
                         beq   Login_LU_Skip
                         ldx   #Array_UsersLogPass               ;if not, load in the User Array
                         ldab  #0
Login_LU_FindUser:       ldaa  4, X+                             ;go through the array at 4
numbers at a time to see if the user number has been entered
                         incb
                         cmpb  #13                               ;if the end of the array is
reached, then there is no such user
                         beq   Login_LU_NoSuchUser
                         cmpa  Variable_HexpadInput              ;if the end is not reached,
check if there is such a user number currently
                         beq   Login_LU_UserFound
                         bra   Login_LU_FindUser
Login_LU_UserFound:      jsr   LCD_PasswordRequest
                         dex
                         dex
```

```
                        dex
                        movb  #0, Counter_GeneralCounter       ;initializes the counter to 0,
this is to be for good PS entries
                        movb  #0, Counter_GeneralCounter1      ;this is for bad PS entries
                        movb  #0, Counter_GeneralCounter2
Login_LU_UserLoop:      ldaa  1, X+                             ;if reached, then a user with
the right number was found
                        jsr   Hexpad_GetInput                     ;calls the hexpad subroutine
so a hexpad entry can be collected
                        jsr   Delay_Debounce
                        inc Counter_GeneralCounter2
                        bra   Login_LU_BigSkip
Login_LU_Skip:          bra   Login_LU_NewUserRequest
Login_LU_BigSkip:            pshd
                        pshx
                        ldaa Counter_GeneralCounter2
                        cmpa #1
                        bne  Login_LU_NotFirst
                        ldd  #LCD_PassEnter1
                        jsr display_string
                        bra  Login_LU_DonePW
Login_LU_NotFirst:      cmpa #2
                        bne  Login_LU_DonePW
                        ldd  #LCD_PassEnter2
                        jsr  display_string
Login_LU_DonePW:        pulx
                        puld
                        cmpa  Variable_HexpadInput              ;once the input is received,
compare it to the current password entry
                        beq   Login_LU_GoodPSEntry              ;if correct, branch
                        bra   Login_LU_BadPSEntry               ;if not, then branch
Login_LU_GoodPSEntry:   inc   Counter_GeneralCounter
                        ldab  Counter_GeneralCounter
                        ldaa  Counter_GeneralCounter1
                        aba
                        cmpa  #3
                        beq   Login_LU_PasswordEnded
                        bra   Login_LU_UserLoop
Login_LU_BadPSEntry:    inc   Counter_GeneralCounter1
                        movb  #1, Bool_BadPassword              ;if not, set the
Bool_BadPassword to 1
                        ldaa  Counter_GeneralCounter1
                        ldab  Counter_GeneralCounter
                        aba
                        cmpa  #3
                        beq   Login_LU_PasswordEnded
                        bra   Login_LU_UserLoop
Login_LU_NoSuchUser:    movb  #1, Bool_BadUser
Login_LU_PasswordEnded: movb #1, Bool_UserPick
                                          puly
                                          pulx
                        puld
                        rts

Login_LU_NewUserRequest: ldx #Array_UsersLogPass                         ;load X with the user
array
```

```
                                            ldab    #0
              ;load B with 0
Login_LU_NewUserGrab: ldaa    4, X+                                                   ;load A with
the current X and increment X by 4
                                            incb
              ;increase B
                                            cmpb    #9
              ;compare B to 9
                                            beq            Login_LU_NoNewSpace
       ;if true, tell the user there is no new space
                                            cmpa    #0
              ;see if the current A is 0 (the space is open)
                                            bne            Login_LU_NewUserGrab
;if no space is found, check the next space
                                            pshd
                                            pshx
                                            ldd #LCD_MadeNewUser
                                            jsr display_string
                                            pulx
                                            puld
Login_LU_AskUser:            jsr            Hexpad_GetInput
;if reached, then call the hexpad subroutine
                        pshb
                        ldab    Variable_HexpadInput
                        cmpb    #$F
                        bne     Login_LU_KeepForward
                        movb    #1, Bool_WrongUserLogin
                        pulb
                        bra     Login_LU_PasswordEnded
Login_LU_KeepForward:   pulb
                        ldy         #Array_UsersLogPass                         ;load Y with
the user array
Login_LU_CheckNewUser:       ldab    4, Y+                                                ;load B
with the current Y and increment Y by 4
                                            cmpb    Variable_HexpadInput
;compare B with the variable hexpad input
                                            beq            Login_LU_NewUserUsed
;if the same, the space is taken
                                            cmpb    #0
              ;if reached, compare B to 0 (is the space open?)
                                            bne            Login_LU_CheckNewUser
;if not 0, then go to the next spot
                                            ldab    Variable_HexpadInput                ;if
reached, then spot is open, so take the input
                                            stab    4, -Y
       ;store it in the open spot
                                            pshd
                                            pshx
                                            ldd #LCD_NewPassword
                                            jsr display_string
                                            pulx
                                            puld
                                            ldaa    #0
              ;set A to 0
Login_LU_GetPassword: jsr           Hexpad_GetInput                                   ;get
the input from user
```

```
                                            ldab   Variable_HexpadInput                ;load B
with the input
                                            stab   1, +Y
       ;store it in the password spot
                                            inca
              ;increment A and see if 3 numbers have been gotten
                                            cmpa   #3
                                            bne           Login_LU_GetPassword
;if not, the get another one
                                            movb   #1, Bool_NewUserLoggedIn
                                            bra           Login_LU_PasswordEnded





Login_LU_NewUserUsed: pshd
                        pshx
                        ldd #LCD_UserNumUsed
                        jsr display_string
                        pulx
                        puld
                                            bra           Login_LU_AskUser

Login_LU_NoNewSpace:  jsr           Login_NotMakeNewUser                ;if no new space is
found, tell the user this
                                            movb   #1, Bool_NewUserLoggedIn

                                            bra           Login_LU_PasswordEnded
       ;then exit the subroutine

;this subroutine displays "No space for new user" to the LCD
Login_NotMakeNewUser:
                        pshd
                        pshx
                        ldd #LCD_NoNewSpace
                        jsr display_string
                        movb #1, Bool_GeneralPauseTF
                        movw #1200, Timer_GeneralPause
Login_NMNU_Wait:        brset   Bool_GeneralPauseTF, #1, Login_NMNU_Wait
                                            ldd #LCD_PressToGo
                                            jsr display_string
                                            pulx
                                            puld
                        jsr     Hexpad_GetInput
                                            rts



;this subroutine is called only when 1 second has passed while a song is playing and
increments through the stepper motor rotation commands, 1 per call.
;these commands are controlled by the STEPMOTO_Increment variable, which goes from 0-3, giving
1 command per line to move the motor 1 tick forward
;it is called from the Interrupt_RTI SP area
STEPMOTO_TurnTheMoto:
                      pshd                          ;push D and X to the stack
                      pshx
                      ldab  Counter_StepperControl  ;load B with the current counter
                      ldx   #Array_StepperControl   ;load X with the stepper array
                      incb                          ;increment B by 1
                      cmpb  #4                      ;if the end of the stepper array would be
passed, go to Around
                      beq   STEPMOTO_TM_Around
STEPMOTO_TM_Execute:  stab  Counter_StepperControl  ;load B to update the control counter
                                 ldaa  B, X                    ;load A with the Bth
element of X
                      staa  Port_P                  ;store A into Port_P to turn the motor 1
turn
                      bra   STEPMOTO_TM_EndOfSM     ;after storing it, go finish

STEPMOTO_TM_Around:   ldab  #0                      ;reached if the stepmotor array is
finished, go back to 0 and start again
                      bra   STEPMOTO_TM_Execute
STEPMOTO_TM_EndOfSM:  pulx                          ;finished with stepmotor routine, pull off
stack and rts
                      puld
                      rts

;this subroutine sets the Bool_WelcomeUser to 2, so it never runs again. It then calls the
subr to print "Welcome" to the LCD, and starts the
;first LED call
State_WelcomeUser:
                             movb #1, Bool_WelcomeUser
                             jsr    LCD_WelcomeUser
                             jsr    LED_WelcomeUser
                             rts


;this subroutine ends the program
ZZEND_THE_PROGRAM:   nop
                                    sei
                     end                                ;this is never reached unless the
battery dies or the player is turned off
```