INCLUDE 'derivative.inc'

XDEF Entry, \_Startup, BIGBOI,song1,IRQButton

XREF \_\_SEG\_END\_SSTACK, init\_LCD, display\_string, pot\_value, read\_pot, SendsChr, PlayTone ; symbol defined by the linker for the end of the stack

my\_variable: SECTION ;set variables for the rest of the code

disp ds.b 33

ledson ds.b 1

localinledseq ds.w 1

bigpress ds.b 1

storage ds.b 1

onessec ds.b 1

tenssec ds.b 1

minutes ds.b 1

user1 ds.b 8

user2 ds.b 8

user3 ds.b 8

deleted ds.b 1

songselect ds.b 1

secpassed ds.b 1

xstorage ds.b 1

displayon ds.b 1

dispon ds.b 1

digitcount ds.b 1

songtimer ds.b 1

notetime ds.b 1

batttimer ds.b 1

ton ds.b 1

toff ds.b 1

shutoffvar ds.b 1

screenofftime ds.b 1

xcounterforthetimer ds.w 1

ystorage ds.w 1

gotosong1now ds.b 1

checkervalue ds.b 1

checkingfirstorsecondpress ds.b 1

stepperseqlocal ds.w 1

secondstorage ds.b 1

countertocomparetotimer ds.w 1

constants: SECTION

portt equ $240 ;define port t

porttddr equ $242 ;define port t ddr

ports equ $248 ;define port s

portsddr equ $24a ;define port s ddr

portu equ $268 ;define port u

ddru equ $26a ;define port u ddr

psru equ $26d ;define port u pull select register

pderu equ $26c ;define port u

portp equ $258 ;define port u pull down

portpddr equ $25A ;define port p ddr

steppersequence dc.b $0a,$0a,$0a,$0a,$0a,$0a,$12,$12,$12,$12,$12,$12, $14, $14,$14, $14,$14, $14,$0c,$0c,$0c,$0c,$0c,$0c ;stepper motor sequence

dcsequence dc.b $70,$b0,$d0,$e0 ;sequence for DC motor

keyseq dc.b $70, $B0, $D0, $E0 ;define the keypad sequence

keypadval dc.b $eb, $77, $7b, $7d, $b7, $bb, $bd, $d7, $db, $dd, $e7, $ed, $7e, $be, $de, $ee ;sequence of key pad values

padval dc.b $0, $1, $2, $3, $4, $5, $6, $7, $8, $9, $a, $b, $c, $d, $e, $f ;values the key pad can resend

LEDseq dc.b $80,$80, $80,$80, $80,$80,$80,$80, $80,$80, $80,$80, $40,$40, $40,$40, $40,$40, $40,$40, $40,$40, $40,$40, $20,$20, $20,$20, $20,$20, $20,$20, $20,$20, $20,$20, $10,$10, $10,$10, $10,$10,$10,$10, $10,$10, $10,$10, $08,$08, $08,$08, $08,$08,$08,$08, $08,$08, $08,$08, $04,$04, $04,$04, $04,$04,$04,$04, $04,$04, $04,$04, $02,$02, $02,$02, $02,$02, $02,$02, $02,$02, $02,$02, $01,$01, $01,$01, $01,$01,$01,$01, $01,$01, $01,$01, $02,$02, $02,$02, $02,$02, $02,$02, $02,$02, $02,$02, $04,$04, $04,$04, $04,$04, $04,$04, $04,$04, $04,$04, $08,$08, $08,$08, $08,$08,$08,$08, $08,$08, $08,$08, $10,$10, $10,$10, $10,$10,$10,$10, $10,$10, $10,$10, $20,$20, $20,$20, $20,$20, $20,$20, $20,$20, $20,$20, $40,$40, $40,$40, $40,$40,$40,$40, $40,$40, $40,$40 ;sequence for the led

user4 dc.b 1,2,3,4,5,6,7,8 ;predefinined user

user5 dc.b 9,$A,$b,$c,$d,$e,$f,0 ;predefined user

sandstormsequence dc.b $9F, $8F, $7F, $6F, $5F, $4F, $3F, $2F, $F ;note sequence for sandstorm

pokemonsequence dc.b $1B, $1B, $1B, $1B, $1B, $16, $12, $15, $14, $1B, $1B, $16, $F, $16, $18, $18, $18, $1B, $16, $F, $F, $1B, $1B, $16, $F, $1B ;not sequence for pokemon theme song

intcr equ $1E

;start of main

MyCode: SECTION

Entry:

\_Startup:

LDS #\_\_SEG\_END\_SSTACK ;set the stack

movb #%00001000, porttddr ;set output of port t

movb #%00011110, portpddr ;set output of port p

movw #0000, localinledseq ;set initial value of variables

movw #0000, stepperseqlocal

movw #0000, countertocomparetotimer

clr secondstorage

clr inbetweennotes

clr notetime

clr bigpress

clr user1

clr user2

clr user3

clr deleted

clr songselect

clr ledson

clr secpassed

clr displayon

clr dispon

clr ystorage

clr xstorage

clr digitcount

clr songtimer

clr shutoffvar

clr xcounterforthetimer

clr gotosong1now

movb #$F0, ddru ;set ddr of port u

movb #$F0, psru ;set pull select register of port u

movb #$0F, pderu ;set pull down of port u

movb #$FF, portsddr ;define s as output

movb #$35, RTICTL ;set interruptt frequency

movb #$80, CRGINT ;enable RTI

movb #%00100000,porttddr ;set output of portt

movb #%01000000,intcr ;set IRQ button

movb #$1e,portpddr ;set output of portp

CLI ;enable interrupts

OHWOW: JSR johnlee1 ;loads introduction to LCD

ldaa shutoffvar ;load the shutoff variable

cmpa #1 ;compare variable to 1

beq OHWOW ;if 1, don't continue with program

Logins: JSR Keypad ;runs keypad to get value

ldaa bigpress ;load keypad val to A

cmpa #1 ;compare a to 1

beq newuserlogin ;if 1, branch to new user

cmpa #2 ;compare a to 2

beq userlogin1 ;if 2, branch to previous user

bra Logins ;if neither, run again

userlogin1: jmp userlogin ;jump to userlogin

newuserlogin:

JSR johnlee2 ;display instructions to LCD

ldaa user1 ;load user1 to a

cmpa #0 ;compare a to 0

beq startit1 ;if equal then start new login

ldaa user2 ;load user2 to a

cmpa #0 ;compare a to 0

beq startit2go ;if equal then start new login

ldaa user3 ;load user3 to a

cmpa #0 ;compare a to 0

beq startit3go ;if equal then start new login

bne writeover ;if none are free, start erasing process

startit2go: jmp startit2 ;jump to new user 2

startit3go: jmp startit3 ;jump to new user 3

writeover: ldaa deleted ;load the deleted value to a

cmpa #0 ;compare to 0

beq delete1 ;if 0 delete 1

cmpa #1 ;compare to 1

beq delete2 ;if 1 delete 2

cmpa #2 ;compare to 2

beq delete3 ;if 2 delete 3

bne uhoh ;if mistake, run this

uhoh: clr deleted ;clear deleted

bra writeover ;run the writeover again

delete1: ldy #0 ;load 0 to y

lupe1: ldaa #0 ;load 0 to a

staa user1,y ;clears user location specified by y

cpy #8 ;compare to see if done

beq soon1 ;if done, jump

iny ;if not increase y

bra lupe1 ;run again

delete2: ldy #0 ;load 0 to y

lupe2: ldaa #0 ;load 0 to a

staa user2,y ;clears user location specified by y

cpy #8 ;compare to see if done

beq soon2 ;if done, jump

iny ;if not increase y

bra lupe2 ;run again

delete3: ldy #0 ;load 0 to y

lupe3: ldaa #0 ;load 0 to a

staa user3,y ;clears user location specified by y

cpy #8 ;compare to see if done

beq soon3 ;if done, jump

iny ;if not increase y

bra lupe3 ;run again

soon1: ldaa deleted ;load deleted to a

adda #1 ;increase a

staa deleted ;store back to deleted

bra startit1 ;jump to start

soon2: ldaa deleted ;load deleted to a

adda #1 ;increase a

staa deleted ;store back to deleted

bra startit2get ;jump to start

startit2get:

jmp startit2 ;jump to beginning of new user 2

soon3: clr deleted ;clear deleted variable

bra startit3get ;jump to start

startit3get: jmp startit3 ;jump to beginning of new user 3

startit1: ldx #0 ;load 0 to x

recheck1: stx xstorage ;store value of x

JSR Keypad ;run keypad

ldaa bigpress ;load pressed value

ldx xstorage ;load x with pointer

staa user1,x ;load press to user

cpx #0 ;check pointer value

beq firsta1 ;if 0 follow branch

cpx #1 ;check pointer value

beq seconda1 ;if 1 follow branch

cpx #2 ;check pointer value

beq thirda1 ;if 2 follow branch

cpx #3 ;check pointer value

beq fourtha1 ;if 3 follow branch

bne startit1 ;if it's broke, send it back

firsta1:

movb #'x',disp+9 ;insert an x to location

bra continuation1 ;branch out

seconda1:

movb #'x',disp+10 ;insert an x to location

bra continuation1 ;branch out

thirda1:

movb #'x',disp+11 ;insert an x to location

bra continuation1 ;branch out

fourtha1:

movb #'x',disp+12 ;insert an x to location

bra continuation1 ;branch out

continuation1:

ldd #disp

pshx ;push x

JSR display\_string ;run display

pulx ;pull x

inx ;increase x

cpx #4 ;compare x to 4

bne recheck1 ;if not 4 keep checking for values

beq forward1 ;if 4 move to password

forward1:

stx xstorage ;store new pointer value

JSR Keypad ;run keypad

ldaa bigpress ;load pressval

ldx xstorage ;load pointer

staa user1,x ;store to user location

cpx #4 ;check pointer value

beq fiftha1 ;if 0 follow branch

cpx #5 ;check pointer value

beq sixtha1 ;if 0 follow branch

cpx #6 ;check pointer value

beq seventha1 ;if 0 follow branch

cpx #7 ;check pointer value

beq eightha1 ;if 0 follow branch

bne startit1go ;if it's broke, send it back

startit1go: jmp startit1 ;jump back

fiftha1:

movb #'x',disp+25 ;insert an x to location

bra continuationa1 ;branch out

sixtha1:

movb #'x',disp+26 ;insert an x to location

bra continuationa1 ;branch out

seventha1:

movb #'x',disp+27 ;insert an x to location

bra continuationa1 ;branch out

eightha1:

movb #'x',disp+28 ;insert an x to location

bra continuationa1 ;branch out

continuationa1:

ldd #disp ;load d value

pshx ;push x

JSR display\_string ;run display sequence

pulx ;pull x

inx ;increase x

cpx #8 ;compare with string size

bne forward1 ;not equal then run again

beq logintime1 ;equal then login

logintime1: jmp logintime ;jump to login

startit2: ldx #0 ;this section of code follows the same logic as startit1 but for user 2

recheck2: stx xstorage

JSR Keypad

ldaa bigpress

ldx xstorage

staa user2,x

cpx #0

beq firsta2

cpx #1

beq seconda2

cpx #2

beq thirda2

cpx #3

beq fourtha2

bne startit2

firsta2:

movb #'x',disp+9

bra continuation2

seconda2:

movb #'x',disp+10

bra continuation2

thirda2:

movb #'x',disp+11

bra continuation2

fourtha2:

movb #'x',disp+12

bra continuation2

continuation2:

ldd #disp

pshx

JSR display\_string

pulx

inx

cpx #4

bne recheck2

beq forward2

forward2: stx xstorage

JSR Keypad

ldaa bigpress

ldx xstorage

staa user2,x

cpx #4

beq fiftha2

cpx #5

beq sixtha2

cpx #6

beq seventha2

cpx #7

beq eightha2

bne yeehaw2

yeehaw2: jmp startit2

fiftha2:

movb #'x',disp+25

bra continuationa2

sixtha2:

movb #'x',disp+26

bra continuationa2

seventha2:

movb #'x',disp+27

bra continuationa2

eightha2:

movb #'x',disp+28

bra continuationa2

continuationa2:

ldd #disp

pshx

JSR display\_string

pulx

inx

cpx #8

bne forward2

beq logintime2

logintime2: jmp logintime

startit3: ldx #0 ;this section of code follows the same logic as startit1 but for user 3

recheck3: stx xstorage

JSR Keypad

ldaa bigpress

ldx xstorage

staa user3,x

cpx #0

beq firsta3

cpx #1

beq seconda3

cpx #2

beq thirda3

cpx #3

beq fourtha3

bne startit3

firsta3:

movb #'x',disp+9

bra continuation3

seconda3:

movb #'x',disp+10

bra continuation3

thirda3:

movb #'x',disp+11

bra continuation3

fourtha3:

movb #'x',disp+12

bra continuation3

continuation3:

ldd #disp

pshx

JSR display\_string

pulx

inx

cpx #4

bne recheck3

beq forward3 ;

forward3:stx xstorage

JSR Keypad

ldaa bigpress

ldx xstorage

staa user3,x

cpx #4

beq fiftha3

cpx #5

beq sixtha3

cpx #6

beq seventha3

cpx #7

beq eightha3

bne yeehaw3

yeehaw3: jmp startit3

fiftha3:

movb #'x',disp+25

bra continuationa3

sixtha3:

movb #'x',disp+26

bra continuationa3

seventha3:

movb #'x',disp+27

bra continuationa3

eightha3:

movb #'x',disp+28

bra continuationa3

continuationa3:

ldd #disp

pshx

JSR display\_string

pulx

inx

cpx #8

bne forward3

beq logintime3

logintime3: jmp logintime ;go to login

logintime bra userlogin ;go to user login

userlogin: JSR johnlee3 ;load LCD display

JSR Keypad ;run keypad

ldaa user1 ;load user1 to a

cmpa bigpress ;compare a to press value

beq login1 ;if equal, run login1

ldaa user2 ;load user2 to a

cmpa bigpress ;compare a to press value

beq login2 ;if equal, run login2

ldaa user3 ;load user3 to a

cmpa bigpress ;compare a to press value

beq login3 ;if equal, run login3

ldaa user4 ;load user3 to a

cmpa bigpress ;compare a to press value

beq login4 ;if equal, run login3

ldaa user5 ;load user3 to a

cmpa bigpress ;compare a to press value

beq login51 ;if equal, run login3

bne senditback ;if none of the three, not a valid login

senditback: JSR loginerror ;run login error message

JSR shhhhdonttell ;let loginerror linger

jmp OHWOW ;back to the beginning

login51: jmp login5 ;jump to user login5

login1: ldx #1 ;load 1 to x

goagain1: stx xstorage ;store the value to x storage

JSR Keypad ;run keypad

ldx xstorage ;load x with the storage value

ldaa user1,x ;load user 1 to a

cmpa bigpress ;compare to pressed value

beq onwardho1 ;if equal, keep going

bne senditback ;otherwies, send it back

onwardho1: inx ;increase x

cpx #8 ;compare to see if finished

beq tologgedin ;if finished login

bne goagain1 ;if not, keep logging in

tologgedin: jmp loggedin ;jump to logged in

login2: ldx #1 ;load 1 to x

goagain2: stx xstorage ;store value to x storage

JSR Keypad ;run keypad

ldx xstorage ;load x storage value

ldaa user2,x ;load user 2 to a

cmpa bigpress ;compare to pressed value

beq onwardho2 ;if equal, keep going

bne senditback ;otherwies, send it back

onwardho2: inx ;increase x

cpx #8 ;compare to see if finished

beq loggedin ;if finished login

bne goagain2 ;if not, keep logging in

login3: ldx #1 ;load 1 to x

goagain3: stx xstorage ;store value to xstorage

JSR Keypad ;run keypad

ldx xstorage ;load x storage value

ldaa user3,x ;load user 3 to a

cmpa bigpress ;compare to pressed value

beq onwardho3 ;if equal, keep going

bne senditback ;otherwies, send it back

onwardho3: inx ;increase x

cpx #8 ;compare to see if finished

beq loggedin ;if finished login

bne goagain3 ;if not, keep logging in

login4: ldx #1 ;load 1 to x

goagain4: stx xstorage ;store the value to x storage

JSR Keypad ;run keypad

ldx xstorage ;load x storage value

ldaa user4,x ;load user 3 to a

cmpa bigpress ;compare to pressed value

beq onwardho3 ;if equal, keep going

bne senditback69 ;otherwies, send it back

onwardho4: inx ;increase x

cpx #8 ;compare to see if finished

beq loggedin ;if finished login

bne goagain4 ;if not, keep logging in

login5: ldx #1 ;load 1 to x

goagain5: stx xstorage ;store value to x storage

JSR Keypad ;run keypad

ldx xstorage ;load x storage value

ldaa user5,x ;load user 3 to a

cmpa bigpress ;compare to pressed value

beq onwardho5 ;if equal, keep going

bne senditback69 ;otherwies, send it back

onwardho5: inx ;increase x

cpx #8 ;compare to see if finished

beq loggedin ;if finished login

bne goagain5 ;if not, keep logging in

senditback69: ;branch for jumping

jmp senditback ;jump back in code

loggedin: JSR Introduction ;say hi to the kids

JSR shhhhdonttell ;jump to subroutine

movb #1,ledson ;turn leds off

switchchk: ldaa portt ;load port t value

cmpa %00000001 ;compare it to on

bne selectsong ;if bit is clear, look at song select

beq playsong ;if bit is set, play song

selectsong:

potread1: clr dispon ;clear the display on value for playing

JSR read\_pot ;read the pot value

ldd pot\_value ;load pot value to d

cpd #40 ;compare pot value

ble daruderead ;if in that region, show darude

bne potread2 ;if not, read next val

potread2: cpd #80 ;compare pot val

ble allstarread ;if in region, show allstar

bne potread3 ;if not read next val

potread3: bra pokemonread ;if not in other two, show pokemon theme song

daruderead: ldaa displayon ;load display on to a

cmpa #1 ;compare to 1

beq daruderead1 ;if 1, then run through the darude read

JSR Daruder ;display darude info on lcd

ldaa #1 ;load 1 to a

staa displayon ;store it to the display select value

daruderead1:

clr songselect ;clear the song select variable

bra switchchk ;branch to switchchk

allstarread:

ldaa displayon ;load display on value

cmpa #2 ;compare to 2

beq allstarread1 ;if 2, run allstar read

JSR allstarr ;display allstar on lcd

ldaa #2 ;load 2 to a

staa displayon ;store to reading variable

allstarread1:

movb #1,songselect ;set songselect variable to 1

bra switchchk ;branch to switchchk

pokemonread:

ldaa displayon ;load display on value to a

cmpa #3 ;compare to 3

beq pokemonread1 ;if 3, then run pokemon read

JSR pokemonr ;display pokemon theme on lcd

ldaa #3 ;load 3 to a

staa displayon ;store 3 to reading variable

pokemonread1:

movb #2,songselect ;set songselect variable to 2

bra switchchk ;branch to switchchk

playsong:

clr displayon ;clear the variable for song select display

ldaa portt ;load port t value to a

cmpa %00000001 ;compare with bit on value

beq playing ;if bit set, playing song

bne pausing ;if bit clear, song paused

playing: ldaa songselect ;load songselect variable to a

cmpa #0 ;compare a to 0

beq darudeplay ;if equal, play darude

cmpa #1 ;compare a to 1

beq allstarplay ;if equal, play all star

cmpa #2 ;compare a to 2

beq pokemonplay ;if equal, play pokemon

bra switchchk ;if for some ungodly reason it doesnt work, sorry

pausing:

bra playsong ;branch to song playing

darudeplay: ldaa dispon ;load song select value

cmpa #1 ;compare with 1

beq switchchk11 ;if 1, jump to switch check

JSR Darudep ;display darude playing

ldaa #1 ;load 1 to a

staa dispon ;store to song select value

bra switchchk11 ;branch to switch check

allstarplay:

ldaa dispon ;load song select value

cmpa #2 ;compare to 2

beq switchchk11 ;if two, jump to switch check

JSR allstarp ;display allstar playing

ldaa #2 ;load 2 to a

staa dispon ;store it to the song select value

bra switchchk11 ;branch to switch check

pokemonplay:

ldaa dispon ;load display value to a

cmpa #3 ;compare to 3

beq switchchk11 ;if 3, branch to switch check

JSR pokemonp ;display pokemon playing

ldaa #3 ;load 3 to a

staa dispon ;store to song select value

bra switchchk11 ;branch to switch check

switchchk11:

JSR SongTiming ;run song timing display code

bra playsong ;jump back to see if song still playing

;end of main

;start keypad and debouncers

shhhhdonttell:

ldy #$FFFF ;load a lot to y

shush: dey ;decrement y

cpy #0 ;compare y to 0

bne shush ;if not 0, keep running

ldy #$FFFF

shuddup: dey

cpy #0

bne shuddup

ldy #$FFFF

shadup: dey

cpy #0

bne shadup

ldy #$FFFF

shedep: dey

cpy #0

bne shedep

ldy #$FFFF

shtinko: dey

cpy #0

bne shtinko

ldy #$FFFF

plinko: dey

cpy #0

bne plinko

ldy #$FFFF

binko: dey

cpy #0

bne binko

RTS ;done running

BOUNCY: ldy #2000 ;load value for 1ms

bigdag: dey ;decrement the value of y

cpy #0 ;compare y to 0

bne bigdag ;if y isnt zero yet, branch to beginning

RTS ;return from subroutine

Keypad:

goagain: ldx #0 ;load 0 to x

mrloop: cpx #5 ;see if sequence is complete

beq goagain ;if complete, restart

bne cont ;if incomplete, continue

cont: ldaa keyseq,x ;load sequence value to a

staa portu ;store the sequence value to u

Jsr BOUNCY ;jump to delay for debounce

inx ;increase x

ldaa portu ;load value of port u to a

staa storage ;save the value in storage1

anda #%00001111 ;mask the upper nibble

cmpa #$F ;compare the lower nibble to see if no button pressed

beq mrloop ;if button not pressed, run through next row

bne nextpart ;if button pressed, check which button

nextpart: ldy #0 ;load 0 to y

ohdaddy: cpy #$10 ;compare y to see how far through sequence

beq goagain ;if value is not in sequence, jump to end

bne yus ;if sequence not complete, continue

yus: ldaa storage ;load a with storage

cmpa keypadval,y ;compare the value in a with the possible keypad values

beq storeitplz ;if equal, store it

bne increaseit ;if not equal, increase y

increaseit: iny ;increase y

bra ohdaddy ;run it again

storeitplz: ldaa padval,y ;put new value into lower nibble

staa bigpress ;stores the pressed keypad value

endgame: ldaa storage ;load storage value to a

ldab portu ;load port u value to b

cba ;compare a and b

beq endgame ;if equal, no new press, keep waiting

bne tookawhile ;if not equal, button released

tookawhile: ldaa storage ;run again for good measure

ldab portu

cba

beq endgame

RTS ;return to main code

;end of keypad and debouncers

;start interrupt

BIGBOI:

;LEDS

ldx localinledseq ;load the location in the led sequence value

ldaa ledson ;load the checking value

cmpa #1 ;compare the checking value to see if leds should be on

beq pushbutton ;if set, leds shouldnt run

ldaa LEDseq,X ;load value from sequence to accumulator a

staa ports ;store value of A to port s LEDs

cpx #164 ;check to see reg x has run through the sequence

beq xtozero ;if x is finished, restart the scan for which sequence

INX ;increase x

stx localinledseq ;store new value of location in sequence

bra enditalllol ;increase x if not finished with sequence

xtozero: ldx #0 ;load zero to x

stx localinledseq ;set value of sequence location back to zero

bra enditalllol ;branch to end of interrupt

enditalllol: ;jump code

jmp enditall ;jump to the end of the code

;PBButton

pushbutton:

ldaa #0 ;load 0 to a

staa ports ;shut off leds

brset portt,#%00100000, pbpress ;check if PB is pressed

bra Songtime ;otherwise branch to song

pbpress:

staa ports ;store a to port s

movw #0, ystorage ;move 0 to y storage for song

;Songs

Songtime:

ldaa portt ;load a with port t

ldab checkervalue ;load b with value to check

cmpb #0 ;compare b to 0

beq dontskip ;if 0, keep doing song

bne skip ;if 1, skip song

comeherepeasants:

cmpa #0 ;compare a to 0

beq enditalllol ;if port t in a is 0, no song playing jump to end

ldaa songselect ;load a with song select value

ldab gotosong1now ;load b with value to go to song

cmpb #0 ;compare b to 0

beq song1first ;if equal, go to song 1

cmpa #0 ;compare a to 0

beq song1 ;if equal go to song one

cmpa #1 ;compare a to 1

beq song2jump ;if equal go to song 2 jump

bne song3jump ;if not 1 or 2 go to song 3 jump

bra enditall69 ;if something doesnt work, jump out of RTI

enditall68:

incb ;increase the value of b

stab notetime ;store it into notetimer

bra enditall69 ;branch to jump

enditall69: jmp enditall ;jump to end of RTI

song3jump: jmp song3 ;jump to song 3

song2jump: jmp song2 ;jump to song 2

dontskip:

movb #1,checkervalue ;move 1 to checker value

bra comeherepeasants ;branch back

skip:

movb #0,checkervalue ;move 0 to checker value

bra enditall69 ;jump out of RTI

song1first:

movw #0,ystorage ;set ystorage value to 0

movb #1,gotosong1now ;set gotosong1now value to 1

ldy ystorage ;load ystorage value to y

song1:

ldy ystorage ;load ystorage value to y

cpy #8 ;compare to 8

bgt resetywhyinthefuckareyousohighdoyouhaveanyideawhattimeitismister ;if its greater than 8, branch

ldaa sandstormsequence,y ;load a with the correct sandstorm sequence value

psha ;push a to use the sequence value

JSR SendsChr ;run the sendschr subroutine

pula ;pull a back

JSR PlayTone ;run the play tone subroutine

cpy #8 ;compare y to 8

bgt resetywhyinthefuckareyousohighdoyouhaveanyideawhattimeitismister ;if greater than 8, branch

iny ;increase y

sty ystorage ;store new value to ystorage

cpy #8 ;compare y to 8 again

beq resetyouryvalue ;if it's equal to 8, branch

bra inbetweennotes ;otherwise, branch

resetywhyinthefuckareyousohighdoyouhaveanyideawhattimeitismister:

movw #0,ystorage ;set ystorage value back to 0

bra inbetweennotes ;branch to inbetween notes

resetyouryvalue:

movw #0,ystorage ;set ystorage value to 0

bra inbetweennotes ;branch to inbetween nbotes

;timing

inbetweennotes:

ldx xcounterforthetimer ;load xcounter to x

inx ;increase value

stx xcounterforthetimer ;store back

CPX #$1000 ;compare to $1000 hex

bne assemblysuseofthejumpcommandmakesmewanttojumpoffabridge ;if not equal, branch

movw #0,xcounterforthetimer ;otherwise set variable to 0

ldaa onessec ;load one second value

adda #1 ;add one

staa onessec ;store back to one second

cmpa #10 ;compare seconds to 10

bne assemblysuseofthejumpcommandmakesmewanttojumpoffabridge ;if not 10, continue

movb #0,onessec ;reset variable

ldaa tenssec ;load tens value for seconds

adda #1 ;add one

staa tenssec ;store it back

cmpa #6 ;compare it to 6

bne assemblysuseofthejumpcommandmakesmewanttojumpoffabridge ;if 6, branch

movb #0,tenssec ;set variable to 0

ldaa minutes ;load minutes to a

bra assemblysuseofthejumpcommandmakesmewanttojumpoffabridge ;branch down

assemblysuseofthejumpcommandmakesmewanttojumpoffabridge:

bra DCSpeedLoop ;jump to dc motor

;DC MOTOR

DCSpeedLoop:

bset portt,#$8 ;set bit 3 of port t

bra enditall96 ;branch to end

ble keepspinningsheep ;branch to spinning

bra stopspinning ;branch to battery dead

WHYDIDTHECHICKENCROSSTHEROADBECAUSEMYPARENTSDONTLOVEME:

keepspinningsheep:

ldx countertocomparetotimer ;load x with counter

inx ;increase x

stx countertocomparetotimer ;store x

bra steppermotortime ;branch to stepper motor

stopspinning:

bclr portt,#$8 ;clear bit 3 of port t

bra steppermotortime ;branch to stepper motor

enditall96: jmp enditall ;jump to end

;steppermoatur

steppermotortime:

runthroughagain:

ldx stepperseqlocal ;load x with stepper sequence pointer

ldaa steppersequence,x ;load a with stepper sequence value

staa portp ;store to port p

inx ;increase pointer

stx stepperseqlocal ;store to variable

cpx #24 ;compare to see if finished

beq seetuup ;if finished branch

bne enditall ;if not finished, branch to end

runit5ever:

bra enditall ;branch to end

seetuup:

movw #0,stepperseqlocal ;set pointer to 0

bra enditall ;branch to end

branchout:

movw #0,stepperseqlocal ;set pointer to 0

bra enditall ;branch to end

itsnotadelay:

ldy #30 ;load value to y

stinkypoo:

cpy 0 ;compare to 0

beq thereturn ;if equal, jump to end

dey ;decrement y

bra stinkypoo ;branch back

thereturn:

bra enditall ;jump to end

;END

enditall: BSET CRGFLG, $80 ;set the interrupt flag agagin

RTI ;exit interrupt

;end of interrupt

;start of LCD display options

johnlee1: ;welcome screen

movb #'N',disp

movb #'e',disp+1

movb #'w',disp+2

movb #' ',disp+3

movb #'U',disp+4

movb #'s',disp+5

movb #'e',disp+6

movb #'r',disp+7

movb #'?',disp+8

movb #' ',disp+9

movb #'P',disp+10

movb #'u',disp+11

movb #'s',disp+12

movb #'h',disp+13

movb #' ',disp+14

movb #'1',disp+15

movb #'L',disp+16

movb #'o',disp+17

movb #'g',disp+18

movb #'i',disp+19

movb #'n',disp+20

movb #'?',disp+21

movb #' ',disp+22

movb #'P',disp+23

movb #'u',disp+24

movb #'s',disp+25

movb #'h',disp+26

movb #' ',disp+27

movb #'2',disp+28

movb #' ',disp+29

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

johnlee2: ;new user screen

movb #'U',disp

movb #'s',disp+1

movb #'e',disp+2

movb #'r',disp+3

movb #'n',disp+4

movb #'a',disp+5

movb #'m',disp+6

movb #'e',disp+7

movb #':',disp+8

movb #' ',disp+9

movb #' ',disp+10 ;username input starting here

movb #' ',disp+11 ;username in

movb #' ',disp+12 ;username in

movb #' ',disp+13 ;username in

movb #' ',disp+14

movb #' ',disp+15

movb #'P',disp+16

movb #'a',disp+17

movb #'s',disp+18

movb #'s',disp+19

movb #'w',disp+20

movb #'o',disp+21

movb #'r',disp+22

movb #'d',disp+23

movb #':',disp+24

movb #' ',disp+25

movb #' ',disp+26 ;password input starting here

movb #' ',disp+27 ;password in

movb #' ',disp+28 ;password in

movb #' ',disp+29 ;password in

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

johnlee3: ;login screen

movb #'L',disp

movb #'o',disp+1

movb #'g',disp+2

movb #'i',disp+3

movb #'n',disp+4

movb #' ',disp+5

movb #'U',disp+6

movb #'s',disp+7

movb #'e',disp+8

movb #'r',disp+9

movb #'n',disp+10 ;username input starting here

movb #'a',disp+11 ;username in

movb #'m',disp+12 ;username in

movb #'e',disp+13 ;username in

movb #':',disp+14

movb #' ',disp+15

movb #'P',disp+16

movb #'a',disp+17

movb #'s',disp+18

movb #'s',disp+19

movb #'w',disp+20

movb #'o',disp+21

movb #'r',disp+22

movb #'d',disp+23

movb #':',disp+24

movb #' ',disp+25

movb #' ',disp+26 ;password input starting here

movb #' ',disp+27 ;password in

movb #' ',disp+28 ;password in

movb #' ',disp+29 ;password in

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

loginerror: ;error message screen

movb #'U',disp

movb #'s',disp+1

movb #'e',disp+2

movb #'r',disp+3

movb #' ',disp+4

movb #'D',disp+5

movb #'o',disp+6

movb #'e',disp+7

movb #'s',disp+8

movb #' ',disp+9

movb #'n',disp+10

movb #'o',disp+11

movb #'t',disp+12

movb #' ',disp+13

movb #'e',disp+14

movb #'x',disp+15

movb #'i',disp+16

movb #'s',disp+17

movb #'t',disp+18

movb #' ',disp+19

movb #'t',disp+20

movb #'r',disp+21

movb #'y',disp+22

movb #' ',disp+23

movb #'a',disp+24

movb #'g',disp+25

movb #'a',disp+26

movb #'i',disp+27

movb #'n',disp+28

movb #' ',disp+29

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

Introduction: ;saying hello screen

movb #'W',disp

movb #'e',disp+1

movb #'l',disp+2

movb #'c',disp+3

movb #'o',disp+4

movb #'m',disp+5

movb #'e',disp+6

movb #' ',disp+7

movb #'t',disp+8

movb #'o',disp+9

movb #' ',disp+10 ;username input starting here

movb #'t',disp+11 ;username in

movb #'h',disp+12 ;username in

movb #'e',disp+13 ;username in

movb #' ',disp+14

movb #' ',disp+15

movb #'M',disp+16

movb #'u',disp+17

movb #'s',disp+18

movb #'i',disp+19

movb #'c',disp+20

movb #' ',disp+21

movb #'p',disp+22

movb #'l',disp+23

movb #'a',disp+24

movb #'y',disp+25

movb #'e',disp+26 ;password input starting here

movb #'r',disp+27 ;password in

movb #'!',disp+28 ;password in

movb #' ',disp+29 ;password in

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

Daruder: ;song select screen 1

movb #' ',disp

movb #' ',disp+1

movb #' ',disp+2

movb #' ',disp+3

movb #' ',disp+4

movb #' ',disp+5

movb #'D',disp+6

movb #'a',disp+7

movb #'r',disp+8

movb #'u',disp+9

movb #'d',disp+10

movb #'e',disp+11

movb #' ',disp+12

movb #' ',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #' ',disp+16

movb #' ',disp+17

movb #' ',disp+18

movb #' ',disp+19

movb #'s',disp+20

movb #'a',disp+21

movb #'n',disp+22

movb #'d',disp+23

movb #'s',disp+24

movb #'t',disp+25

movb #'o',disp+26

movb #'r',disp+27

movb #'m',disp+28

movb #' ',disp+29

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

Darudep: ;song 1 playing screen

movb #'D',disp

movb #'a',disp+1

movb #'r',disp+2

movb #'u',disp+3

movb #'d',disp+4

movb #'e',disp+5

movb #' ',disp+6

movb #' ',disp+7

movb #' ',disp+8

movb #' ',disp+9

movb #' ',disp+10

movb #' ',disp+11

movb #' ',disp+12

movb #' ',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #'S',disp+16

movb #'a',disp+17

movb #'n',disp+18

movb #'d',disp+19

movb #'s',disp+20

movb #'t',disp+21

movb #'o',disp+22

movb #'r',disp+23

movb #'m',disp+24

movb #' ',disp+25

movb #' ',disp+26

movb #' ',disp+27

movb minutes,disp+28

movb #':',disp+29

movb tenssec,disp+30

movb onessec,disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

allstarr: ;song 2 select screen

movb #' ',disp

movb #' ',disp+1

movb #' ',disp+2

movb #' ',disp+3

movb #'S',disp+4

movb #'m',disp+5

movb #'a',disp+6

movb #'s',disp+7

movb #'h',disp+8

movb #' ',disp+9

movb #'M',disp+10

movb #'o',disp+11

movb #'u',disp+12

movb #'t',disp+13

movb #'h',disp+14

movb #' ',disp+15

movb #' ',disp+16

movb #' ',disp+17

movb #'A',disp+18

movb #'l',disp+19

movb #'l',disp+20

movb #' ',disp+21

movb #'S',disp+22

movb #'t',disp+23

movb #'a',disp+24

movb #'r',disp+25

movb #' ',disp+26

movb #' ',disp+27

movb #' ',disp+28

movb #' ',disp+29

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

allstarp: ;song 2 play screen

movb #'S',disp

movb #'m',disp+1

movb #'a',disp+2

movb #'s',disp+3

movb #'h',disp+4

movb #' ',disp+5

movb #'M',disp+6

movb #'o',disp+7

movb #'u',disp+8

movb #'t',disp+9

movb #'h',disp+10

movb #' ',disp+11

movb #' ',disp+12

movb #' ',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #'A',disp+16

movb #'l',disp+17

movb #'l',disp+18

movb #' ',disp+19

movb #'S',disp+20

movb #'t',disp+21

movb #'a',disp+22

movb #'r',disp+23

movb #' ',disp+24

movb #' ',disp+25

movb #' ',disp+26

movb #' ',disp+27

movb minutes,disp+28

movb #':',disp+29

movb tenssec,disp+30

movb onessec,disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

pokemonr: ;song 3 select screen

movb #' ',disp

movb #' ',disp+1

movb #' ',disp+2

movb #' ',disp+3

movb #' ',disp+4

movb #'P',disp+5

movb #'o',disp+6

movb #'k',disp+7

movb #'e',disp+8

movb #'m',disp+9

movb #'o',disp+10

movb #'n',disp+11

movb #' ',disp+12

movb #' ',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #' ',disp+16

movb #' ',disp+17

movb #'T',disp+18

movb #'h',disp+19

movb #'e',disp+20

movb #'m',disp+21

movb #'e',disp+22

movb #' ',disp+23

movb #'S',disp+24

movb #'o',disp+25

movb #'n',disp+26

movb #'g',disp+27

movb #' ',disp+28

movb #' ',disp+29

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

pokemonp: ;song 3 play screen

movb #'P',disp

movb #'o',disp+1

movb #'k',disp+2

movb #'e',disp+3

movb #'m',disp+4

movb #'o',disp+5

movb #'n',disp+6

movb #' ',disp+7

movb #'T',disp+8

movb #'h',disp+9

movb #'e',disp+10

movb #'m',disp+11

movb #'e',disp+12

movb #' ',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #'S',disp+16

movb #'o',disp+17

movb #'n',disp+18

movb #'g',disp+19

movb #' ',disp+20

movb #' ',disp+21

movb #' ',disp+22

movb #' ',disp+23

movb #' ',disp+24

movb #' ',disp+25

movb #' ',disp+26

movb #' ',disp+27

movb minutes,disp+28

movb #':',disp+29

movb tenssec,disp+30

movb onessec,disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

battmessage: ;battery low message

movb #'B',disp

movb #'a',disp+1

movb #'t',disp+2

movb #'t',disp+3

movb #'t',disp+4

movb #'e',disp+5

movb #'r',disp+6

movb #'y',disp+7

movb #' ',disp+8

movb #' ',disp+9

movb #' ',disp+10

movb #' ',disp+11

movb #' ',disp+12

movb #' ',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #'D',disp+16

movb #'e',disp+17

movb #'a',disp+18

movb #'d',disp+19

movb #' ',disp+20

movb #' ',disp+21

movb #' ',disp+22

movb #' ',disp+23

movb #' ',disp+24

movb #' ',disp+25

movb #' ',disp+26

movb #' ',disp+27

movb #'H',disp+28

movb #'e',disp+29

movb #'l',disp+30

movb #'p',disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

SongTiming:

movb minutes,disp+28 ;load minutes to display

movb tenssec,disp+30 ;load tens place to display

movb onessec,disp+31 ;load ones place to display

movb #0,disp+32 ;signal end of display sequence

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;jump to given subroutine

RTS ;return to code

;end of display options

IRQButton:

movw #0,dispon ;clear screen

movb #' ',disp

movb #' ',disp+1

movb #' ',disp+2

movb #' ',disp+3

movb #' ',disp+4

movb #' ',disp+5

movb #' ',disp+6

movb #' ',disp+7

movb #' ',disp+8

movb #' ',disp+9

movb #' ',disp+10

movb #' ',disp+11

movb #' ',disp+12

movb #' ',disp+13

movb #' ',disp+14

movb #' ',disp+15

movb #' ',disp+16

movb #' ',disp+17

movb #' ',disp+18

movb #' ',disp+19

movb #' ',disp+20

movb #' ',disp+21

movb #' ',disp+22

movb #' ',disp+23

movb #' ',disp+24

movb #' ',disp+25

movb #' ',disp+26

movb #' ',disp+27

movb #' ',disp+28

movb #' ',disp+29

movb #' ',disp+30

movb #' ',disp+31

movb #0,disp+32

JSR init\_LCD ;jump to given subroutine

ldd #disp ;load the display value

JSR display\_string ;run display subroutine

ldaa checkingfirstorsecondpress ;load a with order of press value

cmpa #0 ;compare to 0

beq imacodewarriorma ;if zero branch

cmpa #1 ;compare to 1

beq aproudwarrior ;if one, branch

bra bigending ;if messup, jump out of IRQ

imacodewarriorma:

adda #1 ;add 1 to a

staa checkingfirstorsecondpress ;store to variable

movw #1,dispon ;move 1 to display on checker

bra bigending ;jumpt to end

aproudwarrior:

adda #0 ;add 0 to a

staa checkingfirstorsecondpress ;store to variable

movw #0,dispon ;move 0 to display on checker

bigending: RTI ;return from interrupt