; Include derivative-specific definitions

INCLUDE 'derivative.inc'

; export symbols

XDEF Entry, \_Startup

XDEF RTI\_label

XDEF irq\_FUNCTION

XDEF welcomehold

XDEF counter

XDEF lookup

XDEF rows

XDEF PTU

XDEF autoshut3, letknow, autoshut2, autoshut1, gen1cap, gen2cap, gen3cap

XDEF disp1, disp2, disp5

XDEF disp6, time1off, time2off, time3off

XDEF disp7, g1a, g2a, g3a, disp8, disp9, dispa, port\_p, dispb, HOME

XDEF port\_s, HOMEflg

XDEF port\_t, MAX

XDEF clock1, disp

XDEF interval25, disph

XDEF seconds, switchflg, gen2flg, gen3flg, gen1timer, gen2timer, gen3timer, TRACKER, noTRACKER, timercorrection

XDEF setLEDTRACKER, LEDroutine, switchstatus, switchchange, prevswitchstatus, skipswitchcheck

XDEF holdold, gen1off, gen2off, gen3off

XDEF PWMCOUNTER, delayont, delaytimert

XDEF TON, delayons, delaytimers

XDEF TOFF, gen1au, gen2au, gen3au

XDEF delay1, val1, elevator, shutsound

XDEF delayon, timeswrong, locknoise, unlocknoise

XDEF delaytimer, shutoff, gen1stat, gen2stat, gen3stat

XDEF ledswitches, value, sum

XDEF switchchecker, wait, waithold, real\_value

XDEF dispe, dispd, disp3, dispc, dispF, startuptimer, power\_output, real\_power\_output

XDEF shutoff, gen1stat, gen2stat, gen3stat, dispG, potflg, priorvalue, startup, val2, val3

XDEF pushpress, LUT, alarm, Alternator, SoundCounter, mu

XDEF speedup, sendhome, delayonm, delaytimerm, wait1, ledvalue, LEDTABLE, instepper, ledm1, times8

; 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 ; symbol defined by the linker for the end of the stack

XREF SendsChr

XREF PlayTone

XREF initialize

XREF hexkeypad

XREF menu

XREF switch\_file

XREF shutdown

XREF stepper

XREF Sounds

; LCD References

XREF init\_LCD

XREF display\_string

; Potentiometer References

XREF read\_pot

XREF pot\_value

; variable/data section

my\_constant: SECTION

port\_t equ $240

port\_s equ $248

port\_p equ $258

port\_u equ $268

lookup dc.b $EB, $77, $7B, $7D, $B7, $BB, $BD, $D7, $DB, $DD, $E7, $ED, $7E, $BE, $DE, $EE

LUT dc.b $A, $12, $14, $C

LEDTABLE dc.b $1, $2, $4, $8, $10, $20, $40, $80 ;used for fillup to set leading LED to ;flash

rows dc.b $70, $B0, $D0, $E0

welcomehold equ $401 ;Used to add delay for welcome sequence

counter equ $402

setdateandtime equ $405

ledm1: equ $406

times8: equ $408

timeswrong equ $41F

ledvalue equ $407

gen1cap equ $420

gen2cap equ $421

gen3cap equ $422

autoshut1 equ $423

autoshut2 equ $424

autoshut3 equ $425

timehour equ $426

timemin equ $427

clock1 equ $428

interval25 equ $429

seconds equ $43A

switchflg equ $43B

gen2flg equ $43C

gen3flg equ $43D

gen1timer equ $43F

gen2timer equ $440

gen3timer equ $441

TRACKER equ $442

noTRACKER equ $443

PWMCOUNTER equ $44D

TON equ $44F

TOFF equ $44E

startuptimer equ $444

timercorrection equ $445

setLEDTRACKER equ $446

LEDroutine equ $447

switchstatus equ $448

switchchange equ $449

prevswitchstatus equ $44A

skipswitchcheck equ $40B

holdold equ $40C

delay1 equ $40D

delayon equ $40E

delaytimer equ $40F

gen1off equ $453

gen2off equ $452

gen3off equ $451

switchchecker equ $454

ledswitches equ $455

wait equ $456

waithold equ $457

power\_output equ $458

real\_power\_output equ $459

value: equ $45A

real\_value: equ $45B

sum: equ $45C

shutoff: equ $45D

gen1stat: equ $45E

gen2stat: equ $45F

gen3stat: equ $460

potflg: equ $461

priorvalue: equ $462

startup: equ $463

gen1life: equ $464

gen2life: equ $466

gen3life: equ $468

val1: equ $46B

val2: equ $46D

val3: equ $470

pushpress: equ $472

delaytimers: equ $473

delayons: equ $474

speedup: equ $475

sendhome: equ $476

delayonm: equ $477

delaytimerm: equ $478

wait1: equ $479

instepper: equ $404

time3off equ $47A

time2off equ $47B

time1off equ $47C

g1a: equ $47D

g2a: equ $47E

g3a: equ $47F

letknow: equ $480

gen1au: equ $481

gen2au: equ $482

gen3au: equ $483

alarm: equ $484

Alternator: equ $485

SoundCounter: equ $486

locknoise: equ $487

unlocknoise: equ $488

elevator: equ $489

delayont: equ $48A

delaytimert: equ $48B

shutsound: equ $48C

mu: equ $48D

mx: equ $48E

MAX: equ $48F

HOMEflg: equ $491

my\_variable: SECTION

disp: ds.b 33 ; Creating many string variables made it easier to

;check for correct passwords and IDs. It also made certain parts of

; code easier to follow as there is less string manipulation

disp1: ds.b 33

disp2: ds.b 33

disp3: ds.b 33

disp4: ds.b 33

disp5: ds.b 33

disp6: ds.b 33

disp7: ds.b 33

disp8: ds.b 33

disp9: ds.b 33

dispa: ds.b 33

dispb: ds.b 33

dispc: ds.b 33

dispd: ds.b 33

dispe: ds.b 33

dispF: ds.b 33

dispG: ds.b 33

disph: ds.b 33

code section

MyCode: SECTION

Entry:

\_Startup:

LDS #\_\_SEG\_END\_SSTACK ; initialize the stack pointer

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*string initializations\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;The next few lines are strings

;Each one is used to display where you are in the code

;We do change what is in disp later to display smaller messages

; Some of the strings are changed in code to display different things

; An example is the generator menu

; We used on string for all three and changed it based on what generator is selected

movb #' ',disp

movb #' ',disp+1 ;Welcome string, Used in other situations as well

movb #' ',disp+2

movb #' ',disp+3

movb #'W',disp+4

movb #'E',disp+5

movb #'L',disp+6

movb #'C',disp+7

movb #'O',disp+8

movb #'M',disp+9

movb #'E',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 ;string terminator, acts like '\0'

movb #'S',disp1

movb #'e',disp1+1 ;Set date & Time MM/DD/YYYY HH:MM

movb #'t',disp1+2

movb #' ',disp1+3

movb #'d',disp1+4

movb #'a',disp1+5

movb #'t',disp1+6

movb #'e',disp1+7

movb #' ',disp1+8

movb #'&',disp1+9

movb #' ',disp1+10

movb #'t',disp1+11

movb #'i',disp1+12

movb #'m',disp1+13

movb #'e',disp1+14

movb #' ',disp1+15

movb #'M',disp1+16

movb #'M',disp1+17

movb #'/',disp1+18

movb #'D',disp1+19

movb #'D',disp1+20

movb #'/',disp1+21

movb #'Y',disp1+22

movb #'Y',disp1+23

movb #'Y',disp1+24

movb #'Y',disp1+25

movb #' ',disp1+26

movb #'H',disp1+27

movb #'H',disp1+28

movb #':',disp1+29

movb #'M',disp1+30

movb #'M',disp1+31

movb #0,disp1+32

movb #'C',disp2 ;Type A password XXXX

movb #'r',disp2+1

movb #'e',disp2+2

movb #'a',disp2+3

movb #'t',disp2+4

movb #'e',disp2+5

movb #' ',disp2+6

movb #'p',disp2+7

movb #'a',disp2+8

movb #'s',disp2+9

movb #'s',disp2+10

movb #'w',disp2+11

movb #'o',disp2+12

movb #'r',disp2+13

movb #'d',disp2+14

movb #':',disp2+15

movb #'X',disp2+16

movb #'X',disp2+17

movb #'X',disp2+18

movb #'X',disp2+19

movb #' ',disp2+20

movb #' ',disp2+21

movb #' ',disp2+22

movb #' ',disp2+23

movb #' ',disp2+24

movb #' ',disp2+25

movb #' ',disp2+26

movb #' ',disp2+27

movb #' ',disp2+28

movb #' ',disp2+29

movb #' ',disp2+30

movb #' ',disp2+31

movb #0,disp2+32

movb #'T',disp3

movb #'y',disp3+1

movb #'p',disp3+2 ;Type ID XXXX

movb #'e',disp3+3

movb #' ',disp3+4

movb #'I',disp3+5

movb #'D',disp3+6

movb #':',disp3+7

movb #' ',disp3+8

movb #'X',disp3+9

movb #'X',disp3+10

movb #'X',disp3+11

movb #'X',disp3+12

movb #' ',disp3+13

movb #' ',disp3+14

movb #' ',disp3+15

movb #' ',disp3+16

movb #' ',disp3+17

movb #' ',disp3+18

movb #' ',disp3+19

movb #' ',disp3+20

movb #' ',disp3+21

movb #' ',disp3+22

movb #' ',disp3+23

movb #' ',disp3+24

movb #' ',disp3+25

movb #' ',disp3+26

movb #' ',disp3+27

movb #' ',disp3+28

movb #' ',disp3+29

movb #' ',disp3+30

movb #' ',disp3+31

movb #0,disp3+32

movb #'S',disp4

movb #'e',disp4+1

movb #'t',disp4+2 ;Turn on Generator and Max power

movb #' ',disp4+3

movb #'g',disp4+4

movb #'e',disp4+5

movb #'n',disp4+6

movb #'e',disp4+7

movb #'r',disp4+8

movb #'a',disp4+9

movb #'t',disp4+10

movb #'o',disp4+11

movb #'r',disp4+12

movb #'s',disp4+13

movb #' ',disp4+14

movb #' ',disp4+15

movb #'S',disp4+16

movb #'e',disp4+17

movb #'t',disp4+18

movb #' ',disp4+19

movb #'m',disp4+20

movb #'a',disp4+21

movb #'x',disp4+22

movb #' ',disp4+23

movb #'p',disp4+24

movb #'o',disp4+25

movb #'w',disp4+26

movb #'e',disp4+27

movb #'r',disp4+28

movb #' ',disp4+29

movb #' ',disp4+30

movb #' ',disp4+31

movb #0,disp4+32

movb #'H',disp6

movb #'o',disp6+1

movb #'m',disp6+2 ;Home MM/DD/YYYY HH:MM

movb #'e',disp6+3

movb #':',disp6+4

movb #' ',disp6+5

movb #' ',disp6+6

movb #' ',disp6+7

movb #' ',disp6+8

movb #' ',disp6+9

movb #' ',disp6+10

movb #'X',disp6+11

movb #'X',disp6+12

movb #'X',disp6+13

movb #'M',disp6+14

movb #'W',disp6+15

movb #'M',disp6+16

movb #'M',disp6+17

movb #'/',disp6+18

movb #'D',disp6+19

movb #'D',disp6+20

movb #'/',disp6+21

movb #'Y',disp6+22

movb #'Y',disp6+23

movb #'Y',disp6+24

movb #'Y',disp6+25

movb #' ',disp6+26

movb #'H',disp6+27

movb #'H',disp6+28

movb #':',disp6+29

movb #'M',disp6+30

movb #'M',disp6+31

movb #0,disp6+32

movb #'P',disp5

movb #'a',disp5+1

movb #'s',disp5+2 ;Password checking

movb #'s',disp5+3

movb #'w',disp5+4

movb #'o',disp5+5

movb #'r',disp5+6

movb #'d',disp5+7

movb #':',disp5+8

movb #' ',disp5+9

movb #'X',disp5+10

movb #'X',disp5+11

movb #'X',disp5+12

movb #'X',disp5+13

movb #' ',disp5+14

movb #' ',disp5+15

movb #' ',disp5+16

movb #' ',disp5+17

movb #' ',disp5+18

movb #' ',disp5+19

movb #' ',disp5+20

movb #' ',disp5+21

movb #' ',disp5+22

movb #' ',disp5+23

movb #' ',disp5+24

movb #' ',disp5+25

movb #' ',disp5+26

movb #' ',disp5+27

movb #' ',disp5+28

movb #' ',disp5+29

movb #' ',disp5+30

movb #' ',disp5+31

movb #0,disp5+32

movb #'-',disp7

movb #'>',disp7+1

movb #'G',disp7+2 ;Control Menu 1

movb #'e',disp7+3

movb #'n',disp7+4

movb #'e',disp7+5

movb #'r',disp7+6

movb #'a',disp7+7

movb #'t',disp7+8

movb #'o',disp7+9

movb #'r',disp7+10

movb #' ',disp7+11

movb #' ',disp7+12

movb #' ',disp7+13

movb #' ',disp7+14

movb #' ',disp7+15

movb #' ',disp7+16

movb #' ',disp7+17

movb #'P',disp7+18

movb #'a',disp7+19

movb #'s',disp7+20

movb #'s',disp7+21

movb #'w',disp7+22

movb #'o',disp7+23

movb #'r',disp7+24

movb #'d',disp7+25

movb #' ',disp7+26

movb #' ',disp7+27

movb #' ',disp7+28

movb #' ',disp7+29

movb #' ',disp7+30

movb #' ',disp7+31

movb #0,disp7+32

movb #'-',disp8

movb #'>',disp8+1

movb #'D',disp8+2 ;Control Menu 2

movb #'a',disp8+3

movb #'t',disp8+4

movb #'e',disp8+5

movb #'/',disp8+6

movb #'T',disp8+7

movb #'i',disp8+8

movb #'m',disp8+9

movb #'e',disp8+10

movb #' ',disp8+11

movb #' ',disp8+12

movb #' ',disp8+13

movb #' ',disp8+14

movb #' ',disp8+15

movb #' ',disp8+16

movb #' ',disp8+17

movb #'H',disp8+18

movb #'o',disp8+19

movb #'m',disp8+20

movb #'e',disp8+21

movb #' ',disp8+22

movb #' ',disp8+23

movb #' ',disp8+24

movb #' ',disp8+25

movb #' ',disp8+26

movb #' ',disp8+27

movb #' ',disp8+28

movb #' ',disp8+29

movb #' ',disp8+30

movb #' ',disp8+31

movb #0,disp8+32

movb #'-',disp9

movb #'>',disp9+1

movb #'G',disp9+2 ;Control Generators 1 and 2

movb #'e',disp9+3

movb #'n',disp9+4

movb #'e',disp9+5

movb #'r',disp9+6

movb #'a',disp9+7

movb #'t',disp9+8

movb #'o',disp9+9

movb #'r',disp9+10

movb #' ',disp9+11

movb #'1',disp9+12

movb #' ',disp9+13

movb #' ',disp9+14

movb #' ',disp9+15

movb #' ',disp9+16

movb #' ',disp9+17

movb #'G',disp9+18

movb #'e',disp9+19

movb #'n',disp9+20

movb #'e',disp9+21

movb #'r',disp9+22

movb #'a',disp9+23

movb #'t',disp9+24

movb #'o',disp9+25

movb #'r',disp9+26

movb #' ',disp9+27

movb #'2',disp9+28

movb #' ',disp9+29

movb #' ',disp9+30

movb #' ',disp9+31

movb #0,disp9+32

movb #'-',dispa

movb #'>',dispa+1

movb #'G',dispa+2 ;Control Generator 3

movb #'e',dispa+3

movb #'n',dispa+4

movb #'e',dispa+5

movb #'r',dispa+6

movb #'a',dispa+7

movb #'t',dispa+8

movb #'o',dispa+9

movb #'r',dispa+10

movb #' ',dispa+11

movb #'3',dispa+12

movb #' ',dispa+13

movb #' ',dispa+14

movb #' ',dispa+15

movb #' ',dispa+16

movb #' ',dispa+17

movb #'H',dispa+18

movb #'o',dispa+19

movb #'m',dispa+20

movb #'e',dispa+21

movb #' ',dispa+22

movb #' ',dispa+23

movb #' ',dispa+24

movb #' ',dispa+25

movb #' ',dispa+26

movb #' ',dispa+27

movb #' ',dispa+28

movb #' ',dispa+29

movb #' ',dispa+30

movb #' ',dispa+31

movb #0,dispa+32

movb #'G',dispb

movb #'e',dispb+1

movb #'n',dispb+2 ;Generator Menu

movb #'e',dispb+3

movb #'r',dispb+4

movb #'a',dispb+5

movb #'t',dispb+6

movb #'o',dispb+7

movb #'r',dispb+8

movb #' ',dispb+9

movb #' ',dispb+10

movb #' ',dispb+11

movb #' ',dispb+12

movb #' ',dispb+13

movb #' ',dispb+14

movb #' ',dispb+15

movb #'A',dispb+16

movb #'u',dispb+17

movb #'t',dispb+18

movb #'o',dispb+19

movb #' ',dispb+20

movb #'S',dispb+21

movb #'h',dispb+22

movb #'u',dispb+23

movb #'t',dispb+24

movb #'o',dispb+25

movb #'f',dispb+26

movb #'f',dispb+27

movb #' ',dispb+28

movb #' ',dispb+29

movb #' ',dispb+30

movb #' ',dispb+31

movb #0,dispb+32

movb #'I',dispc

movb #'D',dispc+1

movb #':',dispc+2 ;Password checking

movb #' ',dispc+3

movb #'X',dispc+4

movb #'X',dispc+5

movb #'X',dispc+6

movb #'X',dispc+7

movb #' ',dispc+8

movb #' ',dispc+9

movb #' ',dispc+10

movb #' ',dispc+11

movb #' ',dispc+12

movb #' ',dispc+13

movb #' ',dispc+14

movb #' ',dispc+15

movb #' ',dispc+16

movb #' ',dispc+17

movb #' ',dispc+18

movb #' ',dispc+19

movb #' ',dispc+20

movb #' ',dispc+21

movb #' ',dispc+22

movb #' ',dispc+23

movb #' ',dispc+24

movb #' ',dispc+25

movb #' ',dispc+26

movb #' ',dispc+27

movb #' ',dispc+28

movb #' ',dispc+29

movb #' ',dispc+30

movb #' ',dispc+31

movb #0,dispc+32

movb #'W',dispd

movb #'r',dispd+1

movb #'o',dispd+2 ;Password checking

movb #'n',dispd+3

movb #'g',dispd+4

movb #' ',dispd+5

movb #'p',dispd+6

movb #'a',dispd+7

movb #'s',dispd+8

movb #'s',dispd+9

movb #'w',dispd+10

movb #'o',dispd+11

movb #'r',dispd+12

movb #'d',dispd+13

movb #' ',dispd+14

movb #' ',dispd+15

movb #'o',dispd+16

movb #'r',dispd+17

movb #' ',dispd+18

movb #'I',dispd+19

movb #'D',dispd+20

movb #' ',dispd+21

movb #' ',dispd+22

movb #' ',dispd+23

movb #' ',dispd+24

movb #' ',dispd+25

movb #' ',dispd+26

movb #' ',dispd+27

movb #' ',dispd+28

movb #' ',dispd+29

movb #' ',dispd+30

movb #' ',dispd+31

movb #0,dispd+32

movb #'R',dispe

movb #'e',dispe+1

movb #'s',dispe+2 ;Switch warning

movb #'e',dispe+3

movb #'t',dispe+4

movb #' ',dispe+5

movb #'s',dispe+6

movb #'w',dispe+7

movb #'i',dispe+8

movb #'t',dispe+9

movb #'c',dispe+10

movb #'h',dispe+11

movb #'e',dispe+12

movb #'s',dispe+13

movb #' ',dispe+14

movb #' ',dispe+15

movb #' ',dispe+16

movb #' ',dispe+17

movb #' ',dispe+18

movb #' ',dispe+19

movb #' ',dispe+20

movb #' ',dispe+21

movb #' ',dispe+22

movb #' ',dispe+23

movb #' ',dispe+24

movb #' ',dispe+25

movb #' ',dispe+26

movb #' ',dispe+27

movb #' ',dispe+28

movb #' ',dispe+29

movb #' ',dispe+30

movb #' ',dispe+31

movb #0,dispe+32

movb #'T',dispF

movb #'u',dispF+1

movb #'r',dispF+2 ;Switch warning for filling

movb #'n',dispF+3

movb #' ',dispF+4

movb #'o',dispF+5

movb #'f',dispF+6

movb #'f',dispF+7

movb #' ',dispF+8

movb #' ',dispF+9

movb #' ',dispF+10

movb #' ',dispF+11

movb #' ',dispF+12

movb #' ',dispF+13

movb #' ',dispF+14

movb #' ',dispF+15

movb #'G',dispF+16

movb #'e',dispF+17

movb #'n',dispF+18

movb #'e',dispF+19

movb #'r',dispF+20

movb #'a',dispF+21

movb #'t',dispF+22

movb #'o',dispF+23

movb #'r',dispF+24

movb #' ',dispF+25

movb #' ',dispF+26

movb #' ',dispF+27

movb #' ',dispF+28

movb #' ',dispF+29

movb #' ',dispF+30

movb #' ',dispF+31

movb #0,dispF+32

movb #'S',dispG

movb #'Y',dispG+1

movb #'S',dispG+2 ;Emergency shutdown

movb #'T',dispG+3

movb #'E',dispG+4

movb #'M',dispG+5

movb #' ',dispG+6

movb #'S',dispG+7

movb #'H',dispG+8

movb #'U',dispG+9

movb #'T',dispG+10

movb #'D',dispG+11

movb #'O',dispG+12

movb #'W',dispG+13

movb #'N',dispG+14

movb #' ',dispG+15

movb #'I',dispG+16

movb #'D',dispG+17

movb #':',dispG+18

movb #' ',dispG+19

movb #'X',dispG+20

movb #'X',dispG+21

movb #'X',dispG+22

movb #'X',dispG+23

movb #' ',dispG+24

movb #' ',dispG+25

movb #' ',dispG+26

movb #' ',dispG+27

movb #' ',dispG+28

movb #' ',dispG+29

movb #' ',dispG+30

movb #' ',dispG+31

movb #0,dispG+32

movb #'-',disph

movb #'>',disph+1

movb #'I',disph+2 ;Third menu for ID and Mute

movb #'D',disph+3

movb #' ',disph+4

movb #' ',disph+5

movb #' ',disph+6

movb #' ',disph+7

movb #' ',disph+8

movb #' ',disph+9

movb #' ',disph+10

movb #' ',disph+11

movb #' ',disph+12

movb #' ',disph+13

movb #' ',disph+14

movb #' ',disph+15

movb #' ',disph+16

movb #' ',disph+17

movb #'M',disph+18

movb #'u',disph+19

movb #'t',disph+20

movb #'e',disph+21

movb #' ',disph+22

movb #' ',disph+23

movb #' ',disph+24

movb #' ',disph+25

movb #' ',disph+26

movb #'O',disph+27

movb #'f',disph+28

movb #'f',disph+29

movb #' ',disph+30

movb #' ',disph+31

movb #0,disph+32

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*string initialization\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

JSR init\_LCD ;initialize LCD and display welcome

LDD #disp ; Accumulators A and B need the string in order for the LCD display to work

JSR display\_string

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; Write You Code Here

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

JSR initialize ; Subroutine initializes all output peripherals and variables in program

MOVB #$80, CRGINT

MOVB #$10, RTICTL ; initialize interrupt and make interrupt period 0.1ms

MOVB #0, startuptimer ; This sets the startup timer flag to zero so that welcome

;displays for about 3 seconds

CLI ; Clear RTI flag

welcomeloop: LDAA startuptimer ; loop checks if the RTI has incremented the

; staruptimer variable 3 times

CMPA #3

BNE welcomeloop

SEI ; Set RTI flag temporarily as the next part of the

; program initializes things

JSR initialize ; Reinitialize variables as they may have changed

; due to startup loop

; The following routine will set an initial date and time determined by the user

LDD #disp1 ;3 seconds have passed so prompt user to enter date and time

JSR display\_string

LDX #16 ; Load with 16 because that is where the date begins in string

STX $408 ; temp address to save location in string

setdate: LDAA disp1, x ; Load with current character in string starting at the 16th

CMPA #' '

BEQ increx ;Checking for /,:, and spaces. We want to skip these characters

CMPA #'/'

BEQ increx

CMPA #':'

BEQ increx

nope: JSR hexkeypad ; hexkeypad waits for button press

CMPA #$A

BHS nope ; prevent user from using letters in date and time

CMPA #9

BLE number ; These lines up to ‘number’ were coded to display letters. They

; are no longer used and should be deleted

ADDA #$37

STAA disp1, x

STAA disp6, x

BRA increx

number: ADDA #$30 ; need ascii value

STAA disp1, x ; store this number so user can see what they are typing on lcd

STAA disp6, x ; store this number so date correctly displays on home screen

increx: inx ; next value in string

STX $408 ;408 is a placeholder for x

LDD #disp1

JSR display\_string

LDX $408 ; check if at the end of the string

CPX #32

BEQ passwordset ; if the string has been spanned, move to set password

BRA setdate

passwordset: ;Setting Password

LDD #disp2 ; set password prompt

JSR display\_string

LDX #16 ; first character of string that user changes

STX $408 ;408 is a placeholder for x

setpass: LDAA disp2, x

JSR hexkeypad ; wait for user to type password entry

CMPA #9 ; check if a number or letter is pressed so that it can be

; displayed appropriately

BLE number1

ADDA #$37 ; add 37 to convert to ascii for letter

STAA disp2, x ; store value in string

BRA skipnumber1 ; skip to skipnumber1 so dont alter correct key press

number1: ADDA #$30 ; add 30 to convert to ascii for number

STAA disp2, x

skipnumber1: inx ; move to next value in the string

STX $408

LDD #disp2 ; display string on LCD with user character added

JSR display\_string

LDX $408

CPX #20 ; load x with place in string and move to ID set if 4 characters

; pressed

BEQ IDset

BRA setpass

IDset: ;Setting ID. Refer to password set for details as this is the same

;process

LDD #disp3

JSR display\_string

LDX #9

STX $408 ;408 is a placeholder for x

setID: LDAA disp3, x

JSR hexkeypad

CMPA #9

BLE number2

ADDA #$37

STAA disp3,x

BRA skipnumber2

number2: ADDA #$30

STAA disp3, x

skipnumber2: inx

iny

STX $408 ;408 is a placeholder for x

LDD #disp3

JSR display\_string

LDX $408

CPX #13

BEQ check

BRA setID

check:

; check routine verifies user has the plant at max output before

; rest of program starts

LDD #disp4 ;prompt user to make sure all generators are on by having first 3 switches

;up

JSR display\_string

on: LDAA port\_t ;Checking if swithces are up

CMPA #$0F

BNE check7

BRA max1

check7: CMPA #$07

BNE on

max1: JSR read\_pot ;Checking is power is max

LDD pot\_value ; potentiometer indicates plant output percentage

CPD #$D0 ; max value of pot changes on different boards so this is

; harcoded to ensure program continues

BNE max1

BSET $4, INTCR

CLI ;start interrupts

MOVB #$F2, MAX ; makes $F2 max value of plant output

MOVB #0, $48E

MOVB #0, startup ; tell program we are done with initialize routine

Bra HOME

;The homescreen does a few things

;It updates the time and the power output

;It also just waits for you to push F

HOME: MOVB #1, HOMEflg ; tell program we are in the home screen

MOVB #0, sendhome ; reset flag that sent us home if it did

MOVB #1, startup ; in home

LDAA TON ; Load value of plant output

CMPA #$E8 ; due to inaccuracy of pot, a value of $E8 indicates max

; output

BLS nofull

MOVB #'3', disp6+11 ;300 indicates maximum amount of power output

MOVB #'0', disp6+12

MOVB #'0', disp6+13

BRA skiplcdupdate

nofull: LDAA #100 ; if pot value isnt max, need to convert digits to correct

; values

LDAB TON

MUL ; multiply pot value by 100

LDX $48E

IDIV ; divide pot value by max potential pot value

TFR x, d ; transfer quotient to d

LDAA #3

MUL ; multiply quotient by 3

LDX #100

IDIV ; divide product by 100

TFR x,a ; transfer product to a

ADDA #$30

STAA disp6+11 ; this gives us the 100s place

SUBA #$30 ; get value back to non-ascii

LDAA #0

LDX #10

IDIV ; divide remainder by 10

TFR x, a ; transfer remainder to a

ADDA #$30

STAA disp6+12 ; accumulator a contains 10s place

ADDB #$30

STAB disp6+13 ; accumulator b contains 1s place

skiplcdupdate:MOVB #0, port\_s

LDAA gen1off ; check if generator 1 is on

CMPA #1

BEQ generator2

LDAA port\_s ; if generator 1 is on, turn led 1 on

ORAA #$1

STAA port\_s

generator2: LDAA gen2off

CMPA #1 ; check if generator 2 is on

BEQ generator3

LDAA port\_s

ORAA #$2 ; if generator 2 is on, turn led 2 on

STAA port\_s

generator3: LDAA gen3off

CMPA #1

BEQ conwait ; check if generator 3 is on

LDAA port\_s

ORAA #$4 ; turn led 3 on if generator 3 is on

STAA port\_s

conwait: MOVB #0, waithold

LDD #disp6

JSR display\_string ; display home screen

JSR hexkeypad ; wait for user input

MOVB #0, TRACKER ; if program fell out of hexkeypad routine due to tracker, set

; it to zero

LDAB switchchange ; check if a switch change made program fall out of keypad

; routine

CMPB #1

MOVB #0, startup ; leaving home

BEQ switch\_file1 ; file performs dip switch functions

CMPA #$F ; check if user pressed ‘F’

BNE HOMEJMP

JSR menu ; performs menu functions

BRA HOMEJMP

switch\_file1: JSR switch\_file

BRA HOMEJMP

HOMEJMP:

JMP HOME ; too far for normal branch

; This is the start of the RTI

; The RTI does a ton in the program

; The main task of the RTI is to keep track of time

; These timers are used for the music, the clock, and the speed of burning coal.

; It also updates these things every time it needs too

; Most of the RTI is skipped and does not do anything until it has run a certain number of times

; This keeps the RTI running quickly and not lagging around

RTI\_label: LDAA delayons ; checks if program is using a short delay

CMPA #1

BNE skipsdelay

LDAA delaytimers ; increment short delay timer

INCA

STAA delaytimers

skipsdelay: LDAA clock1 ; program clock

INCA ; keep track of how many times entered RTI

STAA clock1

LDAA clock1

CMPA #$FF ; increment interval25 every 25 milliseconds

BNE ENDRTIjmp ; end RTI if program hasnt passed 25ms

BRA yay

ENDRTIjmp: JMP ENDRTI

yay: MOVB #0, clock1 ; reset 25ms clock

LDAA interval25

INCA ; keep track of how many intervals of 25 milliseconds have occured

STAA interval25

LDAA waithold ; check if program needs to change led status

CMPA #1

BNE skipwaithold

LDAA wait1

INCA

STAA wait1

CMPA #5

BNE skipwaithold

MOVB #0, wait1

LDAA wait

INCA

STAA wait

CMPA #$2

BNE turnoffleds

LDAA instepper

CMPA #1

BNE doo

LDAA ledm1 ; if instepper is high, perform this special process to slowly ‘increase’

; leds

LSLA

INCA

STAA port\_s

BRA doo1

doo: MOVB ledswitches, port\_s

doo1: MOVB #0, wait ; reset wait timer

BRA skipwaithold

turnoffleds: LDAA instepper

CMPA #1

BEQ skipwaithold1

MOVB #0, port\_s

BRA skipwaithold

skipwaithold1: MOVB ledm1, port\_s ; display stepper special case on leds

skipwaithold:LDAA delayon ; check if program is attempting mid sized delay

CMPA #1

BNE skipdelay

LDAA delaytimer

INCA

STAA delaytimer

skipdelay: LDAA delayont ; check if program is attempting mid sized edlay

CMPA #1

BNE skipdelayt

LDAA delaytimert

INCA

STAA delaytimert

skipdelayt: LDAA interval25

CMPA #31 ; increment seconds after every 40 25 millisecond intervals

BNE ENDRTIJMP

BRA skipendrti

ENDRTIJMP:

JMP CHECKSWITCHES

skipendrti: MOVB #0, interval25

LDAA startuptimer ; increment startup timer

INCA

STAA startuptimer

LDAA delayonm ; check if program is attempting a long delay

CMPA #1

BNE skipdelaym

LDAA delaytimerm

INCA

STAA delaytimerm

skipdelaym: LDAA seconds ; increment program clock

INCA

STAA seconds

LDAA shutdown ; check if a shutdown is occuring

CMPA #1

BEQ ENDRTIjp

BRA skipjmp

ENDRTIjp: JMP ENDRTI

skipjmp: LDAA startuptimer ; startup lasts 3 seconds only

CMPA #3

BLE ENDRTIJMP

MOVB #3, startuptimer

LDAA gen1timer ; next 9 lines increment generator timers as they slowly run

; out of fuel

INCA

STAA gen1timer

LDAA gen2timer

INCA

STAA gen2timer

LDAA gen3timer

INCA

STAA gen3timer

gen1timerdec1:LDAA real\_power\_output ; check if pot is greater than zero

CMPA #0

BEQ continuejmp

BRA gen1timerdec

continuejmp: JMP continue

gen1timerdec:

LDAA g1a

CMPA #1

BNE dc1

LDAA time1off

CMPA #0

BEQ skipdec1

DECA

STAA time1off

CMPA #0

BNE checkif1

skipdec1: MOVB #1, gen1off

MOVB #0, g1a

MOVB #0, autoshut1

MOVB #1, gen1au

MOVB #1, letknow

checkif1: LDAA gen1off

CMPA #1

BEQ gencheck2

dc1:

LDAA #$FF

LDAB #15

MUL

TFR d, x

LDD real\_power\_output

TAB

LDAA #0

STAB $46C

TFR x, d

LDX val1

IDIV

CPX #$100

BLO okay1

LDX #$FF

okay1: TFR x, a

STAA gen1life

LDAA gen1timer

CMPA gen1life

BLS gencheck2

LDAA gen1cap

LSRA

MOVB #1, setLEDTRACKER

STAA gen1cap

LDAA gen1cap

CMPA #$0

BNE gen1on

MOVB #1, gen1off

gen1on: MOVB #0, gen1timer

gencheck2: LDAA g2a

CMPA #1

BNE dc2

LDAA time2off

CMPA #0

BEQ dec2

DECA

STAA time2off

CMPA #0

BNE che2

dec2: MOVB #1, gen2off

MOVB #0, g2a

MOVB #1, gen2au

MOVB #0, autoshut2

MOVB #1, letknow

che2: LDAA gen2off

CMPA #1

BEQ gencheck3

dc2:

LDAA #$FF

LDAB #30

MUL

TFR d, x

LDD real\_power\_output

TAB

LDAA #0

STAB $46E

TFR x, d

LDX val2

IDIV

CPX #$100

BLO okay2

LDX #$FF

okay2: TFR x, a

STAA gen2life

LDAA gen2timer

CMPA gen2life

BLS gencheck3

LDAA gen2cap

LSRA

MOVB #1, setLEDTRACKER

STAA gen2cap

LDAA gen2cap

CMPA #$0

BNE gen2on

MOVB #1, gen2off

gen2on: MOVB #0, gen2timer

gencheck3: LDAA g3a

CMPA #1

BNE dc3

LDAA time3off

CMPA #0

BEQ dec3

DECA

STAA time3off

CMPA #0

BNE cheo3

dec3: MOVB #1, gen3off

MOVB #0, g3a

MOVB #1, gen3au

MOVB #0, autoshut3

MOVB #1, letknow

cheo3: LDAA gen3off

CMPA #1

BEQ continue

dc3:

LDAA #$FF

LDAB #45

MUL

TFR d, x

LDD real\_power\_output

TAB

LDAA #0

STAB $471

TFR x, d

LDX val3

IDIV

CPX #$100

BLO okay3

LDX #$FF

okay3: TFR x, a

STAA gen3life

LDAA gen3timer

CMPA gen3life

BLS continue

LDAA gen3cap

LSRA

MOVB #1, setLEDTRACKER

STAA gen3cap

LDAA gen3cap

CMPA #$0

BNE gen3on

MOVB #1, gen3off

STAA gen3cap

gen3on: MOVB #0, gen3timer

continue: LDAA seconds ; keep track of how many seconds have occured

CMPA #60

BNE CHECKSWITCHES

LDAA noTRACKER

CMPA #0

BEQ skiptracker

MOVB #1, TRACKER

skiptracker: MOVB #0, seconds

LDAA disp6+31 ; once 60 seconds has occured, increment the ones digit in minutes

SUBA #$30

INCA

CMPA #$A

BGE INCNEXT

ADDA #$30

STAA disp6+31

BRA CHECKSWITCHES

INCNEXT: MOVB #'0', disp6+31 ; if the ones digit in minutes is a 9, reset to 0 and increment tens digit

LDAA disp6+30

SUBA #$30

INCA

CMPA #6 ; if tens digit in minutes is a 5

BGE INCHOUR

ADDA #$30

STAA disp6+30

BRA CHECKSWITCHES

INCHOUR: MOVB #'0', disp6+30

LDAA disp6+28

SUBA #$30

INCA

CMPA #4

BGT INCNEXTHOUR

ADDA #$30

STAA disp6+28

BRA CHECKSWITCHES

INCNEXTHOUR: MOVB #'0', disp6+28

LDAA disp6+27

SUBA #$30

INCA

CMPA #2

BGT CHECKSWITCHES

ADDA #$30

STAA disp6+27

BRA CHECKSWITCHES

ENDRTI:

CHECKSWITCHES:

LDAA PWMCOUNTER ; increment PWM value

ADDA #1

STAA PWMCOUNTER

CMPA TON ; check if PWM value is in the on stage

BLS setex

CMPA MAX ; check if PWM value has reached max value of output

BLS clrex

MOVB #0, PWMCOUNTER ; reset PWM value

clrex: BCLR port\_t, #$8 ; make DC motor stop

BRA switches

setex: BSET port\_t, #$8 ; make DC motor start

BRA switches

switches: LDAA holdold ; check if switch value can be ignored

CMPA #1

BEQ ENDRTIBLAH

LDAA port\_t ; check if last or first 3 switches have been changed

ANDA #$87

CMPA switchstatus

BEQ ENDRTIBLAH

MOVB #1, switchchange ; let program know a switch change has occured

ENDRTIBLAH:

LDAA mu ; check if user has muted the program

CMPA #1

BEQ ending

LDAA alarm ; check if shutdown has occured to initiate alarm sound

CMPA #1

BNE checkun

JSR Sounds

checkun: LDAA unlocknoise ; check if unlock noise has been initiated

CMPA #1

BNE checklo

JSR Sounds

checklo: LDAA locknoise ; check if lock noise has been initiated

CMPA #1

BNE checkele

JSR Sounds

checkele:LDAA elevator ; check if filling noise has been initiated

CMPA #1

BNE checkshut

JSR Sounds

checkshut:LDAA shutsound ; check if auto shutoff noise has been initiated

CMPA #1

BNE ending

JSR Sounds

ending: BSET CRGFLG, #$80 ; reset RTI flag

RTI

;This is the IRQ

;All it does is set a flag when you push the button

;The rest is done in the shutdown file

irq\_FUNCTION:

MOVB #1, shutoff ; let program know IRQ was pressed

RTI