Lee Sichello

Spring 2011

**‘Smart Power Bar’**

**Background**

There are a variety of energy meters available on the market but consumers are very limited in their options. The vast majority include only one outlet and are incapable of performing the cost calculations that many consumers find challenging.

**Proposal**

**Description**

A power bar will be created to measure the power usage for two 110 VAC electrical outlets on the module. Measurements will need to be logged in memory for averaging and cost calculations. To allow for input of local power costs, there will be a series of pushbuttons for user control. Users will have an option to input local power costs as well as a series of easily selectable options for what is displayed on the LCD screen. All displayed measurements and calculations must be within 10% error.

**Overview and Operation**

The user selectable options on the LCD screen for each device must include:

* Instantaneous current use
* Instantaneous power use
* Average energy use
* Total energy use
* Estimated cost per day of operation

Users will be able to page through these options at will.

Put simply, electrical power usage at any instant is the product of voltage and current. Since this device will be used with 110 V, 50Hz AC, only current will be measured. Any deviations from true 110 VAC residential power will be made up for by my 10% error margin.

Current sensing will be done via a series of current transformers (CT's). Current will be sampled, via and ADC, at a much higher frequency than the 50Hz supply power. By logging many samples each cycle, an RMS current calculation can be made and saved at regular time intervals (at least once a second), then samples may be cleared from memory. Each time RMS current is calculated, energy use will be calculated via:

**E = P**rms \* **Time Elapsed**

Rather than keeping an indefinite running total of consumed energy, the unit will accumulate up to 0.01 kWh (the resolution of the screen) then increment. By keeping track of the energy consumed and the cost per kWh, it is straight forward to calculate the cumulative cost or forecast the cost per day of an appliance.

The unit will be powered up and acquiring data whenever plugged into a wall socket. If unplugged, all former data will be cleared from memory. This can also be achieved by means of a master reset button. Users can also reset the data for individual channels using the pushbuttons.

Upon startup, the device will enter a default mode of operation and remain there until the user interrupts it via the inputs. Interrupts include selecting display options and entering the “PROGRAM” mode to input local power costs.

**Operating Conditions and Safety**

Being a high voltage device, safety is a major concern. For this reason, the unit will be electrically isolated from the 110 VAC with the exception of the step-down transformer used in the power supply. The current transformers will provide this isolation.

To protect from over-current, the main input will be equipped with either a fuse or circuit breaker. The unit must be able to operate between 0 and 50 degrees Celsius (domestic use) with humidity up to 80%.

**Physical Packaging**

The prototype will be housed completely within a 10x6x3.5” aluminum project box. The box will have a single power cord for 110 VAC power connection.

**Budget**

The budget for the initial working prototype is a fixed cost of $3000. The production power bars must be under $100 though. If it is not built by the deadline, a discount of 10% per month will be given until the project is finished.

**Optional Additional Features**

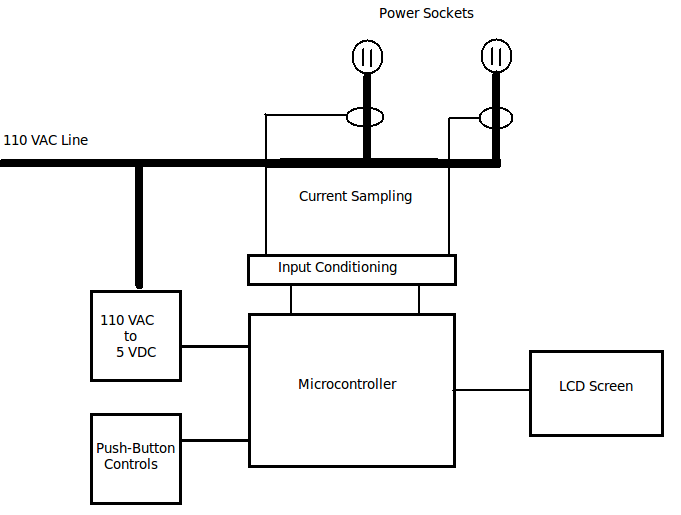
For safety considerations, surge protection may be added to cut power in such an event. Also, ground-fault interruption could be implemented by measuring return current on the neutral main line. If this current is not equal to the sum of the currents in the devices, power should be cut and a signal LED triggered.

Time permitting, an on-board battery powered clock could be added to the unit. Then users could set the time and schedule times for devices to turn off/on throughout the day.

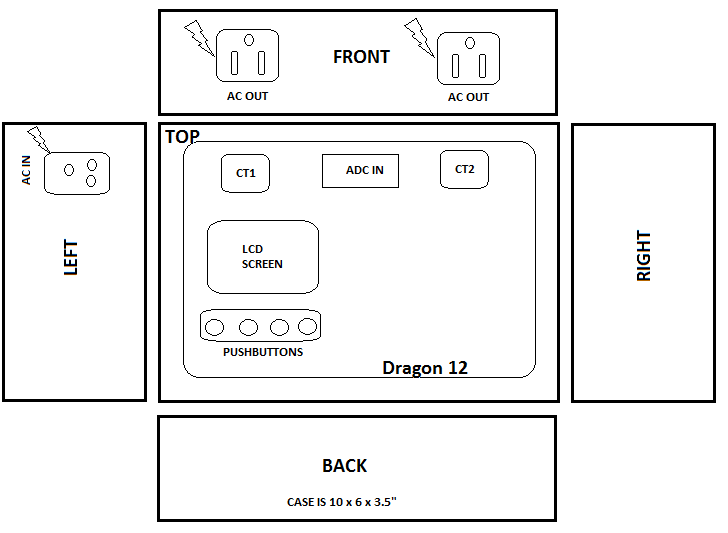
**Bill Of Materials**

1. Master Control Unit
   1. Microprocessor
      1. Wytec Dragon12 module (selected for its second hand availability)
   2. Analog Inputs
      1. Push Buttons
      2. Analog to Digital Conversion
         1. HC9S12 A/D 8 or 10 bit Sigma Delta ADC
         2. Input conditioning
            1. LM358 Low Power, Single Supply, Dual Operational Amp
            2. Resistors and capacitors as required to filter and scale input
   3. Digital Outputs
      1. Hitachi 44780 LCD Controller
   4. Power Supply
      1. Universal AC/DC Power Adaptor Supplies +5 to +7VDC power
2. Current Sense
   1. Current Transformers - (2) CT1015 , 1:1000, 15Amp
3. Wiring and Connectors
   1. Power Cable (110VAC) – 14 Guage, LIVE/NTRL/GND stranded wire’
   2. 1 Male, 2 Female Standard 110VAC Plugs
   3. Signal Jumpers – Connect Current Sense to MCU
4. Safety
   1. Circuit Breaker (8 Ohm)

**Logical and Physical Layout**

****

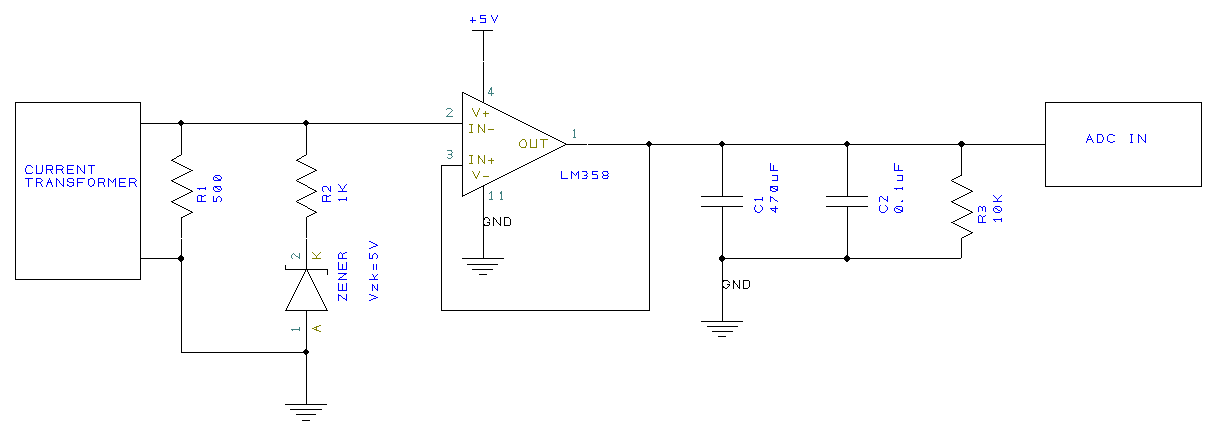
Logical Connection Diagram



Physical Layout Diagram

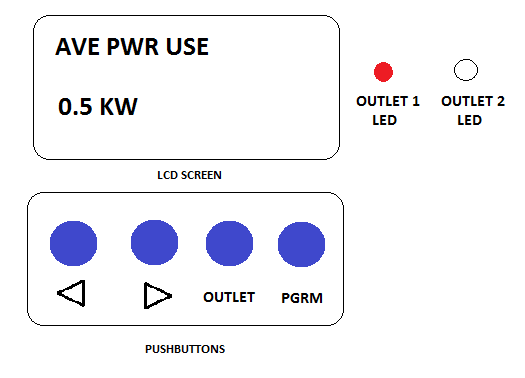
**Input Conditioning**

The output from the CT’s is an AC voltage proportional to the current being used. In order to interface this with the ADC inputs, it must be scaled from 0-5V. To do this, the CT voltage is half-wave rectified, then passed through a single supply (+5V) op-amp which acts as a protective buffer. The half-waves are then smoothed to a DC signal by capacitors. The circuit looks as follows:



**LCD/Pushbutton Interface**

When plugged in, the unit will boot to a default mode. In this mode, it will simply display the average power use for outlet 1, as indicated below.

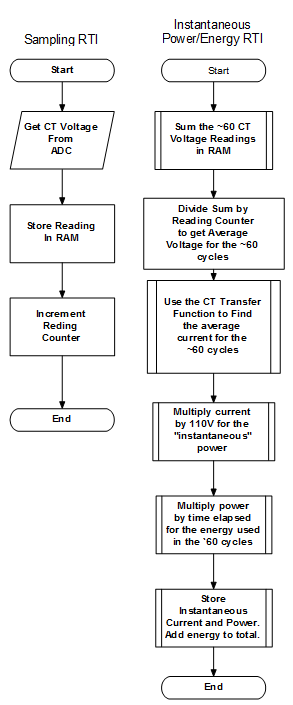
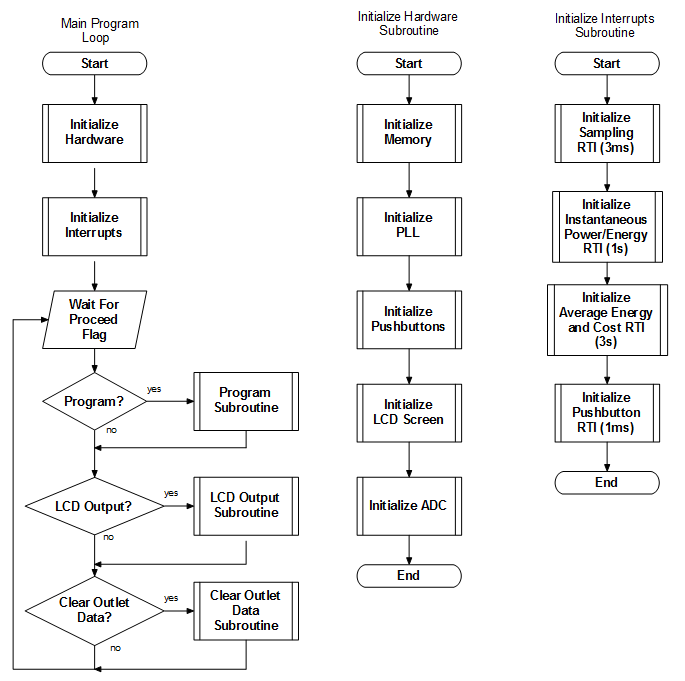


To change between outlets (indicated by LED’s) simply hit the OUTLET button. In order to clear the data for the present outlet, hold down the OUTLET button for 4 seconds. To toggle display options for the present outlet use the LEFT and RIGHT arrows. In order to program in the cost per kWh you are being charged, press the PGRM button then use the LEFT and RIGHT arrows to set the value. Once the value is set, hit PGRM again to return to normal display, or wait for 10 seconds.

**Software Design**

The main program of the firmware will initialize the hardware and interrupts, then wait for interrupts to set a proceed flag to continue. The interrupt routines will be used to perform measurements, do calculations and set flags for use in the main program. The interrupts used will be:

* Real-Time Interrupts
  + Sample current 5 times per 60Hz AC cycle (that is, every 3 ms), log data.
  + Calculate instantaneous current & power use, and update total energy use every 60 AC cycles (every 1s)
  + Estimate average energy use and cost per day every 180 cycles (every 3s)
  + Check pushbutton status (every 1ms)

****

Flow Chart Of Main Program and Some RTI’s (more to come).

**System Testing**

Testing will be done using an in-line AC ammeter (to measure actual current flow) and a load with variable resistance. The current will be ramped up and the ammeter readings will be compared to the instantaneous current readings on the unit to ensure that the two are within a 10% agreement.

**Detailed Schedule**

1. Complete Input Conditioning circuit – by Mar 12
   1. Finish prototyping
   2. Develop transfer function for current vs. voltage output
   3. Solder together
2. Assemble project in aluminum housing – by Mar 17
   1. Wire in AC circuitry
   2. Fix input conditioning board and Dragon 12 in place
3. Push button interrupts – by Mar 20
   1. Program proper reaction for each button
4. Real-time interrupts – by Mar 25
   1. Aquire current samples
   2. Do power and cost calculations
5. Troubleshoot and Complete Unit – by Apr 3

**Completed Unit**



**Design vs Reality**

I am extremely pleased with the end result. All of the deliverables layed out in the functional specification have been met by the ‘Smart Power Bar.’The unit is capable of sampling and displaying the current for two outlets within a 10% error. The unit is capable of taking user input for the local cost per kilowatt-hour.Using the current sample and local cost, it can calculate and display the instantaneous power use, average power use, total energy use from startup, total cost incurred and the estimated cost for a full day of operation for each outlet independantly. Display and user input protocol are a match with the initial design.

The unit is packaged in an aluminum box which is smaller that the maximum specs mentioned. The case is grounded and utilizes a 15 Amp breaker for safety considerations. I changed from a 8Amp to 15 Amp breaker since the loads I wished to drive would’ve been popping the breaker.

Things which I did not mention in my design which have been included in the final product are that my 5V DC power supply is external to the aluminum case (so it can be reused) and there are holes bored in the left side of the case so that a serial connection can be made. Also, there are holes in the case for access to the Dragon12 Bootloaded switches and reset button.

**Hours Spent on Project**

Based on the assumption that I have spent 10 hours per week on the project since the beginning of reading week, plus 10 hours on initial design and another 10 on documentation; I estimate that I’ve spent around **80 to 90** hours working on my project. This is a lot of time but I think it has been reflected in the final result.

**Testing Plan With Results**

As mentioned, the testing plan was to be done using an in-line AC ammeter (to measure actual current flow) and a load with variable resistance. The current was ramped up and the ammeter readings were compared to the instantaneous current readings on the unit to ensure that the two are within a 10% agreement. In reality, I was closed to a maximum error of about 2%, but it was nice to have that extra padding. Also, once the variance of the incoming 120V power was folded into the mix, the error may have been closer to 10%.

**Conclusions/Lessons Learned**

Again, I am quite pleased with my results for the project. It was time consuming and frustrating at times but also very rewarding in the end. The biggest lessons I learned, aside from the obvious hardware interfacing and assembly code, were:

* Always backup working versions of your code in case something goes awry and you can’t fix it.
* Break things up into small chunks and focus on them one at a time or it’s easy to get overwhelmed.
* The least important aspects and little details are often the most time consuming in a project like this.

**Final Block Diagram**

**C:\Users\Lee\Documents\School\ENEL387\Project\Docs\Final_Spec\hrdwr_block.tif**

**Detailed Schematic**

**C:\Users\Lee\Documents\School\ENEL387\Project\Docs\Final_Spec\mian_circuit.tif**

**Source Code:**

**Main**

;Project: Smart Power Bar

;Creator: Lee Sichello

;Date: Jan-Mar, 2011

;Class: ENEL 387

;Description: The following is the assembly code written to control the

;'smart power bar' module. IT is written for the HC9S12

;and implemented on a Dragon12 board from WYTEC along with a

;custom designed current sampling circuit. The program makes use

;of the LCD screen for output and uses the pushbuttons for user

;control. Two ADC channels are the only external interfaces.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

#include D\_variant\_registers.inc

REGBLK: equ $0000

STACK: equ $2000

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

org $1000

counter: rmb 1

pbtn\_status: rmb 1

pbtn\_sample: rmb 1

pbtn\_count: rmb 2

disp\_no: rmb 1

hold\_count: rmb 2

pgm\_count: rmb 2

intFLG: rmb 1 ;used to enter into main loop

cost: rmb 2

daily\_buffer: rmb 7

cost\_buffer: rmb 7

v\_buffer: rmb 6

i\_buffer: rmb 6

p\_buffer: rmb 7

Pave\_buffer: rmb 7

totE\_buffer: rmb 7

totC\_buffer: rmb 7

n\_nullsamp1: rmb 2

sum\_samp1: rmb 2 ;sum of 1ms voltage samples

n\_samp1: rmb 2 ;number of 1ms samples

sum\_ave1: rmb 2 ;sum of average reading for 60 cycles

n\_ave1: rmb 2 ;number of 60 cycle averages

v\_inst1: rmb 2 ;'instantaneous' voltage = sum\_ave50cyc/n\_ave50cyc(taken every 2000 samples or 2s)

i\_inst1: rmb 2 ;instantneous current

p\_inst1: rmb 2

n\_nullsamp2: rmb 2

sum\_samp2: rmb 2 ;sum of 1ms voltage samples

n\_samp2: rmb 2 ;number of 1ms samples

sum\_ave2: rmb 2 ;sum of average reading for 60 cycles

n\_ave2: rmb 2 ;number of 60 cycle averages

v\_inst2: rmb 2 ;'instantaneous' voltage = sum\_ave50cyc/n\_ave50cyc(taken every 2000 samples or 2s)

i\_inst2: rmb 2 ;instantneous current

p\_inst2: rmb 2

Pave1: rmb 2

Pave2: rmb 2

Iave1: rmb 2

n\_Iave1: rmb 2

Iave2: rmb 2

n\_Iave2: rmb 2

daily1: rmb 2

daily2: rmb 2

tcount1: rmb 2

tcount2: rmb 2

seconds1: rmb 2

seconds2: rmb 2

totE1: rmb 2

totE2: rmb 2

totcost1: rmb 2

totcost2: rmb 2

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

org $4000 ;begin program code

handler\_0: rti

start: lds #STACK

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

#include init\_hrdwr.inc ;initialize the hardware

#include init\_intrpt.inc ;initialize the RTI

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

main:

movw #$0000,v\_inst1 ;initialize variables

movw #$0000,v\_inst2

movw #$0000,i\_inst1

movw #$0000,i\_inst2

movw #$0000,p\_inst1

movw #$0000,p\_inst2

movw #$0000,n\_nullsamp1

movw #$0000,sum\_ave1

movw #$0000,n\_samp1

movw #$0000,sum\_samp1

movw #$0000,n\_ave1

movw #$0000,sum\_ave1

movw #$0000,n\_nullsamp2

movw #$0000,sum\_ave2

movw #$0000,n\_samp2

movw #$0000,sum\_samp2

movw #$0000,n\_ave2

movw #$0000,sum\_ave2

movb #$00,intFLG

movw #$0000,cost

movb #$00,pbtn\_status

movb #$01,disp\_no

movw #$0000,Iave1

movw #$0000,n\_Iave1

movw #$0000,Iave2

movw #$0000,n\_Iave2

movw #$0000,Pave1

movw #$0000,Pave2

movw #$0000,daily1

movw #$0000,daily2

movw #$0000,hold\_count

movw #$0000,tcount1

movw #$0000,tcount2

movw #$0000,seconds1

movw #$0000,seconds2

movw #$0000,totE1

movw #$0000,totE2

movw #$0000,totcost1

movw #$0000,totcost2

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

back:

jsr DISPLAY ;display proper output

check\_pgm:

brclr pbtn\_status,#$01,check\_out ;program cost if needed

jsr PROGRAM

check\_out:

brclr pbtn\_status,#$02,check\_done ;change/reset outlet

jsr OUTLET ;if needed

check\_done:

jmp back ;repeat

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

#include CNTRL\_FNCTNS.inc ;functions for flow control

#inclde PRINT\_CMNDS.inc ;various output commands

#include BUFF\_PREP.inc ;ascii buffer preparation

#include LCD\_cmnds.inc ;basic LCD commands

#include RTI\_ISR.inc ;Real Time Interrupt

#include DATA.inc ;output strings

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

#include vector\_table.inc

End

**;CNTRL\_FNCTNS.inc**

;Lee Sichello

;Purpose: A set of subroutines which alter the flow of control and user flags in the program. Makes output decisions

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

DISPLAY:

brset pbtn\_status,#$04,inc\_display

brset pbtn\_status,#$08,dec\_display

jmp display\_set

dec\_display: ldaa #$01

cmpa disp\_no

beq set\_disp6

dec disp\_no

bra display\_set

set\_disp6:

ldaa #6

staa disp\_no

bra display\_set

inc\_display: ldaa #$06

cmpa disp\_no

beq set\_disp1

inc disp\_no

bra display\_set

set\_disp1:

ldaa #1

staa disp\_no

bra display\_set

display\_set:

movb #$00,pbtn\_status

jsr clear\_LCD

disp\_1: ldaa #$01

cmpa disp\_no

bne disp\_2

jsr print\_i

rts

disp\_2: ldaa #$02

cmpa disp\_no

bne disp\_3

jsr print\_p

rts

disp\_3: ldaa #$03

cmpa disp\_no

bne disp\_4

jsr printPave

rts

disp\_4: ldaa #$04

cmpa disp\_no

bne disp\_5

jsr print\_totE

rts

disp\_5: ldaa #$05

cmpa disp\_no

bne disp\_6

jsr print\_totC

rts

disp\_6: jsr print\_daily

rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

PROGRAM:

bclr pbtn\_status,#$01

jsr clear\_LCD

ldx #str\_pgm

jsr putsLCD

ldaa #$C0

jsr cmd2LCD

ldd cost

ldy #cost\_buffer

jsr prep\_1dp\_buff

ldx #cost\_buffer

jsr putsLCD

ldx #str\_cents

jsr putsLCD

movw #$0000,pgm\_count

bclr intFLG,#$03

pgm\_back:

brclr intFLG,#$03,pgm\_check\_stat ;check pbtns until time is up

bclr intFLG,#$03 ;clear timeout flag

rts

pgm\_check\_stat:

brset pbtn\_status,#%00001000,pgm\_L ;go to left routine if pressed

brset pbtn\_status,#%00000100,pgm\_R ;go to right routine if pressed

jmp pgm\_back

pgm\_L:

ldd cost

cpd #0

beq print\_new\_cost

subd #5

std cost

bclr pbtn\_status,#%00001000

bra print\_new\_cost

pgm\_R:

ldd cost

cpd #500

beq print\_new\_cost

ldd cost

addd #5

std cost

bclr pbtn\_status,#%00000100

jmp print\_new\_cost

print\_new\_cost:

ldaa #$C0

jsr cmd2LCD

ldd cost

ldy #cost\_buffer

jsr prep\_1dp\_buff

ldx #cost\_buffer

jsr putsLCD

ldx #str\_cents

jsr putsLCD

movw #$0000,pgm\_count

bclr intFLG,#$02

jmp pgm\_back

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

OUTLET:

movw #$0000,hold\_count

outlet\_back:

brclr pbtn\_status,#$02,toggle\_outlet

ldy hold\_count

cpy #2000

bne outlet\_back

brset intFLG,#08,reset2

movw #$0000,Iave1

movw #$0000,n\_Iave1

movw #$0000,Pave1

movw #$0000,daily1

movw #$0000,tcount1

movw #$0000,seconds1

movw #$0000,totE1

movw #$0000,totcost1

bra outpt\_rst

reset2: movw #$0000,Iave2

movw #$0000,n\_Iave2

movw #$0000,Pave2

movw #$0000,daily2

movw #$0000,tcount2

movw #$0000,seconds2

movw #$0000,totE2

movw #$0000,totcost2

bra outpt\_rst

toggle\_outlet:

brset intFLG,#$08,clear\_outflg

bset intFLG,#$08 ;toggles the outlet

rts

outpt\_rst:

jsr clear\_LCD

ldx #str\_rst

jsr putsLCD

jsr print\_out

jsr de2s

movb #$0000,hold\_count

brset pbtn\_status,#$02,\*

rts

clear\_outflg:

bclr intFLG,#$08

rts

**;PRINT\_CMNDS.inc**

;Lee Sichello

;purpose: subroutines for printing the instantaneous voltage/current/power and average power and daily cost

print\_v: ldx #str\_voltage

jsr putsLCD

jsr print\_out

print\_v\_back:

brclr pbtn\_status,#$0F,print\_v\_back2

rts

print\_v\_back2: brclr intFLG,#$C0,print\_v\_back ;stay until print flag or no load goes high

ldaa #$C0

jsr cmd2LCD

brset intFLG,#$08,load\_v2

bclr intFLG,#$40

brset intFLG,#$10,NoLoad\_v

ldd v\_inst1

bra load\_v\_buff

load\_v2:

bclr intFLG,#$80

brset intFLG,#$20,NoLoad\_v

ldd v\_inst2

load\_v\_buff:

ldy #v\_buffer

jsr prep\_2dp\_buff

ldx #v\_buffer

jsr putsLCD

ldx #str\_Volts

jsr putsLCD

jmp print\_v\_back

NoLoad\_v: jsr print\_NoLoad

jmp print\_v\_back

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

print\_i: ldx #str\_current

jsr putsLCD

jsr print\_out

print\_i\_back:

brclr pbtn\_status,#$0F,print\_i\_back2

rts

print\_i\_back2: brclr intFLG,#$C0,print\_i\_back ;stay until print flag or no load goes high

ldaa #$C0

jsr cmd2LCD

brset intFLG,#$08,load\_i2

bclr intFLG,#$40

brset intFLG,#$10,NoLoad\_i

ldd i\_inst1

bra load\_i\_buff

load\_i2:

bclr intFLG,#$80

brset intFLG,#$20,NoLoad\_i

ldd i\_inst2

load\_i\_buff:

ldy #i\_buffer

jsr prep\_2dp\_buff

ldx #i\_buffer

jsr putsLCD

ldx #str\_Amps

jsr putsLCD

bclr intFLG,#$40

jmp print\_i\_back

NoLoad\_i: jsr print\_NoLoad

jmp print\_i\_back

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

print\_p: ldx #str\_power

jsr putsLCD

jsr print\_out

print\_p\_back:

brclr pbtn\_status,#$0F,print\_p\_back2

rts

print\_p\_back2: brclr intFLG,#$C0,print\_p\_back ;stay until print flag or no load goes high

ldaa #$C0

jsr cmd2LCD

cacl\_p12: ldd i\_inst1 ;current in mA in d

ldy #110 ;assumed voltage

emul ;d\*y->y:d contains power in mW

ldx #1000

ediv ;d:y/x->y (y contains power in W)

sty p\_inst1

ldd i\_inst2 ;current in mA in d

ldy #110 ;assumed voltage

emul ;d\*y->y:d contains power in mW

ldx #1000

ediv ;d:y/x->y (y contains power in W)

sty p\_inst2

brset intFLG,#$08,load\_p2

bclr intFLG,#$40

brset intFLG,#$10,NoLoad\_p

ldd p\_inst1

bra load\_p\_buff

load\_p2:

bclr intFLG,#$80

brset intFLG,#$20,NoLoad\_p

ldd p\_inst2

load\_p\_buff:

ldy #p\_buffer

jsr prep\_3dp\_buff

ldx #p\_buffer

jsr putsLCD

ldx #str\_kW

jsr putsLCD

bclr intFLG,#$40

jmp print\_p\_back

NoLoad\_p: jsr print\_NoLoad

jmp print\_i\_back

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

printPave: ldx #str\_Pave

jsr putsLCD

jsr print\_out

printPav\_back:

brclr pbtn\_status,#$0F,printPav\_back2

rts

printPav\_back2: brclr intFLG,#$C0,printPav\_back ;stay until print flag or no load goes high

ldaa #$C0

jsr cmd2LCD

calc\_Pav12: ldd Iave1 ;current in mA in d

ldy #110 ;assumed voltage

emul ;d\*y->y:d contains power in mW

ldx #1000

ediv ;d:y/x->y (y contains power in W)

sty Pave1

ldd Iave2 ;current in mA in d

ldy #110 ;assumed voltage

emul ;d\*y->y:d contains power in mW

ldx #1000

ediv ;d:y/x->y (y contains power in W)

sty Pave2

brset intFLG,#$08,loadPav2

;bclr intFLG,#$40

;brset intFLG,#$10,NoLoad\_Pav

ldd Pave1

bra loadPave\_buff

loadPav2: ldd Pave2

;bclr intFLG,#$80

;brset intFLG,#$20,NoLoad\_Pav

loadPave\_buff:

ldy #Pave\_buffer

jsr prep\_3dp\_buff

ldx #Pave\_buffer

jsr putsLCD

ldx #str\_kW

jsr putsLCD

bclr intFLG,#$40

jmp printPav\_back

NoLoad\_Pav: jsr print\_NoLoad

jmp printPav\_back

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

print\_daily:

ldaa #$80

jsr cmd2LCD

ldx #str\_daily

jsr putsLCD

jsr print\_out

daily\_back:

brclr pbtn\_status,#$0F,is\_cost\_set

rts

is\_cost\_set:

ldx cost

cpx #$0000

bne daily\_back2

ldaa #$C0

jsr cmd2LCD

ldx #str\_NoCost

jsr putsLCD

bra daily\_back

daily\_back2: brclr intFLG,#$C0,daily\_back ;stay until print flag or no load goes high

calc\_daily: ldd Iave1 ;current in mA in d

ldy #110 ;assumed voltage

emul ;d\*y->y:d contains power in mW

ldx #1000

ediv ;d:y/x->y (y contains power in W)

sty Pave1

ldy Pave1

ldd #24

emul ;y:d contains ave power(w) \* 24 hrs

ldy cost

emul

ldx #1000

ediv ;y:d contains cost in Wh\*(cents/kWh) = cents/K

sty daily1

ldd Iave2 ;current in mA in d

ldy #110 ;assumed voltage

emul ;d\*y->y:d contains power in mW

ldx #1000

ediv ;d:y/x->y (y contains power in W)

sty Pave2

ldd #24

emul ;y:d contains ave power(w) \* 24 hrs

ldy cost

emul

ldx #1000

ediv ;y:d contains cost in Wh\*(cents/kWh) = cents/K

sty daily2

ldaa #$C1

jsr cmd2LCD

brset intFLG,#$08,load\_daily2

ldd daily1

bra ld\_daily\_buff

load\_daily2: ldd daily2

ld\_daily\_buff:

cpd #10

bge outpt\_daily

ldaa #$C0

jsr cmd2LCD

ldx #str\_below0

jsr putsLCD

jmp print\_daily

outpt\_daily:

ldy #daily\_buffer

jsr prep\_2dp\_buff

ldx #daily\_buffer

jsr putsLCD

ldx #str\_11spaces

jsr putsLCD

ldaa #$C0

jsr cmd2LCD

ldaa #$24

jsr putcLCD

bclr intFLG,#$40

jmp daily\_back

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

print\_totE: ldx #str\_tot

jsr putsLCD

jsr print\_out

printtot\_back:

brclr pbtn\_status,#$0F,printtot\_back2

rts

printtot\_back2: brclr intFLG,#$C0,printtot\_back ;stay until print flag or no load goes high

ldaa #$C0

jsr cmd2LCD

calc\_tot12: ldd Iave1 ;current in mA in d

ldy #110 ;assumed voltage

emul ;d\*y->y:d contains power in mW

ldx #1000

ediv ;d:y/x->y (y contains power in W)

sty Pave1

ldd seconds1

emul ;d\*y->y:d = energy in W\*s

ldx #3600

ediv :y=total w\*hrs

sty totE1

ldd Iave2 ;current in mA in d

ldy #110 ;assumed voltage

emul ;d\*y->y:d contains power in mW

ldx #1000

ediv ;d:y/x->y (y contains power in W)

sty Pave2

ldd seconds2

emul ;d\*y->y:d = energy in W\*s

ldx #3600

ediv :y=total w\*hrs

sty totE2

brset intFLG,#$08,loadtot2

;bclr intFLG,#$40

;brset intFLG,#$10,NoLoad\_tot

ldd totE1

bra loadtot\_buff

loadtot2: ldd totE2

;bclr intFLG,#$80

;brset intFLG,#$20,NoLoad\_tot

loadtot\_buff:

ldy #totE\_buffer

jsr prep\_3dp\_buff

ldx #totE\_buffer

jsr putsLCD

ldx #str\_kWh

jsr putsLCD

bclr intFLG,#$40

jmp printtot\_back

NoLoad\_tot: jsr print\_NoLoad

jmp printtot\_back

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

print\_totC:

ldaa #$80

jsr cmd2LCD

ldx #str\_totcost

jsr putsLCD

jsr print\_out

printC\_back:

brclr pbtn\_status,#$0F,is\_cost\_setC

rts

is\_cost\_setC:

ldx cost

cpx #$0000

bne printC\_back2

ldaa #$C0

jsr cmd2LCD

ldx #str\_NoCost

jsr putsLCD

bra printC\_back

printC\_back2: brclr intFLG,#$C0,printC\_back ;stay until print flag or no load goes high

calc\_C12: ldd Iave1 ;current in mA in d

ldy #110 ;assumed voltage

emul ;d\*y->y:d contains power in mW

ldx #1000

ediv ;d:y/x->y (y contains power in W)

sty Pave1

ldd seconds1

emul ;d\*y->y:d = energy in W\*s

ldx #3600

ediv :y=total w\*hrs

sty totE1

ldd cost

emul ;y\*d->y:d contains total cost in thousands of dollars

ldx #1000

ediv ;y contains cost in cents

sty totcost1

ldd Iave2 ;current in mA in d

ldy #110 ;assumed voltage

emul ;d\*y->y:d contains power in mW

ldx #1000

ediv ;d:y/x->y (y contains power in W)

sty Pave2

ldd seconds2

emul ;d\*y->y:d = energy in W\*s

ldx #3600

ediv :y=total w\*hrs

sty totE2

ldd cost

emul ;y\*d->y:d contains total cost in thousands of dollars

ldx #1000

ediv ;y contains cost in cents

sty totcost2

ldaa #$C1

jsr cmd2LCD

brset intFLG,#$08,loadC2

;bclr intFLG,#$40

;brset intFLG,#$10,NoLoad\_C

ldd totcost1

bra loadC\_buff

loadC2: ldd totcost2

;bclr intFLG,#$80

;brset intFLG,#$20,NoLoad\_C

loadC\_buff:

cpd #10

bge outpt\_totC

ldaa #$C0

jsr cmd2LCD

ldx #str\_below0

jsr putsLCD

jmp print\_totC

outpt\_totC:

ldy #totC\_buffer

jsr prep\_2dp\_buff

ldx #totC\_buffer

jsr putsLCD

ldx #str\_11spaces

jsr putsLCD

ldaa #$C0

jsr cmd2LCD

ldaa #$24

jsr putcLCD

bclr intFLG,#$40

jmp printC\_back

NoLoad\_C: jsr print\_NoLoad

jmp printC\_back

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

print\_out:

ldaa #$8E

jsr cmd2LCD

ldaa #$23

jsr putcLCD

brset intFLG,#$08,print\_out2

print\_out1:

ldaa #$31

jsr putcLCD

rts

print\_out2:

ldaa #$32

jsr putcLCD

rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

print\_NoLoad: ldx #str\_NoLoad

jsr putsLCD

rts

**;buff\_prep.inc**

;Lee Sichello

;prepares buffers with various amounts of decimal places in ASCII for LCD output

prep\_1dp\_buff: pshx ;expects value to convert in d , beginning of 6 bit buffer address in y

pshy

pshd

ldx #100 ;

idiv ;d/x->x , rem->d

pshd

cpx #$0000

beq div\_10

xgdx

addb #$30 ;convert d to ascii

stab 0,Y

iny

div\_10:

puld

ldx #10 ;

idiv ;d/x->x , rem->d

pshd

xgdx

addb #$30 ;convert d to ascii

stab 0,Y

ldaa #46 ;decimal point

staa 1,Y

puld

addb #$30

stab 2,Y

ldaa #$20 ;space character

staa 3,Y

ldaa #$00 ;null character

staa 4,Y

puld

puly

pulx

rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

prep\_2dp\_buff: pshx ;expects value to convert in d , beginning of 6 bit buffer address in y

pshy

pshd

ldx #10000 ;

idiv ;d/x->x , rem->d

pshd

cpx #$0000

beq div\_1000

xgdx

addb #$30 ;convert d to ascii

stab 0,Y

iny

div\_1000:

puld

ldx #1000 ;

idiv ;d/x->x , rem->d

pshd

xgdx

addb #$30 ;convert d to ascii

stab 0,Y

ldaa #46 ;decimal point

staa 1,Y

puld

ldx #100

idiv ;(10xrem)/51->x (first didgit of fraction), rem->d

pshd

xgdx

addb #$30

stab 2,Y

puld

ldx #10

idiv

xgdx

addb #$30

stab 3,Y

ldaa #$20 ;space character

staa 4,Y

ldaa #$00 ;null character

staa 5,Y

puld

puly

pulx

rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

prep\_3dp\_buff: pshx ;expects value to convert in d , beginning of 6 bit buffer address in y

pshy

pshd

ldx #1000 ;

idiv ;d/x->x , rem->d

pshd

xgdx

addb #$30 ;convert d to ascii

stab 0,Y

ldaa #46 ;decimal point

staa 1,Y

puld

ldx #100

idiv ;(10xrem)/51->x (first didgit of fraction), rem->d

pshd

xgdx

addb #$30

stab 2,Y

puld

ldx #10

idiv

pshd

xgdx

addb #$30

stab 3,Y

puld

addb #$30

stab 4,Y

ldaa #$20 ;space character

staa 5,Y

ldaa #$00 ;null character

staa 6,Y

puld

puly

pulx

rts

**;LCD\_cmnds.inc**

lcd\_dat: equ PORTK

lcd\_dir: equ DDRK

lcd\_E equ $02

lcd\_RS equ $01

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

clear\_LCD: psha

ldaa #$01

jsr cmd2LCD

pula

rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

cmd2LCD: pshy

psha

psha

bclr lcd\_dat,lcd\_RS

bset lcd\_dat,lcd\_E

anda #$F0

lsra

lsra

oraa #lcd\_E

staa lcd\_dat

nop

nop

nop

bclr lcd\_dat,lcd\_E

pula

anda #$0F

lsla

lsla

bset lcd\_dat,lcd\_E

oraa #lcd\_E

staa lcd\_dat

nop

nop

nop

bclr lcd\_dat,lcd\_E

ldy #1

jsr de50us

pula

puly

rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

openLCD: pshy

psha

movb #$FF,lcd\_dir

ldy #10

jsr de10ms

ldaa #$28

jsr cmd2lcd

ldaa #$0F

jsr cmd2lcd

ldaa #$06

jsr cmd2lcd

ldaa #$01

jsr cmd2lcd

ldy #2

jsr de10ms

pula

puly

rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

putcLCD: pshy

psha

psha

bset lcd\_dat,lcd\_RS

bset lcd\_dat,lcd\_E

anda #$F0

lsra

lsra

oraa #$03

staa lcd\_dat

nop

nop

nop

bclr lcd\_dat,lcd\_E

pula

anda #$0F

lsla

lsla

bset lcd\_dat,lcd\_E

oraa #$03

staa lcd\_dat

nop

nop

nop

bclr lcd\_dat,lcd\_E

ldy #1

jsr de50us

pula

puly

rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

putsLCD:

ldaa 1,x+

beq donePS

jsr putcLCD

bra putsLCD

donePS: rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

de10ms: ldaa #10 ; delay 10 ms

staa counter

de10ms1: ldy #6000 ; 6000 x 4 = 24,000 cycles = 1ms

de10ms2: dey ; this instruction takes 1 cycle

bne de10ms2 ; this instruction takes 3 cycles

dec counter

bne de10ms1 ; not 250ms yet, delay again

rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

de50us: ldaa #250 ; delay 250 us

staa counter

de50us1: ldy #6 ; 6 x 4 = 24 cycles = 1us

de50us2: dey ; this instruction takes 1 cycle

bne de50us2 ; this instruction takes 3 cycles

dec counter

bne de50us1 ; not 250ms yet, delay again

rts

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

de2s: ldaa #50 ; delay 250 us

staa counter

de2s1: ldy #60000 ; 60000 x 4 = 240000 cycles = 0.1s

de2s2: dey ; this instruction takes 1 cycle

bne de2s2 ; this instruction takes 3 cycles

dec counter

bne de2s1 ; not 250ms yet, delay again

rts

**;init\_hrwr.inc**

;hardware initialization for adc and pushbuttons

init\_hrdwr:

movb #$00,INITRG ;configure registers(lab 6 handout)

movb #$11,INITRM

bclr CLKSEL,#$80 ;configure clock (lab 6 handout)

bset PLLCTL,#$40

movb #$2,SYNR

movb #$0,REFDV

nop

nop

bclr CRGFLG,#$08,\*

bset CLKSEL,#$80

ldx #REGBLK

ldaa #$ff

staa ddrj,x ; make port J an output port

staa ddrb,x ; make port B an output port

staa ddrp,x ; make port P an output port

staa ptp,x ; turn off 7-segment LED display

movb #$FF,ptp ; turn off 7-segment LED display

jsr openLCD ; turn on/enable LCD screen

movb #$00,DDRH

ldaa #$0C

jsr cmd2LCD

movb #$C0,atd0ctl2 ;configure adc registers

movb #$08,atd0ctl3

movb #$05,atd0ctl4

**;init\_intrpt.inc**

;rti interrupts initialized here

init\_intrpt:

movb #%01000000,RTICTL ;set RTICLOCK to 2^13osclks , or 1ms

bset CRGINT,#$80 ;enable RTI locally

cli ;enable interrupt globally

**;RTI\_ISR.inc**

;Lee Sichello

;interrupt service routine that samples pushbuttons and current

RTI\_ISR: bset CRGFLG,#$80

;---------------------------------------

TIMING\_STUFF:

ldx tcount1

inx

stx tcount1

cpx #1000

bne TIMING2

movw #$0000,tcount1

ldx seconds1

inx

stx seconds1

TIMING2:

ldx tcount2

inx

stx tcount2

cpx #1000

bne PGM\_STUFF

movw #$0000,tcount2

ldx seconds2

inx

stx seconds2

;---------------------------------------

PGM\_STUFF:

ldx pgm\_count

cpx #5000

bne inc\_pgm\_count

movw #$0000,pgm\_count

bset intFLG,#$03 ;set the timeout bit

jmp CHECK\_RST\_HOLD

inc\_pgm\_count:

inx

stx pgm\_count

jmp CHECK\_RST\_HOLD

;---------------------------------------

CHECK\_RST\_HOLD:

ldy hold\_count

iny

sty hold\_count

ldaa pth ;retrieve pushbttn status in bits 0-3 (0=pressed)

coma ;1 = pressed

anda #$0F ;clear upper bits

staa pbtn\_sample

brclr pbtn\_sample,#$02,PBTN\_STUFF

movb #$02,pbtn\_status

done\_RST: rti

;---------------------------------------

PBTN\_STUFF:

cmpa #$00

beq nothing\_pressed ;if no button is down, go to nonthing pressed

ldx #750

cpx pbtn\_count

bne low\_pbtn\_count ;if pbtn\_cout != 500, go to nothing pressed

bra pbtn\_pressed

nothing\_pressed:

movb #$00,pbtn\_status

ldx #750

stx pbtn\_count

bra ADC\_STUFF

low\_pbtn\_count:

ldx pbtn\_count

inx

stx pbtn\_count

bra ADC\_STUFF

pbtn\_pressed:

staa pbtn\_status

movw #$00,pbtn\_count

jmp ADC\_STUFF

;--------------------------------------

ADC\_STUFF:

;========

adc\_out1:

movb #$86,atd0ctl5 ;tells adc to take reading on an6

brclr atd0stat0,#$80,\* ;wait until reading is finished

ldaa atd0dr0h ;retrieve sample from data register

ldab atd0dr0l ;get upper half of sample into D

cpd #5

bge keep\_sample1 ;if sample is greater than or = #$01

null\_sample1: ldx n\_nullsamp1

inx

stx n\_nullsamp1

ldx #1000

cpx n\_nullsamp1

ble no\_load1 ;if 40 of 50 samples are null, there is no load

jmp adc\_out2

;set v\_inst = 0

no\_load1: movw #$0000,v\_inst1

movw #$0000,i\_inst1

movw #$0000,n\_nullsamp1

bset intFLG,#$10

bset intFLG,#$40 ;set print flag

jmp calc\_Iave1

keep\_sample1: addd sum\_samp1 ;add to sum of samples

std sum\_samp1 ;store new sum

ldx n\_samp1 ;increment sample count

inx

stx n\_samp1

ldx #50

cpx n\_samp1

ble get\_ave1 ;if n\_samples > 50 return get\_ave

jmp adc\_out2

get\_ave1: ldd sum\_samp1 ;if n\_samples >= 50 take sum/count=average

idiv ;d/x->x where x = average for 50 samples

xgdx ;put average into d

addd sum\_ave1 ;add 50cycle ave to sum

std sum\_ave1

ldx n\_ave1 ;increment sample count

inx

stx n\_ave1

movw #$0000,n\_samp1 ;reset the cycle count

movw #$0000,sum\_samp1

movw #$0000,n\_nullsamp1

bclr intFLG,#$10

ldx #10

cpx n\_ave1

ble get\_v\_inst1 ;if n\_samples > 50 return get\_ave

jmp adc\_out2

get\_v\_inst1: ldd sum\_ave1 ;if n\_samples >= 40 take sum/count=average

idiv ;d/x->x where x = average for 2000 samples

xgdx ;put average into d

ldy #1000

emul ;d\*1000 into y:d

ldx #205

ediv ;y:d/x into Y, remainder into D (y=voltage in mV)

sty v\_inst1

ldd #357 ;conversion factor times 100

emul ;v\_inst\*395 into y:d (result fits in d though)

ldx #100

ediv ;v\_vinst\*395/100 into y gives current (in mA)

sty i\_inst1

movw #$0000,n\_ave1 ;reset the cycle count

movw #$0000,sum\_ave1

bset intFLG,#$40 ;set the intFLG

calc\_Iave1:

ldd Iave1

ldy n\_Iave1

emul ;Iave\*n\_Iave->y:d (weighting)

addd i\_inst1 ;add new entry to product

bcc no\_carry1 ;incremement y if needed

iny

no\_carry1: ldx n\_Iave1

inx

stx n\_Iave1

ediv ;[(old ave)\*entries + new instantaneous]/entries+1 -> y

sty Iave1

;=========

adc\_out2:

movb #$00,atd0dr0h

movb #$00,atd0dr0h

movb #$82,atd0ctl5 ;tells adc to take reading on an5

brclr atd0stat0,#$80,\* ;wait until reading is finished

ldaa atd0dr0h ;retrieve sample from data register

ldab atd0dr0l ;get upper half of sample into D

cpd #5

bge keep\_sample2 ;if sample is greater than or = #$01

null\_sample2: ldx n\_nullsamp2

inx

stx n\_nullsamp2

ldx #1000

cpx n\_nullsamp2

ble no\_load2 ;if 40 of 50 samples are null, there is no load

jmp done\_adc

;set v\_inst = 0

no\_load2: movw #$0000,v\_inst2

movw #$0000,i\_inst2

movw #$0000,n\_nullsamp2

bset intFLG,#$20

bset intFLG,#$80 ;set print flag

jmp calc\_Iave2

keep\_sample2: addd sum\_samp2 ;add to sum of samples

std sum\_samp2 ;store new sum

ldx n\_samp2 ;increment sample count

inx

stx n\_samp2

ldx #50

cpx n\_samp2

ble get\_ave2 ;if n\_samples > 50 return get\_ave

jmp done\_adc

get\_ave2: ldd sum\_samp2 ;if n\_samples >= 50 take sum/count=average

idiv ;d/x->x where x = average for 50 samples

xgdx ;put average into d

addd sum\_ave2 ;add 50cycle ave to sum

std sum\_ave2

ldx n\_ave2 ;increment sample count

inx

stx n\_ave2

movw #$0000,n\_samp2 ;reset the cycle count

movw #$0000,sum\_samp2

movw #$0000,n\_nullsamp2

bclr intFLG,#$20

ldx #10

cpx n\_ave2

ble get\_v\_inst2 ;if n\_samples > 50 return get\_ave

jmp done\_adc

get\_v\_inst2: ldd sum\_ave2 ;if n\_samples >= 40 take sum/count=average

idiv ;d/x->x where x = average for 2000 samples

xgdx ;put average into d

ldy #1000

emul ;d\*1000 into y:d

ldx #205

ediv ;y:d/x into Y, remainder into D (y=voltage in mV)

sty v\_inst2

ldd #379 ;conversion factor times 100

emul ;v\_inst\*395 into y:d (result fits in d though)

ldx #100

ediv ;v\_vinst\*395/100 into y gives current (in mA)

sty i\_inst2

movw #$0000,n\_ave2 ;reset the cycle count

movw #$0000,sum\_ave2

bset intFLG,#$80 ;set the intFLG

calc\_Iave2:

ldd Iave2

ldy n\_Iave2

emul ;Iave\*n\_Iave->y:d (weighting)

addd i\_inst2 ;add new entry to product

bcc no\_carry2 ;incremement y if needed

iny

no\_carry2: ldx n\_Iave2

inx

stx n\_Iave2

ediv ;[(old ave)\*entries + new instantaneous]/entries+1 -> y

sty Iave2

done\_adc: rti

**;DATA.inc**

;Lee Sichello

;A set of strings for LCD output

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

str\_pgm: fcc 'SET LOCAL COST '

fcb $00

str\_cents: fcc 'cents/kWh '

fcb $00

str\_voltage: fcc 'VOLTAGE '

fcb $00

str\_current: fcc 'CURRENT '

fcb $00

str\_power: fcc 'POWER '

fcb $00

str\_Pave: fcc 'AVERAGE POWER '

fcb $00

str\_daily: fcc 'DAILY COST '

fcb $00

str\_$: fcc '$'

fcb $00

str\_Amps: fcc 'Amps '

fcb $00

str\_Volts: fcc 'Volts '

fcb $00

str\_kWh fcc 'kWh '

fcb $00

str\_kW: fcc 'kW '

fcb #$00

str\_NoLoad: fcc '\*\*NO LOAD\*\*'

fcb $00

str\_rst: fcc 'RESET OUTLET '

fcb $00

str\_NoCost: fcc 'Must Set $/kWh '

fcb $00

str\_below0: fcc 'Under 1 cent '

fcb $00

str\_tot: fcc 'TOTAL ENERGY'

fcb $00

str\_11spaces: fcc ' '

fcb $00

str\_totcost: fcc 'TOTAL COST'

fcb $00

; **D\_variant\_registers.inc**

; Register addresses in this file can be relocated by setting REGBASE to a valid offset.

;(YOU MUST ACTUALLY RELOCATE THE 9S12 REGISTER BLOCK for this to work)

REGBASE equ $0

\*

\*

\* HC12 i/o register locations (Not all subsystems exist in all D versions)

\*

\*

PORTA equ REGBASE+0 ;port a = address lines a8 - a15

PORTB equ REGBASE+1 ;port b = address lines a0 - a7

DDRA equ REGBASE+2 ;port a direction register

DDRB equ REGBASE+3 ;port b direction register

PORTE equ REGBASE+8 ;port e = mode,irq and control signals

DDRE equ REGBASE+9 ;port e direction register

PEAR equ REGBASE+$a ;port e assignments

MODE equ REGBASE+$b ;mode register

PUCR equ REGBASE+$c ;port pull-up control register

RDRIV equ REGBASE+$d ;port reduced drive control register

EBICTL equ REGBASE+$e ;e stretch control

INITRM equ REGBASE+$10 ;ram location register

INITRG equ REGBASE+$11 ;register location register

INITEE equ REGBASE+$12 ;eeprom location register

MISC equ REGBASE+$13 ;miscellaneous mapping control

MTST0 equ REGBASE+$14 ; reserved

ITCR equ REGBASE+$15 ;interrupt test control register

ITEST equ REGBASE+$16 ;interrupt test register

MTST1 equ REGBASE+$17 ; reserved

PARTIDH equ REGBASE+$1a ;part id high

PARTIDL equ REGBASE+$1b ;part id low

MEMSIZ0 equ REGBASE+$1c ;memory size

MEMSIZ1 equ REGBASE+$1d ;memory size

IRQCR equ REGBASE+$1e ;interrupt control register

INTCR equ REGBASE+$1e ;interrupt control register (old name)

HPRIO equ REGBASE+$1f ;high priority reg

BKPCT0 equ REGBASE+$28 ;break control register

BKPCT1 equ REGBASE+$29 ;break control register

BKP0X equ REGBASE+$2a ; break 0 index register

BKP0H equ REGBASE+$2b ; break 0 pointer high

BRP0L equ REGBASE+$2c ; break 0 pointer low

BKP1X equ REGBASE+$2d ; break 1 index register

BKP1H equ REGBASE+$2e ; break 1 pointer high

BRP1L equ REGBASE+$2f ; break 1 pointer low

PPAGE equ REGBASE+$30 ;program page register

PORTK equ REGBASE+$32 ;port k data

DDRK equ REGBASE+$33 ;port k direction

SYNR equ REGBASE+$34 ; synthesizer / multiplier register

REFDV equ REGBASE+$35 ; reference divider register

CTFLG equ REGBASE+$36 ; reserved

CRGFLG equ REGBASE+$37 ; pll flags register

CRGINT equ REGBASE+$38 ; pll interrupt register

CLKSEL equ REGBASE+$39 ; clock select register

PLLCTL equ REGBASE+$3a ; pll control register

RTICTL equ REGBASE+$3b ;real time interrupt control

COPCTL equ REGBASE+$3c ;watchdog control

FORBYP equ REGBASE+$3d ;

CTCTL equ REGBASE+$3e ;

ARMCOP equ REGBASE+$3f ;cop reset register

TIOS equ REGBASE+$40 ;timer input/output select

CFORC equ REGBASE+$41 ;timer compare force

OC7M equ REGBASE+$42 ;timer output compare 7 mask

OC7D equ REGBASE+$43 ;timer output compare 7 data

TCNT equ REGBASE+$44 ;timer counter register hi

\*TCNT equ REGBASE+$45 ;timer counter register lo

TSCR equ REGBASE+$46 ;timer system control register (Old Name)

TSCR1 equ REGBASE+$46 ;timer system control register

TTOV equ REGBASE+$47 ;reserved

TCTL1 equ REGBASE+$48 ;timer control register 1

TCTL2 equ REGBASE+$49 ;timer control register 2

TCTL3 equ REGBASE+$4a ;timer control register 3

TCTL4 equ REGBASE+$4b ;timer control register 4

TMSK1 equ REGBASE+$4c ;timer interrupt mask 1 (Old Name)

TIE equ REGBASE+$4c ;timer interrupt mask 1

TMSK2 equ REGBASE+$4d ;timer interrupt mask 2 (Old Name)

TSCR2 equ REGBASE+$4d ;timer interrupt mask 2

TFLG1 equ REGBASE+$4e ;timer flags 1

TFLG2 equ REGBASE+$4f ;timer flags 2

TC0 equ REGBASE+$50 ;timer capture/compare register 0

\*TC0 equ REGBASE+$51 ; low byte

TC1 equ REGBASE+$52 ;timer capture/compare register 1

\*TC1 equ REGBASE+$53 ;; low byte

TC2 equ REGBASE+$54 ;timer capture/compare register 2

\*TC2 equ REGBASE+$55 ; low byte

TC3 equ REGBASE+$56 ;timer capture/compare register 3

\*TC3 equ REGBASE+$57 ; low byte

TC4 equ REGBASE+$58 ;timer capture/compare register 4

\*TC4 equ REGBASE+$59 ; low byte

TC5 equ REGBASE+$5a ;timer capture/compare register 5

\*TC5 equ REGBASE+$5b ; low byte

TC6 equ REGBASE+$5c ;timer capture/compare register 6

\*TC6 equ REGBASE+$5d ; low byte

TC7 equ REGBASE+$5e ;timer capture/compare register 7

\*TC7 equ REGBASE+$5f ; low byte

PACTL equ REGBASE+$60 ;pulse accumulator controls

PAFLG equ REGBASE+$61 ;pulse accumulator flags

PACN3 equ REGBASE+$62 ;pulse accumulator counter 3

PACN2 equ REGBASE+$63 ;pulse accumulator counter 2

PACN1 equ REGBASE+$64 ;pulse accumulator counter 1

PACN0 equ REGBASE+$65 ;pulse accumulator counter 0

MCCTL equ REGBASE+$66 ;modulus down conunter control

MCFLG equ REGBASE+$67 ;down counter flags

ICPAR equ REGBASE+$68 ;input pulse accumulator control

DLYCT equ REGBASE+$69 ;delay count to down counter

ICOVW equ REGBASE+$6a ;input control overwrite register

ICSYS equ REGBASE+$6b ;input control system control

TIMTST equ REGBASE+$6d ;timer test register

PBCTL equ REGBASE+$70 ; pulse accumulator b control

PBFLG equ REGBASE+$71 ; pulse accumulator b flags

PA3H equ REGBASE+$72 ; pulse accumulator holding register 3

PA2H equ REGBASE+$73 ; pulse accumulator holding register 2

PA1H equ REGBASE+$74 ; pulse accumulator holding register 1

PA0H equ REGBASE+$75 ; pulse accumulator holding register 0

MCCNT equ REGBASE+$76 ; modulus down counter register

\*MCCNTL equ REGBASE+$77 ; low byte

TCOH equ REGBASE+$78 ; capture 0 holding register

\*TCOH equ REGBASE+$79 ; low byte

TC1H equ REGBASE+$7a ; capture 1 holding register

\*TC1H equ REGBASE+$7b ; low byte

TC2H equ REGBASE+$7c ; capture 2 holding register

\*TC2H equ REGBASE+$7d ; low byte

TC3H equ REGBASE+$7e ; capture 3 holding register

\*TC3H equ REGBASE+$7f ; low byte

ATD0CTL0 equ REGBASE+$80 ;adc control 0 (reserved)

ATD0CTL1 equ REGBASE+$81 ;adc control 1 (reserved)

ATD0CTL2 equ REGBASE+$82 ;adc control 2

ATD0CTL3 equ REGBASE+$83 ;adc control 3

ATD0CTL4 equ REGBASE+$84 ;adc control 4

ATD0CTL5 equ REGBASE+$85 ;adc control 5

ATD0STAT0 equ REGBASE+$86 ;adc status register 0

ATD0TEST0 equ REGBASE+$88 ;adc test (reserved)

ATD0TEST1 equ REGBASE+$89 ;adc test (reserved)

ATD0STAT1 equ REGBASE+$8b ;adc status register 1

ATD0DIEN equ REGBASE+$8d ;

PORTAD0 equ REGBASE+$8f ;port adc = input only

ATD0DR0H equ REGBASE+$90 ;adc result 0 register

ATD0DR0L equ REGBASE+$91 ; low byte

ATD0DR1H equ REGBASE+$92 ;adc result 1 register

ATD0DR1L equ REGBASE+$93 ; low byte

ATD0DR2H equ REGBASE+$94 ;adc result 2 register

ATD0DR2L equ REGBASE+$95 ; low byte

ATD0DR3H equ REGBASE+$96 ;adc result 3 register

ATD0DR3L equ REGBASE+$97 ; low byte

ATD0DR4H equ REGBASE+$98 ;adc result 4 register

ATD0DR4L equ REGBASE+$99 ; low byte

ATD0DR5H equ REGBASE+$9a ;adc result 5 register

ATD0DR5L equ REGBASE+$9B ; low byte

ATD0DR6H equ REGBASE+$9c ;adc result 6 register

ATD0DR6L equ REGBASE+$9D ; low byte

ATD0DR7H equ REGBASE+$9e ;adc result 7 register

ATD0DR7L equ REGBASE+$9F ; low byte

PWME equ REGBASE+$a0 ;pwm enable

PWMPOL equ REGBASE+$a1 ;pwm polarity

PWMCLK equ REGBASE+$a2 ;pwm clock select register

PWMPRCLK equ REGBASE+$a3 ;pwm prescale clock select register

PWMCAE equ REGBASE+$a4 ;pwm center align select register

PWMCTL equ REGBASE+$a5 ;pwm control register

PWMTST equ REGBASE+$a6 ;reserved

PWMPRSC equ REGBASE+$a7 ;reserved

PWMSCLA equ REGBASE+$a8 ;pwm scale a

PWMSCLB equ REGBASE+$a9 ;pwm scale b

PWMSCNTA equ REGBASE+$aa ;reserved

PWMSCNTB equ REGBASE+$ab ;reserved

PWMCNT0 equ REGBASE+$ac ;pwm channel 0 counter

PWMCNT1 equ REGBASE+$ad ;pwm channel 1 counter

PWMCNT2 equ REGBASE+$ae ;pwm channel 2 counter

PWMCNT3 equ REGBASE+$af ;pwm channel 3 counter

PWMCNT4 equ REGBASE+$b0 ;pwm channel 4 counter

PWMCNT5 equ REGBASE+$b1 ;pwm channel 5 counter

PWMCNT6 equ REGBASE+$b2 ;pwm channel 6 counter

PWMCNT7 equ REGBASE+$b3 ;pwm channel 7 counter

PWMPER0 equ REGBASE+$b4 ;pwm channel 0 period

PWMPER1 equ REGBASE+$b5 ;pwm channel 1 period

PWMPER2 equ REGBASE+$b6 ;pwm channel 2 period

PWMPER3 equ REGBASE+$b7 ;pwm channel 3 period

PWMPER4 equ REGBASE+$b8 ;pwm channel 4 period

PWMPER5 equ REGBASE+$b9 ;pwm channel 5 period

PWMPER6 equ REGBASE+$ba ;pwm channel 6 period

PWMPER7 equ REGBASE+$bb ;pwm channel 7 period

PWMDTY0 equ REGBASE+$bc ;pwm channel 0 duty cycle

PWMDTY1 equ REGBASE+$bd ;pwm channel 1 duty cycle

PWMDTY2 equ REGBASE+$be ;pwm channel 2 duty cycle

PWMDTY3 equ REGBASE+$bf ;pwm channel 3 duty cycle

PWMDTY4 equ REGBASE+$c0 ;pwm channel 4 duty cycle

PWMDTY5 equ REGBASE+$c1 ;pwm channel 5 duty cycle

PWMDTY6 equ REGBASE+$c2 ;pwm channel 6 duty cycle

PWMDTY7 equ REGBASE+$c3 ;pwm channel 7 duty cycle

PWMSDN equ REGBASE+$c4 ;pwm shutdown register

SCI0BDH equ REGBASE+$c8 ;sci 0 baud reg hi byte

SCI0BDL equ REGBASE+$c9 ;sci 0 baud reg lo byte

SCI0CR1 equ REGBASE+$ca ;sci 0 control1 reg

SCI0CR2 equ REGBASE+$cb ;sci 0 control2 reg

SCI0SR1 equ REGBASE+$cc ;sci 0 status reg 1

SCI0SR2 equ REGBASE+$cd ;sci 0 status reg 2

SCI0DRH equ REGBASE+$ce ;sci 0 data reg hi

SCI0DRL equ REGBASE+$cf ;sci 0 data reg lo

SCI1BDH equ REGBASE+$d0 ;sci 1 baud reg hi byte

SCI1BDL equ REGBASE+$d1 ;sci 1 baud reg lo byte

SCI1CR1 equ REGBASE+$d2 ;sci 1 control1 reg

SCI1CR2 equ REGBASE+$d3 ;sci 1 control2 reg

SCI1SR1 equ REGBASE+$d4 ;sci 1 status reg 1

SCI1SR2 equ REGBASE+$d5 ;sci 1 status reg 2

SCI1DRH equ REGBASE+$d6 ;sci 1 data reg hi

SCI1DRL equ REGBASE+$d7 ;sci 1 data reg lo

SPI0CR1 equ REGBASE+$d8 ;spi 0 control1 reg

SPI0CR2 equ REGBASE+$d9 ;spi 0 control2 reg

SPI0BR equ REGBASE+$da ;spi 0 baud reg

SPI0SR equ REGBASE+$db ;spi 0 status reg hi

SPI0DR equ REGBASE+$dd ;spi 0 data reg

IBAD equ REGBASE+$e0 ;i2c bus address register

IBFD equ REGBASE+$e1 ;i2c bus frequency divider

IBCR equ REGBASE+$e2 ;i2c bus control register

IBSR equ REGBASE+$e3 ;i2c bus status register

IBDR equ REGBASE+$e4 ;i2c bus message data register

DLCBCR1 equ REGBASE+$e8 ;bdlc control regsiter 1

DLCBSVR equ REGBASE+$e9 ;bdlc state vector register

DLCBCR2 equ REGBASE+$ea ;bdlc control register 2

DLCBDR equ REGBASE+$eb ;bdlc data register

DLCBARD equ REGBASE+$ec ;bdlc analog delay register

DLCBRSR equ REGBASE+$ed ;bdlc rate select register

DLCSCR equ REGBASE+$ee ;bdlc control register

DLCBSTAT equ REGBASE+$ef ;bdlc status register

SPI1CR1 equ REGBASE+$f0 ;spi 1 control1 reg

SPI1CR2 equ REGBASE+$f1 ;spi 1 control2 reg

SPI1BR equ REGBASE+$f2 ;spi 1 baud reg

SPI1SR equ REGBASE+$f3 ;spi 1 status reg hi

SPI1DR equ REGBASE+$f5 ;spi 1 data reg

SPI2CR1 equ REGBASE+$f8 ;spi 2 control1 reg

SPI2CR2 equ REGBASE+$f9 ;spi 2 control2 reg

SPI2BR equ REGBASE+$fa ;spi 2 baud reg

SPI2SR equ REGBASE+$fb ;spi 2 status reg hi

SPI2DR equ REGBASE+$fd ;spi 2 data reg

FCLKDIV equ REGBASE+$100 ;flash clock divider

FSEC equ REGBASE+$101 ;flash security register

FTSTMOD equ REGBASE+$102 ;TEST register

FCNFG equ REGBASE+$103 ;flash configuration register

FPROT equ REGBASE+$104 ;flash protection register

FSTAT equ REGBASE+$105 ;flash status register

FCMD equ REGBASE+$106 ;flash command register

FADDRHI equ REGBASE+$108 ;flash addr register hi

FADDRLO equ REGBASE+$109 ;flash addr register lo

FDATAHI equ REGBASE+$10a ;flash data register hi

FDATALO equ REGBASE+$10b ;flash data register lo

ECLKDIV equ REGBASE+$110 ;eeprom clock divider

ECNFG equ REGBASE+$113 ;eeprom configuration register

EPROT equ REGBASE+$114 ;eeprom protection register

ESTAT equ REGBASE+$115 ;eeprom status register

ECMD equ REGBASE+$116 ;eeprom command register

EADDRHI equ REGBASE+$118 ;eeprom addr register hi

EADDRLO equ REGBASE+$119 ;eeprom addr register lo

EDATAHI equ REGBASE+$11a ;eeprom data register hi

EDATALO equ REGBASE+$11b ;eeprom data register lo

ATD1CTL0 equ REGBASE+$120 ;adc1 control 0 (reserved)

ATD1CTL1 equ REGBASE+$121 ;adc1 control 1 (reserved)

ATD1CTL2 equ REGBASE+$122 ;adc1 control 2

ATD1CTL3 equ REGBASE+$123 ;adc1 control 3

ATD1CTL4 equ REGBASE+$124 ;adc1 control 4

ATD1CTL5 equ REGBASE+$125 ;adc1 control 5

ATD1STAT0 equ REGBASE+$126 ;adc1 status register hi

ATD1TEST0 equ REGBASE+$128 ;adc1 test (reserved)

ATD1TEST1 equ REGBASE+$129 ;adc1 test (reserved)

ATD1STAT1 equ REGBASE+$12b ;adc1 status register lo

ATD1DIEN equ REGBASE+$12d ;adc1 input enable register

PORTAD1 equ REGBASE+$12f ;port adc1 = input only

ATD1DR0H equ REGBASE+$130 ;adc1 result 0 register

ATD1DR0L equ REGBASE+$131 ;low byte

ATD1DR1H equ REGBASE+$132 ;adc1 result 1 register

ATD1DR1L equ REGBASE+$133 ;low byte

ATD1DR2H equ REGBASE+$134 ;adc1 result 2 register

ATD1DR2L equ REGBASE+$135 ;low byte

ATD1DR3H equ REGBASE+$136 ;adc1 result 3 register

ATD1DR3L equ REGBASE+$137 ;low byte

ATD1DR4H equ REGBASE+$138 ;adc1 result 4 register

ATD1DR4L equ REGBASE+$139 ;low byte

ATD1DR5H equ REGBASE+$13a ;adc1 result 5 register

ATD1DR5L equ REGBASE+$13b ;low byte

ATD1DR6H equ REGBASE+$13c ;adc1 result 6 register

ATD1DR6L equ REGBASE+$13d ;low byte

ATD1DR7H equ REGBASE+$13e ;adc1 result 7 register

ATD1DR7L equ REGBASE+$13f ;low byte

CAN0CTL0 equ REGBASE+$140 ;can0 control register 0

CAN0CTL1 equ REGBASE+$141 ;can0 control register 1

CAN0BTR0 equ REGBASE+$142 ;can0 bus timing register 0

CAN0BTR1 equ REGBASE+$143 ;can0 bus timing register 1

CAN0RFLG equ REGBASE+$144 ;can0 receiver flags

CAN0RIER equ REGBASE+$145 ;can0 receiver interrupt enables

CAN0TFLG equ REGBASE+$146 ;can0 transmit flags

CAN0TIER equ REGBASE+$147 ;can0 transmit interrupt enables

CAN0TARQ equ REGBASE+$148 ;can0 transmit message abort control

CAN0TAAK equ REGBASE+$149 ;can0 transmit message abort status

CAN0TBSEL equ REGBASE+$14a ;can0 transmit buffer select

CAN0IDAC equ REGBASE+$14b ;can0 identfier acceptance control

CAN0RXERR equ REGBASE+$14e ;can0 receive error counter

CAN0TXERR equ REGBASE+$14f ;can0 transmit error counter

CAN0IDAR0 equ REGBASE+$150 ;can0 identifier acceptance register 0

CAN0IDAR1 equ REGBASE+$151 ;can0 identifier acceptance register 1

CAN0IDAR2 equ REGBASE+$152 ;can0 identifier acceptance register 2

CAN0IDAR3 equ REGBASE+$153 ;can0 identifier acceptance register 3

CAN0IDMR0 equ REGBASE+$154 ;can0 identifier mask register 0

CAN0IDMR1 equ REGBASE+$155 ;can0 identifier mask register 1

CAN0IDMR2 equ REGBASE+$156 ;can0 identifier mask register 2

CAN0IDMR3 equ REGBASE+$157 ;can0 identifier mask register 3

CAN0IDAR4 equ REGBASE+$158 ;can0 identifier acceptance register 4

CAN0IDAR5 equ REGBASE+$159 ;can0 identifier acceptance register 5

CAN0IDAR6 equ REGBASE+$15a ;can0 identifier acceptance register 6

CAN0IDAR7 equ REGBASE+$15b ;can0 identifier acceptance register 7

CAN0IDMR4 equ REGBASE+$15c ;can0 identifier mask register 4

CAN0IDMR5 equ REGBASE+$15d ;can0 identifier mask register 5

CAN0IDMR6 equ REGBASE+$15e ;can0 identifier mask register 6

CAN0IDMR7 equ REGBASE+$15f ;can0 identifier mask register 7

CAN0RXFG equ REGBASE+$160 ;can0 rx foreground buffer thru +$16f

CAN0TXFG equ REGBASE+$170 ;can0 tx foreground buffer thru +$17f

CAN1CTL0 equ REGBASE+$180 ;can1 control register 0

CAN1CTL1 equ REGBASE+$181 ;can1 control register 1

CAN1BTR0 equ REGBASE+$182 ;can1 bus timing register 0

CAN1BTR1 equ REGBASE+$183 ;can1 bus timing register 1

CAN1RFLG equ REGBASE+$184 ;can1 receiver flags

CAN1RIER equ REGBASE+$185 ;can1 receiver interrupt enables

CAN1TFLG equ REGBASE+$186 ;can1 transmit flags

CAN1TIER equ REGBASE+$187 ;can1 transmit interrupt enables

CAN1TARQ equ REGBASE+$188 ;can1 transmit message abort control

CAN1TAAK equ REGBASE+$189 ;can1 transmit message abort status

CAN1TBSEL equ REGBASE+$18a ;can1 transmit buffer select

CAN1IDAC equ REGBASE+$18b ;can1 identfier acceptance control

CAN1RXERR equ REGBASE+$18e ;can1 receive error counter

CAN1TXERR equ REGBASE+$18f ;can1 transmit error counter

CAN1IDAR0 equ REGBASE+$190 ;can1 identifier acceptance register 0

CAN1IDAR1 equ REGBASE+$191 ;can1 identifier acceptance register 1

CAN1IDAR2 equ REGBASE+$192 ;can1 identifier acceptance register 2

CAN1IDAR3 equ REGBASE+$193 ;can1 identifier acceptance register 3

CAN1IDMR0 equ REGBASE+$194 ;can1 identifier mask register 0

CAN1IDMR1 equ REGBASE+$195 ;can1 identifier mask register 1

CAN1IDMR2 equ REGBASE+$196 ;can1 identifier mask register 2

CAN1IDMR3 equ REGBASE+$197 ;can1 identifier mask register 3

CAN1IDAR4 equ REGBASE+$198 ;can1 identifier acceptance register 4

CAN1IDAR5 equ REGBASE+$199 ;can1 identifier acceptance register 5

CAN1IDAR6 equ REGBASE+$19a ;can1 identifier acceptance register 6

CAN1IDAR7 equ REGBASE+$19b ;can1 identifier acceptance register 7

CAN1IDMR4 equ REGBASE+$19c ;can1 identifier mask register 4

CAN1IDMR5 equ REGBASE+$19d ;can1 identifier mask register 5

CAN1IDMR6 equ REGBASE+$19e ;can1 identifier mask register 6

CAN1IDMR7 equ REGBASE+$19f ;can1 identifier mask register 7

CAN1RXFG equ REGBASE+$1a0 ;can1 rx foreground buffer thru +$1af

CAN1TXFG equ REGBASE+$1b0 ;can1 tx foreground buffer thru +$1bf

CAN2CTL0 equ REGBASE+$1c0 ;can2 control register 0

CAN2CTL1 equ REGBASE+$1c1 ;can2 control register 1

CAN2BTR0 equ REGBASE+$1c2 ;can2 bus timing register 0

CAN2BTR1 equ REGBASE+$1c3 ;can2 bus timing register 1

CAN2RFLG equ REGBASE+$1c4 ;can2 receiver flags

CAN2RIER equ REGBASE+$1c5 ;can2 receiver interrupt enables

CAN2TFLG equ REGBASE+$1c6 ;can2 transmit flags

CAN2TIER equ REGBASE+$1c7 ;can2 transmit interrupt enables

CAN2TARQ equ REGBASE+$1c8 ;can2 transmit message abort control

CAN2TAAK equ REGBASE+$1c9 ;can2 transmit message abort status

CAN2TBSEL equ REGBASE+$1ca ;can2 transmit buffer select

CAN2IDAC equ REGBASE+$1cb ;can2 identfier acceptance control

CAN2RXERR equ REGBASE+$1ce ;can2 receive error counter

CAN2TXERR equ REGBASE+$1cf ;can2 transmit error counter

CAN2IDAR0 equ REGBASE+$1d0 ;can2 identifier acceptance register 0

CAN2IDAR1 equ REGBASE+$1d1 ;can2 identifier acceptance register 1

CAN2IDAR2 equ REGBASE+$1d2 ;can2 identifier acceptance register 2

CAN2IDAR3 equ REGBASE+$1d3 ;can2 identifier acceptance register 3

CAN2IDMR0 equ REGBASE+$1d4 ;can2 identifier mask register 0

CAN2IDMR1 equ REGBASE+$1d5 ;can2 identifier mask register 1

CAN2IDMR2 equ REGBASE+$1d6 ;can2 identifier mask register 2

CAN2IDMR3 equ REGBASE+$1d7 ;can2 identifier mask register 3

CAN2IDAR4 equ REGBASE+$1d8 ;can2 identifier acceptance register 4

CAN2IDAR5 equ REGBASE+$1d9 ;can2 identifier acceptance register 5

CAN2IDAR6 equ REGBASE+$1da ;can2 identifier acceptance register 6

CAN2IDAR7 equ REGBASE+$1db ;can2 identifier acceptance register 7

CAN2IDMR4 equ REGBASE+$1dc ;can2 identifier mask register 4

CAN2IDMR5 equ REGBASE+$1dd ;can2 identifier mask register 5

CAN2IDMR6 equ REGBASE+$1de ;can2 identifier mask register 6

CAN2IDMR7 equ REGBASE+$1df ;can2 identifier mask register 7

CAN2RXFG equ REGBASE+$1e0 ;can2 rx foreground buffer thru +$1ef

CAN2TXFG equ REGBASE+$1f0 ;can2 tx foreground buffer thru +$1ff

CAN3CTL0 equ REGBASE+$200 ;can3 control register 0

CAN3CTL1 equ REGBASE+$201 ;can3 control register 1

CAN3BTR0 equ REGBASE+$202 ;can3 bus timing register 0

CAN3BTR1 equ REGBASE+$203 ;can3 bus timing register 1

CAN3RFLG equ REGBASE+$204 ;can3 receiver flags

CAN3RIER equ REGBASE+$205 ;can3 receiver interrupt enables

CAN3TFLG equ REGBASE+$206 ;can3 transmit flags

CAN3TIER equ REGBASE+$207 ;can3 transmit interrupt enables

CAN3TARQ equ REGBASE+$208 ;can3 transmit message abort control

CAN3TAAK equ REGBASE+$209 ;can3 transmit message abort status

CAN3TBSEL equ REGBASE+$20a ;can3 transmit buffer select

CAN3IDAC equ REGBASE+$20b ;can3 identfier acceptance control

CAN3RXERR equ REGBASE+$20e ;can3 receive error counter

CAN3TXERR equ REGBASE+$20f ;can3 transmit error counter

CAN3IDAR0 equ REGBASE+$210 ;can3 identifier acceptance register 0

CAN3IDAR1 equ REGBASE+$211 ;can3 identifier acceptance register 1

CAN3IDAR2 equ REGBASE+$212 ;can3 identifier acceptance register 2

CAN3IDAR3 equ REGBASE+$213 ;can3 identifier acceptance register 3

CAN3IDMR0 equ REGBASE+$214 ;can3 identifier mask register 0

CAN3IDMR1 equ REGBASE+$215 ;can3 identifier mask register 1

CAN3IDMR2 equ REGBASE+$216 ;can3 identifier mask register 2

CAN3IDMR3 equ REGBASE+$217 ;can3 identifier mask register 3

CAN3IDAR4 equ REGBASE+$218 ;can3 identifier acceptance register 4

CAN3IDAR5 equ REGBASE+$219 ;can3 identifier acceptance register 5

CAN3IDAR6 equ REGBASE+$21a ;can3 identifier acceptance register 6

CAN3IDAR7 equ REGBASE+$21b ;can3 identifier acceptance register 7

CAN3IDMR4 equ REGBASE+$21c ;can3 identifier mask register 4

CAN3IDMR5 equ REGBASE+$21d ;can3 identifier mask register 5

CAN3IDMR6 equ REGBASE+$21e ;can3 identifier mask register 6

CAN3IDMR7 equ REGBASE+$21f ;can3 identifier mask register 7

CAN3RXFG equ REGBASE+$220 ;can3 rx foreground buffer thru +$22f

CAN3TXFG equ REGBASE+$230 ;can3 tx foreground buffer thru +$23f

PTT equ REGBASE+$240 ;portt data register

PTIT equ REGBASE+$241 ;portt input register

DDRT equ REGBASE+$242 ;portt direction register

RDRT equ REGBASE+$243 ;portt reduced drive register

PERT equ REGBASE+$244 ;portt pull device enable

PPST equ REGBASE+$245 ;portt pull polarity select

PTS equ REGBASE+$248 ;ports data register

PTIS equ REGBASE+$249 ;ports input register

DDRS equ REGBASE+$24a ;ports direction register

RDRS equ REGBASE+$24b ;ports reduced drive register

PERS equ REGBASE+$24c ;ports pull device enable

PPSS equ REGBASE+$24d ;ports pull polarity select

WOMS equ REGBASE+$24e ;ports wired or mode register

PTM equ REGBASE+$250 ;portm data register

PTIM equ REGBASE+$251 ;portm input register

DDRM equ REGBASE+$252 ;portm direction register

RDRM equ REGBASE+$253 ;portm reduced drive register

PERM equ REGBASE+$254 ;portm pull device enable

PPSM equ REGBASE+$255 ;portm pull polarity select

WOMM equ REGBASE+$256 ;portm wired or mode register

MODRR equ REGBASE+$257 ;portm module routing register

PTP equ REGBASE+$258 ;portp data register

PTIP equ REGBASE+$259 ;portp input register

DDRP equ REGBASE+$25a ;portp direction register

RDRP equ REGBASE+$25b ;portp reduced drive register

PERP equ REGBASE+$25c ;portp pull device enable

PPSP equ REGBASE+$25d ;portp pull polarity select

PIEP equ REGBASE+$25e ;portp interrupt enable register

PIFP equ REGBASE+$25f ;portp interrupt flag register

PTH equ REGBASE+$260 ;porth data register

PTIH equ REGBASE+$261 ;porth input register

DDRH equ REGBASE+$262 ;porth direction register

RDRH equ REGBASE+$263 ;porth reduced drive register

PERH equ REGBASE+$264 ;porth pull device enable

PPSH equ REGBASE+$265 ;porth pull polarity select

PIEH equ REGBASE+$266 ;porth interrupt enable register

PIFH equ REGBASE+$267 ;porth interrupt flag register

PTJ equ REGBASE+$268 ;portj data register

PTIJ equ REGBASE+$269 ;portj input register

DDRJ equ REGBASE+$26a ;portj direction register

RDRJ equ REGBASE+$26b ;portj reduced drive register

PERJ equ REGBASE+$26c ;portj pull device enable

PPSJ equ REGBASE+$26d ;portj pull polarity select

PIEJ equ REGBASE+$26e ;portj interrupt enable register

PIFJ equ REGBASE+$26f ;portj interrupt flag register

CAN4CTL0 equ REGBASE+$280 ;can4 control register 0

CAN4CTL1 equ REGBASE+$281 ;can4 control register 1

CAN4BTR0 equ REGBASE+$282 ;can4 bus timing register 0

CAN4BTR1 equ REGBASE+$283 ;can4 bus timing register 1

CAN4RFLG equ REGBASE+$284 ;can4 receiver flags

CAN4RIER equ REGBASE+$285 ;can4 receiver interrupt enables

CAN4TFLG equ REGBASE+$286 ;can4 transmit flags

CAN4TIER equ REGBASE+$287 ;can4 transmit interrupt enables

CAN4TARQ equ REGBASE+$288 ;can4 transmit message abort control

CAN4TAAK equ REGBASE+$289 ;can4 transmit message abort status

CAN4TBSEL equ REGBASE+$28a ;can4 transmit buffer select

CAN4IDAC equ REGBASE+$28b ;can4 identfier acceptance control

CAN4RXERR equ REGBASE+$28e ;can4 receive error counter

CAN4TXERR equ REGBASE+$28f ;can4 transmit error counter

CAN4IDAR0 equ REGBASE+$290 ;can4 identifier acceptance register 0

CAN4IDAR1 equ REGBASE+$291 ;can4 identifier acceptance register 1

CAN4IDAR2 equ REGBASE+$292 ;can4 identifier acceptance register 2

CAN4IDAR3 equ REGBASE+$293 ;can4 identifier acceptance register 3

CAN4IDMR0 equ REGBASE+$294 ;can4 identifier mask register 0

CAN4IDMR1 equ REGBASE+$295 ;can4 identifier mask register 1

CAN4IDMR2 equ REGBASE+$296 ;can4 identifier mask register 2

CAN4IDMR3 equ REGBASE+$297 ;can4 identifier mask register 3

CAN4IDAR4 equ REGBASE+$298 ;can4 identifier acceptance register 4

CAN4IDAR5 equ REGBASE+$299 ;can4 identifier acceptance register 5

CAN4IDAR6 equ REGBASE+$29a ;can4 identifier acceptance register 6

CAN4IDAR7 equ REGBASE+$29b ;can4 identifier acceptance register 7

CAN4IDMR4 equ REGBASE+$29c ;can4 identifier mask register 4

CAN4IDMR5 equ REGBASE+$29d ;can4 identifier mask register 5

CAN4IDMR6 equ REGBASE+$29e ;can4 identifier mask register 6

CAN4IDMR7 equ REGBASE+$29f ;can4 identifier mask register 7

CAN4RXFG equ REGBASE+$2a0 ;can4 rx foreground buffer thru +$2af

CAN4TXFG equ REGBASE+$2b0 ;can4 tx foreground buffer thru +$2bf

\* end registers

**;vector\_table.inc**

; ----------------------------------------------------

; This vector table is intended for use with code

; downloaded into Flash using the resident bootloader

; in the Dragon12 board.

; ------------------ VECTOR TABLE --------------------

org $EF8C

fdb Handler\_0 ;$EF8C: PWM emergency shutdown

fdb Handler\_0 ;$EF8E: PortP

fdb Handler\_0 ;$EF90: CAN4TX

fdb Handler\_0 ;$EF92: CAN4RX

fdb Handler\_0 ;$EF94: CAN4ERR

fdb Handler\_0 ;$EF96: CAN4WU

fdb Handler\_0 ;$EF98: CAN3TX

fdb Handler\_0 ;$EF9A: CAN3RX

fdb Handler\_0 ;$EF9C: CAN3ERR

fdb Handler\_0 ;$EF9E: CAN3WU

fdb Handler\_0 ;$EFA0: CAN2TX

fdb Handler\_0 ;$EFA2: CAN2RX

fdb Handler\_0 ;$EFA4: CAN2ERR

fdb Handler\_0 ;$EFA6: CAN2WU

fdb Handler\_0 ;$EFA8: CAN1TX

fdb Handler\_0 ;$EFAA: CAN1RX

fdb Handler\_0 ;$EFAC: CAN1ERR

fdb Handler\_0 ;$EFAE: CAN1WU

fdb Handler\_0 ;$EFB0: CAN0TX

fdb Handler\_0 ;$EFB2: CAN0RX

fdb Handler\_0 ;$EFB4: CAN0ERR

fdb Handler\_0 ;$EFB6: CAN0WU

fdb Handler\_0 ;$EFB8: FLASH

fdb Handler\_0 ;$EFBA: EEPROM

fdb Handler\_0 ;$EFBC: SPI2

fdb Handler\_0 ;$EFBE: SPI1

fdb Handler\_0 ;$EFC0: IIC Bus

fdb Handler\_0 ;$EFC2: BDLC

fdb Handler\_0 ;$EFC4: CRG self-clock-mode

fdb Handler\_0 ;$EFC6: CRG PLL Lock

fdb Handler\_0 ;$EFC8: Pulse Accumulator B Overflow

fdb Handler\_0 ;$EFCA: Modulus Down Counter underflow

fdb Handler\_0 ;$EFCC: PORTH

fdb Handler\_0 ;$EFCE: PORTJ

fdb Handler\_0 ;$EFD0: ATD1

fdb Handler\_0 ;$EFD2: ATD0

fdb Handler\_0 ;$EFD4: SCI1 Serial System

fdb Handler\_0 ;$EFD6: SCI0 Serial System

fdb Handler\_0 ;$EFD8: SPI0

fdb Handler\_0 ;$EFDA: Pulse Accumulator Input Edge

fdb Handler\_0 ;$EFDC: Pulse Accumulator A Overflow

fdb Handler\_0 ;$EFDE: Enhanced Capture Timer Overflow

fdb Handler\_0 ;$EFE0: Enhanced Capture Timer Channel 7

fdb Handler\_0 ;$EFE2: Enhanced Capture Timer Channel 6

fdb Handler\_0 ;$EFE4: Enhanced Capture Timer Channel 5

fdb Handler\_0 ;$EFE6: Enhanced Capture Timer Channel 4

fdb Handler\_0 ;$EFE8: Enhanced Capture Timer Channel 3

fdb Handler\_0 ;$EFEA: Enhanced Capture Timer Channel 2

fdb Handler\_0 ;$EFEC: Enhanced Capture Timer Channel 1

fdb Handler\_0 ;$EFEE: Enhanced Capture Timer Channel 0

fdb RTI\_ISR ;$EFF0: Real Time Interrupt (RTI)

fdb Handler\_0 ;$EFF2: IRQ

fdb Handler\_0 ;$EFF4: XIRQ

fdb Handler\_0 ;$EFF6: Software Interrupt (SWI)

fdb Handler\_0 ;$EFF8: Illegal Instruction Trap

fdb start ;$EFFA: COP Failure Reset

fdb Handler\_0 ;$EFFC: Clock Monitor Fail Reset

fdb start ;$EFFE: /RESET

**Data Sheets:**

**C:\Users\Lee\Documents\School\ENEL387\Project\Docs\Final_Spec\32172E7Cd01.tifC:\Users\Lee\Documents\School\ENEL387\Project\Docs\Final_Spec\CT1015.tifC:\Users\Lee\Documents\School\ENEL387\Project\Docs\Final_Spec\zener_1N4731A.tif**