Tufts University
Department of Electrical Engineering
and Computer Science

NERD GIRLS
Maximum Power Point Tracker

Stephanie Chin
Jeanell Gadson
Katie Nordstrom

Project Advisor: Karen Panetta
Project Consultants: Matthew Heller, Richard Colombo, Michael Quaglia

Senior Design Project 2003
Final Report
May 12, 2003
TABLE OF CONTENTS

1. Purpose

2. Introduction
   2.1. Photovoltaic Cells and Array Research
   2.2. Power Supply Research
   2.3. MPPT Research

3. Basic Design
   3.1. Why are we building a MPPT?
   3.2. How does it work?

4. Implementation
   4.1. Overall Design Considerations
   4.2. Hardware
       4.2.1. Components
       4.2.2. Voltage Control
       4.2.3. Charging Unit
       4.2.4. Solar Array Protection
   4.3. Software
       4.3.1. Menu Structure
       4.3.2. Algorithm

5. Assessment
   5.1. Hardware
   5.2. Software

6. Conclusion
   6.1. Future Work

7. Appendix
   7.1. Hardware Schematics
       7.1.1. MPPT Block Diagram
       7.1.2. Circuitry Schematic
   7.2. Software Flowcharts
       7.2.1. Menu Structure
       7.2.2. Algorithm
       7.2.3. Algorithm Submenu Function
   7.3. Code
       7.3.1. solargirls.asm
       7.3.2. Lcd.asm
       7.3.3. Math.asm
7.3.4. p2plsp18.lkr
7.4. Datasheets
  7.4.1. PIC Microcontroller
  7.4.2. DC/DC Converter PT4122A
  7.4.3. DC/DC Converter TPS6734IP
  7.4.4. PWM TL598CN
  7.4.5. Diode 16CTU04S
  7.4.6. LTC DAC 1451CN8
  7.4.7. MOSFET IXFX90N20Q
  7.4.8. MOSFET Driver MAX4420CPA

8. References

9. Acknowledgements
1. PURPOSE

The objective of the project was to design a Maximum Power Point Tracker (MPPT) for a solar-powered vehicle. This component optimized the amount of power obtained from the photovoltaic array and charged the power supply. The solar car will be constructed by the 2003/2004 Nerd Girls Team and will incorporate the Maximum Power Point Tracker unit into the final design.

2. INTRODUCTION

Developed by Professor Karen Panetta, the Tufts University Nerd Girls Project brings together a team of multidisciplinary undergraduate female engineers. Their mission is to build and race a solar-powered vehicle in Fall 2003 and to use it as an outreach tool to introduce engineering to young students.

2.1 PHOTOVOLTAIC CELLS AND ARRAY RESEARCH

Photovoltaic cells are devices that absorb sunlight and convert that solar energy into electrical energy.

Solar cells are commonly made of silicon, one of the most abundant elements on Earth. Pure silicon, an actual poor conductor of electricity, has four outer valence electrons that form tetrahedral crystal lattices.

The electron clouds of the crystalline sheets are stressed by adding trace amounts of elements that have three or five outer shell electrons that will enable electrons to move. The nuclei of these elements fit well in the crystal lattice, but with only three outer shell electrons, there are too few electrons to balance out, and "positive holes" float in the electron cloud. With five outer shell electrons, there are too many electrons. The process of adding these impurities on purpose is called "doping." When doped with an element with five electrons, the resulting silicon is called N-type ("n" for negative) because of the prevalence of free electrons. Likewise, when doped with an element of three electrons, the silicon is called P-type. The absence of electrons (the "holes") define P-type.

The combination of N-type and P-type silicon cause an electrostatic field to form at the junction. At the junction, electrons from the sides mix and form a barrier, making it hard for electrons on the N side to cross to the P side. Eventually equilibrium is reached, and an electric field separates the sides.

When photons (sunlight) hit a solar cell, its energy frees electron-holes pairs. The electric field will send the free electron to the N side and hole to the P side. This causes further disruption of electrical neutrality, and if an external current path is provided,
electrons will flow through the path to their original side (the P side) to unite with holes that the electric field sent there, doing work for us along the way. The electron flow provides the current, and the cell's electric field causes a voltage. With both current and voltage, we have power, which is the product of the two.

Three solar cell types are currently available: monocrystalline, polycrystalline, and thin film, discerned by material, efficiency, and composition.

By wiring solar cells in series, the voltage can be increased; or in parallel, the current. Solar cells are wired together to form a solar panel. Solar panels can be joined to create a solar array.

2.2 POWER SUPPLY RESEARCH

A battery is a source portable electric power. A storage battery is a reservoir, which may be used repeatedly for storing energy. Energy is charged and drained from the reservoir in the form of electricity, but it is stored as chemical energy. The most common storage battery is the lead-acid battery that is widely used in automobiles. They represent about 60% of all batteries sold worldwide and are usually more economical and have a high tolerance for abuse. Lead-acid batteries are inexpensive, relatively safe and easily recyclable, but have a low energy-to-weight ratio, which is a serious limitation when trying to build lightweight vehicles.

New battery technologies are constantly being explored that can offer better energy-to-weight ratios, lower costs and increased battery life. The nickel-metal-hydride battery has received a great deal of attention as a near future solution. Nickel-metal-hydride batteries offer about twice the energy capacity for the same weight as a current lead-acid battery. Another battery type with an even greater energy density is Lithium ion.

2.3 MPPT RESEARCH

The Maximum Power Point Tracker (MPPT) is needed to optimize the amount of power obtained from the photovoltaic array to the power supply.

The output of a solar module is characterized by a performance curve of voltage versus current, called the I-V curve. See Figure 1. The maximum power point of a solar module is the point along the I-V curve that corresponds to the maximum output power possible for the module. This value can be determined by finding the maximum area under the current versus voltage curve.

![Figure 1: I-V Curve](image-url)
3. **Basic Design**

3.1 **Why are we building a MPPT?**

There are commercially available MPPTs which are typically used for home solutions and buildings. These are not designed to withstand the harsh, fast-changing environmental conditions of solar car racing. Design of the customized MPPT will ensure that the system operates as closely to the Maximum Power Point (MPP) while being subjected to the varying lighting and temperature.

3.2 **How does it work?**

The inputs of the MPPT consisted of the photovoltaic voltage and current outputs. The adjusted voltage and current output of the MPPT charges the power supply. See Figure 2.

A microcontroller was utilized to regulate the integrated circuits (ICs) and calculate the maximum power point, given the output from the solar array. Hardware and software integration was necessary for the completion of this component.
4. IMPLEMENTATION

4.1 OVERALL DESIGN CONSIDERATIONS

Many factors influenced the component selection and the design of the MPPT.

- In terms of optimal functionality, the theory of power conservation needed to be applied. The input and output voltage and current were calculated such that the power into and out of the MPPT was equal.

- To protect the photovoltaic array from damage, protection diodes were employed.

- Two 48V lead acid battery banks were utilized. Only one battery bank will be charged at a time. (The other will be employed to run other components of the car).

- In order to trickle charge the batteries, a voltage exceeding 48V must be fed to the bank. In this design, 50V was chosen to charge the power supply.

- To prevent damage and overcharging of the power supply, a FET was employed.

4.2 HARDWARE

The MPPT circuitry consisted of three sections – Voltage Control, Charging Unit, and Solar Array Protection. See Appendix 7.1.1. The Voltage Control block consisted of two DC to DC converters that stepped down the solar array voltage. The converters supplied the necessary voltage to run the various components of the system. Secondly, the Charging Unit consisted of the PIC microcontroller, PWM, MOSFET, and protection diodes. It computed the maximum power point and regulated the various integrated circuits that charged the 48V power supply. Lastly, the Solar Array Protection block consisted of the protection diodes used to prevent solar panel damage.

4.2.1 COMPONENTS

Table 1 shows the components used for each of the three sections of the hardware design. See Appendix 7.4 for datasheets.
### 4.2.2 Voltage Control

The DC/DC Buck Converter stepped down the solar array output voltage (approximately 48V) to 5v in order to power the PIC, DACs, and RS-232. The DC/DC Boost Converter stepped up the 5v output from the Buck Converter to 12v in order to power the PWM.

### 4.2.3 Charging Unit

The charging unit consisted of multiple components, which worked together to power the battery array. This unit contained the ADCs, DACs, PIC microcontroller, PWM, MOSFET, MOSFET driver, inductor, and protection diodes.

The ADC changed the analog output of the solar array into a digital signal to be manipulated by the PIC microcontroller. The DAC worked in the opposite direction of the ADC. It changed the digital output from the PIC to an analog signal, which regulated the PWM.

The PIC microcontroller performed all of the calculations necessary to obtain the maximum power point. The PIC received the input voltage directly from the solar array and converted the value to a digital signal via the ADCs. In order to determine the input current, the output voltage of the voltage divider was sent to the PIC as a digital signal via the ADCs. From there, knowing the resistance of the voltage divider, the calculations were performed within the PIC. Having both the input voltage (V) and current (I) from the solar array, the power could be determined (P=V*I). Keeping the theory of power conservation in mind, the output power from the PIC needed to equal the input power from the solar array. At the same time, the charging voltage must exceed the battery array voltage, 48V; therefore 50V was assumed for the output voltage. The output current was calculated using the input power and the output voltage. This value was then converted to an analog signal via the DACs and sent to the PWM.

### Table 1: Components

<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>PART NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC Microcontroller</td>
<td>PICF458</td>
</tr>
<tr>
<td>DC to DC Converter (5V)</td>
<td>PT4122A</td>
</tr>
<tr>
<td>DC to DC Converter (12V)</td>
<td>TPS6734IP</td>
</tr>
<tr>
<td>Pulse Width Modulator (PWM)</td>
<td>TL598CN</td>
</tr>
<tr>
<td>Diode</td>
<td>16CTU04S</td>
</tr>
<tr>
<td>Digital to Analog Converter (DAC)</td>
<td>LTC1451CN8</td>
</tr>
<tr>
<td>MOSFET</td>
<td>IXFX90N20Q</td>
</tr>
<tr>
<td>MOSFET driver</td>
<td>MAX4420CPA</td>
</tr>
</tbody>
</table>
The PWM received the adjusted voltage and current from the PIC, and changed its duty cycle accordingly. This duty cycle controlled the MOSFET.

The MOSFET acted like a switch. When it was on, it closed the circuit and sent the power to ground, preventing the overcharging of the battery array. At this time, current built up in the inductor and it was able to charge. When it was off, the circuit opened, and the power was sent through the protection diodes to the battery array. At this time, the inductor discharged.

The protection diodes prevented current from flowing back from the batteries and potentially damaging the solar array. By placing the diodes in parallel, the overall resistance decreased, and allowed a greater amount of current to pass through.

4.2.4 Solar Array Protection Block

The voltage divider took the voltage from the solar array and stepped it down to a maximum voltage of 4.08V. This prevented the ADC from “blowing out.” Without the voltage divider, the solar array would send too large of a voltage for the ADC to handle. Protection diodes were utilized to prevent the current from flowing back to the solar array and causing damage to it.

Figure 4: MPPT Circuit Board
4.3 SOFTWARE

The PIC Microcontroller chosen had sufficient memory to meet the demands of the design. The ADCs were also included in the PIC, which reduced the amount of additional external parts.

Programming was completed in MPASM Assembly. See Appendix 7.2 and 7.3 for Software flowcharts and code.

4.3.1 MENU STRUCTURE

The PIC contains a LCD screen, which enabled us to display the input and output voltages and currents. This enabled us to confirm the results of the calculations performed by the PIC. The structure of the LCD output was laid out as a menu. There were four main menu items, Voltage input from the solar array, current input from the solar array, voltage output from the MPPT and current output from the MPPT. See Figure 5.

Initially, the welcoming note was displayed on the LCD followed by the voltage input from the solar array menu item. A register called which_menu was used to organize the information about which menu item the user was viewing. Bit 0 of the which_menu register indicated whether or not the user was within the first menu item. If the bit value was 1, this meant the user was looking at the input voltage from the solar array. A 0 bit value meant the user was not within this menu item. The same system was set up for the rest of the menu items. Bit 1 was allocated to the input current from the solar array menu item. Bit 2 was allocated to the output voltage from the MPPT menu item. Finally, bit 3 was allocated to the output current from the MPPT menu item.

By pressing RA4, the user could scroll through the main menu items. By pushing RB0, the user could view the submenu of each main menu item. For example, if the user wanted to see the changing input voltage values, the user would scroll through the menu (using the RA4 button) until the Vin Solar menu item was displayed. Then, the user would select this (pushing RB0) and the voltage would be displayed on the LCD. The user could return to the main menu by pushing RB0 again. The which_menu register bit values were used to determine the return location on the main menu.

The final design was set up to perform the calculations to determine the output power each time the user selected the output current from the MPPT menu item. In order to test the functionality of the calculation code, values were hard-coded for the input voltage, input current and output voltage. For example, if the voltage input was 5V and the current input was 10mA, the two values were multiplied together to determine the power. If we wanted a 2V output, this value would be hard-coded as the output voltage. The input power would be divided by the 2V and the result would be the output current. So, in this example, the output current would be displayed as 25mA. This way the power output from the MPPT remained the same as the power input from the solar
panels, but the voltage and current were adjusted so that enough voltage would be sent to a power supply to charge it. See Appendix 7.2.1.

Figure 5: PIC Microcontroller LCD Menu Display
The topmost figure shows the welcome screen. The left screens are the scrollable main menus that display a submenu containing input/output data if RBO is selected. Sample inputs were used to test the calculation algorithm, as shown.
4.3.2 ALGORITHM

When the program started running, the first steps taken were to configure the PIC ports being used for inputs and outputs and to set the A/D conversion information. See Appendix 7.2.2. From there, the output voltage was given a set value. This value should be 50V, as this was the amount of voltage needed to charge the 48V battery array.

The welcome note was then displayed to inform the user that the program was running. Following this, the first item on the main menu was displayed (Vin Solar). At this point the user had the option to either select the item using the RB0 button (and the value would be displayed on the LCD) or to scroll through the four menu items using the RA4 button.

When the user selected one of the menu items by pressing RB0, the program first cleared the which_menu bit that was previously 1 (indicating the last menu item that was viewed). See Appendix 7.2.3. The label was then displayed on the LCD screen and the which_menu bit allocated to the current menu item was set to 1.

The program then took the data and either converted the value to a digital signal (if the data was received from port A) and stored the value in a register, or just stored the hard-coded value in a register. This was the only information needed to display the values for the first three menu items.

If the user selected the current output of the MPPT menu item, the output current was calculated using the input voltage, input current and output voltage values stored in the registers. The result was then printed to the LCD screen.

In order to return to the correct menu item, the program checked the bit values of the which_menu. For example, if bit 0 of which_menu was equal to the value of 1, the program would return to the first menu item, Vin Solar.
5. **ASSESSMENT**

5.1 **HARDWARE**

DIP packaging was used because they are easier to wire wrap. Wire wrapping for a majority of the circuitry was chosen instead of soldering because it will facilitate future changes.

Chip sockets were used instead of wire wrapping directly to the chip; thus if the chip goes bad, it can be replaced and does not have to be rewired.

The voltage divider circuitry was determined by assuming that the maximum output voltage of the solar array is 75V, and the maximum input of the ADC is 5 volts. See Figure 6. The following resistor values were used in order to obtain a maximum output of 4.08V: \( R_1 = 620\,\Omega \), \( R_2 = 68\,\Omega \), \( R_L = 75\,\Omega \)

![Figure 6: Voltage Divider Circuitry](image)

Extra diodes were not needed for the Solar Protection Array. Diode protection to \( V_{DD} \) and \( V_{SS} \) were included in the ADCs on the PIC microcontroller.

The capacitors used do not support high voltages for an extended period of time, therefore they will have a short lifespan.

The packaging for the MOSFET and diodes made it difficult to attach to the circuit board.

The circuitry was placed on multiple boards. This made it easier to visualize the layout, but greatly increased the overall size of the complete device. If the final device was packaged, the wiring and chips would be protected from damage. Also, the input and output wires would be easily accessible.
5.2 SOFTWARE

The calculation section of the program worked with only a few flaws. We were able to calculate the input power and then determine the output current knowing the output voltage desired and the input power. However, the code produced incorrect results once the test values were increased to numbers large enough to produce results greater than 256. The multiplication function was set up to multiply an 8-bit number by another 8-bit number and the result would be 16 bits total, stored in two 8-bit registers. When the two numbers being multiplied produced a result greater than 256, the value stored in the high bit register was incorrect. At the same time, we came across problems when the result of the division function included a fraction. The code was set up to print three decimal values to the LCD (up to 256). Several different steps were taken in an attempt to print out correct results with fractions; however, the goal was never achieved.

The design was set up so that the PIC would receive an input voltage and current from the solar array. However, there were difficulties when it came to reading the input
values. Knowing port A was the port used for A/D conversions, it was set up so that there could be two inputs for voltage and current. There were two registers used to configure the A/D conversion information, ADCON0 and ADCON1. ADCON0 bit 0 was set to enable the A/D conversion and bits 3-5 were used to determine the channel from which the PIC was reading the input to convert. Eventually, it should be set up so that bits 5-3 are switched between 000 and 001, taking turns reading the input from channel 0 and channel 1. In order to test this, however, the bits were hard-coded to 000. ADCON1 bits 3-0 were set for two inputs (1101). With two inputs, there needed to be voltage references to ground and +5V. Ideally, with this test, an input between 0 and 5 volts would be used as the voltage input from the solar array (smaller test values at first). However, the program constantly shutdown when this design was attempted.

In order to show how the A/D conversion would work, though, the potentiometer values were used as the voltage input. The potentiometer was defaulted with a link to channel 0 of port A and it seemed that this was the only way to test the A/D conversions. It was set to convert numbers 0 through 15. So, in the final design, the user could rotate the knob of the potentiometer to test different values (from 0 to 15) that acted as the input voltage.

Overall, the program was able to meet the requirements of the design, but only to a certain degree. The final integration of the hardware and software was unable to work due to the troubles encountered when attempting to input or output a voltage to or from the PIC. The A/D conversion and the calculations could be tested with the final program however. The finished program consisted of a hard-coded value of 4mA for the input current and 2V for the output voltage. The user could test the program by rotating the potentiometer value (acting as the input voltage) and the result could be viewed under the Iout MPPT menu item. For example, the user could turn the potentiometer so that the value of the input voltage was 5V. The program would calculate the power using this and the 4mA hard-coded. The output current would then be determined using this power value and the output voltage of 2V. The result in this case would be 10mA.
6. CONCLUSION

In order to charge a power source at its maximum efficiency, a Maximum Power Point Tracker (MPPT) device is utilized. The MPPT design incorporated three systems - the Voltage Divider, Charging Unit, and Solar Array Protection.

Although the final MPPT did not completely function as planned, the software algorithm did complete the correct calculation to find the Maximum Power Point. As the project came to an end, various changes could have been made which could benefit the design and implementation process. A smaller output range of the solar array would have helped to design a more efficient MPPT. Allowance of ample time is necessary. Many problems with the component purchasing and software were encountered.

There were a few weaknesses in the code. First, the PIC was not programmed to continuously loop. A program that automatically checks and updates the maximum power point could improve the design. Secondly, the program did not successfully communicate with the hardware. Working communication is absolutely crucial in the final device that will be incorporated into the solar-powered vehicle.

Use of space in the car is also an important factor, as it can be critical to the overall design. A more organized circuitry layout on only one board would enable the device to be simply set into the car.

6.1 FUTURE WORK

Fast-switching components are necessary to operate the device intended for solar car racing. The component choice is key in the design of the MPPT. High power efficiency is attained by carefully researching and selected the right components.
7. APPENDIX

CONTENTS

7.1 Hardware Schematics
   7.1.1 MPPT Block Diagram
   7.1.2 Circuitry Schematic

7.2 Software Flowcharts
   7.2.1 Menu Structure
   7.2.2 Algorithm
   7.2.3 Algorithm Submenu Function

7.3 Code
   7.3.1 solargirls.asm
   7.3.2 Lcd.asm
   7.3.3 Math.asm
   7.3.4 p2plsp18.lkr

7.4 Datasheets
   7.4.1 PIC Microcontroller
   7.4.2 DC/DC Converter PT4122A
   7.4.3 DC/DC Converter TPS6734IP
   7.4.4 PWM TL598CN
   7.4.5 Diode 16CTU04S
   7.4.6 LTC DAC 1451CN8
   7.4.7 MOSFET IXFX90N20Q
   7.4.8 MOSFET Driver MAX4420CPA
APPENDIX 7.1 HARDWARE SCHEMATICS
APPENDIX 7.1.2 CIRCUITRY SCHEMATIC
APPENDIX 7.2 SOFTWARE FLOWCHARTS
APPENDIX 7.2.1 MENU STRUCTURE

![Diagram of LCD Menu Structure]

- Start
  - Display Welcome Menu
    - Display VIN (Solar) Menu
      - Display VIN Solar data in submenu
    - Display IIN (Solar) Menu
      - Display IIN (Solar) data in submenu
    - Display VOUT (MPPT) Menu
      - Display VOUT (MPPT) data in submenu
    - Display IOUT (MPPT) Menu
      - Display IOUT (MPPT) data in submenu
APPENDIX 7.2.2 ALGORITHM

Start

Configure ports

Configure A/D Converter

Set MPPT Vout= 50V

Display Welcome Menu on PIC Microcontroller LCD

Display Main Menus (Vin Solar, Iin Solar, Vout MPPT, Iout MPPT)

RA4 selected?

Yes

RB0 selected?

Yes

Complete submenu function
APPENDIX 7.2.3 ALGORITHM SUBMENU FUNCTION

Start

Clear designated bits of which-menu register

Display label (volts=, curr=)

Set which_menu register

Get data from ports

Do A/D conversion

Store value in register

Preform calculation (as needed)

Print value to LCD screen

Test bits of which_menu

Return to main menu

Nerd Girls Solar/Mppt Group
Submenu Flowchart
APPENDIX 7.3  CODE
APPENDIX 7.3.1  SOLAR GIRLS.ASM

;************************************************************************
;* Microchip Technology Inc. 2002
;* Assembler version: 2.0000
;* Filename:
;*  solargirls.asm (main routine)
;* Dependents:
;*  p18lcd.asm
;*  p18math.asm
;*  16f877.lkr
;************************************************************************

;MAXIMUM POWER POINT TRACKER PIC CODE
;STEPHANIE, KATIE, JEANELL
;************************************************************************

list p=18f452
#include p18f452.inc

;Program Configuration Registers
__CONFIG    _CONFIG2L, _BOR_OFF_2L & _PWRT_ON_2L
__CONFIG    _CONFIG4L, _STVR_OFF_4L & _LVP_OFF_4L & _DEBUG_OFF_4L
__CONFIG    _CONFIG5L, _CP0_OFF_5L & _CP1_OFF_5L & _CP2_OFF_5L & _CP3_OFF_5L
__CONFIG    _CONFIG6L, _WRT0_OFF_6L & _WRT1_OFF_6L & _WRT2_OFF_6L & _WRT3_OFF_6L
__CONFIG    _CONFIG7L, _EBTR0_OFF_7L & _EBTR1_OFF_7L & _EBTR2_OFF_7L & _EBTR3_OFF_7L

#define scroll_dir TRISA,4
#define scroll  PORTA,4  ;Push-button RA4 on PCB
#define select_dir TRISB,0
#define select  PORTB,0  ;Push-button RB0 on PCB

EXTERN LCDInit, temp_wr, d_write , i_write, LCDLine_1, LCDLine_2
EXTERN UMUL0808L, UDIV1608L, AARGB0, AARGB 1, BARGB0, BARGB1, AARGB5, REMB0, REMB1, TEMP

ssprw macro    ;check for idle SSP module routine
movlw 0x00
andwf SSPCON2,W
sublw 0x00
btfss STATUS,Z
bra $-8
btfsc SSPSTAT,R_W
bra $-2
endm

variables UDATA
which_menu RES 1
ptr_pos    RES 1
ptr_count  RES 1
temp_1     RES 1
temp_2     RES 1
temp_3     RES 1
cmd_byte   RES 1
temperature RES 1
LSD        RES 1
MsD        RES 1
MSD        RES 1
seconds    RES 1
minutes RES 1
hours RES 1
NumH RES 1
NumL RES 1
TenK RES 1
Thou RES 1
Hund RES 1
Tens RES 1
Ones RES 1
volt_in RES 1
curr_in RES 1
batt_volt RES 1
batt_curr RES 1

STARTUP CODE
NOP
goto start
NOP
NOP
NOP

PROG1 CODE

stan_table ;table for standard code
    "XXXXXXXXXXXXXXXX"
    ptr:
data " Vin (Solar) " :0
data " Iin (Solar) " :16
data " Vout (MPPT) " :32
data " Iout (MPPT) " :48
data "RA4=Next RB0=Sel" :64
data " Nerd Girls " :80
data " MPPT Rocks " :96
data "RA4=Set RB0=Menu" :112
data "RA4= --> RB0= ++" :128
data " RB0 = Exit " :144
data "Volts = " :160
data "Current = " :176
data " " :192

start
    call LCDInit

    movlw B'10100100' ;initialize USART
    movwf TXSTA ;8-bit, Async, High Speed
    movlw .25
    movwf SPBRG ;9.6kbaud @ 4MHz
    movlw B'10010000'
    movwf RCSTA
    bcf TRISC,2 ;configure CCP1 module for buzzer
    bcf TRISC,6
    movlw 0x80
    movwf PR2 ;initialize PWM period
    movlw 0x80 ;initialize PWM duty cycle
    movwf CCPRL
    bcf CCP1CON,CCP1X
    bcf CCP1CON,CCP1Y
movlw 0x05 ;postscale 1:1, prescaler 4, Timer2 ON
movwf T2CON
bsf TRISA,4 ;make switch RA4 an Input

;ADDITIONS FOR A/D CONVERTING
clr PORTB ;Clear PORTB
clr TRISB ;PORTB all outputs, display 4 MSB's
;of A/D result on LEDs
bsf TRISB,0 ;make switch RB0 an Input
movlw B'01000001' ;Fosc/8, A/D enabled
movwf ADCON0
movlw B'00001110' ;B'00001110';Left justify,1 analog channel
movwf ADCON1 ;VDD and VSS references

; test register value print by putting a value into curr_in

; movlw B'00000011' ; put value in register W (35)
; movwf volt_in ; put value of reg. W into volt_in
movlw B'00000100' ;B'01100100'
movwf curr_in ; put value of reg. W into curr_in
movlw B'00000010' ; put value 50 (50v output to batt) in reg. W
movwf batt_volt ;put value of reg. W (50) into batt_volt reg.

;*************** STANDARD CODE MENU SELECTION **************
;Introduction
movlw .80 ;send "Nerd Girls" to LCD
movwf ptr_pos
call stan_char_1
movlw .96 ;send "MPPT Rocks" to LCD
movwf ptr_pos
call stan_char_2
call delay_1s ;delay for display
call delay_1s

call delay_1s

call delay_1s

call delay_1s

menu
;------------------ VOLTAGE IN (SOLAR)---------------------
bcf which_menu, 3 ;bit 3 of register which_menu is cleared to 0
btfss scroll ;wait for RA4 release
goto $-2
btfss select ;wait for RB0 release
goto $-2
movlw 0x00 ;Displays "Solar Vout" (.0) to LCD
movwf ptr_pos
call stan_char_1
movlw .64 ;RA4=Next RB0=Sel
movwf ptr_pos
call stan_char_2
v_wait
bsf which_menu, 0
btfss select ;voltmeter measurement ??
bra voltmeter
btfsb scroll ;next mode ??
bra v_wait ;NO
btfsb scroll ;YES
bra $-2 ;wait for RA4 release

;------------- SOLAR CURRENT OUTPUT -------------------------------

menu_buz
  bcf which_menu, 0 ;bit 0 of register which_menu is cleared to 0
  btfsb scroll ;wait for RA4 release
  goto $-2
  btfsb select ;wait for RB0 release
  bra $-2

  movlw .16 ;Displays "Solar Iout" to LCD
  movwf ptr_pos
call stand_char_1

  movlw .64 ;RA4=Next RB0=Sel
  movwf ptr_pos
call stand_char_2

b_wait
  bsf which_menu, 1 ;bit 1 of register which_menu is cleared to 0
  btfsb select ;current measurement??
  bra voltmeter
  btfsb scroll ;next mode??
  bra b_wait ;NO
  btfsb scroll ;YES
  bra $-2 ;wait for RA4 release

;---------------- MPPT VOLTAGE OUTPUT ----------------------

menu_temp
  bcf which_menu, 1 ;bit 1 of register which_menu is cleared to 0
  btfsb scroll ;wait for RA4 release
  bra $-2
  btfsb select ;wait for RB0 release
  bra $-2

  movlw .32 ;Display "MPPT Vout" to LCD
  movwf ptr_pos
call stand_char_1

  movlw .64 ;RA4=Next RB0=Sel
  movwf ptr_pos
call stand_char_2

t_wait
  bsf which_menu, 2 ;bit 2 of register which_menu is cleared to 0
  btfsb select ;current measurement??
  bra voltmeter
  btfsb scroll ;next mode??
  bra t_wait ;NO
  btfsb scroll ;YES
  bra $-2 ;wait for RA4 release

;------------- -- MPPT CURRENT OUTPUT --------------------------

menu_clock
  bcf which_menu, 2 ;bit 2 of register which_menu is cleared to 0
  btfsb scroll ;wait for RA4 release
  bra $-2
  btfsb select ;wait for RB0 release
bra $-2
movlw .48 ;Display "MPPT Iout" to LCD
movwf ptr_pos
call stan_char_1
movlw .64 ;RA4=Next RB0=Sel
movwf ptr_pos
call stan_char_2

;---------------------don't need clock stuff---------------------
c_wait
bsf which_menu, 3
btfss select ;current measurement??
bra voltmeter
btfsc scroll ;next mode??
bra c_wait ;NO
btfss scroll ;YES
bra $-2 ;wait for RA4 release
btfss select ;goto time ??
bra $-2
btfsc scroll ;NO, next mode ??
bra c_wait ;NO
btfss scroll ;YES
bra $-2 ;wait for release

;------------------- ---end of clock stuff------------------------
bra menu ;begining of menu
return

;************** STANDARD USER CODE ******************************
;----------- Voltmeter------------------------------------------
voltmeter
btfss select ;wait for RB0 release
bra $-2
btfsc scroll ;NO, next mode ??
bra c_wait ;NO
btfss scroll ;YES
bra $-2 ;wait for release

;ADDITIONS FOR A/D CONVERTING
;write in 001 for bits 5-3 of adcon0
bsf ADCON0,GO ;Start A/D conversion (changes bit 2 of ADCON0 to 1)
Wait
btfss PIR1,ADIF ;Wait for conversion to complete
goto Wait
swapf ADRESH,W ;Swap A/D result nibbles
andlw 0x0f ;Mask off lower 4 bits
movwf volt_in ;Write A/D result to PORTB

;********************************************

;perform calculations
btfsc which_menu, 0 ;if selected solar voltage output (bit 0 of reg. which_menu would then be 1)
goto temp_inputvoltprint
btfsc which_menu, 2 ;if selected mppt voltage output (bit 2 of reg. which_menu would then be 1)
goto temp_outputvoltprint
btfsc which_menu, 1 ;if selected solar current output (bit 1 of reg. which_menu would then be 1)
goto temp_inputcurrprint
btfsc which_menu, 3 ;if selected mppt current output (bit 3 of reg. which_menu would then be 1)
goto temp_outputcurrprint ;send "Current = " to the LCD
movwf ptr_pos
call stan_char_1

temp_inputvoltprint

call LCDLine_1
;movlw 0x20  ;space
;movwf temp_wr
;call d_write
;movlw 0x20  ;space
;movwf temp_wr
;call d_write
movlw A"V"    ;print "V"
movwf temp_wr
call d_write
movlw A"O"    ;print "O"
movwf temp_wr
call d_write
movlw A"L"
movwf temp_wr
call d_write
movlw A"T"
movwf temp_wr
call d_write
movlw A"S"
movwf temp_wr
call d_write
movlw 0x20
;space
call d_write
movlw A="
;print ";"
call d_write
movlw 0x20
;space
call d_write
movf  volt_in, W  ;print Digital Input test value
call bin_bcd
;get temp ready for LCD
movf MSD,W
;send high digit
call d_write
movf MsD,W
;send middle digit
call d_write
movf LSD,W
;send low digit
call d_write
movf A"V"
call d_write
movlw 0x20
;space
call d_write
;end of sending unit to LCD
movwf  temp_wr
call  d_write
goto  volts_again

;-------------------------------------------
temp_inputcurrprint
    call  LCDLine_1
;movlw 0x20  ;space
;movwf  temp_wr
;call  d_write
;movlw 0x20  ;space
;movwf  temp_wr
;call  d_write
movlw A’C’    ;print "C"
omw   temp_wr
    call  d_write
movlw A’U’    ;print "U"
omw   temp_wr
    call  d_write
movlw A’R’
omw   temp_wr
    call  d_write
movlw A’R’
omw   temp_wr
    call  d_write
movlw A’R’
omw   temp_wr
    call  d_write
movlw A’E’
omw   temp_wr
    call  d_write
movlw 0x20  ;space
movwf  temp_wr
    call  d_write
movlw A’I’
omw   temp_wr
    call  d_write
movlw A’N’
omw   temp_wr
    call  d_write
movlw 0x20  ;space
movwf  temp_wr
    call  d_write
movlw A’=’    ;print "="
omw   temp_wr
    call  d_write
movlw 0x20  ;space
movwf  temp_wr
    call  d_write
movf  curr_in, W      ;print Digital Input test value
    call  bin_bcd   ;get temp ready for LCD
movf  MSD,W   ;send high digit
movwf  temp_wr
    call  d_write
movf  MsD,W   ;send middle digit
movwf  temp_wr
    call  d_write
movf  LSD,W   ;send low digit
movwf  temp_wr
    call  d_write
movlw A’m’
omw   temp_wr
call d_write
movlw A'\n'
movwf temp_wr
call d_write
movlw 0x20 ;space
movwf temp_wr
call d_write
;end of sending unit to LCD
movwf temp_wr
call d_write
goto volts_again

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

 ;----------------------------------------------------------------------
temp_outputvoltprint
call LCDLine_1
 ;movlw 0x20 ;space
 ;movwf temp_wr
 ;call d_write
 ;movlw 0x20 ;space
 ;movwf temp_wr
 ;call d_write
movlw A"V" ;print "C"
movwf temp_wr
call d_write
movlw A"O" ;print "U"
movwf temp_wr
call d_write
movlw A"L"
movwf temp_wr
call d_write
movlw A"T"
movwf temp_wr
call d_write
movlw A"S"
movwf temp_wr
call d_write
movlw 0x20 ;space
movwf temp_wr
call d_write
movlw A"O"
movwf temp_wr
call d_write
movlw A"U"
movwf temp_wr
call d_write
movlw A"T"
movwf temp_wr
call d_write
movlw 0x20 ;space
movwf temp_wr
call d_write
movlw A"=" ;print ":=
movwf temp_wr
call d_write
movlw 0x20 ;space
movwf temp_wr
call d_write
movlw batt_volt, W ;print Digital Input test value
call bin_bcd ; get temp ready for LCD
movf MSD,W ; send high digit
movwf temp_wr
call d_write
movf MsD,W ; send middle digit
movwf temp_wr
call d_write
movf LSD,W ; send low digit
movwf temp_wr
call d_write
movlw A"V"
movwf temp_wr
call d_write
movlw 0x20 ; space
movwf temp_wr
call d_write

; end of sending unit to LCD
movwf temp_wr
call d_write
goto volts_again

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

;------------------------------------------------------------------------------
temp_outputcurrprint
call LCDLine_1
; movlw 0x20 ; space
; movwf temp_wr
; call d_write
; movlw 0x20 ; space
; movwf temp_wr
; call d_write
; movlw A"C" ; print "C"
; movwf temp_wr
; call d_write
; movlw A"U" ; print "U"
; movwf temp_wr
; call d_write
movlw A"R"
movwf temp_wr
call d_write
movlw A"R"
movwf temp_wr
call d_write
movlw A"E"
movwf temp_wr
call d_write
movlw 0x20 ; space
movwf temp_wr
call d_write
movlw A"O"
movwf temp_wr
call d_write
movlw A"U"
movwf temp_wr
call d_write
movlw A"T"
movwf temp_wr
call d_write
movlw 0x20   ;space
movwf temp_wr
call d_write
movlw A="   ;print "="
movwf temp_wr
call d_write
movlw 0x20   ;space
movwf temp_wr
call d_write
movf volt_in, W
movwf AARGB0   ;voltage in
movf curr_in, W
movwf BARG0    ;current in
call UMUL0808L  ;multiply BARG0 by AARGB0 (result stored in BARGB1 (high) and
AARGB1 (low)

movf BARGB1, W  ;high bit register result of mult stored in BARGB0
movf BARG0
movf AARGB1, W  ;low bit register result of mult stored in AARGB0
movf AARG0
movf batt_volt, W  ;store 50V (volt. output to batteries) in BARGB1
movf BARGB1
call UDIV1608L   ;[BARGB0][AARGB0][BARGB1] result stored in AARGB1
movf AARGB1, W  ;storing result in AARGB1 and sending to reg. W to print

movf MSD,W
movff temp_wr
call d_write
movf MsD,W
movff temp_wr
call d_write
movf LSD,W   ;send high digit from the LSD #.xx
movff temp_wr
call d_write

movf volt_in, W  ;moves input voltage into reg. W
mulwf curr_in   ;multiplies the input volt. and input curr, stores result in W
movf PRODL, W
movw AARGB0
movf PRODH, W
movw AARG0
movf batt_volt, W  ;6
movw BARGB0
mov w BARG0
movf AARGB0, W  ;prepare for 16-bit binary to BCD
movf NumH
movf AARGB1, W
movf NumL
mov w bin16_bcd  ;get volts ready for LCD
mov w LCDLine_2  ;display A/D result on 2nd line
movf Hund,W  ;get hunds
mov w bin_bcd
movf MsD,W
movw temp_wr
;call    d_write  ;send high digit from the LSD #.xx
;movf    LSD,W
;movwf   temp_wr
;call    d_write

movf    AARG1, W
;call    bin_bcd

movf    MSD,W   ;send high digit
movwf    temp_wr
;call    d_write
movf    Msd,W   ;send middle digit
movwf    temp_wr
;call    d_write
movf    LSD,W   ;send low digit
movwf    temp_wr
;call    d_write
movlw    A'm'
movwf    temp_wr
call    d_write
movlw    A'A'
movwf    temp_wr
call    d_write
movlw    0x20   ;space
movwf    temp_wr
;end of sending unit to LCD
call    d_write
goto    volts_again

;-------------------------------
volts_again
movlw    .144   ;Display "RB0 = Exit" to LCD
movwf    ptr_pos
;move data into TXREG
;carriage return
btfss    TXSTA, TRMT  ;wait for data TX
bra      $-2
btfsc    select ;exit volt measurement ?? - if register select is 0, then skip next instruction and exit
bra      voltmeter ;NO, do conversion again
btfsc    which_menu, 0 ;YES, if bit 0 of register which_menu is 0, then skip next instruction
bra      menu       ;branches to next menu item (solar current output)
btfsc    which_menu, 1 ;YES, if bit 1 of reg. which_menu is 0, skips next instr
bra      menu_buz   ;branches to next menu item (mppt voltage output)
btfsc    which_menu, 2 ;YES, if bit 2 of reg. which_menu is 0, skips next instr
bra      menu_temp  ;branches to next menu item (mppt current output)
btfsc    which_menu, 3 ;YES, if bit 3 of reg. which_menu is 0, skips next instr
bra      menu_clock

;----------------- CLOCK -----------------------------
clock
btfss    select ;wait for RB0 button release
bra      $-2
movlw 0x0F   ;initialize TIMER1
movwf T1CON
clf seconds
clf minutes
clf hours
overflow
bcf PIR1,TMR1IF
movlw 0x80
movwf TMR1H   ;load regs for 1 sec overflow
clf TMR1L
incf seconds,F  ;increment seconds
movf seconds,W
sublw .60
btfss STATUS,Z  ;increment minutes ?
bra clk_done
incf minutes,F
clf seconds
movf minutes,W
sublw .60
btfss STATUS,Z  ;increment hours ?
bra clk_done
incf hours,F
clf minutes
movf hours,W
sublw .13
btfss STATUS,Z
bra clk_done
movlw .1
movwf hours   ;start a new 12 hour period
clk_done
movf hours,W   ;send hours to LCD
call bin_bcd
call LCDLine_1  ;place time on line 1
movf MsD,W   ;send middle digit
movwf temp_wr
call d_write
movf LSD,W   ;send low digit
call d_write
movlw 0x3A   ;send : colon
call d_write
movf minutes,W  ;send minutes to LCD
call bin_bcd
movf MsD,W   ;send middle digit
movwf temp_wr
call d_write
movf LSD,W   ;send low digit
call d_write
movlw 0x3A   ; send : colon
call d_write
movf seconds,W ;send seconds to LCD
call bin_bcd

movf MsD,W ;send middle digit
movwf temp_wr
call d_write

movf LSD,W ;send low digit
movwf temp_wr
call d_write

movlw 0x20 ;send 3 spaces after 00:00:00
call d_write

movlw 0x20
call d_write

movlw 0x20
call d_write

movlw 0x20
call d_write

movlw 0x20
call d_write

movlw .112 ;send "RA4=Dn RB0=Menu" to LCD
call stan_char_2

btfss scroll ;set time ?
bra set_time

btfss select ;return to main menu ?
bra menu

btfss PIR1,TMR1IF ;has timer1 overflowed ?
bra $-2 ;NO, wait til overflow
bra overflow ;YES

return

***************************** ****************************

;************************** ROUTINES *************************
;*************************************************************
;*******************************************************;
;----Standard code, Place characters on line-1-----------------------

斯坦字符_1

call LCDLine_1 ;move cursor to line 1
movlw .16 ;1-full line of LCD
movwf ptr_count
movlw UPPER stan_table
movwf TBLPTRU
movlw HIGH stan_table
movwf TBLPTRH
movlw LOW stan_table
movwf TBLPTRL
movf ptr_pos,W
addwf TBLPTRL,F
clrf WREG
addwf TBLPTRH,F
addwf TBLPTRU,F
```assembly
tan_next_char_1
  tblr
  movff TABLET,temp_wr
  call d_write ;send character to LCD
  decfsz ptr_count,F ;move pointer to next char
  bra tan_next_char_1
  movlw "\n" ;move data into TXREG
  movwf TXREG ;next line
  btfss TXSTA,TRMT ;wait for data TX
  goto $-2
  movlw "\r" ;move data into TXREG
  movwf TXREG ;carriage return
  btfss TXSTA,TRMT ;wait for data TX
  goto $-2
  return

;----Standard code, Place characters on line-2----------------------
tan_char_2
  call LCDLine_2 ;move cursor to line 2
  movlw .16 ;1-full line of LCD
  movwf ptr_count
  movlw UPPER stan_table
  movwf TBLPTRU
  movlw HIGH stan_table
  movwf TBLPTRH
  movlw LOW stan_table
  movwf TBLPTRL
  movf ptr_pos,W
  addwf TBLPTRL,F
  clrf WREG
  addwfc TBLPTRH,F
  addwfc TBLPTRU,F

stan_next_char_2
  tblr
  movff TABLET,temp_wr
  call d_write ;send character to LCD
  decfsz ptr_count,F ;move pointer to next char
  bra tan_next_char_2
  movlw "\n" ;move data into TXREG
  movwf TXREG ;next line
  btfss TXSTA,TRMT ;wait for data TX
  goto $-2
  movlw "\r" ;move data into TXREG
  movwf TXREG ;carriage return
  btfss TXSTA,TRMT ;wait for data TX
  goto $-2
  return

;--------------------------------------------- --------------------
;------------------ 100ms Delay -----------------------------
delay_100ms
  movlw 0xFF
```

movwf temp_1
movlw 0x83
movwf temp_2
d100l1
  decfsz temp_1,F
  bra d100l1
decfsz temp_2,F
  bra d100l1
return

;---------------- 1s Delay -----------------------------
delay_1s
  movlw 0xFF
  movwf temp_1
  movwf temp_2
  movlw 0x05
  movwf temp_3
d11
  decfsz temp_1,F
  bra d11
  decfsz temp_2,F
  bra d11
  decfsz temp_3,F
  bra d11
return

;---------------- Set Current Time ---------------------
set_time
  movlw .128
  ;send "RA4= --> RBO= ++" to LCD
  movwf ptr_pos
  call stan_char_2
  set_time_again
  btfss scroll
  ;wait for button release
  bra $-2
  call LCDLine_1
  ;start at 0x00 on LCD
  btfss select
  ;wait for RB0 button release
  bra $-2
  call delay_100ms
  btfss select
  ;increment hours (tens) ?
  bra inc_hours
  bra next_digit
inc_hours
  incf hours
  movf hours,W
  ;check if hours has passed 12 ?
  sublw .13
  btfss STATUS,Z
  ;YES, reset hours to 00
  bra next_digit
next_digit
  btfss scroll
  ;move to next digit
  bra inc_mins
  movf hours,W
  call bin_bcd
  ;get hours ready for display
  movf MsD,W
  ;send tens digit
  movwf temp_wr
call d_write
movf LSD,W ;send ones digit
movwf temp_wr
call d_write
movlw 0x3A ;send : colon
movf temp_wr
call d_write
bra set_time_again

inc_mins
btfss scroll ;wait for RA4 button release
bra $-2
call LCDLine_1
movlw 0x14 ;shift cursor to right 3 places
movf temp_wr
call i_write
movlw 0x14
movf temp_wr
call i_write
movlw 0x14
movf temp_wr
call i_write
movf minutes,W ;check if hours has passed 12 ?
sublw .60
btfss STATUS,Z
bra next_digit?
clf minutes

next_digit?
btfss scroll ;move to next digit
bra set_time_done
movf minutes,W

call bin_bcd ;get minutes ready for display
movf MsD,W ;send tens digit
movf temp_wr
call d_write
movf LSD,W ;send ones digit
movf temp_wr
call d_write
movlw 0x3A ;send : colon
movf temp_wr
call d_write
bra inc_mins

set_time_done
btfss scroll ;wait for RA4 button release
bra $-2
bra overflow
;---------------- Binary (8-bit) to BCD ------------------
;  255 = highest possible result

bin_bcd
  clrf MSD
  clrf MsD
  movwf LSD  ; move value to LSD

  ghundredth
  movlw .100  ; subtract 100 from LSD
  subwf LSD,W
  btfss STATUS,C  ; is value greater than 100
  bra gtenth  ; NO goto tenths
  movwf LSD  ; YES, move subtraction result into LSD
  incf MSD,F  ; increment hundredths
  bra ghundredth

  gtenth
  movlw .10  ; take care of tenths
  subwf LSD,W
  btfss STATUS,C
  bra over  ; finished conversion
  movwf LSD
  incf MsD,F  ; increment tenths position
  bra gtenth

over
  movf MSD,W   ; get BCD values ready for LCD display
  xorlw 0x30
  movwf MSD
  movf MsD,W
  xorlw 0x30
  movwf MsD
  movf LSD,W
  xorlw 0x30
  movwf LSD

  retlw 0

;---------------- Binary (16-bit) to BCD -----------------------
;  xxx = highest possible result

bin16_bcd
  ; Takes number in NumH:NumL
  ; Returns decimal in TenK:Thou:Hund:Tens:Ones

  swapf NumH,W
  andlw 0x0F
  addlw 0xF0
  movwf Thou
  addwf Thou,F
  addlw 0xE2
  movwf Hund
  addlw 0x32
  movwf Ones

  movf NumH,W
  andlw 0x0F
  addwf Hund,F
  addwf Hund,F
  addlw 0xE9
  movwf Tens
  addwf Tens,F
  addwf Tens,F
  retlw 0
swapf NumL,W
andlw 0x0F
addwf Tens,F
addwf Ones,F

rlcf Tens,F
rlcf Ones,F
comf Ones,F
rlcf Ones,F

movf NumL,W
andlw 0x0F
addwf Ones,F
rlcf Thou,F

movlw 0x07
movwf TenK
movlw 0x0A ; Ten
Lb1:
decf Tens,F
addwf Ones,F
btfss STATUS,C
bra Lb1
Lb2:
decf Hund,F
addwf Tens,F
btfss STATUS,C
bra Lb2
Lb3:
decf Thou,F
addwf Hund,F
btfss STATUS,C
bra Lb3
Lb4:
decf TenK,F
addwf Thou,F
btfss STATUS,C
bra Lb4

retlw 0

;---------------------------- EEPROM WRITE -----------------------------
write_eeprom
bsf SSPCON2,SEN ;start bit
btfsc SSPCON2,SEN
goto $-2
movlw B'10100000' ;send control byte (write)
movwf SSPBUF
ssprw
btfsc SSPCON2,ACKSTAT ;ack?
goto $-2
movlw 0x00 ;send slave address HIGH byte
movwf SSPBUF
ssprw
btfsc SSPCON2,ACKSTAT ;ack?
goto $-2
movlw 0x05   ;send slave address LOW byte(0x0005)
movwf SSPBUF
ssprw
btfsc SSPCON2,ACKSTAT  ;ack?
goto $-2

movf temperature, w  ;send slave DATA = temperature
movwf SSPBUF
ssprw
btfsc SSPCON2,ACKSTAT  ;ack?
goto $-2

bsf SSPCON2,PEN  ;stop bit
btfsc SSPCON2,PEN
  goto $-2

bcf PIR1,TMR1IF  ;clear TIMER1 overflow flag
clr TMR1H  ;clear registers for next overflow
clr TMR1L

return

;**************************** ****************************
end
APPENDIX 7.3.1  LCD.ASM

;********************************************************************
;* Microchip Technology Inc. 2002
;* Assembler version: 2.0000
;* Filename:
;*   p18lcd.asm (main routine)
;* Dependents:
;*   p18demo.asm
;*   p18math.asm
;*   16f877.lkr
;********************************************************************

list p=18f452
#include p18f452.inc

#define LCD_D4 PORTD, 0 ; LCD data bits
#define LCD_D5 PORTD, 1
#define LCD_D6 PORTD, 2
#define LCD_D7 PORTD, 3
#define LCD_D4_DIR TRISD, 0 ; LCD data bits
#define LCD_D5_DIR TRISD, 1
#define LCD_D6_DIR TRISD, 2
#define LCD_D7_DIR TRISD, 3
#define LCD_E PORTA, 1 ; LCD E clock
#define LCD_RW PORTA, 2 ; LCD read/write line
#define LCD_RS PORTA, 3 ; LCD register select line
#define LCD_E_DIR TRISA, 1
#define LCD_RW_DIR TRISA, 2
#define LCD_RS_DIR TRISA, 3
#define LCD_INS 0
#define LCD_DATA 1

D_LCD_DATA   UDATA
COUNTER    res 1
delay       res 1
temp_wr     res 1
temp_rd     res 1

GLOBAL temp_wr

PROG1 CODE

;***************************************************************************

LCDLine_1
  movlw 0x80
  movwf temp_wr
  rcall i_write
  return

GLOBAL LCDLine_1

LCDLine_2
movlw 0xC0
movwf temp_wr
rcall i_write
return
GLOBAL LCDLine_2

;write data
d_write
movff temp_wr,TXREG
btfss TXSTA,TRMT
goto $-2
rcall LCDBusy
bsf STATUS, C
rcall LCDWrite
return
GLOBAL d_write

;write instruction
i_write
rcall LCDBusy
bcf STATUS, C
rcall LCDWrite
return
GLOBAL i_write

rlcd macro MYREGISTER
IF MYREGISTER == 1
bsf STATUS, C
rcall LCDRead
ELSE
bcf STATUS, C
rcall LCDRead
ENDIF
endm

;********************************************************************
;*******************************************************************
LCDInit
clrf PORTA
bcf LCD_E_DIR ;configure control lines
bcf LCD_RW_DIR
bcf LCD_RS_DIR
movlw b'00001110'
movwf ADCON1
movlw 0xff   ; Wait ~15ms @ 20 MHz
movwf COUNTER
lil1
movlw b'00001110'
movwf ADCON1
movlw 0xff
movwf COUNTER
lil1
movlw b'00110000'  ;#1 Send control sequence
movwf temp_wr
bcf STATUS,C
rcall LCDWriteNibble
movlw 0xff               ;Wait ~4ms @ 20 MHz
movwf COUNTER
lil2
movlw 0xFF
movwf delay
rcall DelayXCycles
decfsz COUNTER,F
bra lil2
movlw b'00110000'        ;#2 Send control sequence
movwf temp_wr
bcf STATUS,C
rcall LCDWriteNibble
movlw 0xFF               ;Wait ~100us @ 20 MHz
movwf delay
rcall DelayXCycles
movlw b'0011000'         ;#3 Send control sequence
movwf temp_wr
bcf STATUS,C
rcall LCDWriteNibble
         ;test delay
movlw 0xFF               ;Wait ~100us @ 20 MHz
movwf delay
rcall DelayXCycles
movlw b'00100000'        ;#4 set 4-bit
movwf temp_wr
bcf STATUS,C
rcall LCDWriteNibble
rcall LCDBusy             ;Busy?
movlw b'00101000'        ;#5 Function set
movwf temp_wr
rcall i_write
movlw b'00001101'        ;#6 Display = ON
movwf temp_wr
rcall i_write
movlw b'00000001'        ;#7 Display Clear
movwf temp_wr
rcall i_write
movlw b'00000110'        ;#8 Entry Mode
movwf temp_wr
rcall i_write
movlw b'10000000'        ;DDRAM addresss 0000
movwf temp_wr
rcall i_write
return
GLOBAL LCDInit

; *******************************************************************
;********************************************************************
LCDWriteNibble

btfss STATUS, C ; Set the register select
bcf LCD_RS
btfsc STATUS, C
bsf LCD_RS

bcf LCD_RW ; Set write mode

bcf LCD_D4_DIR ; Set data bits to outputs
bcf LCD_D5_DIR
bcf LCD_D6_DIR
bcf LCD_D7_DIR

NOP ; Small delay
NOP

bsf LCD_E ; Setup to clock data

btfss temp_wr, 7 ; Set high nibble
bcf LCD_D7
btfsc temp_wr, 7
bsf LCD_D7
btfss temp_wr, 6
bcf LCD_D6
btfsc temp_wr, 6
bsf LCD_D6
btfss temp_wr, 5
bcf LCD_D5
btfsc temp_wr, 5
bsf LCD_D5
btfss temp_wr, 4
bcf LCD_D4
btfsc temp_wr, 4
bsf LCD_D4

NOP
NOP

bcf LCD_E ; Send the data

return

; *******************************************************************
;********************************************************************

GLOBAL LCDWrite

; rcall LCDBusy
rcall LCDWriteNibble
swapf temp_wr,F
rcall LCDWriteNibble
swapf temp_wr,F

return
LCDRead
  bsf   LCD_D4_DIR           ; Set data bits to inputs
  bsf   LCD_D5_DIR
  bsf   LCD_D6_DIR
  bsf   LCD_D7_DIR

  btfss   STATUS, C          ; Set the register select
  bcf    LCD_RS
  btfs   STATUS, C
  bsf    LCD_RS

  bsf    LCD_RW              ; Read = 1
  NOP
  NOP

  bsf    LCD_E               ; Setup to clock data
  NOP
  NOP
  NOP
  NOP

  btfss   LCD_D7             ; Get high nibble
  bcf    temp_rd, 7
  btfs   LCD_D7
  bsf    temp_rd, 7
  btfs   LCD_D6
  bcf    temp_rd, 6
  btfs   LCD_D6
  bsf    temp_rd, 6
  btfs   LCD_D5
  bcf    temp_rd, 5
  btfs   LCD_D5
  bsf    temp_rd, 5
  btfs   LCD_D4
  bcf    temp_rd, 4
  btfs   LCD_D4
  bsf    temp_rd, 4

  bcf    LCD_E                ; Finished reading the data
  NOP
  NOP
  NOP
  NOP
  NOP
  NOP
  NOP

  bsf    LCD_E                ; Setup to clock data
  NOP
  NOP

  btfss   LCD_D7              ; Get low nibble
  bcf    temp_rd, 3
Tufts University Senior Design 2003
Stephanie Chin, Jeanell Gadson, Katie Nordstrom
Nerd Girls Solar/MPPT Group

btfsc LCD_D7
bsf temp_rd, 3
btfss LCD_D6
bcf temp_rd, 2
btfsc LCD_D6
bsf temp_rd, 2
btfss LCD_D5
bcf temp_rd, 1
btfsc LCD_D5
bsf temp_rd, 1
btfss LCD_D4
bcf temp_rd, 0
btfsc LCD_D4
bsf temp_rd, 0
bcf LCD_E ; Finished reading the data

FinRd
return

; ************************************************************************
; ************************************************************************

LCDBusy
; Check BF
rlcd LCD_INS
btfsc temp_rd, 7
bra LCDBusy
return

GLOBAL LCDBusy

; ************************************************************************
; ************************************************************************

DelayXCycles

decfsz delay,F
bra DelayXCycles
return

; ************************************************************************

END


APPENDIX 7.3.3 MATH.ASM

************************************************************************
* Microchip Technology Inc. 2002
* Assembler version: 2.0000
* Filename:
* p18math.asm (main routine)
* Designed to run at 4MHz
* PICDEM 2 PLUS DEMO code
************************************************************************

list p=18f452
#include p18f452.inc
#define _C STATUS,0

MATH_VAR UDATA
AARGB0 RES 1
AARGB1 RES 1
AARGB5 RES 1
BARGB0 RES 1
BARGB1 RES 1
REMB0 RES 1
REMB1 RES 1
TEMP RES 1
LOOPCOUNT RES 1

GLOBAL AARGB0, AARGB1, BARGB0, BARGB1, REMB0, AARGB5, REMB1, TEMP

PROG2 CODE
;---------------- 8 * 8 UNSIGNED MULTIPLY -----------------------
;       Max Timing:     3+12+6*8+7 = 70 clks
;       Min Timing:     3+7*6+5+3 = 53 clks
;       PM: 19            DM: 4
UMUL0808L
    CLRF    AARGB1
    MOVLW   0x08
    MOVWF   LOOPCOUNT
    MOVF    AARGB0,W
    LOOPUM0808A
    RRCF     BARGB0, F
    BTFSC   _C
    bra    LUM0808NAP
    DECFSZ  LOOPCOUNT, F
    bra    LOOPUM0808A
    CLRF    AARGB0
    RETLW   0x00
LUM0808NAP
    BCF     _C
    bra    LUM0808NA
LOOPUM0808
    RRCF    BARGB0, F
    BTFSC   _C
    ADDWF   AARGB0, F
LUM0808NA  RRCF  AARGB0, F
  RRCF  AARGB1, F
  DECFSZ  LOOPCOUNT, F
  bra  LOOPUM0808
  return
GLOBAL  UMUL0808L

;----------------  16/8 UNSIGNED DIVIDE   ------------------------

; Max Timing: 2+7*12+11+3+7*24+23 = 291 clks
; Min Timing: 2+7*11+10+3+7*17+16 = 227 clks
; PM: 39  DM: 7

UDIV1608L
GLOBAL  UDIV1608L
  CLRF  REMB0 ; clears contents of register REMB0
  MOVLW  8 ; moves 8 into register LOOPCOUNT
  MOVWF  LOOPCOUNT
  UDIV1608A
  RLCF  AARGB0,W ; contents of reg. AARGB0 rotated one bit to left through carry flag (result in W)
  RLCF  REMB0, F ; contents of reg. REMB0 rotated one bit to left through carry flag
  MOVF  BARGB0,W ; moves contents of BARGB0 to reg. W
  SUBWF  REMB0, F
  BTFSC  _C
  bra  UOK68A
  ADDWF  REMB0, F
  BCF  _C
  UOK68A
  RLCF  AARGB0, F
  DECFSZ  LOOPCOUNT, F
  bra  LOOPU1608A
  CLRF  TEMP
  MOVLW  8
  MOVWF  LOOPCOUNT

LOOPU1608B
  RLCF  AARGB1,W
  RLCF  REMB0, F
  RLCF  TEMP, F
  MOVF  BARGB0,W
  SUBWF  REMB0, F
  CLRF  AARGB5
  CLRW
  BTFFS  _C
  INCFSZ  AARGB5,W
  SUBWF  TEMP, F
  BTFFS  _C
  bra  UOK68B
  MOVF  BARGB0,W
  ADDWF  REMB0, F
  CLRF  AARGB5
  CLRW
  BTFFS  _C
  INCFSZ  AARGB5,W
  ADDWF  TEMP, F
  BCF  _C
UOK68B RLCF AARGB1, F

DECSZ LOOPCOUNT, F
bra LOOPU1608B

return
GLOBAL UDIV1608L

end
APPENDIX 7.3.4  P2PLSP18.LKR

// Sample linker command file for 18F452i used with MPLAB ICD 2
// $Id: 18f452i.lkr,v 1.1 2002/02/26 16:55:21 sealep Exp$

LIBPATH .

CODEPAGE NAME=vectors START=0x0 END=0x29 PROTECTED
CODEPAGE NAME=page START=0x2A END=0x7DBF
CODEPAGE NAME=debug START=0x7DC0 END=0x7FFF PROTECTED
CODEPAGE NAME=idlocs START=0x200000 END=0x200007 PROTECTED
CODEPAGE NAME=config START=0x300000 END=0x30000D PROTECTED
CODEPAGE NAME=devid START=0x3FFFFE END=0x3FFFFF PROTECTED
CODEPAGE NAME=eedata START=0xF00000 END=0xF000FF PROTECTED

ACCESSBANK NAME=accessram START=0x0 END=0x7F
DATABANK   NAME=gpr0 START=0x80 END=0xFF
DATABANK   NAME=gpr1 START=0x100 END=0x1FF
DATABANK   NAME=gpr2 START=0x200 END=0x2FF
DATABANK   NAME=gpr3 START=0x300 END=0x3FF
DATABANK   NAME=gpr4 START=0x400 END=0x4FF
DATABANK   NAME=gpr5 START=0x500 END=0x5FF
DATABANK   NAME=dbgspr START=0x5F4 END=0x5FF PROTECTED
ACCESSBANK NAME=accesssfr START=0xF80 END=0xFFF PROTECTED

SECTION NAME=STARTUP ROM=vectors
SECTION NAME=PROG1 ROM=page
APPENDIX 7.4 Datasheets
APPENDIX 7.4.1  PIC MICROCONTROLER
APPENDIX 7.4.2 DC/DC CONVERTER PT4122A
APPENDIX 7.4.3  DC/DC CONVERTER TPS6734IP
APPENDIX 7.4.4  PWM TL598CN
APPENDIX 7.4.4 PWM TL598CN
APPENDIX 7.4.5 DIODE 16CTU04S
APPENDIX 7.4.6  LTC DAC 1451CM8
APPENDIX 7.4.7  MOSFET IXFX90N20Q
APPENDIX 7.4.8  MOSFET DRIVER MAX4420CPA
8. REFERENCES


“IXYS HiPerFET Power MOSFETs IXFX 90N20Q Data Sheet.” IXYS, 2002.


”Maxim High-Speed, 6A Single MOSFET Drivers Data Sheet.” Maxim Integrated Products, 1992.


9. ACKNOWLEDGEMENTS

We would like to thank our project advisor, Professor Karen Panetta, for her support throughout the year and for giving us the opportunity to work on a challenging and unique project with an amazing team of engineers.

We would also like to thank our project consultants, Matthew Heller, Richard Colombo, and Michael Quaglia, for their generous time, patience, and guidance with the MPPT design. We have learned valuable engineering project skills that we can apply to future endeavors.

Many thanks also go out to Project Manager Larisa Schelkin, Professor Steven Morrison, George Preble, and Warren Gagosian for their undying willingness to help with any aspect of the project.