Column #50, June 1999 by Lon Glazner:

Dual Digital Power Supply — Part 2

For those of you who were tuned into last month's episode of "Engineers With Brain Spasms," you would have seen me go through the process of defining a user interface for a digitally-controlled power supply. The user interface went together pretty quickly and with relatively little pain. I wish I could say the same about this month's power supply design. I went to great lengths to keep the design as simple as possible, and was thus hampered by my own design constraints. That being said, I feel that there are quite a few useful bits and pieces that can be filtered out of the final product.

So even though I came close to scrapping this implementation of a digitally-controlled dual power supply at least five times over the last month, in the end I felt that it provided enough useful information to push it through to completion.

In electronic design, there are a few fields that are clearly viewed as arcane arts by engineers not employed in those fields. A sampling of these disciplines would include radio frequency design, high-speed digital design, antenna theory and, of course, switching power supply design. I don't pretend to be an expert in any of those fields. So, when it came time to design a power supply circuit for this article, I made sure to make use of circuits that were relatively simple. The power supply circuit that I used is inexpensive, requires few parts, and is extremely useful.

I have found this circuit to be quite useful in previous designs, although its effectiveness in this particular design was somewhat limited.

I also felt that this article should focus on implementing a digital control technique in conjunction with a user interface. I was wary of introducing too many new electronic components, and thus making the article too difficult for beginners to understand. Given the amount of space I have available, it's very difficult to provide an accurate overview of a design if I have to describe in detail a large number of electronic parts. What's the point of my diatribe you ask? Well, in order to keep things simple, I resorted to using the DS1267-010 Dallas Semiconductor digital potentiometer, and the ADC0831 National Semiconductor eight-bit A/D. Both of these parts were used in the April '99 Stamp Applications article. So, if you are a regular reader, you should be familiar with those parts. But by selecting these parts, I placed some serious limitations on the power supply's capabilities.

Defining the Design

In my initial design, I was shooting for a digitally-controlled dual power supply with on output range of 3-20V and about 2A current source capability. A linear regulator, such as the National Semiconductor LM317, could be used. But if I used a linear regulator, cooling and heatsinking would be mandatory. For example, an 8Vdc output linear regulator, providing 500mA of current, with an input voltage of 24Vdc would have to dissipate (24Vdc-8Vdc)*0.5A = 8 watts. Even with external pass elements, I would have to account for considerable power dissipation.

On the other hand, a switching power supply would provide me with the efficiency necessary to minimize power dissipation concerns. It was primarily due to power dissipation considerations that I selected the MAX726 (Maxim Integrated Products: 1-800-998-8800 for samples) step-down, PWM, switch-mode DC-DC regulator, as a power supply. You can approximate the power dissipation requirements for the MAX726 by multiplying the load current by 1.1Vdc. This is described in the MAX726 data sheet. Therefore, by using a MAX726 instead of a linear regulator, your 8Vdc supply would only have to dissipate 1.1Vdc*0.5A = 0.55 watts; which is a significant improvement over the linear regulator.

Furthermore, the MAX726 has an output voltage range of 2.5Vdc-35Vdc, and the ability to source 2A of continuous current. Package power dissipation should still be addressed at higher load currents. These features coupled with some internal short circuit protection make the MAX726 ideal for this design.

But how do you adjust the output voltage with this regulator? Take a glance at Figure 50.1. From the MAX726 data sheet, I know that the output voltage is selected by R7 and the value of the potentiometer. I also know from the equation for a voltage divider that the feedback voltage at pin 1 of the MAX726 can be described by …

Vfb = Vout*(Rpot/(Rpot+R7)

But unlike your average voltage divider, the feedback voltage, Vfb, is pre-set to 2.21Vdc internally by the MAX726. This has two important effects on our design. The first effect is that with Vfb set to 2.21V, the current through our digital pot will not exceed its maximum rating (5mA). Additionally, the voltage on our digital pot is kept within its specifications (7Vdc maximum). All in all, it means that a digital pot is an effective means of adjusting the output voltage generated by the MAX726.

There are a few points that must be stated here. The digital potentiometer (DS1267-010) was very effective in controlling the output voltage of the MAX726 over a short range of voltages. Specifically, I had luck controlling the MAX726 between 3Vdc-10Vdc. Over 10Vdc, there was not enough resolution available in the DS1267-010 to allow accurate voltage control. So at this point, I reduced the power supply output to 10Vdc maximum, and modified the user interface code to reflect this change. I also realized that whenever I reduced the output voltage below 4Vdc, I was violating a recommendation in the MAX726 data sheet. The data sheet specifically states that the resistor between the FB pin and ground should not exceed 4K ohms (this is where the DS1267-010 is located). In this system, any voltage output between 3Vdc and 4Vdc is generated by a DS1267-010 setting of greater than 4K ohms. I tested the voltages under load and didn't find any serious degradation in performance. So I left the minimum output voltage limit set to 3Vdc.

The limitations forced onto this design really did revolve around the capabilities of the DS1267-010. If, for example, the digital pot had 10 bits of resolution, then I think many of these shortcomings would disappear. But for simplicity and, due to a nearing publication deadline, I decided to press on with the design.

As part of a feedback system, I included two eight-bit A/D converters (ADC0831). So the way the system worked could be described in five steps.

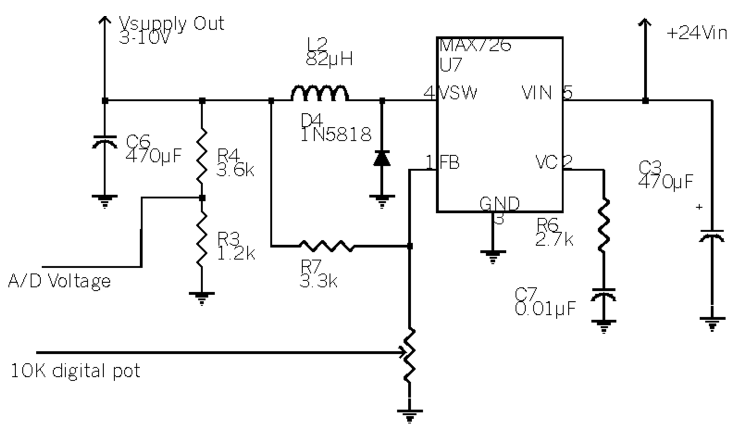
1. The user enters the desired voltage output.
2. The BASIC Stamp2 (BS2) converts the desired output to a desired A/D reading.
3. The BS2 calculates an approximate setting for the digital potentiometer.

4. The BS2 measures the actual output voltage and trims the DS1267-010 until the desired A/D matches the actual A/D.
5. If the actual output voltage never matches the desired output voltage, then the user is notified, otherwise the system waits for the next user update.

The A/D inputs were originally read through a voltage divider circuit that divides down the actual output voltage by four. This was done in order to measure 20Vdc maximums with 5Vdc A/Ds. When I changed the output voltage range of the supply from 3Vdc-20Vdc to 3Vdc-10Vdc, I could have changed the voltage divider from a divide by four configuration to a divide by two configuration. Making this change would give greater resolution in the A/D measurement results. While this would be good, it doesn't change the fact that output voltage resolution is limited at higher output voltages due to the characteristics of the DS1267-010.

So, after all of the give and take, what is left over is a dual supply with 3Vdc-10Vdc regulated output and a 1A source capability. The current handling capability of this circuit was limited to 1A by the inductor selected.

Figure 50.1: Hookup diagram

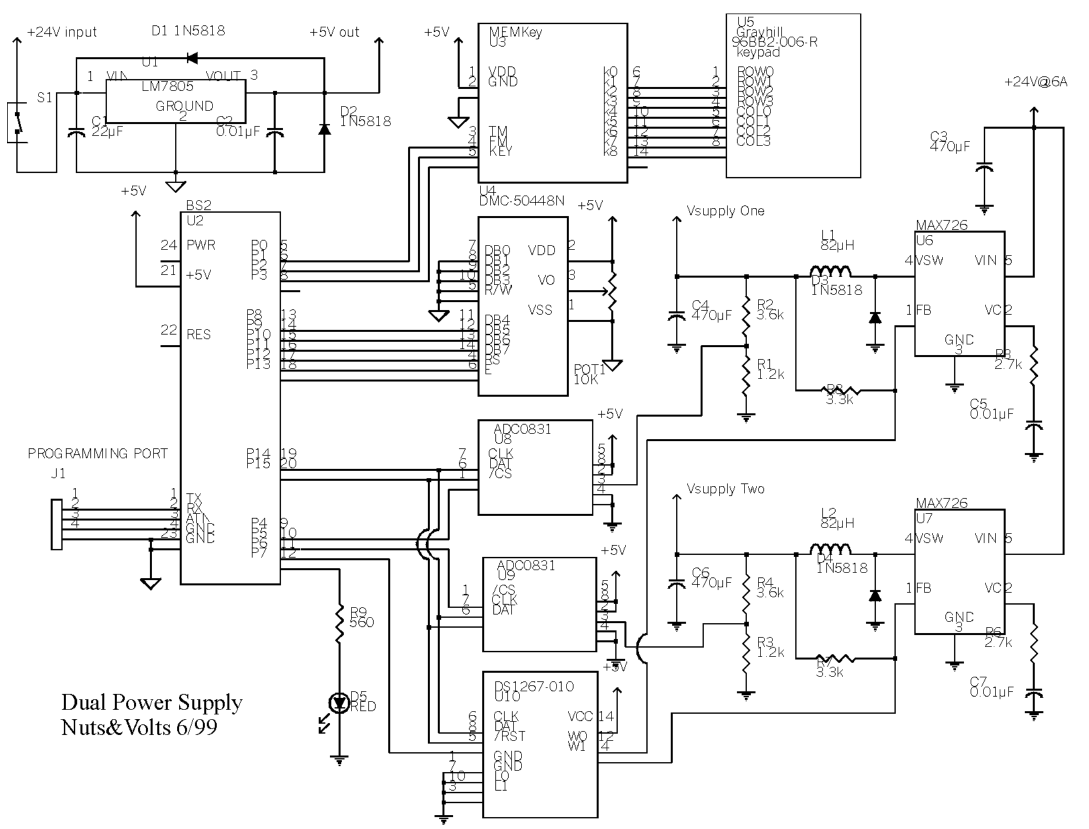


Connecting the Parts

The two ADC0831s and the DS1267-010 were all connected to the same clock and data lines. Each chip had a separate chip select control line. Using the LCD in four-bit mode — as well as the MEMKey for serial keypad encoding — freed up enough I/O lines to allow this design to get done. The MEMKey could be used in single wire communication mode by connecting the TM and FM pins together to one BS2 I/O pin. This would free up another I/O line. Also, if the out of regulation indicator LED was omitted from the design, then three I/O lines could be available for other use.

Program memory is pretty much used up at this point, although moving the design to a BS2SX would eliminate that problem. A complete system schematic is detailed in Figure 50.2.

Figure 50.2: User interface and power supply together



Power Supply Limitations

The major limitations of this power supply were imposed by using the DS1267-010. If I had this design to do over again, I would probably remove the digital potentiometer and add two 5K ohm mechanical potentiometers for voltage setting. Then you could change the eight-bit A/Ds to 10-bit A/Ds. You might even add a couple more A/Ds and some current measurement capability. The keypad could also be removed in lieu of a couple of buttons for selecting voltage or current displays. If you wanted to get really tricky, you could design an active load by biasing a BJT in its active region through adjustments to its base resistance. This could potentially replace the DS1267-010.

An important thing to consider when using the MAX726 is which inductor to use. The inductor is in the high current path. So your inductor must have a continuous current rating equal to, or greater than, the maximum load current that your supply will provide. An inexpensive, relatively high current inductor is the Toko 8RHB type which can be found at Digi-Key. I had some 82uH inductors laying around which were rated for 1.1A, and they seemed to do the trick. There are higher current rated inductors that are available, but I really like the packaging of the Toko parts. They take up less room than some electrolytic capacitors.

Lastly, I didn't heatsink my MAX726 regulators. They handled about a watt of power (900mA load) without overheating. I used the three second test for my thermal modeling. The three second test consists of licking your finger and placing it on a potentially hot device. If you hear a sizzle (that sounds like saliva turning to steam), then it would be prudent to withdraw the exposed digit. If no sizzle occurs, and you can keep you finger on the device for three seconds without feeling any pain, then the part is probably not in danger of overheating. Of course, my finger has been calibrated by a national laboratory at great expense. Therefore, your own results may vary.

Writing the Code

The code was a bit of a bear. The concept for implementing this power supply was pretty straightforward. But since the digital potentiometer was not able to give me good resolution at the higher voltages, I had to throw in a slew of If…Then statements. In the end, I think I was able to get about 80% of the functionality that I was looking for. There are a couple of equations that I use in the "UpdateSupply" subroutine. These equations relate the digital potentiometer settings to the desired output voltage. When the equation is implemented, the value in the Analog register is a binary value that relates the output voltage to an A/D measurement result. Here is one of the equations of which I speak:

DSpots1 = (2376/(Analog1 - 28)) -14

And here is how it is arrived at. From the MAX726 data sheet, we know that the output voltage can be related to the digital potentiometer with the following equation:

Vout = (R7 +Rpot)*(2.21V/Rpot)

where
R7 = 3300 ohms
Rpot= 9000*Dspots1/256bits + 500 (for wiper) ohms
and Vout= Analog1*5Vdc/256bits*4 (for dividing circuit at A/D input) = 20*Analog1/256

Substituting this result for Vout gives …

20*Analog1/256 =3300*2.21/
((9000Dspots1/256)+500) + 2.21
20*Analog1/256 =7293
((9000Dspots1/256)+500) + 2.21
Analog1 = (256/20)*((7293/(35*Dspots1
+500)) + 2.21)
Analog1 = (93359/(35*Dspots1 + 500)) + 28
Analog1 = 2667/(Dspots1 + 14) + 28
Now solve for Dspots1…
Dspots1 = (2667/(Analog1 -28)) - 14

After this relationship was derived, I fine-tuned the equations for both of the output voltages. This was most easily done by adjusting the constant 2667 until the output voltage created by a new potentiometer setting closely matched the desired A/D reading that is stored in the Analog registers.

A little explanation of the how the registers were used in this program may clarify the functionality of the software. The Analog registers store an eight-bit value that is boiled down from the desired output voltage as it is displayed on the LCD. This eight-bit value is then used to derive a value for setting the digital potentiometer, as described by the equations above. After the digital potentiometer has been updated, the program trims the output voltage by taking A/D readings (stored in AD_in registers), and comparing them to the desired value (again in the Analog registers). Any readings that do not meet the requirements set forth in the program cause the potentiometer setting to be adjusted up or down until a closer match occurs.

If, after several adjustments, the desired value does not match the actual value, then the actual value is loaded into the display and the “out of regulation” LED is lit.
This implementation of a digitally-controlled dual power supply required most of the BS2’s RAM and EEPROM program memory to implement. However, the code is far

from fine-tuned and could be refined to a great degree.

In Closing

I'm probably going to have a second go at this power supply circuit. The keypad and digital potentiometer will likely be removed. Instead a four-line LCD, four separate 10 bit A/Ds, and a couple of manual potentiometers will be used. The extra A/Ds would be used for current measurement. Further testing is also required to determine regulation under load. I would also like to locate a higher current inductor to maximize the current source capability of the design. It may be feasible to parallel a couple of the Toko inductors that I'm currently using to reach the 2A capability that the MAX726 can provide.

For those of you in need of a higher current system, Maxim also has the MAX724. This part is identical in functionality to the MAX726, but can handle up to 5A of current. This was one of those designs that seem to force limitations on me. Or maybe I should say that I forced them on myself by sticking with the DS1267-010 as a means of controlling the output voltage. But with those changes that I mentioned above, I think I can tweak this into a successful and useful electronic design.

Regardless, I learned a little, and didn't blow anything up.

```
'Program Listing 50.1: june99_4.bs2
'
'JUNE99.BS2 - Dual power supply and user interface code listing. This
'source code implements a user interface consisting of a 2x8 LCD screen
'operating in 4 bit mode, and a 4x4 keypad. The LCD is driven directly
'with a BASIC Stamp 2 while the keypad is encoded by a MEMKey serial
'keypad encoder. LCD display data is stored in the MEMKey's user
'accessible EEPROM. Further modifications to this code were included to
'implement a digitally controlled dual power supply. The supplies
'parameters were reduced from a maximum of 20.0V to 10.0V in order to
'maintain resolutions with eight-bit A/D and D/A (digital pot) devices.
'
'
' LCD constants
RS      CON     12              ' Register Select (1 = char)
E       CON     13              ' LCD Enable pin (1 = enabled)
'
'LCD control characters
ClrLCD  CON     $01             ' clear the LCD
CrsrHm  CON     $02             ' move cursor to home position
CrsrLf  CON     $10             ' move cursor left
CrsrRt  CON     $14             ' move cursor right
DispLf  CON     $18             ' shift displayed chars left
DispRt  CON     $1C             ' shift displayed chars right
DDRam   CON     $80             ' Display Data RAM control
'
' LCD Variables
Char    VAR     Byte            ' Character sent to LCD
'
' MEMKey pin assignments
TM      CON     0               ' To Master
FM      CON     1               ' From Master
KEY     CON     2               ' Key press notification pin
'
' MEMKey variables
Index   VAR     Byte            ' For next loop variable
KeyVal  VAR     Byte            ' Storage for key values
B_1     VAR     Byte            ' Variable storage byte
B_2     VAR     Byte            ' Variable storage byte
B_3     VAR     Byte            ' Variable storage byte
B_4     VAR     Byte            ' Variable storage byte
B_5     VAR     Byte            ' Variable storage byte
B_6     VAR     Byte            ' Variable storage byte
B_7     VAR     Byte            ' Variable storage byte
B_8     VAR     Byte            ' Variable storage byte
B_9     VAR     Byte            ' Variable storage byte
'
' MEMKey constants
Baud    CON     396             ' Baud rate = 2400
```

```
PConfig CON     $0E             ' Program configuration command
Config  CON     $00             ' Disable typematic, disable auto
PDBounce CON    $04             ' Program debounce command
DBounce CON     $0A             ' Set debounce for 25ms
Peeprom CON     $08             ' Program user EEPROM command
Reeprom CON     $09             ' Read user EEPROM command
Pkeyval CON     $0A             ' Program key value command
Rkeyval CON     $0B             ' Read key value command
Default CON     $11             ' Resets MEMKey to default values
Rbuffer CON     $00             ' Read key in buffer
'
'Serial Device Pin Assignments
Clk     CON     14              ' Serial clock control pin
Dat     CON     15              ' Serial data control pin
CS_ad1  CON     4               ' Chip select for A/D one
CS_ad2  CON     5               ' Chip select for A/D two
CS_pot  CON     6               ' Chip select for digital pot.

'
'Serial Device Variables
DSpots  VAR     WORD            ' Storage word for pot values
DSpots1 VAR     DSpots.lowbyte  ' Voltage control pot for V1
DSpots2 VAR     DSpots.highbyte' Voltage control pot for V2
Analog1 VAR     WORD            ' Analog working register
Analog2 VAR     WORD            ' Analog working register
AD_in1  VAR     BYTE            ' Results from A/D read of V1
AD_in2  VAR     BYTE            ' Results from A/D read of V2
Working VAR     WORD            ' Working register f

'
'LED pin assignment
LED_control     CON     7       'Out of regulation LED indicator

'
'**************************************************************************
' This routine initializes the BASIC Stamp, LCD, DS1267-010, and MEMKey.

' Initialize the BS2
BS2_ini:
        DirH = %01111111        ' set pins 8-15 direction
        OutH = %00000000        ' clear the pins
        DirL = %11110010        ' set pins 0-7 direction
        OutL = %10110010
'Initialize digital potentiometers
        DSpots = $FFFF ' Initialize V1 and V2 to 3V each
        GOSUB SetPotValue
'
' Initialize the LCD (Hitachi HD44780 controller)
LCD_ini:
        OutC = %0011            ' 8-bit mode
        PULSOUT E, 1
```

```
        PAUSE 5
        PULSOUT E, 1
        PULSOUT E, 1
        OutC = %0010            ' 4-bit mode
        PULSOUT E, 1
        Char = 40               ' Set for 2 line operation
        GOSUB LCDcmd
        Char = 12               ' Shift cursor right
        GOSUB LCDcmd
        Char = 6                ' Increment DDRAM after write
        GOSUB LCDcmd
        Char = 1                ' Clear LCD screen
        GOSUB LCDcmd

'
' Initialize the MEMKey
MEMKey_ini:
        HIGH   FM               ' Make sure FM is high
        PAUSE 2000              ' Let the system power settle
        SEROUT FM,Baud,[PConfig,Config]
                                ' Configure MEMKey for Polled Mode
        PAUSE  15               ' Pause 10ms for EEPROM access
        SEROUT FM,Baud,[PDBounce,DBounce]
                                ' Program debounce value
        PAUSE  15               ' Pause 10ms for EEPROM access
'       GOSUB  Reset            ' Run this when using a new MEMKey
'
'
'Initialize output voltage and display
        GOSUB  DisplayLCD       ' Recall display
        GOSUB  UpdateSupply     ' Modify output voltages
'*************************************************************************
MainProgram:
        If IN2 <> 1 Then MainProgram ' Check the KEY pin for a logic high
        GOSUB  KeyFind' Poll for a key press
        GOSUB  ModeSelect       ' Start user interface interaction
        GOSUB  UpdateSupply
        GOTO   MainProgram
'
'
'************************************************************************
' -----[ Subroutines ]---------------------------------------------------
'************************************************************************
'LCD commands, such as address pointer, are sent via LCDcmd, characters
'are sent with the LCDwr routine. This routine and LCD initialization
'routines taken from a previous author's Stamp Applications code listing.
'I believe it was Jon Williams who wrote the original code.
'
LCDcmd:
        LOW RS                  ' Enter command mode
                                ' then write the character
```

```
LCDwr:
       OutC = Char.HIGHNIB   ' Output high nibble
       PULSOUT E, 1          ' Strobe the Enable line
       OutC = Char.LOWNIB     ' Output low nibble
       PULSOUT E, 1
       HIGH RS               ' Return to character mode
       RETURN
'***********************************************************************
'All display data including voltage levels is stored in the MEMkey's
'EEPROM in locations $00-$0F. Whenever the display is updated each
'character is read from EEPROM and then sent to the LCD. The leading zeros
'for the voltage evels one both line one and line two are displayed as
'ASCII " "(space). This makes the display look a little nicer. If this
'display update routine is too slow for your design you can just update
'the voltage levels.
'
DisplayLCD:
Line1:
       Char = $80            ' Display line one
       GOSUB  LCDcmd
       For Index     = $00 to $07
              SEROUT FM,Baud,[Reeprom,Index]
                             ' Read EEPROM command
              SERIN     TM,Baud,[B_1]      ' Display value is read
              If Index <> $04 then TestAn1A
' Translate ASCII tens value into decimal
              Analog1 = (B_1 - 48) * 100
TestAn1A:
              If Index <> $05 then TestAn1B
' Translate ASCII ones value into decimal
              Analog1 = Analog1 + (B_1 -48 * 10)
TestAn1B:
              If Index <> $07 then TestAn1C
' Translate ASCII tenths value into decimal
              Analog1 = Analog1 + (B_1 -48)
TestAn1C:
              Char      = B_1
              If Index <> $04 then Continue1
' Test for leading zero
              If Char <> $30 then Continue1
' If ASCII zero then
              Char = " "            ' replace with blank space
Continue1:
              GOSUB LCDwr
       Next
Line2:
       Char   =          $C0        ' Display line two
       GOSUB  LCDcmd
       For Index = $08 to $0F
              SEROUT FM,Baud,[Reeprom,Index]
```

```
' Read EEPROM command
               SERIN    TM,Baud,[B_1]       ' Display value is read
               Char     = B_1
               If Index <> $0C then TestAn2A
' Translate ASCII tens value into decimal
               Analog2 = (B_1 - 48) * 100
TestAn2A:
               If Index <> $0D then TestAn2B
' Translate ASCII ones value into decimal
               Analog2 = Analog2 + (B_1 -48 * 10)
TestAn2B:
               If Index <> $0F then TestAn2C
' Translate ASCII tenths value into decimal
               Analog2 = Analog2 + (B_1 -48)
TestAn2C:
       If Index <> $0C then Continue2       ' Test for leading zero
               If Char <> $30 then Continue2' If ASCII zero then
               Char = " "     ' replace with blank space
Continue2:
               GOSUB LCDwr
               Next
       Analog1 = Analog1 * 256 / 200' Analog1 is the desired AD_in1 value
       Analog2 = Analog2 * 256 / 200' Analog2 is the desired AD_in2 value
       RETURN
'***********************************************************************
' The Reset subroutine returns the MEMKey to it's initial settings. It
'also resets the LCD display values. Upon initial power up of this design
'a "GOSUB Reset" should be placed prior to entering the MainProgram code
'space.After the EEPROM has been initialized the "GOSUB Reset" may be
'commented out or deleted. By doing this the last values displayed, prior
'to a power down, will be the values loaded on power up.
'
Reset:
       SEROUT FM,Baud,[Default]      'Reset MEMKey to default settings
       PAUSE   200
       SEROUT FM,Baud,[PConfig,Config] 'Configure MEMKey for Polled Mode

       PAUSE  15      ' Pause 10ms for EEPROM access
       SEROUT FM,Baud,[PDBounce,DBounce] ' Program debounce value
       PAUSE  15      ' Pause 10ms for EEPROM access

                     ' Update key values and display values
       For Index = $00 to $0F
LOOKUP Index,
[$00,$01,$02,$03,$04,$05,$06,$07,$08,$09,$0A,$0B,$0C,$0D,$0E,$0F],B_1
LOOKUP
Index,["1","2","3","A","4","5","6","B","7","8","9","C","*","0","#","D"],B_
2
LOOKUP Index,["V","1",":"," ","0","5",".","0","V","2",":","
","0","5",".","0"],B_3
               SEROUT FM,Baud,[Peeprom,B_1,B_3]
```

```
               PAUSE    15
               SEROUT FM,Baud,[Pkeyval,B_1,B_2]
               PAUSE    15
       Next
       RETURN
'************************************************************************
'ModeSelect determines which voltage is being adjusted, or if a reset
'command has been implemented. The scroll-up and scroll-down modes have
'not yet been implemented. The scroll ' functions can be defined once the
'voltage control method has been proved in the lab.a
'
ModeSelect:
       If KeyVal = "A" then AdjustV1
' Adjust voltage one has been selected
       If KeyVal = "B" then AdjustV2
' Adjust voltage two has been selected
       If KeyVal = "D" then ResetSupply
' A reset command has been entered
       RETURN
' Any key other than A,B, or D was pressed and can be ignored
AdjustV1:
       SEROUT FM,Baud,[Peeprom,$03,">"]     ' Load a ">" into EEPROM
       PAUSE  15
       GOSUB  DisplayLCD ' Display the ">" next to voltage being adjusted
       B_2 = $04          ' Start EEPROM address at voltage values
AdjV1Continue:
       GOSUB  KeyFind' Wait for the next keypress
       If KeyVal = "C" then AdjV1Done
' If a "C" was pressed then were done adjusting
       If KeyVal > "9" then AdjV1Over
' Check to if A,,B,C,D key was pressed by accident
       If KeyVal = "#" then AdjV1Over
' Check to see if pound key was preseed
       If KeyVal = "*" then AdjV1Over
' Check to see if asterisk key was pressed
       SEROUT FM,Baud,[Peeprom,B_2,KeyVal]
' Store numeric 0-9 in current EEPROM address
       PAUSE  15
       GOTO AdjV1Again
AdjV1Over:
       SEROUT FM,Baud,[Peeprom,B_2,"0"]
' If keypress was "A","B", or "D" return a "0"
       PAUSE  15
AdjV1Again:
       GOSUB DisplayLCD      ' Update display with the latest key press
       B_2 = B_2 + 1         ' Increment EEPROM address pointer
       If B_2 = $06 then AdjV1Again
' If EEPROM pointed at "." then increment again
       If B_2 = $08 then AdjustV1
' If line2 pointed at then reset EEPROM pointer
       GOTO   AdjV1Continue
```

```
AdjV1Done:
       GOSUB Limits   ' Make sure values are within limits
       RETURN
AdjustV2:
       SEROUT FM,Baud,[Peeprom,$0B,">"]
' Comments for second line are the same except all
       PAUSE  15      ' EEPROM pointers are 8 greater than line 1
       GOSUB  DisplayLCD
       B_2 = $0C
AdjV2Continue:
       GOSUB  KeyFind
       If KeyVal = "C" then AdjV2Done
       If KeyVal > "9" then AdjV2Over
       If KeyVal = "#" then AdjV2Over
       If KeyVal = "*" then AdjV2Over
       SEROUT FM,Baud,[Peeprom,B_2,KeyVal]
       PAUSE  15
       GOTO AdjV2Again
AdjV2Over:
       SEROUT FM,Baud,[Peeprom,B_2,"0"]
       PAUSE  15
AdjV2Again:
       GOSUB DisplayLCD
       B_2 = B_2 + 1
       If B_2 = $0E then AdjV2Again
       If B_2 = $10 then AdjustV2
       GOTO   AdjV2Continue
AdjV2Done:
       GOSUB Limits
       RETURN
ResetSupply:
       GOSUB  Reset   ' Implement reset command
       GOSUB DisplayLCD      ' Update display
       RETURN
'*********************************************************************
'Keyfind looks for a logic high on the KEY pin (IN2). If one is present
'then the Read Key Buffer command for the MEMKey is implemented. There is
'enough RAM allotted in the SERIN command to read a maximum buffer size of
'8 bytes. It is likely that only one key value will be returned since the
'typematic rate has been truned off in the MEMKey. The SERIN "escape
'clause" has been set to 50ms to ensure that the serial communication does
'not hang up the program. Once this subroutine is entered it will not
'exit until a key has been pressed. When it does exit the key value will
'be loaded into the KeyVal variable.
'
KeyFind:
              If IN2 <> 1 Then KeyFind
' Check the KEY pin for a logic high
              SEROUT FM,Baud,[Rbuffer]
' Read buffer command
              SERIN
```

```
TM,Baud,50,DoneBuffer,[KeyVal,B_3,B_4,B_5,B_6,B_7,B_8,B_9]
DoneBuffer:
               SEROUT FM,Baud,[Rbuffer]
' If there is a fast key press don't accept it
               PAUSE    40
               If IN2 <> 0 then DoneBuffer
' Wait for KEY pin to go to logic low
               RETURN
'*********************************************************************
' The Limits subroutine checks entered values for out of range or mis-
'keyed entries. If either voltage input is greater than 10.0 or less than
'03.0 then the values are forced to either the minimum or maximum values
'accepted. Prior to this routine being exited the ">" character
'that was loaded next to the adjusted voltage is replaced with a space
'character.
'
Limits:
               SEROUT FM,Baud,[Reeprom,$04]
' Read tens character of voltage on line 1
               SERIN TM,Baud,[B_1]
               If B_1 > "0" then ZeroOnesV1
' Check to see if tens is greater than "1"
               If B_1 = "0" then TestOnesV1
' Check to see if tens is a zero
               GOTO    NextLimit
ZeroOnesV1:
               SEROUT FM,Baud,[Peeprom,$04,"1"]
' If tens is greater than "1" then force display
               PAUSE    15          ' to "10.0". The "." is not adjusted
               SEROUT FM,Baud,[Peeprom,$05,"0"]
               PAUSE    15
               SEROUT FM,Baud,[Peeprom,$07,"0"]
               PAUSE    15
               GOTO NextLimit
TestOnesV1:
               SEROUT FM,Baud,[Reeprom,$05]
' If tens is "0" the read ones character
               SERIN TM,Baud,[B_1]
               If B_1 > "2" then NextLimit
' Check to see if ones is less than "3"
               SEROUT FM,Baud,[Peeprom,$05,"3"]
' If so then force display to "03.0"
               PAUSE    15
               SEROUT FM,Baud,[Peeprom,$07,"0"]
               PAUSE    15
NextLimit:
               SEROUT FM,Baud,[Reeprom,$0C]
' Test voltage on line 2 as line one was tested
               SERIN TM,Baud,[B_1]
               If B_1 > "0" then ZeroOnesV2
               If B_1 = "0" then TestOnesV2
```

```
              GOTO    DoneLimit
ZeroOnesV2:
              SEROUT FM,Baud,[Peeprom,$0C,"1"]
              PAUSE    15
              SEROUT FM,Baud,[Peeprom,$0D,"0"]
              PAUSE    15
              SEROUT FM,Baud,[Peeprom,$0F,"0"]
              PAUSE    15
              GOTO    DoneLimit
TestOnesV2:
              SEROUT FM,Baud,[Reeprom,$0D]
              SERIN TM,Baud,[B_1]
              If B_1 > "2" then DoneLimit
              SEROUT FM,Baud,[Peeprom,$0D,"3"]
              PAUSE    15
              SEROUT FM,Baud,[Peeprom,$0F,"0"]
              PAUSE    15
DoneLimit:
              SEROUT FM,Baud,[Peeprom,$03," "]
' Replace either ">" with a " "(space)
              PAUSE    15
              SEROUT FM,Baud,[Peeprom,$0B," "]
              PAUSE    15
              GOSUB    DisplayLCD
              RETURN
'*********************************************************************
' The UpdateSupply routine uses the Analog1 and Analog2 values to
'approximate the appropriate digital potentiometer settings. Once the
'potentiometer is adjusted a For Next loop is used to trim the output
'voltages. If the desired A/D measurement stored in the Analog register
'matches the actual A/D reading in the AD_in register the no adjustments
'to the digital pot are made. Otherwise the potentiometer setting is
'incremented or decremented to trim the voltage until the A/D reading is
'in line with the desired value. Trimming is attempted 10 times, with 50ms
'allotted for settling time between adjustments. I found that this system
'lacked resolution at voltages above about 8.5V. For this reason if the
'digital pot setting is less than 20 I settle for a an A/D match within 2
'bits of the desired value (± 160mV). If, after trimming is attempted, the
'desired A/D value is not reached then the program jumps to
'an out of regulation section. This portion of the code updates the
'display to the actual voltage output, and lights the "out of regulation"
'LED to inform the user that the output voltage has been adjusted.
'
UpdateSupply:
       LOW    LED_control
       DSpots1 = (2376/(Analog1 - 28)) -14
' Equation to estimate the desired pot setting(V1)
       DSpots2 = (2304/(Analog2 - 28)) -14
' Equation to estimate the desired pot setting(V2)
       GOSUB  SetPotValue           ' Update pot settings
'
```

```
TrimV1:
       For Index = 1 to 10          ' Trim voltage 10 times
              GOSUB  ReadAnalogs
              If AD_in1 = Analog1 then DoneV1
' If on first 7 trims then adj. by 1s
              If Index < 8 then NoMinV1
              If DSpots1 > 20 then NoMinV1
' If pot setting > 20 then adj. by 1s
              If AD_in1 = (Analog1+2) then DoneV1
' Otherwise accept +/- 2 bits as accurate
              If AD_in1 = (Analog1-2) then DoneV1
NoMinV1:
              If AD_in1 > Analog1 then IncDSpots1
' Reduce output voltage
              DSpots1 = DSpots1 - 1
              GOTO   DoneV1
IncDSpots1:
              DSpots1 = DSpots1 + 1
' Increase output voltage

DoneV1:
              If AD_in2 = Analog2 then DoneV2
' V2 follows same adjustment rules as V1
              If Index < 8 then NoMinV2
              If DSpots2 > 20 then NoMinV2
              If AD_in2 = (Analog2+2) then DoneV2
              If AD_in2 = (Analog2-2) then DoneV2
NoMinV2:
              If AD_in2 > Analog2 then IncDSpots2
              DSpots2 = DSpots2 - 1
              GOTO   DoneV2
IncDSpots2:
              DSpots2 = DSpots2 + 1

DoneV2:
              GOSUB  SetPotValue    ' Update pot settings after trimming
              Pause  50             ' Allow 50ms for voltages to settle
       NEXT                         ' Trim up to 10 times
'
       GOSUB  ReadAnalogs           ' Read in analog voltages
       If AD_in1 = Analog1 then TestOutOfReg2
' Find the differences between desired
       Working = Analog1 - AD_in1   ' and actual A/D readings
       If Working < 3 then TestOutOfReg2    ' If < 3 then all OK
       Working = ~Working
' If > 3 see if AD_in was > Analog by complimenting
       If Working < 3 then TestOutOfReg2
' If compliment < 3 then all OK
       GOTO OutOfReg1               ' Not OK so out of regulation
TestOutOfReg2:
       If AD_in2 = Analog2 then DoneOOR2
```

```
' Test analog and AD_in 2 for settings
       Working = Analog2 - AD_in2
       If Working < 2 then DoneOOR2
       Working = ~Working
       If Working < 2 then DoneOOR2
       GOTO OutOfReg2
DoneOOR2:
       GOSUB  DisplayLCD            ' Update display values
       RETURN
'
OutOfReg1:
       HIGH   LED_control                  ' Light out of range LED
       If AD_in1 < 129 then NoChangeAD_in1  ' Make sure A/D is under 10V
       AD_in1 = 128
NoChangeAD_in1:
       Working = (AD_in1 * 200) / 256
' Translate actual A/D value to ASCII
       B_1 = (Working / 100) + 48
' Calculate ASCII hundreds digit from AD meas.
       SEROUT FM,Baud,[Peeprom,$04,B_1]
' Program MEMKey EEPROM with new value
       PAUSE  15
       B_2 = (Working / 10)
' Calculate ASCII tens digit from AD meas.
       If B_2 < 10 then SkipSub1
       B_2 = B_2 - 10
SkipSub1:
       B_2 = B_2 + 48
       SEROUT FM,Baud,[Peeprom,$05,B_2]
' Program MEMKey EEPROM with new value
       PAUSE  15
       If Working < 100 then SkipSub2
       Working = Working - 100
SkipSub2:
'
       B_3 = (Working - ((Working/10)*10)) + 48
' Calculate ASCII ones digit from AD meas.
       SEROUT FM,Baud,[Peeprom,$07,B_3]
' Program MEMKey EEPROM with new value
       PAUSE  15
       GOTO TestOutOfReg2

OutOfReg2:
       If AD_in2 < 129 then NoChangeAD_in2  ' Update V2 as V1 was updated
       AD_in2 = 128
NoChangeAD_in2:
       Working = (AD_in2 * 200) / 256
       B_1 = (Working / 100) + 48
       SEROUT FM,Baud,[Peeprom,$0C,B_1]
       PAUSE  15
       B_2 = (Working / 10)
```

```
        If B_2 < 10 then SkipSub3
        B_2 = B_2 - 10
SkipSub3:
        B_2 = B_2 + 48
        SEROUT FM,Baud,[Peeprom,$0D,B_2]
        PAUSE   15
        If Working < 100 then SkipSub4
        Working = Working - 100
SkipSub4:

        B_3 = (Working - ((Working/10)*10)) + 48
        SEROUT FM,Baud,[Peeprom,$0F,B_3]
        PAUSE   15
        GOTO DoneOOR2
'
'*************************************************************************
' The SetPotValue shifts the pots settings out to the DS1267-010.
'
SetPotValue:
        HIGH CS_Pot
        PULSOUT Clk,10
        SHIFTOUT Dat,Clk,msbfirst,[DSpots\16]
        LOW CS_Pot
        PAUSE   10
        RETURN
'*************************************************************************
' The ReadAnalogs routine reads in the actual output voltages from the
'ADC0831 8 bit A/Ds.
'
ReadAnalogs:
        LOW CS_ad1
        SHIFTIN Dat,Clk,2,[AD_in1\9]
        HIGH    CS_ad1
'
        LOW CS_ad2
        SHIFTIN Dat,Clk,2,[AD_in2\9]
        HIGH    CS_ad2
        RETURN
'*************************************************************************
END:
'
```