Using the STK-500 with a four-digit seven segment common anode LED display to program the ATtiny26 ADC

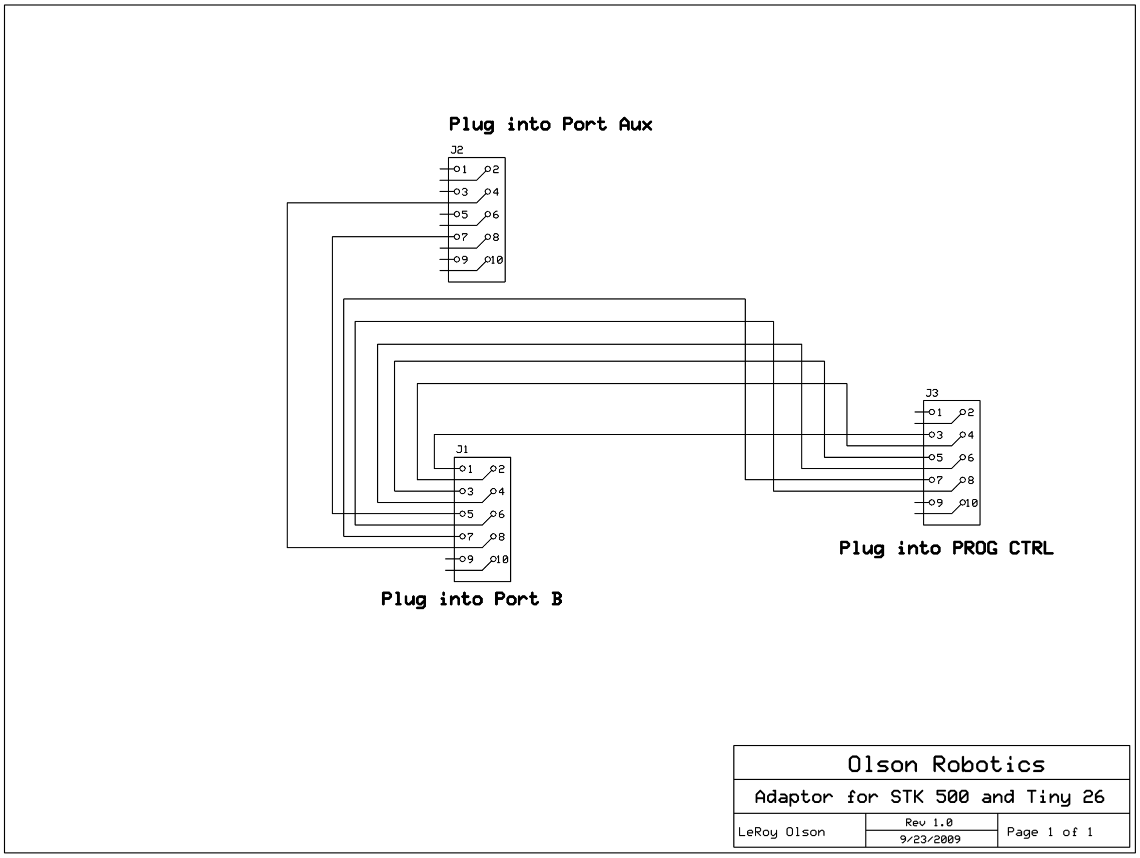
This document is an account of my re introduction into AVR programming. Several years ago I had worked with the AVR but then went onto other things. Recently my interest in the AVR was rekindled and I decided that I wanted to build a voltmeter using the Tiny26 after reading an article in AVR Freaks about the Mega Meter. I had an old STK-500 board collecting dust, I pulled the programmer out made sure it worked, I then installed the newest version of AVR Studio and WIN Avr after which I installed the update on my STK-500 board. I was now ready to go until I looked in the manual to setup the STK-500 for programming the Attiny26. That's when I discovered that you had to make a special jumper in order to program the chip and also that some of the jumper pin connections were setup different. I also realized that I needed a display to use in my project to display the results. After some deep thought and looking at the Electronic Gold mine catalog online I seen the LTC-5851Y 4 digit 7-segment common anode display on sale so I order six of them (I ordered extra just incase along with some other stuff I might need). I then designed the circuit board layout etched my board and soldered together my display that I could easily plug into the STK-500 board.

The first couple of pages show the jumper plug and the settings for the STK-500 board, and the following pages show the schematic and connectors for the LED display.
Next are pictures of the STK-500 board with LED operating, and program running on Attiny26. At the end of this document I have the source code written with Micro C Pro. I have also included the hex file within this zip file.

I started this endeavor using assembly language then went to Win Avr but I had trouble finding example code, there were many hex files available but little source code. I then stumbled across Mikro C on the Internet. I checked it out and there documentation looked good and included some circuit examples. I gave it a try and have come to like it although I prefer AVR Studios debugger. I still must use AVR Studio to program my chip, as Mikro C has no interface for the STK-500 board. To do this I just keep AVR Studio minimized until I need it then load the code I want into the programmer and write it to the Tiny26.
The circuits and board layout were drawn with Express PCB, which is available on the Internet free.

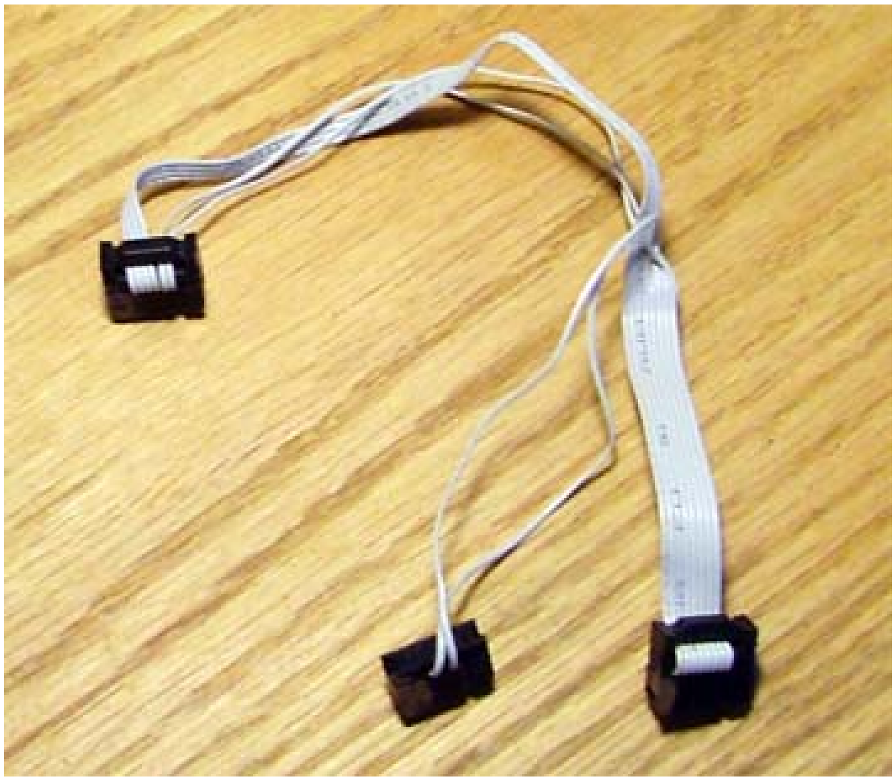
Below is a schematic of the jumper needed to program the tiny 26 with the STK-500 programmer board. In making the header I used old 10 pin serial connectors from computers by carefully removing the header clips with an Ex-acto knife then connecting them as in the drawing. I also marked each socket on the side facing the front of the STK-500 board so that they would all be facing the front of the board when plugged in properly.

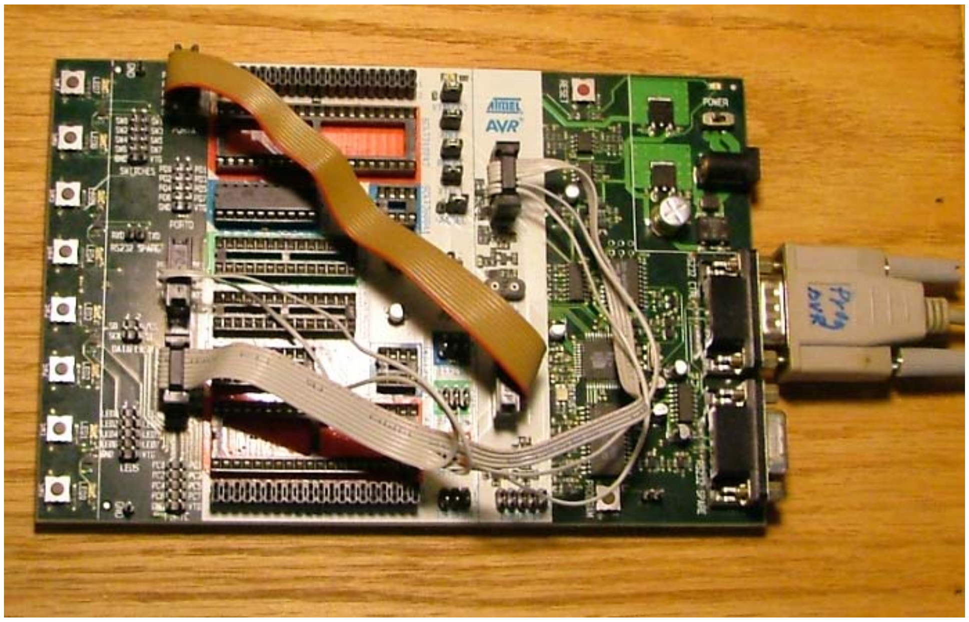
Schematic of modified header cable to program Attiny26 with STK 500 board



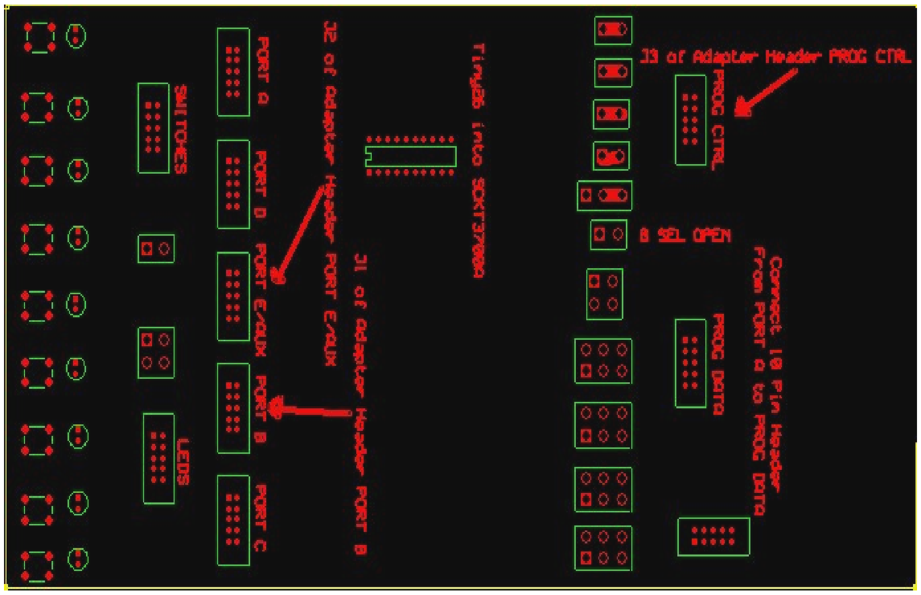
J1 of the header is plugged into the Port B socket; J2 is plugged into PORTE/AUX, and J3 into PROG CTRL. Also one of the ten-pin header cables is plugged into PORTA and the other end is plugged into PROG DATA, Make sure to maintain correct polarity.

Here is a picture of how the finished header should look. On the left side of each header you can see where I marked them with nail polish to help assure I plugged them in correctly.





On the Right the STK-500 board with cables and Tiny26 plugged in, all the jumpers set up, and ready to program.


SWITCHES
LEDS
PORT A
PORT D
PORT E/AUX
PORT B
PORT C
J2 of Adapter Header PORT E/AUX
J1 of Adapter Header PORT B
Tiny26 into SCKT3700D1
J3 of Adapter Header PROG CTRL
PROG CTRL
B SEL OPEN
PROG DATA
Connect 10 Pin Header from PORT A to PROG DATA

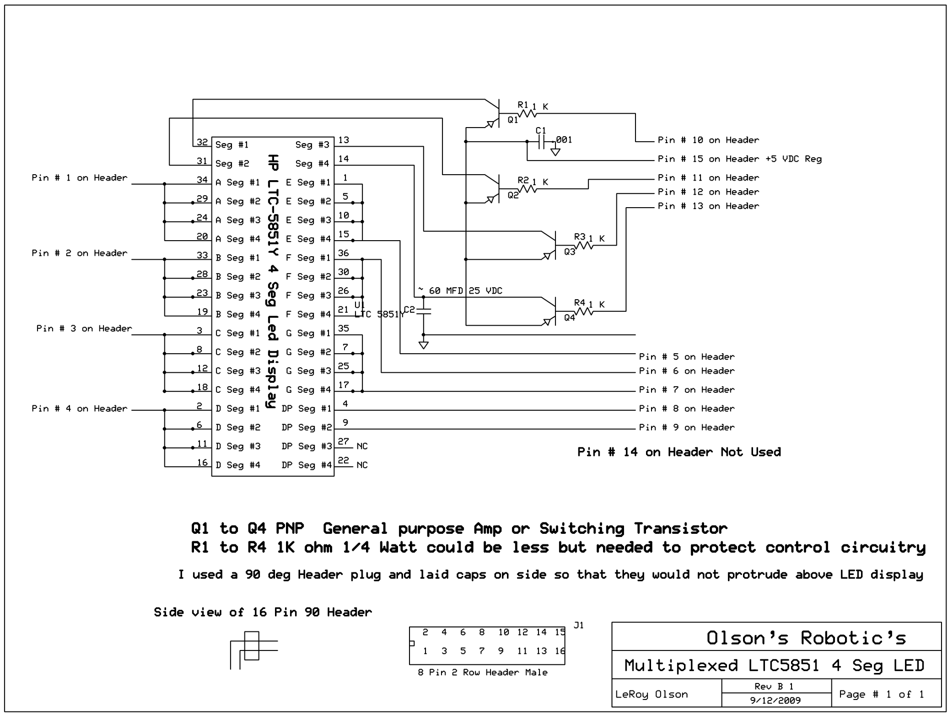
On the Left is a drawing of the STK-500 with proper jumper connection for programming Tiny26.

The LTC-5851Y circuit and board.

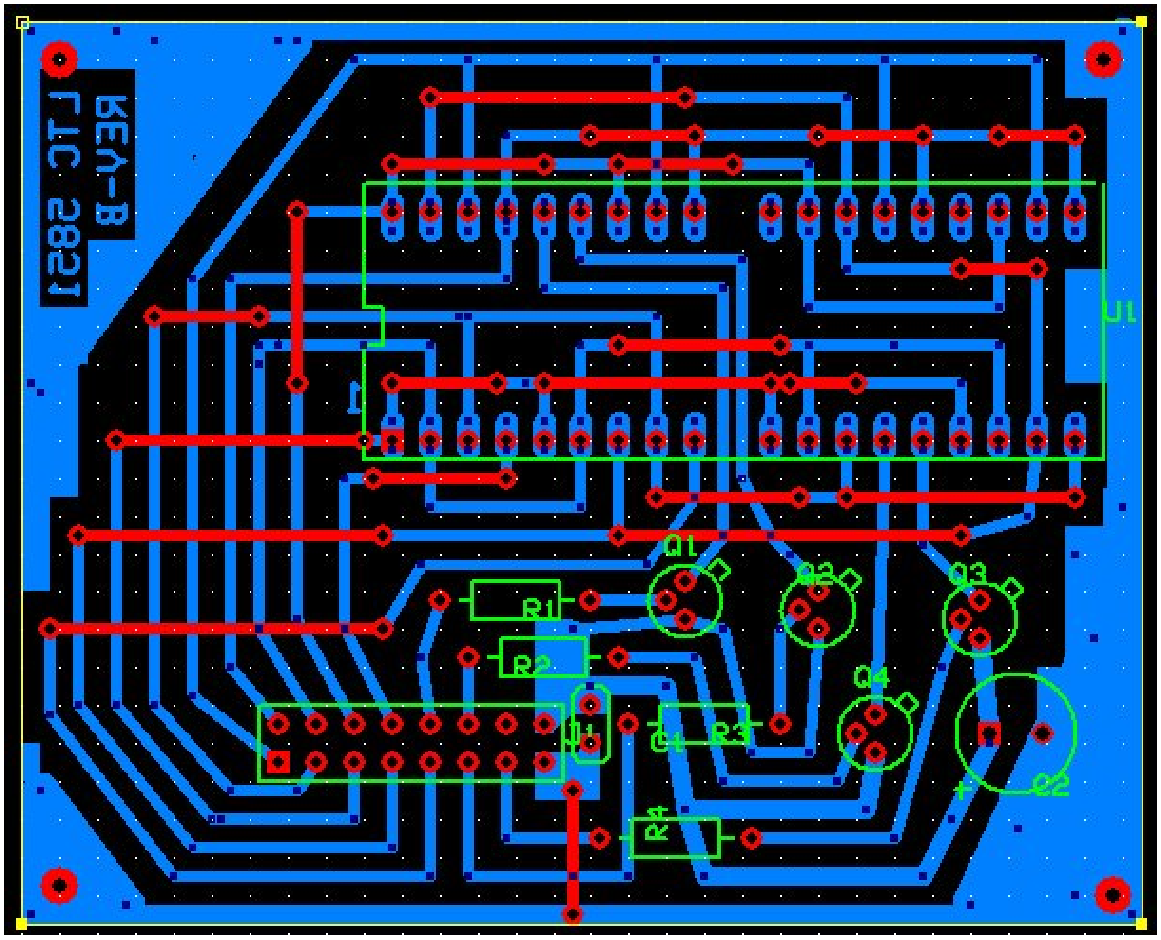
This drawing is of the schematic and pin layout for the LTC-5851Y LED display I used with the STK-500

This schematic was used to build the board; I later replaced Q1 through Q4 PNP tranistors with 2N5306 Darlington NPN transistors, connecting the collector to the 5 volt supply and the emiter to the positive terminal for each digit on the LED. I then changed R1 through R4 to 3.7K resistors.

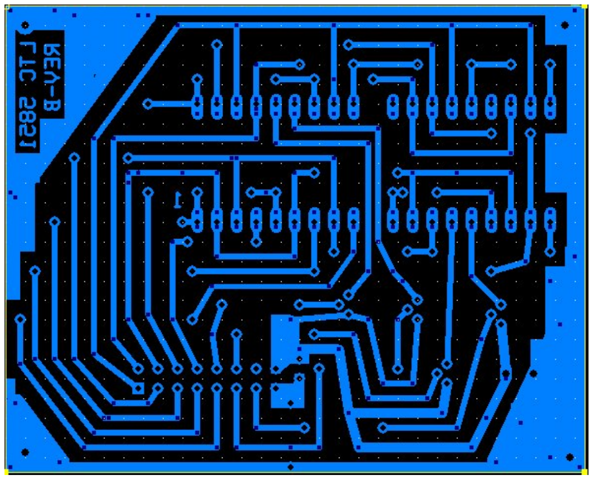
Pins 1 to 8 are wired into a 10-pin header, and pins 10 to 13 into a 4-pin header. I then connected the ground to the STK-500 and the 5 VDC to another supply as they dray near ½ amp and I did not want to overload the STK-500.



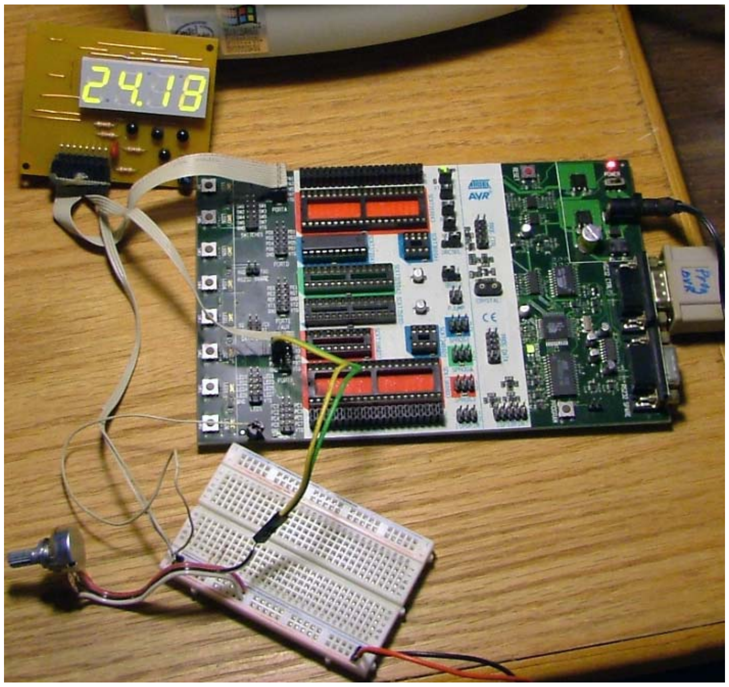
Below is the LTC-5851 circuit board layout. Blue represents the bottom traces, Red represents the top traces or in my case the jumper wires. Green are the parts. The square pin is #1.



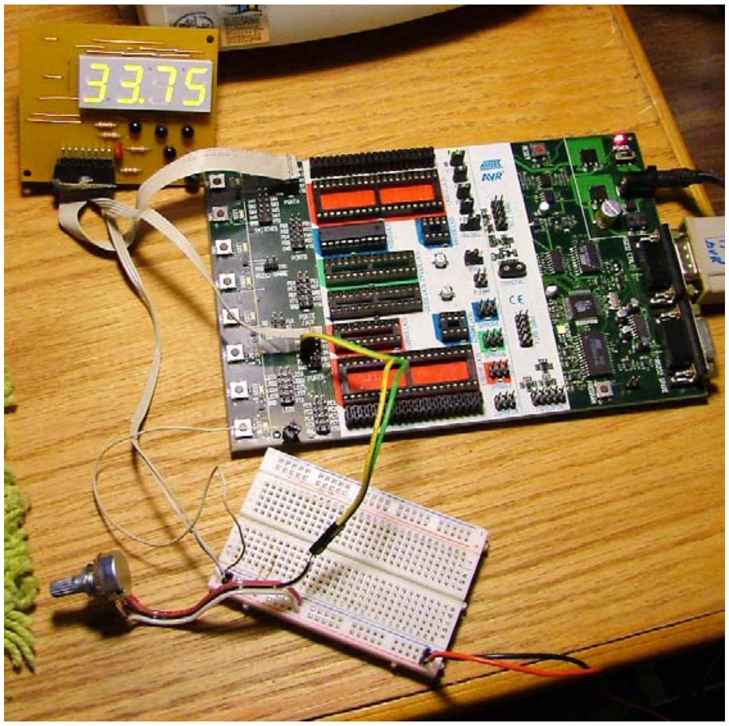
Below is the foil trace for the bottom of the LED display board. This layout was drawn with Express PCB which is free and available on the internet.



Here are pictures of the program running on the STK500 board with the LED display, a variable resistor connected on one side to ground and the other side to VCC with the center connection connected to pin 7 of the Tiny26. In these two pictures please note that the decimal point is on but it has no meaning and is only on because I did not disable it. The actual read out is 2.418 in picture on left and 3.375 in the lower picture.



In these two pictures you can see how I connected my 7 segment plug to Port A and my 4 digit plug to Port B. For these plugs I used old hard drive headers I then cut them to size with my dremmel tool. This makes it convenient to unplug when programming the chip and they can also be used for other port configurations with this board



The C source Code

When programming the fuse bits I used 8 Mhz internal osc with 64 cycle startup delay. The internal oscillator could be 4 Mhz also but with the delay set to any other value then stated above I had problems with the program starting unless I would ground the reset pin in which case the program would then start. Below is the C source code for Attiny26 ADC, this was compiled with Mikro C into the included hex file. I have made some modifications to this code for my finished project and I am sure there are more changes that could be implemented to make the code more efficient and compact, but this was the working code for my original project.

```
/*******************************************************************************************
** Routine to read ADC and display voltage as XXXX on 4 digit common anode 7 seg LED display **
****** Using ATtiny26 chip and LTC 5851Y LED package from Electronic Goldmine. ***************
********************************** Written by LeRoy Olson *************************************
************************************* September 22 2009 ****************************************
***** Written and compiled with Mikro Pro C and programmed with AVR Studio and STK - 500 ******
********************** Program Name T26-ADC.C    Version 1.00 Rev 2A ************************
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
****************** The 7 Segments A through G are parallel and connected to pins **************
*** PAO to 6. A to 0, B to 2, CV to 3, etc. The digits 1 to 4 are driven by PNP transistors ***
*************** and connected to PBO through PB3. The input is PB4 (ADC7) *********************
This displays reasonably accurate readings considering what is designed for it is very good. *
*********************************************************************************************/

#include <built_in.h>

unsigned short dmask(unsigned short dnum);
void Digit_Select();
unsigned short lmask(unsigned short lnum);

//float fnumber;  // used to store value from adc read
unsigned short digit, scount, dcount; //dcount used to track which digit to display 1000, 100, 10, or 1
unsigned short digt_array[4]; //digt_array stores the value of the four digit number
int number;//unsigned int number;


void main() {

 DDRA = 0x7F;  //Set POA0 to POA6 as outputs
 DDRB = 0x0F;   // Set POB0 to POB3 as outputs
 PORTA = 0xFF;   // Turn off PORTA Display
 PORTB = 0x0F;  // Turn off PORTB Display

 do
  {
  float fnumber;
   fnumber = ADC_Read(7);    // get ADC value from ADC chanel 7 Pin# 7
   number = (fnumber * 5.2); // Multiply by voltage applied to AVCC and save as int
   digit = number / 1000;  //extract 1000 s digit
   digt_array[3]= lmask(digit);//store in digit_array
   digit = (number / 100) %10; //extract 100s digit
   digt_array[2] = lmask(digit);
   digit = (number / 10) %10;  //extract 10s digit
   digt_array[1] = lmask(digit);
   digit = number %10;         // extract 1s digit
   digt_array[0] = lmask(digit);

   Delay_ms (1);

   Digit_Select(); // jump to display numbers on Led display
  }
   while(1);
}
```

```
void Digit_Select()
{
 dcount = dmask(dcount);
 PORTB = dcount; // Turn on appropiate led digit 1 to 4
 PORTA = digt_array[scount];  // Display number for approiate digit

 scount ++;
 if (scount > 3u)
    scount = 0;
 dcount = scount;
}

 unsigned short dmask(unsigned short dnum) // turn off all but selected digit
 {
  switch (dnum)
         {
         case 0: return 0x07;
         case 1: return 0x0B;
         case 2: return 0x0D;
         case 3: return 0x0E;
         }
 }


  unsigned short lmask(unsigned short lnum) // setup led to matrix display a number 0-9
{
 switch (lnum)
 {
  case 0: return 0xC0; // display zero
  case 1: return 0xF9; //display one
  case 2: return 0xA4; //display two
  case 3: return 0xB0; //display three
  case 4: return 0x99; //display four
  case 5: return 0x92; //display five
  case 6: return 0x82; //display six
  case 7: return 0xF8; //display seven
  case 8: return 0x80; //display eight
  case 9: return 0x98; //display nine
  default: return 0x86; // display error E
  }
  //end case
}
```

In closing I have discovered many uses for this chip, such as a voltmeter, amp meter, or an RPM meter with the proper input configuration. All of which could be used for single phase monitoring or three phase monitoring of voltage, current, or speed with warning indicators or alarms that could also shut the equipment down if a value was exceeded. Its amazing that one little 20 pin chip with the proper program can do so much and yet even much much more with the proper program and hardware.

My current project is to incorporate this into a 5 ¼ X 5 ¼ X 2 inch high box with 4 digit read out, three inputs, and J-Tag plug for in circuit programming with an over voltage or current alarm and equipment shutdown and reset.

I hope that this document will be of help to others just starting with AVR programming. Thank You LeRoy Olson. October 1 2009.