Updated: September 25, 2015

EE 354 **September 16, 2016**

C Sample Programs

```
//DataArray
/* This program creates an array of data in code memory
 * that is 32 bytes long.  Fill this array with the ascii codes for
 * the capital letters plus the digits 0, 1, 2, 3, 4, 5, and 6.
 * Write a forever loop which inputs a byte from port 0 and uses
 * the lower 5 bits of this byte to access one of the array
 * elements and display it on P1.
*/
#include <t89c51ac2.h>
code unsigned char mArray[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
code unsigned char dArray[] = {0x1, 0x2, 0x3, 0x4, 0x5, 0x6};
void main()
  {unsigned char aIndex;
   while(1)
     {aIndex = P0 & 0x1F;
      P1 = mArray[aIndex];
     }
  }
```

```
/*Bits
   This program illustrates how to define bits in C and how to define
   special function registers (sfr).  Note that since this program
   defines its own sfr's we don't need an include file.
*/
unsigned char bdata x;   //sbit and bdata must be global
sbit bit0 = x ^ 0;       //bdata is bit data location 20h to 2fh
sbit bit7 = x ^ 7;       //  in internal data memory
sfr P0 = 0x80;           //special function registers for ports
sfr P1 = 0x90;           //  sfr range from 0x80 to 0xff.  These
sfr P2 = 0xA0;           //  definitions are normally part of
sfr P3 = 0xB0;           //  #include <reg51.h>
sbit P1_0 = P1^0;        //Can define sbits in ports
sbit P1_2 = P1^2;        //  Note that C uses the caret (^) instead
sbit P1_3 = P1^3;        //  of the assemblers period as in P1.0

void main()
  {bit y;                // y is not global and thus not know in any
subprograms
   while(1)
    {bit0 = 1;
       bit7 = 1;
       P0 = 0xff;          // sends byte data to the ports
       P2 = 0xaa;
       P1 = 0xff;
       P1_0 = 0;           // sends bit data to the ports
       P1_2 = 0;
     x = P1_0;           // port bit data to global bit data
     y = x;              // global bit data to local bit data
       P1_3 = y;
      }
  }
```

Updated: September 25, 2015

```
//LogicOps
/* Logical Operations
 *This example taken from the text (Stewart and Miao) p. 153.
 * It illustrates the use of bit operators to do sequential logic
 *
*/
#include<t89c51ac2.h>
//Global bit definitions using bits of P1
sbit x0 = P1^0;
sbit x1 = P1^1;
sbit x2 = P1^2;
sbit y0 = P1^3;
sbit y1 = P1^4;

void main()
  {while (1)
     {y0 = ~y1 & x0 & ~x1 & x2;    //~ is the inverse bit operator
        y1 = ~y0 & ~x0 & ~x1 & ~x2;  //& logically ANDS two bits
      }
  }
```

```
//LongInt.c
// This program inputs 4 bytes from P0, adds them up, and finds
// the average.  The average is converted to BCD and displayed
// on P2 and P1.  Long ints are used to preserve precision
// so that the display always gives 3 digits for the average
// in the form of ddd., dd.d, or d.dd
//Run this program on the simulator to see results
#include<t89c51ac2.h>
void Convert2BCD(long avg);
void main(void)
  {unsigned char w, x, y, z;
   long sum, avg;
   w = P0;           //Input 4 bytes as unsigned chars
   x = P0;
   y = P0;
   z = P0;
   sum = w + x + y + z;  //Add them up in a long;
   avg = 100*sum/4;      //*100 and divide by 4 for avg
                         //  this preserves precision
   Convert2BCD(avg);     //Convert to BCD and display
   while(1);
  }
//
void Convert2BCD(long avg)
  {unsigned char d0, d1, d2, d3, d4;
   d0 = avg % 10;  //Convert to BCD by using mod function
   avg = avg/10;   // Largest avg = 255*100 = 25500 so
   d1 = avg % 10;  // five digits
   avg = avg/10;
   d2 = avg % 10;
   avg = avg/10;
   d3 = avg % 10;
   avg = avg/10;
   d4 = avg %10;
   //P2 is most significant
   if(d4 != 0)     //If d4 is not 0 display d4,d3,d2
     {P2 = d4;     //  decimal at the end
      P1 = d2 + d3*16;
     }
   else if(d3 != 0) //If d4 = 0 but d3 != 0 then display
     {P2 = d3;      //  d3 d2.d1
      P1 = d1 + d2*16;
     }
   else             //if d4 = 0 and d3 = 0 then display
     {P2 = d2;      //  d3.d2,d1
      P1 = d0 + d1*16;
     }
  }
```

Updated: September 25, 2015

```
//Shift.c
//This program illustrates shifting and rotating.
//  It rotates two bits to the right for port 1.
#include <t89c51ac2.h>
void main()
  {unsigned char x;
   x = 3;
   while(1)
     {x = x << 1;    //Shift operator is <<
        if(x == 0x80)  //check if bit is going off the end
          x = 0x81;
        if(x == 2)
          x = 3;
        P1 = x;
       }
  }
```

```
//TimerInt.c
// This program produces a 50KHz square wave on P3.2 using
//  Timer 0 in the 8-bit autoreload mode.
//  50KHz corresponds to a 20usec period but interrupt complements
//  P3.2 at twice that rate so we need a 10 usec period for T0.
#include<t89c51ac2.h>
void SqWave();
void main(void)
   {CKCON = 0x01;   // x2 mode
    TMOD = 0x02;    //Timer 0 mode = not gated, internal clock, 8-bit, auto reload
    //For fosc = 29.4MHz in x2 mode timer is clocked at 29.4Mhz/6 = 4.9MHz so
    // period is 1/4.9Mhz = .20408 usec.  To get 10 usec we need 10/.20408 =
    // 49 counts.  256 - 49 = 205 = 0xcd.
    TH0 = 0xcd;     //Timer 0 high set to 205 for 10 micro-seconds
    TR0 = 1;        //Timer 0 run control bit in TCON
    ET0 = 1;        //Timer 0 interrupt enable
    EA = 1;         //Global interrupt enable
    while(1);
   }
//
//Timer 0 comes in on Interrupt 1.
void SqWave() interrupt 1 using 1
  {P3 = P3 ^ 4;  //Exclusive or with 00000100 for bit 3.2
  }
```

```
//TimerCC03Polled.c
// This program produces a square wave with a 2 msec high time and 1 msec low
//  time on P3.2 using timer 0 and timer 1 in a polled mode.
#include <at89c51cc03.h>
//For fosc = 28.2076MHz in x2 mode timer is clocked at 28.2076Mhz/6 =
4.701MHz
// so period is 1/4.701Mhz = .212708 usec.  To get 1 msec we need
// 1000/.212708 = 4701 counts.  65536 - 4701 = 60835 = 0xEDA3.
const char TH = 0xED;
const char TL = 0xA3;
void main(void)
   {CKCON = 0x01;   //double clocked
    CKCON &= 0xFD;
    CKCON |= 0x04;
    TMOD = 0x11;    //Timer 0 mode = not gated, internal clock, 16-bit mode
                    //Timer 1 mode = not gated, internal clock, 16-bit mode
    TH0 = TH;
    TL0 = TL;
    TH1 = TH;
    TL1 = TL;
    while(1)
      {TF0 = 0;
       TR0 = 1;         //Timer 0 run control bit in TCON
       P3 = P3 | 4;     //Set bit 3.2 to 1
       while(TF0 == 0); //Wait for time 0 to time out
       TR0 = 0;         //Turn off timer 0
       TH0 = TH;      //Reload timer 0
       TL0 = TL;
       TF1 = 0;
       TR1 = 1;         //Timer 1 run control bit in TCON
       P3 = P3 & 0xFB;  //Set bit 3.2 to 0
       while(TF1 == 0); //Wait for Timer 1 overflow
       TR1 = 0;         //Turn Timer 1 off
       TH1 = TH;      //Reset timer 1
       TL1 = TL;
      }
   }
```

```
//TimerInts.c
// This program produces a square wave with a 2 msec high time and 1 msec low
//  time on P3.2 using timer 0 and timer 1 in an interrupt mode.
#include<t89c51ac2.h>

void main(void)
   {CKCON = 0x01;   // x2 mode
    TMOD = 0x11;    //Timer 0 mode = not gated, internal clock, 16-bit mode
                    //Timer 1 mode = not gated, internal clock, 16-bit mode
    //For fosc = 29.4MHz in x2 mode timer is clocked at 29.4Mhz/6 = 4.9MHz so
    // period is 1/4.9Mhz = .20408 usec.  To get 1 msec we need 1000/.20408
    // = 4900 counts.  65536 - 4900 = 60636 = 0xECDC.  For 2 msec we need
    // 9800 counts.  65536 - 9800 = 55736 = 0xD9B8.

    TH0 = 0xD9;     //Timer 0 set to D9B8 - 55736
    TL0 = 0xB8;

    TH1 = 0xEC;     //Timer 1 set to D8F0 = 60636
    TL1 = 0xDC;
    TR0 = 1;
    ET0 = 1;        //Timer 0 interrupt enable
    EA = 1;         //Global interrupt enable
    while(1);
   }
//
void T0Int() interrupt 1 using 1
  {P3 = P3 | 4;  //bit P3.2 to 1
   TR0 = 0;      //Turn timer 0 off
   TH1 = 0xD9;   //Timer 1 set to D9B8 = 55736
   TL1 = 0xB8;
   TR1 = 1;      //Turn timer 1 on
   ET0 = 0;      //Timer 0 interrupt off
   ET1 = 1;      //Timer 1 interrupt on
  }
//
void T1Int(void) interrupt 3 using 1
  {P3 = P3 & 0xB1;   //bit P3.1 to 0
   TR1 = 0;          //Turn timer 1 off
   TH0 = 0xEC;       //Timer 0 set to ECDC = 57536
   TL0 = 0xDC;
   TR0 = 1;          //Turn timer 0 on
   ET1 = 0;          //Timer 1 interrupt off
   ET0 = 1;          //Timer 0 interrupt on
  }
```

```
//SerialOut.c
/* Sends out a single line on the serial port using Timer 2 in a
    double clocked mode with a baud rate of 19,200.  The message
    ends in CRLF.

   This program is meant to work with either a C# program or using
   Hyperterminal on a standard PC reading the serial port.
*/
#include <t89c51ac2.h>
char msg [] = "Hello Mom!";
void main (void)
  {unsigned char i;
   CKCON = 0x01;      // x2 mode, T2 clock is same as crystal
   SCON = 0X40;       // Mode 2, 8 bit uart transmit only, uses T2
   RCLK=1;            // Turn on receive clock in T2CON
   TCLK=1;            // Turn on transmit clock in T2CON
   //Baud rate = fCrystal/(32*(65536 - (RCAP2H, RCAP2L))
   // If double clocked, multiply fCrystal by 2.
   RCAP2H=0XFF;       //19.2k baud @ 29.4 Mhz
   RCAP2L=0xA0;       //
   TR2=1;             // TCON bit to start Timer 2
   REN=0;             // Transmit only
   RI = 0;            // Clear the receive interrupt flag
   i = 0;
   while(msg[i] != 0) //Char string ends in 0
     {TI = 0;
      SBUF = msg[i];
      i++;
      while (TI == 0);   // Wait for write to be done
     }
   TI = 0;
   SBUF = 0x0D;       //CR
   while (TI == 0);   // Wait for write to be done
   TI = 0;
   SBUF = 0x0A;       //LF
   while (TI == 0);   // Wait for write to be done
  }
```

```
//SerialIO.c
/* This program works with either C# or Hyperterminal connected to the serial
   port.  It can be run with the simulator as well.  (Use View to pull up the
   serial window for simulation.)
   The program transmits "Ready!!!" on reset and then waits for characters
   to be received on the serial port at 19,200.  If the character is 1, 2,
   or 3 a bit is set on port 1.
*/
#include <t89c51ac2.h>
code char msg [] = "Ready!!\r\n";
unsigned char c;
void main (void)
  {unsigned char i;
   CKCON = 0x01;      // x2 mode, T2 clock is same as crystal
   SCON = 0X40;       // Mode 2, 8 bit uart transmit only, uses T2
   RCLK=1;            // Turn on receive clock in T2CON
   TCLK=1;            // Turn on transmit clock in T2CON
   //Baud rate = fCrystal/(32*(65536 - (RCAP2H, RCAP2L))
   // If double clocked, multiply fCrystal by 2.
   RCAP2H=0XFF;       //19.2k baud @ 29.4 Mhz
   RCAP2L=0xA0;       //
   TR2=1;             // TCON bit to start Timer 2
   REN=1;             // Receive and transmit
   RI = 0;            // Clear the receive interrupt flag
   EA = 1;            //Turn on global interrupt flag
   ES = 1;            //Turn no serial interrupt
   i = 0;
   while(msg[i] != 0) //Char string ends in 0
     {TI = 0;         // Send out initial message
      SBUF = msg[i];
      i++;
      while (TI == 0);   // Wait for write to be done
     }
   c = 0;
   while(c == 0);        //Wait for a character to come in
   while(1)              // strip off upper four bits of each
      {switch (c & 0x0F) // character and set appropriate bit
         {case 1:        // on P1 if 1, 2, or 3
             P1 = 1;
             break;
          case 2:
             P1 = 2;
             break;
          case 3:
             P1 = 4;
          default:
             break;
         }
      }
  }
```

```
//Serial interrupt that happens when character is received.
void SerialInt() interrupt 4 using 1
  {if(RI)                           //If receiving data
      {c = SBUF & 0x7F;             //strip off parity flag
       SBUF = c;                    //Echo
       while(TI == 0);              //Wait here until transmit complete
       RI = 0;                      //Turn off receive interrupt flag
       TI = 0;
       SBUF = 0x0D;       //CR
       while (TI == 0);   // Wait for write to be done
       TI = 0;
       SBUF = 0x0A;       //LF
       while (TI == 0);   // Wait for write to be done
       TI = 0;
      }
  }
```

```
//Code memory example
//Takes data stored in code memory and sends it to P2.  The data is in an
//  array called dots[].  The index into the array is taken from P1
#include<at89c51cc03.h>
 code unsigned char dots[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80,
                              0x03, 0x06, 0x0C, 0x18, 0x30, 0x60, 0xC0, 0x81,
                              0x81, 0x42, 0x24, 0x18, 0x18, 0x24, 0x42, 0x81
                             };
unsigned char flag = 0;
void ISR();

void main()
  {unsigned char indx, i;
   CKCON = 0x01; // x2 mode
   TMOD = 0x02; //Timer 0 mode = not gated, internal clock, 8-bit, auto reload
   //For fosc = 28.2076MHz in x2 mode timer is clocked at 28.2076Mhz/6 = 4.701MHz
   // so period is 1/4.701Mhz = .2127086 usec. To get 10 usec we need
   // 50/.2127608 = 235 counts. 256 - 235 = 21 = 0x15
   TH0 = 0x15; //Timer 0 high set to 209 for 10 micro-seconds
   TR0 = 1; //Timer 0 run control bit in TCON
   ET0 = 1; //Timer 0 interrupt enable
   EA = 1; //Global interrupt enable

   while(1)
     {indx = (P1 & 0x03)*8;
      for(i=indx;i<indx+8;i++)
         {P2 = dots[i];
          while(flag == 0);
          flag = 1;
         }
     }
  }

//Timer 0 comes in on Interrupt 1.
void ISR() interrupt 1 using 1
 {flag = 1;
 }
```

Updated: September 25, 2015

```
//Generates 10 random integer 1 to 6 and sends it to P1
#include<at89c51cc03.h>
#include <stdlib.h>
void main()
  {int i;
   int r;
   srand(345);   //This is the random number generator seed
       for (i = 0; i < 10; i++)
         {r = rand();
          P1 = (r % 6) + 1;  //Numbers 1, 2, .. 6
         }
  }
```