8051 programming in C

cuu duong than cong . com

Why program 8051 in C

- Compilers produce hex files that is downloaded to ROM of microcontroller
  - The size of hex file is the main concern
    - ✓ Microcontrollers have limited on-chip ROM
    - ✓ Code space for 8051 is limited to 64K bytes
- C programming is less time consuming, but has larger hex file size
- The reasons for writing programs in C
  - It is easier and less time consuming to write in C than Assembly
  - C is easier to modify and update
  - You can use code available in function libraries
  - C code is portable to other microcontroller with little of no modification

Data types

- A good understanding of C data types for 8051 can help programmers to create smaller hex files
  - Unsigned char
  - Signed char
  - Unsigned int
  - Signed int
  - Sbit (single bit)
  - Bit and sfr

Unsigned char

- The character data type is the most natural choice
  - 8051 is an 8-bit microcontroller
- Unsigned char is an 8-bit data type in the range of 0 – 255 (00 – FFH)
  - One of the most widely used data types for the 8051
    - ✓ Counter value
    - ✓ ASCII characters
- C compilers use the signed char as the default if we do not put the keyword unsigned

Write an 8051 C program to send values 00 – FF to port P1.

Solution:

```
#include <reg51.h>
void main(void)
   {
     unsigned char z;
     for (z=0;z<=255;z++)
        P1=z;
   }
```

1. Pay careful attention to the size of the data
2. Try to use unsigned *char* instead of *int* if possible

Write an 8051 C program to send hex values for ASCII characters of 0, 1, 2, 3, 4, 5, A, B, C, and D to port P1.

Solution:

```
#include <reg51.h>
void main(void)
   {
     unsigned char mynum[]="012345ABCD";
     unsigned char z;
     for (z=0;z<=10;z++)
        P1=mynum[z];
   }
```

Signed char

- The signed char is an 8-bit data type
  - Use the MSB D7 to represent – or +
  - Give us values from –128 to +127
- We should stick with the unsigned char unless the data needs to be represented as signed numbers
- Example: Write an 8051 C program to send values of –4 to +4 to port P1.


```
//Singed numbers
#include <reg51.h>
void main(void)
   {
     char mynum[]={+1,-1,+2,-2,+3,-3,+4,-4};
     unsigned char z;
     for (z=0;z<=8;z++)
        P1=mynum[z];
   }
```

Unsigned and Signed int

- The unsigned int is a 16-bit data type
  - Takes a value in the range of 0 to 65535 (0000 – FFFFH)
  - Define 16-bit variables such as memory addresses
  - Set counter values of more than 256
  - Since registers and memory accesses are in 8-bit chunks, the misuse of int variables will result in a larger hex file
- Signed int is a 16-bit data type
  - Use the MSB D15 to represent – or +
  - We have 15 bits for the magnitude of the number from –32768 to +32767

BIT and SFR

- The bit data type allows access to single bits of bit-addressable memory spaces 20 – 2FH
- To access the byte-size SFR registers, we use the sfr data type

Data Type	Size in Bits	Data Range/Usage
unsigned char	8-bit	0 to 255
(signed) char	8-bit	-128 to +127
unsigned int	16-bit	0 to 65535
(signed) int	16-bit	-32768 to +32767
sbit	1-bit	SFR bit-addressable only
bit	1-bit	RAM bit-addressable only
sfr	8-bit	RAM addresses 80 – FFH only

Write an 8051 C program to toggle bit D0 of the port P1 (P1.0) 50,000 times.

Solution:

```
#include <reg51.h>
sbit MYBIT=P1^0;

void main(void)
  {
    unsigned int z;
    for (z=0;z<=50000;z++)
       {
         MYBIT=0;
         MYBIT=1;
       }
  }
```

sbit keyword allows access to the single bits of the SFR registers

Time delay

- There are two way s to create a time delay in 8051 C
  - Using the 8051 timer
  - Using a simple for loop
- Be mindful of three factors that can affect the accuracy of the delay:
  - The 8051 design
    - ✓ The number of machine cycle
    - ✓ The number of clock periods per machine cycle
  - The crystal frequency connected to the X1 – X2 input pins
  - Compiler choice
    - ✓ C compiler converts the C statements and functions to Assembly language instructions
    - ✓ Different compilers produce different code

Time delay

Write an 8051 C program to toggle bits of P1 continuously forever with some delay.

Solution:

```
//Toggle P1 forever with some delay in between
//"on" and "off"
#include <reg51.h>
void main(void)
  {
    unsigned int x;
    for (;;)                    //repeat forever
      {
        p1=0x55;
        for (x=0;x<40000;x++); //delay size
                               //unknown
        p1=0xAA;
        for (x=0;x<40000;x++);
      }
  }
```

We must use the oscilloscope to measure the exact duration

Write an 8051 C program to toggle bits of P1 ports continuously with a 250 ms.

Solution:

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
   {
     while (1)                    //repeat forever
        {
          p1=0x55;
          MSDelay(250);
          p1=0xAA;
          MSDelay(250);
        }
   }

void MSDelay(unsigned int itime)
   {
     unsigned int i,j;
     for (i=0;i<itime;i++)
        for (j=0;j<1275;j++);
   }
```

I/O Programming

LEDs are connected to bits P1 and P2. Write an 8051 C program that shows the count from 0 to FFH (0000 0000 to 1111 1111 in binary) on the LEDs.

Solution:

```
#include <reg51.h>
#defind LED P2;

void main(void)
   {
     P1=00;              //clear P1
     LED=0;              //clear P2
     for (;;)            //repeat forever
        {
          P1++;          //increment P1
          LED++;         //increment P2
        }
   }
```

Ports P0 – P3 are byte-accessable and we use the P0 – P3 labels as defined in the 8051/52 header file.

I/O Programming

Write an 8051 C program to get a byte of data form P1, wait 1/2 second, and then send it to P2.

Solution:

```
#include <reg51.h>
void MSDelay(unsigned int);
                cuu duong than cong . com
void main(void)
  {
    unsigned char mybyte;
    P1=0xFF;                    //make P1 input port
    while (1)
       {      cuu duong than cong . com
         mybyte=P1;             //get a byte from P1
         MSDelay(500);
         P2=mybyte;             //send it to P2
       }
  }
```

I/O Programming

Write an 8051 C program to get a byte of data form P0. If it is less than 100, send it to P1; otherwise, send it to P2.

Solution:

```
#include <reg51.h>

void main(void)
  {
    unsigned char mybyte;
    P0=0xFF;                    //make P0 input port
    while (1)
       {
         mybyte=P0;             //get a byte from P0
         if (mybyte<100)
            P1=mybyte;          //send it to P1
         else
            P2=mybyte;          //send it to P2
       }
  }
```

I/O Programming

Write an 8051 C program to toggle only bit P2.4 continuously without disturbing the rest of the bits of P2.

Solution:

```
//Toggling an individual bit
#include <reg51.h>
sbit mybit=P2^4;

void main(void)
  {
    while (1)
       {
         mybit=1;              //turn on P2.4
         mybit=0;              //turn off P2.4
       }
  }
```

Ports P0 – P3 are bit-addressable and we use *sbit* data type to access a single bit of P0 - P3

Use the Px^y format, where x is the port 0, 1, 2, or 3 and y is the bit 0 – 7 of that port

I/O Programming

Write an 8051 C program to monitor bit P1.5. If it is high, send 55H to P0; otherwise, send AAH to P2.

Solution:

```
#include <reg51.h>
sbit mybit=P1^5;

void main(void)
  {
    mybit=1;                    //make mybit an input
    while (1)
       {
         if (mybit==1)
            P0=0x55;
         else
            P2=0xAA;
       }
  }
```

I/O Programming

Write an 8051 C program to toggle all the bits of P0, P1, and P2 continuously with a 250 ms delay. Use the sfr keyword to declare the port addresses.

Another way to access the SFR RAM space 80 – FFH is to use the *sfr* data type

Solution:

```
//Accessing Ports as SFRs using sfr data type
sfr P0=0x80;
sfr P1=0x90;
sfr P2=0xA0;
void MSDelay(unsigned int);

void main(void)
  {
    while (1)
       {
         P0=0x55;
         P1=0x55;
         P2=0x55;
         MSDelay(250);
         P0=0xAA;
         P1=0xAA;
         P2=0xAA;
         MSDelay(250);
       }
  }
```

I/O Programming

Write an 8051 C program to turn bit P1.5 on and off 50,000 times.

Solution:

```
sbit MYBIT=0x95;

void main(void)
  {
    unsigned int z;
    for (z=0;z<50000;z++)
       {
         MYBIT=1;
         MYBIT=0;
       }
  }
```

We can access a single bit of any SFR if we specify the bit address

Notice that there is no #include <reg51.h>. This allows us to access any byte of the SFR RAM space 80 – FFH. This is widely used for the new generation of 8051 microcontrollers.

I/O Programming

Write an 8051 C program to get the status of bit P1.0, save it, and send it to P2.7 continuously.

Solution:

```
#include <reg51.h>
sbit inbit=P1^0;
sbit outbit=P2^7;
bit membit;              //use bit to declare
                         //bit- addressable memory

void main(void)
  {
    while (1)
       {
         membit=inbit;     //get a bit from P1.0
         outbit=membit;    //send it to P2.7
       }
  }
```

We use bit data type to access data in a bit-addressable section of the data RAM space 20 – 2FH

Logic operations

- Logical operators :
  - AND (&&), OR (||), and NOT (!)
- Bit-wise operators :
  - AND (&), OR (|), EX-OR (^), Inverter (~), Shift Right (>>), and Shift Left (<<)
    - ✓ These operators are widely used in software engineering for embedded systems and control

Bit-wise Logic Operators for C

		AND	OR	EX-OR	Inverter
A	B	A&B	A|B	A^B	~B
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	
1	1	1	1	0	

Logic Operations

```
#include <reg51.h>

void main(void)
  {
    P0=0x35 & 0x0F;          //ANDing
    P1=0x04 | 0x68;          //ORing
    P2=0x54 ^ 0x78;          //XORing
    P0=~0x55;                //inversing
    P1=0x9A >> 3;            //shifting right 3
    P2=0x77 >> 4;            //shifting right 4
    P0=0x6 << 4;             //shifting left 4
  }
```

Logic Operations

Write an 8051 C program to toggle all the bits of P0 and P2 continuously with a 250 ms delay. Using the inverting and Ex-OR operators, respectively.

Solution:

```
#include <reg51.h>
void MSDelay(unsigned int);

void main(void)
   {
     P0=0x55;
     P2=0x55;
     while (1)
        {
          P0=~P0;
          P2=P2^0xFF;
          MSDelay(250);
        }
   }
```

Logic Operations

Write an 8051 C program to get bit P1.0 and send it to P2.7 after inverting it.

Solution:

```
#include <reg51.h>
sbit inbit=P1^0;
sbit outbit=P2^7;
bit membit;

void main(void)
  {
    while (1)
       {
         membit=inbit;    //get a bit from P1.0
         outbit=~membit; //invert it and send
                          //it to P2.7
       }
  }
```

Data conversion

Write an 8051 C program to convert 11111101 (FD hex) to decimal and display the digits on P0, P1 and P2.

Solution:

```
#include <reg51.h>

void main(void)
  {
    unsigned char x,binbyte,d1,d2,d3;
    binbyte=0xFD;
    x=binbyte/10;
    d1=binbyte%10;
    d2=x%10;
    d3=x/10;
    P0=d1;
    P1=d2;
    P2=d3;
  }
```

RAM data

- The 8051 C compiler allocates RAM locations
  - Bank 0 – addresses 0 – 7
  - Individual variables – addresses 08 and beyond
  - Array elements – addresses right after variables
    - ✓ Array elements need contiguous RAM locations and that limits the size of the array due to the fact that we have only 128 bytes of RAM for everything
  - Stack – addresses right after array elements

RAM data

Compile and single-step the following program on your 8051 simulator. Examine the contents of the 128-byte RAM space to locate the ASCII values.

Solution:

```
#include <reg51.h>

void main(void)
  {
    unsigned char mynum[]="ABCDEF"; //RAM space
    unsigned char z;
    for (z=0;z<=6;z++)
       P1=mynum[z];
  }
```

Timer in C

- Ex: Write an 8051 C program to toggle all the bits of port P1 continuously with some delay in between. Use Timer 0, 16-bit mode to generate the delay.

```
#include <reg51.h>
void T0Delay(void);
void main(void){
  while (1) {
    P1=0x55;
    T0Delay();
    P1=0xAA;
    T0Delay();
  }
}
void T0Delay(){
  TMOD=0x01;
  TL0=0x00;
  TH0=0x35;
  TR0=1;
  while (TF0==0);
  TR0=0;
  TF0=0;
}
```

FFFFH – 3500H = CAFFH
= 51967 + 1 = 51968

51968 × 1.085 µs = 56.384 ms is the approximate delay

Timer in C

- Ex: Write an 8051 C program to toggle only bit P1.5 continuously every 50 ms. Use Timer 0, mode 1 (16-bit) to create the delay.

```
#include <reg51.h>
void T0M1Delay(void);
sbit mybit=P1^5;
void main(void){
  while (1) {
    mybit=~mybit;
    T0M1Delay();
  }
}
void T0M1Delay(void){
  TMOD=0x01;
  TL0=0xFD;
  TH0=0x4B;
  TR0=1;
  while (TF0==0);
  TR0=0;
  TF0=0;
}
```

FFFFH – 4BFDH = B402H

= 46082 + 1 = 46083

46083 × 1.085 μs = 50 ms

Timer in C

- Ex: A switch is connected to pin P1.2. Write an 8051 C program to monitor SW and create the following frequencies on pin P1.7: SW=0: 500Hz, SW=1: 750Hz, use Timer 0, mode 1 for both of them.

```
#include <reg51.h>
sbit mybit=P1^5;
sbit SW=P1^7;
void T0M1Delay(unsigned char);
void main(void){
  SW=1;
  while (1) {
    mybit=~mybit;
    if (SW==0)
       T0M1Delay(0);
    else
       T0M1Delay(1);
  }
}
```

```
void T0M1Delay(unsigned char c){
  TMOD=0x01;
  if (c==0) {
     TL0=0x67;
     TH0=0xFC;
  }
  else {
     TL0=0x9A;
     TH0=0xFD;
  }
  TR0=1;
  while (TF0==0);
  TR0=0;
  TF0=0;
}
```

Timer in C

- Ex: Write an 8051 C program to toggle only pin P1.5 continuously every 250 ms. Use Timer 0, mode 2 (8-bit auto-reload) to create the delay.

```
#include <reg51.h>
void T0M2Delay(void);
sbit mybit=P1^5;
void main(void){
  unsigned char x,y;
  while (1) {
    mybit=~mybit;
    for (x=0;x<250;x++)
        for (y=0;y<36;y++) //we put 36, not 40
            T0M2Delay();
  }
}
void T0M2Delay(void){
  TMOD=0x02;
  TH0=-23;
  TR0=1;
  while (TF0==0);
  TR0=0;
  TF0=0;
}
```

Due to overhead of the for loop in C, we put 36 instead of 40

256 – 23 = 233

23 × 1.085 µs = 25 µs and

25 µs × 250 × 40 = 250 ms

Timer in C

- Ex: Write an 8051 C program to create a frequency of 2500 Hz on pin P2.7. Use Timer 1, mode 2 to create delay.

```
#include <reg51.h>
void T1M2Delay(void);
sbit mybit=P2^7;
void main(void){
  unsigned char x;
  while (1) {
    mybit=~mybit;
    T1M2Delay();
  }
}
void T1M2Delay(void){
  TMOD=0x20;
  TH1=-184;
  TR1=1;
  while (TF1==0);
  TR1=0;
  TF1=0;
}
```

1/2500 Hz = 400 µs

400 µs /2 = 200 µs

200 µs / 1.085 µs = 184

Interrupt - Programming in C

- The 8051 compiler have extensive support for the interrupts
  - They assign a unique number to each of the 8051 interrupts
  - It can assign a register bank to an ISR
    - ✓ This avoids code overhead due to the pushes and pops of the R0 – R7 registers

Interrupt	Name	Numbers
External Interrupt 0	(INT0)	0
Timer Interrupt 0	(TF0)	1
External Interrupt 1	(INT1)	2
Timer Interrupt 1	(TF1)	3
Serial Communication	(RI + TI)	4
Timer 2 (8052 only)	(TF2)	5

Interrupt - Programming in C

- Ex: Write a C program that continuously gets a single bit of data from P1.7 and sends it to P1.0, while simultaneously creating a square wave of 200 μs period on pin P2.5. Use Timer 0 to create the square wave. Assume that XTAL = 11.0592 MHz.

```
#include <reg51.h>
sbit SW   =P1^7;
sbit IND  =P1^0;
sbit WAVE =P2^5;
void timer0(void) interrupt 1 {
  WAVE=~WAVE;  //toggle pin
}
void main() {
  SW=1;           //make switch input
  TMOD=0x02;
  TH0=0xA4;       //TH0=-92
  IE=0x82;        //enable interrupt for timer 0
  while (1) {
    IND=SW;       //send switch to LED
  }
}
```

Interrupt - Programming in C

- Ex: Write a C program using interrupts to do the following:
  (a) Generate a 10 KHz frequency on P2.1 using T0 8-bit auto-reload
  (b) Use timer 1 as an event counter to count up a 1-Hz pulse and display it on P0. The pulse is connected to EX1.

```
#include <reg51.h>
sbit WAVE =P2^1;
Unsigned char cnt;

void timer0() interrupt 1 {
  WAVE=~WAVE;   //toggle pin
}

void timer1() interrupt 3 {
  cnt++;          //increment counter
  P0=cnt;         //display value on pins
}

void main() {
  cnt=0;          //set counter to 0
  TMOD=0x42;
  TH0=0x-46;      //10 KHz
  IE=0x86;        //enable interrupts
  TR0=1;          //start timer 0
  while (1);      //wait until interrupted
}
```