ROHINI COLLEGE OF ENGINEERING AND TECHNOLOGY

Kanyakumari Main Road, near Anjugramam, Palkulam, Anjugramam, Tamil Nadu 629401

Department of Electronics and Communication Engineering

&

Masha Innovation Centre
Kamaraj Building,
Nagercoil-629001

VALUE ADDED COURSE ON EMBEDDED C PROGRAMMING

COURSE MATERIAL

8051 Embedded 'C' Programming

Definition: An embedded system is an application that contains at least one programmable computer (typically in the form of a microcontroller, a microprocessor or digital signal processor chip) and which is used by individuals who are, in the main, unaware that the system is computer-based.

Introduction to Embedded C

Looking around, we find ourselves to be surrounded by various types of embedded systems. Be it a digital camera or a mobile phone or a washing machine, all of them has some kind of processor functioning inside it. Associated with each processor is the embedded software. If hardware forms the body of an embedded system, embedded processor acts as the brain, and embedded software forms its soul. It is the embedded software which primarily governs the functioning of embedded systems.

During infancy years of microprocessor based systems, programs were developed using assemblers and fused into the EPROMs. There used to be no mechanism to find what the program was doing. LEDs, switches, etc. were used to check correct execution of the program. Some 'very fortunate' developers had In-circuit Simulators (ICEs), but they were too costly and were not quite reliable as well.

As time progressed, use of microprocessor-specific assembly-only as the programming language reduced and embedded systems moved onto C as the embedded programming language of choice. C is the most widely used programming language for embedded processors/controllers. Assembly is also used but mainly to implement those portions of the code where very high timing accuracy, code size efficiency, etc. are prime requirements.

Advantages of C:

Use of C in embedded systems is driven by following advantages.

- It is small and reasonably simpler to learn, understand, program and debug. C Compilers are available for almost all embedded devices in use today, and there is a large pool of experienced C programmers.
- Unlike assembly, C has advantage of processor-independence and is not specific to any particular microprocessor/ microcontroller or any system. This makes it convenient for a user to develop programs that can run on most of the systems.
- As C combines functionality of assembly language and features of high level languages, C is treated as a 'middle-level computer language' or 'high level assembly language'. C language is fairly efficient and supports access to I/O and provides ease of management of large embedded projects.
- Well proven compilers are available for every embedded processor (8-bit to 32-bit).
- C x 51 Cross Compiler supports all of the ANSI Standard C directives.

C Versus Embedded 'C'

C Language	Embedded 'C'
'C' is a well structured, well defined and standardized general purpose programming language.	Embedded 'C' can be considered as a subset of 'C' language.
'C' is used for desktop computers	Embedded 'C' is for microcontroller based applications
'C' has the luxury to use resources of a desktop PC like memory, OS, etc. on desktop systems	Embedded 'C' has to use with the limited resources (RAM, ROM, I/Os) on an embedded processor
Compilers for 'C' (ANSI C) typically generate OS dependant executables	Embedded 'C' requires compilers (Cross Compilers) to create files to be downloaded to the microcontrollers / microprocessors where it needs to run

Compilers Vs Cross Compiler

Compiler is a software tool that converts a source code written in a high level language to machine code. Generally compilers are used for desktop applications.

A cross compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is run. Cross compiler tools are generally found in use to generate compiles for embedded system

Storage Classes

A storage class decides scope, visibility and lifetime of a variable. 'C' supports the following storage classes.

- Auto (automatic variables)
- Extern (external variables)
- Static (static variables)
- Register (register variables)

Auto: A variable declared inside a function without any storage class specification, is by default an Auto (automatic) variable. They are created when a function is called and are

destroyed automatically when the function exits. Automatic variables can also be called local variables because they are local to a function. By default they are assigned garbage value by the compiler.

Extern (External or Global variable): A variable that is declared outside of any function is a Global variable. Global variables remain available throughout the program. One important thing to remember about global variable is that their values can be changed by any function in the program. The extern keyword is used before a variable to inform the compiler that this variable is declared somewhere else. The extern declaration does not allocate storage for variables.

Static: A static variable tells the compiler to persist the variable until the end of program. Instead of creating and destroying a variable every time when it comes into and goes out of scope. Static is initialized only once and remains into existence till the end of program. Static variables are assigned '0' (zero) as default value by the compiler.

Register variable inform the compiler to store the variable in register instead of memory. Register variable has faster access than normal variable. Frequently used variables are kept in register. Only few variables can be placed inside register. Note that we can never get the address of such variables.

Data Types

There are various type of Data types in C :

- 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 and unsigned char is also an 8-bit data type in the range of 0 –255 (00 –FFH).C compilers use the signed char as the default data types if we do not put the keyword unsigned char.

We always use unsigned char in program until and unless we don't need to represent signed numbers for example Temperature.

signed char :

The signed char is an 8-bit data type. signed char use the MSB D7 to represent –or +. signed char give us values from –128 to +127.

Unsigned 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:

- 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 :

- The bit data type allows access to single bits of bit-addressable memory spaces 20 –2FH

sfr :

- To access the byte-size SFR registers, we use the sfr data type.

Embedded C Programs

1. An embedded C program to load a number into Accumulator.

```
# include <reg51.h>
void main( )
{
    Acc = 0x25;
}
```

2. Write a program to load three numbers into Accumulator and send them to port 1

```
# include <reg51.h>
void main( )
{
    Acc = 0x25;
    P1 = Acc;
    Acc = 0x46;
    P1 = Acc;
    Acc = 0x92;
    P1 = Acc;
}
```

3. Write an 8051 C program to toggle all the bits of P1 continuously.

```
//Toggle P1 forever
#include <reg51.h>
void main( )
{
        for ( ; ;  )
        {
                P1=0 x 55;
                P1=0 x AA;
        }
}
```

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

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

5. Write an 8051 C program to send hex values for ASCII characters of 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D and E to port P1.

```
#include <reg51.h>
void main(void)
{
        unsigned char mynum[ ] = "0123456789ABCDE";
        unsigned char z;
        for (z = 0; z <= 15; z++)
        P1 = mynum[z];
}
```

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

```
#include <reg51.h>
void MSDelay(unsigned int); //delay routine definition
void main( )
{
        while (1) //repeat forever
        {
                P1= 0x55;
                MSDelay(250);
                P1= 0xAA;
                MSDelay(250);
        }
}

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

7. Write an 8051 C program to get a byte of data form P1, wait ½ second (i.e., 500 ms) and then send it to P2.

```
#include <reg51.h>
void MSDelay(unsigned int);
void main( )
{
        unsigned char mybyte;
        P1= 0x FF; //make P1 input port
```

```
        while (1)
        {
                mybyte = P1; //get a byte from P1
                MSDelay(500); //wait for ½ second.
                P2 = mybyte; //send it to P2
        }
}

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

8. Write a C program for 8051 to transfer the letter "A" serially at 9600 baud continuously. Use 8-bit data and 1 stop bit.

```
#include <reg51.h>
void main( )
{
        TMOD = 0x20; //use Timer 1, mode 2
        TH1= 0xFD; //9600 baud rate
        SCON = 0x50;    //configure SCON
        TR1 = 1; // start the timer 1
        while (1)
        {
                SBUF = 'A'; //place value in buffer
                while (TI = = 0);
                 TI = 0;
        }
}
```

9. Write an 8051 C program to transfer the message “SJC” serially at 9600 baud, 8-bit data, 1 stop bit. Do this continuously.

```
# include <reg51.h>
void sendChar(unsigned char);
void main(void)
{
    TMOD=0x20; //use Timer 1, mode 2
    TH1=0xFD; //9600 baud rate
    SCON=0x50;
    TR1=1; //start timer
    while (1)
    {
            sendChar(‘S’);
            sendChar (‘J’);
            sendChar (‘C’);
    }
}

void sendChar(unsigned char  x)
{
    SBUF = x; //place value in buffer
    while (TI = = 0); //wait until transmitted
    TI = 0;
}
```

10. Program the 8051 in C to receive bytes of data serially and put them in P1. Set the baud rate at 9600, 8-bit data, and 1 stop bit.

```
#include <reg51.h>
void main( )
{
        unsigned char mybyte;
        TMOD = 0 x 20; //use Timer 1, mode 2
        TH1= 0xFD; //9600 baud rate
        SCON = 0x50;
        TR1 = 1; //start timer
        while (1) //repeat forever
        {
                while (RI = = 0); //wait to receive
                mybyte = SBUF; //save value
                P1 = mybyte; //write value to port
                RI = 0;
        }
}
```

11. Write an 8051 C Program to send the two messages “first name” and “last name” to the serial port. If SW = 0, send first name else if SW = 1, send last name. Set the baud rate at 9600, 8-bit data, and 1 stop bit.

```
# include <reg51.h>
unsigned  char  SW;
void main(void)
{
        unsigned char z;
        unsigned char first_name[ ]= “LNRAO”; //use your first name
```

```
        unsigned char last_name[ ] = “MERUGU”; // use your second name
        TMOD=0x20; //use Timer 1, mode 2
        TH1=0xFD; //9600 baud rate
        SCON=0x50;
        TR1=1; //start timer

        if(SW = = 0)
        {
                for (z = 0; z < 5; z++)
                {
                        SBUF = first_name[z]; //place value in buffer
                        while(TI==0); //wait for transmit
                        TI=0;
                }
        }
        else
        {

                for (z = 0; z < 6; z++)
                 {
                        SBUF = last_name[z]; //place value in buffer
                        while(TI==0); //wait for transmit
                        TI=0;
                }
        }
}
```