**Unit II- Programming for Embedded Systems using C**

Embedded C Programming is the soul of the processor functioning inside each and every [embedded system](https://www.elprocus.com/ieee-projects-on-embedded-systems/) we come across in our daily life, such as mobile phone, washing machine, and digital camera.

Each processor is associated with an embedded software. The first and foremost thing is the embedded software that decides functioning of the embedded system. Embedded C language is most frequently used to [program the microcontroller](https://www.elprocus.com/how-to-program-the-microcontroller/).



Earlier, many embedded applications were developed using assembly level programming. However, they did not provide portability. This disadvantage was overcome by the advent of various high level languages like C, Pascal, and COBOL. However, it was the C language that got extensive acceptance for embedded systems, and it continues to do so. The C code written is more reliable, scalable, and portable; and in fact, much easier to understand.

**About C Language**

C language was developed by Dennis Ritchie in 1969. It is a collection of one or more functions, and every function is a collection of statements performing a specific task.
C language is a middle-level language as it supports high-level applications and low-level applications. Before going into the details of embedded C programming, we should know about RAM memory organization.

**Salient features of the language**

* C language is a software designed with different keywords, data types, variables, constants, etc.
* Embedded C is a generic term given to a programming language written in C, which is associated with a particular hardware architecture.
* Embedded C is an extension to the C language with some additional header files. These header files may change from controller to controller.
* The [microcontroller 8051](https://www.elprocus.com/8051-microcontroller-architecture-and-applications/) #include<reg51.h> is used.

The embedded system designers must know about the hardware architecture to write programs. These programs play prominent role in monitoring and controlling external devices. They also directly operate and use the internal architecture of the microcontroller, such as interrupt handling, timers, serial communication and other available features.

**Differences between C and Embedded C**



The basic additional features of the embedded software

**Data types :**The data type refers to an extensive system for declaring variables of different types like integer, character, float, etc. The embedded C software uses four data types that are used to store data in the memory.

The ‘char’ is used to store any single character; ‘int’ is used to store integer value, and ‘float’ is used to store any precision floating point value.

The size and range of different data types on a 32-bit machine is given in the following table. The size and range may vary on machines with different word sizes.



**Keywords**

There are certain words that are reserved for doing specific tasks. These words are known as keywords. They are standard and predefined in the Embedded C.

Keywords are always written in lowercase. These keywords must be defined before writing the main program. The basic keywords of an embedded software are given below:



sbit: This data type is used in case of accessing a single bit of SFR register.

Syntax: sbit variable name = SFR bit ;

Ex: sbit a=P2^1;

Explanation: If we assign p2.1 as ‘a’ variable, then we can use ‘a’ instead of p2.1 anywhere in the program, which reduces the complexity of the program.

Bit: This data type is used for accessing the bit addressable memory of RAM (20h-2fh).

Syntax: bit variable name;

Ex: bit c;

Explanation: It is a bit sequence setting in a small data area that is used by a program to remember something.

SFR: This data type is used for accessing a SFR register by another name. All the SFR registers must be declared with capital letters.

Syntax: SFR variable name = SFR address of SFR register;

Ex: SFR port0=0x80;

Explanation: If we assign 0x80 as ‘port0’, then we can use 0x80 instead of port0 anywhere in the program, which reduces the complexity of the program.

SFR Register: The SFR stands for ‘Special Function Register’. Microcontroller 8051 has 256 bytes of RAM memory. This RAM is divided into two parts: the first part of 128 bytes is used for data storage, and the other of 128 bytes is used for SFR registers. All peripheral devices like I/O ports, timers and counters are stored in the SFR register, and each element has a unique address.

**The Structure of an Embedded C Program**

comments

preprocessor directives

global variables

main() function

{

local variables

statements

…………..

…………..

}

fun(1)

{

local variables

statements

…………..

…………..

}

**Comments:** In embedded C programming language, we can place comments in our code which helps the reader to understand the code easily.

C=a+b; /\* add two variables whose value is stored in another variable C\*/

**Preprocessor directives:** All the functions of the embedded C software are included in the preprocessor library like “#includes<reg51.h>, #defines”. These functions are executed at the time of running the program.

**Global variable :**A global variable is a variable that is declared before the main function, and can be accessed on any function in the program.



A local variable is a variable declared within a function, and it is valid only to be used within that function.



**Main () function :**The execution of a program starts with the main function. Every program uses only one main () function.

**Advantages of embedded C program**

Its takes less time to develop application program.

It reduces complexity of the program.

It is easy to verify and understand.

It is portable in nature from one controller to another.

**Examples of a few Embedded C Programs**

The following are a few simple Embedded C programs used for microcontroller-based projects.









A device driver is a program that controls a particular type of device that is attached

to your computer. There are device drivers for printers, displays, CD-ROM readers,

diskette drives, and so on.

A driver provides a software interface to hardware devices, enabling operating

systems and other computer programs to access hardware functions without

needing to know precise details of the hardware being used.

A driver typically communicates with the device through the computer bus or

communications subsystem to which the hardware connects. When a calling

program invokes a routine in the driver, the driver issues commands to the device.

Once the device sends data back to the driver, the driver may invoke routines in the

original calling program. Drivers are hardware-dependent and operating-system-

specific. They usually provide the interrupt handling required for any necessary

asynchronous time-dependent hardware interface.

A device driver is a program that controls a particular type of device that is attached

to your computer. There are device drivers for printers, displays, CD-ROM readers,

diskette drives, and so on.

A driver provides a software interface to hardware devices, enabling operating

systems and other computer programs to access hardware functions without

needing to know precise details of the hardware being used.

A driver typically communicates with the device through the computer bus or

communications subsystem to which the hardware connects. When a calling

program invokes a routine in the driver, the driver issues commands to the device.

Once the device sends data back to the driver, the driver may invoke routines in the

original calling program. Drivers are hardware-dependent and operating-system-

specific. They usually provide the interrupt handling required for any necessary

asynchronous time-dependent hardware interface.

A device driver is a program that controls a particular type of device that is attached

to your computer. There are device drivers for printers, displays, CD-ROM readers,

diskette drives, and so on.

A driver provides a software interface to hardware devices, enabling operating

systems and other computer programs to access hardware functions without

needing to know precise details of the hardware being used.

A driver typically communicates with the device through the computer bus or

communications subsystem to which the hardware connects. When a calling

program invokes a routine in the driver, the driver issues commands to the device.

Once the device sends data back to the driver, the driver may invoke routines in the

original calling program. Drivers are hardware-dependent and operating-system-

specific. They usually provide the interrupt handling required for any necessary

asynchronous time-dependent hardware interface.