**Basics of Embedded C Program**

Embedded C is one of the most popular and most commonly used Programming Languages in the development of Embedded Systems. So, in this article, we will see some of the Basics of Embedded C Program and the Programming Structure of Embedded C.

Embedded C is perhaps the most popular languages among Embedded Programmers for programming Embedded Systems. There are many popular programming languages like Assembly, BASIC, C++, Python etc. that are often used for developing Embedded Systems but Embedded C remains popular due to its efficiency, less development time and portability.

Before digging in to the basics of Embedded C Program, we will first take a look at what an Embedded System is and the importance of Programming Language in Embedded Systems.

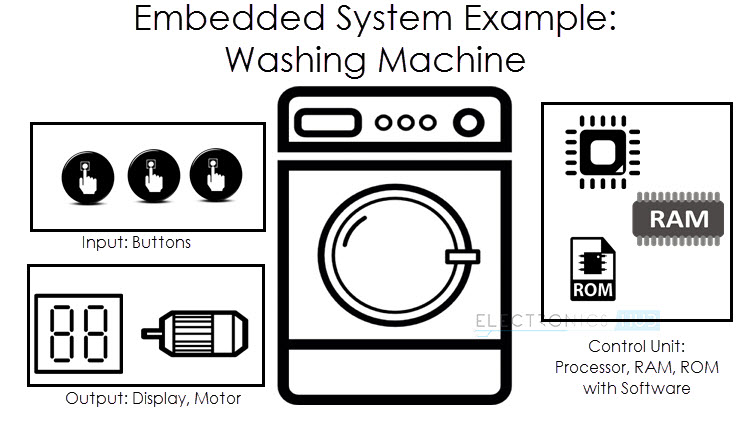
Outline

* [What is an Embedded System?](https://www.electronicshub.org/basics-of-embedded-c-program/#What_is_an_Embedded_System)
* [Programming Embedded Systems](https://www.electronicshub.org/basics-of-embedded-c-program/#Programming_Embedded_Systems)
* [Factors for Selecting the Programming Language](https://www.electronicshub.org/basics-of-embedded-c-program/#Factors_for_Selecting_the_Programming_Language)
* [Introduction to Embedded C Programming Language](https://www.electronicshub.org/basics-of-embedded-c-program/#Introduction_to_Embedded_C_Programming_Language)
* [Difference between C and Embedded C](https://www.electronicshub.org/basics-of-embedded-c-program/#Difference_between_C_and_Embedded_C)
* [Basics of Embedded C Program](https://www.electronicshub.org/basics-of-embedded-c-program/#Basics_of_Embedded_C_Program)
  + [Keywords in Embedded C](https://www.electronicshub.org/basics-of-embedded-c-program/#Keywords_in_Embedded_C)
  + [Data Types in Embedded C](https://www.electronicshub.org/basics-of-embedded-c-program/#Data_Types_in_Embedded_C)
* [Basic Structure of an Embedded C Program (Template for Embedded C Program)](https://www.electronicshub.org/basics-of-embedded-c-program/#Basic_Structure_of_an_Embedded_C_Program_Template_for_Embedded_C_Program)
  + [Different Components of an Embedded C Program](https://www.electronicshub.org/basics-of-embedded-c-program/#Different_Components_of_an_Embedded_C_Program)
* [Basic Embedded C Program](https://www.electronicshub.org/basics-of-embedded-c-program/#Basic_Embedded_C_Program)
  + [Example of Embedded C Program](https://www.electronicshub.org/basics-of-embedded-c-program/#Example_of_Embedded_C_Program)

**What is an Embedded System?**

An Embedded System can be best described as a system which has both the hardware and software and is designed to do a specific task. A good example for an Embedded System, which many households have, is a Washing Machine.

We use washing machines almost daily but wouldn’t get the idea that it is an embedded system consisting of a Processor (and other hardware as well) and software.



It takes some inputs from the user like wash cycle, type of clothes, extra soaking and rinsing, spin rpm, etc., performs the necessary actions as per the instructions and finishes washing and drying the clothes. If no new instructions are given for the next wash, then the washing machines repeats the same set of tasks as the previous wash.

Embedded Systems can not only be stand-alone devices like Washing Machines but also be a part of a much larger system. An example for this is a Car. A modern day Car has several individual embedded systems that perform their specific tasks with the aim of making a smooth and safe journey.

Some of the embedded systems in a Car are Anti-lock Braking System (ABS), Temperature Monitoring System, Automatic Climate Control, Tyre Pressure Monitoring System, Engine Oil Level Monitor, etc.

Also read [EMBEDDED SYSTEMS & ITS REAL TIME APPLICATIONS](https://www.electronicshub.org/embedded-system-real-time-applications/).

**Programming Embedded Systems**

As mentioned earlier, Embedded Systems consists of both Hardware and Software. If we consider a simple Embedded System, the main Hardware Module is the Processor. The Processor is the heart of the Embedded System and it can be anything like a Microprocessor, Microcontroller, DSP, CPLD (Complex Programmable Logic Device) or an FPGA (Field Programmable Gated Array).

All these devices have one thing in common: they are programmable i.e., we can write a program (which is the software part of the Embedded System) to define how the device actually works.

Embedded Software or Program allow Hardware to monitor external events (Inputs / Sensors) and control external devices (Outputs) accordingly. During this process, the program for an Embedded System may have to directly manipulate the internal architecture of the Embedded Hardware (usually the processor) such as Timers, Serial Communications Interface, Interrupt Handling, and I/O Ports etc.

From the above statement, it is clear that the Software part of an Embedded System is equally important as the Hardware part. There is no point in having advanced Hardware Components with poorly written programs (Software).

There are many programming languages that are used for Embedded Systems like Assembly (low-level Programming Language), C, C++, JAVA (high-level programming languages), Visual Basic, JAVA Script (Application level Programming Languages), etc.

In the process of making a better embedded system, the programming of the system plays a vital role and hence, the selection of the Programming Language is very important.

**Factors for Selecting the Programming Language**

The following are few factors that are to be considered while selecting the Programming Language for the development of Embedded Systems.

* **Size:** The memory that the program occupies is very important as Embedded Processors like Microcontrollers have a very limited amount of ROM (Program Memory).
* **Speed**: The programs must be very fast i.e., they must run as fast as possible. The hardware should not be slowed down due to a slow running software.
* **Portability:** The same program can be compiled for different processors.
* Ease of Implementation
* Ease of Maintenance
* Readability

Earlier Embedded Systems were developed mainly using Assembly Language. Even though Assembly Language is closest to the actual machine code instructions and produces small size hex files, the lack of portability and high amount of resources (time and man power) spent on developing the code, made the Assembly Language difficult to work with.

There are other high-level programming languages that offered the above mentioned features but none were close to C Programming Language. Some of the benefits of using Embedded C as the main Programming Language:

* Significantly easy to write code in C
* Consumes less time when compared to Assembly
* Maintenance of code (modifications and updates) is very simple
* Make use of library functions to reduce the complexity of the main code
* You can easily port the code to other architecture with very little modifications

**Introduction to Embedded C Programming Language**

Before going in to the details of Embedded C Programming Language and basics of Embedded C Program, we will first talk about the C Programming Language.

The C Programming Language, developed by Dennis Ritchie in the late 60’s and early 70’s, is the most popular and widely used programming language. The C Programming Language provided low level memory access using an uncomplicated compiler (a software that converts programs to machine code) and achieved efficient mapping to machine instructions.

The C Programming Language became so popular that it is used in a wide range of applications ranging from Embedded Systems to Super Computers.

Embedded C Programming Language, which is widely used in the development of Embedded Systems, is an extension of C Program Language. The Embedded C Programming Language uses the same syntax and semantics of the C Programming Language like main function, declaration of datatypes, defining variables, loops, functions, statements, etc.

The extension in Embedded C from standard C Programming Language include I/O Hardware Addressing, fixed point arithmetic operations, accessing address spaces, etc.

**Difference between C and Embedded C**

There is actually not much difference between C and Embedded C apart from few extensions and the operating environment. Both C and Embedded C are ISO Standards that have almost same syntax, datatypes, functions, etc.

Embedded C is basically an extension to the Standard C Programming Language with additional features like Addressing I/O, multiple memory addressing and fixed-point arithmetic, etc.

C Programming Language is generally used for developing desktop applications, whereas Embedded C is used in the development of Microcontroller based applications.

**Basics of Embedded C Program**

Now that we have seen a little bit about Embedded Systems and Programming Languages, we will dive in to the basics of Embedded C Program. We will start with two of the basic features of the Embedded C Program: Keywords and Datatypes.

**Keywords in Embedded C**

A Keyword is a special word with a special meaning to the compiler (a C Compiler for example, is a software that is used to convert program written in C to Machine Code). For example, if we take the Keil’s Cx51 Compiler (a popular C Compiler for 8051 based Microcontrollers) the following are some of the keywords:

* bit
* sbit
* sfr
* small
* large

The following table lists out all the keywords associated with the Cx51 C Compiler.

|  |  |  |
| --- | --- | --- |
| \_at\_ | alien | bdata |
| bit | code | compact |
| data | far | idata |
| interrupt | large | pdata |
| \_priority\_ | reentrant | sbit |
| sfr | sfr16 | small |
| \_task\_ | using | xdata |

**Data Types in Embedded C**

Data Types in C Programming Language (or any programming language for that matter) help us declaring variables in the program. There are many data types in C Programming Language like signed int, unsigned int, signed char, unsigned char, float, double, etc. In addition to these there few more data types in Embedded C.

The following are the extra data types in Embedded C associated with the Keil’s Cx51 Compiler.

* bit
* sbit
* sfr
* sfr16

The following table shows some of the data types in Cx51 Compiler along with their ranges.

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Bits (Bytes)** | **Range** |
| bit | 1 | 0 or 1 (bit addressable part of RAM) |
| signed int | 16 (2) | -32768 to +32767 |
| unsigned int | 16 (2) | 0 to 65535 |
| signed char | 8 (1) | -128 to +127 |
| unsigned | 8 (1) | 0 to 255 |
| float | 32 (4) | ±1.175494E-38 to ±3.402823E+38 |
| double | 32 (4) | ±1.175494E-38 to ±3.402823E+38 |
| sbit | 1 | 0 or 1 (bit addressable part of RAM) |
| sfr | 8 (1) | RAM Addresses (80h to FFh) |
| sfr16 | 16 (2) | 0 to 65535 |

**Basic Structure of an Embedded C Program (Template for Embedded C Program)**

The next thing to understand in the Basics of Embedded C Program is the basic structure or Template of Embedded C Program. This will help us in understanding how an Embedded C Program is written.

The following part shows the basic structure of an Embedded C Program.

* + Multiline Comments . . . . . Denoted using /\*……\*/
  + Single Line Comments . . . . . Denoted using //
  + Preprocessor Directives . . . . . #include<…> or #define
  + Global Variables . . . . . Accessible anywhere in the program
  + Function Declarations . . . . . Declaring Function
  + Main Function . . . . . Main Function, execution begins here  
    {  
    Local Variables . . . . . Variables confined to main function  
    Function Calls . . . . . Calling other Functions  
    Infinite Loop . . . . . Like while(1) or for(;;)  
    Statements . . . . .  
    ….  
    ….  
    }
  + Function Definitions . . . . . Defining the Functions  
    {  
    Local Variables . . . . . Local Variables confined to this Function  
    Statements . . . . .  
    ….  
    ….  
    }

Before seeing an example with respect to 8051 Microcontroller, we will first see the different components in the above structure.

**Different Components of an Embedded C Program**

**Comments:** Comments are readable text that are written to help us (the reader) understand the code easily. They are ignored by the compiler and do not take up any memory in the final code (after compilation).

There are two ways you can write comments: one is the single line comments denoted by // and the other is multiline comments denoted by /\*….\*/.

**Preprocessor Directive:** A Preprocessor Directive in Embedded C is an indication to the compiler that it must look in to this file for symbols that are not defined in the program.

In C Programming Language (also in Embedded C), Preprocessor Directives are usually represented using # symbol like #include… or #define….

In Embedded C Programming, we usually use the preprocessor directive to indicate a header file specific to the microcontroller, which contains all the SFRs and the bits in those SFRs.

In case of 8051, Keil Compiler has the file “reg51.h”, which must be written at the beginning of every Embedded C Program.

**Global Variables:** Global Variables, as the name suggests, are Global to the program i.e., they can be accessed anywhere in the program.

**Local Variables:** Local Variables, in contrast to Global Variables, are confined to their respective function.

**Main Function:** Every C or Embedded C Program has one main function, from where the execution of the program begins.

Related Post: “[EMBEDDED SYSTEM PROJECT IDEAS](https://www.electronicshub.org/embedded-systems-projects-ideas/)“.

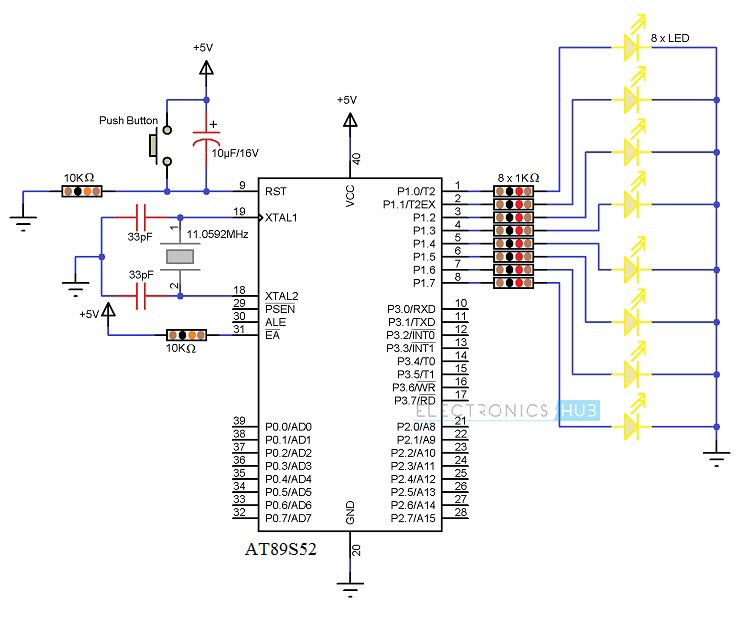
**Basic Embedded C Program**

Till now, we have seen a few Basics of Embedded C Program like difference between C and Embedded C, basic structure or template of an Embedded C Program and different components of the Embedded C Program.

Continuing further, we will explore in to basics of Embedded C Program with the help of an example. In this example, we will use an 8051 Microcontroller to blink LEDs connected to PORT1 of the microcontroller.

**Example of Embedded C Program**

The following image shows the circuit diagram for the example circuit. It contains an 8051 based Microcontroller (AT89S52) along with its basic components (like RESET Circuit, Oscillator Circuit, etc.) and components for blinking LEDs (LEDs and Resistors).



In order to write the Embedded C Program for the above circuit, we will use the Keil C Compiler. This compiler is a part of the Keil µVision IDE. The program is shown below.

**COMPONENTS**

#include<reg51.h>*// Preprocessor Directive*  
void delay (int);*// Delay Function Declaration*

void main(void)*// Main Function*  
{  
P1 = 0x00;  
*/\* Making PORT1 pins LOW. All the LEDs are OFF.*  
*\* (P1 is PORT1, as defined in reg51.h) \*/*

while(1)*// infinite loop*  
{  
P1 = 0xFF;*// Making PORT1 Pins HIGH i.e. LEDs are ON.*  
delay(1000);  
*/\* Calling Delay function with Function parameter as 1000.*  
*\* This will cause a delay of 1000mS i.e. 1 second \*/*

P1 = 0x00;*// Making PORT1 Pins LOW i.e. LEDs are OFF.*  
delay(1000);  
}  
}

void delay (int d)*// Delay Function Definition*  
{  
unsigned int i=0;*// Local Variable. Accessible only in this function.*  
  
*/\* This following step is responsible for causing delay of 1000mS*  
*\* (or as per the value entered while calling the delay function) \*/*

for(; d>0; d–)  
{  
for(i=250; i>0; i – -);  
for(i=248; i>0; i – -);  
}  
}