Basics of Embedded C Program

Embedded C

Embedded C the most popular and most commonly used Programming Languages in the development of Embedded Systems.

This is not a programming module so we will just look at some of the Basics of an Embedded C Program and the Programming Structure of Embedded C.

Programming Embedded Systems

- Embedded Software or Program allow Hardware to monitor external events (Inputs) 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.

Introduction to Embedded C Programming Language

- Embedded C Programming Language 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.
- 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

- **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

  These are few of the many keywords associated with the Cx51 C Compiler along with the standard C Keywords.

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

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 . . . . .
  ….
  ….
  }

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 #include… or #define….

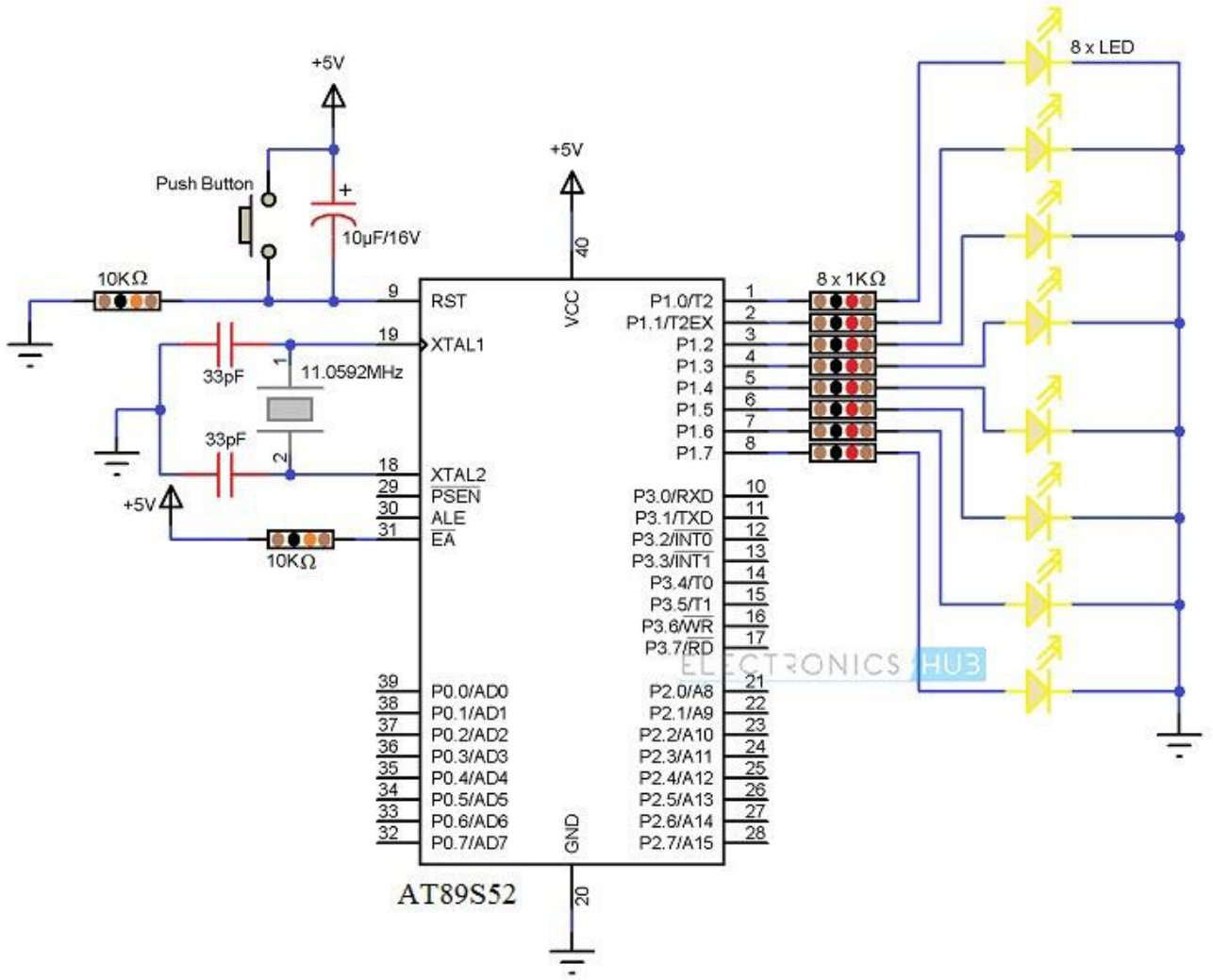
In Embedded C Programming, we usually use the preprocessor directive to indicate a header file specific to the microcontroller, which contains all the SFRs and the bits in those SFRs.

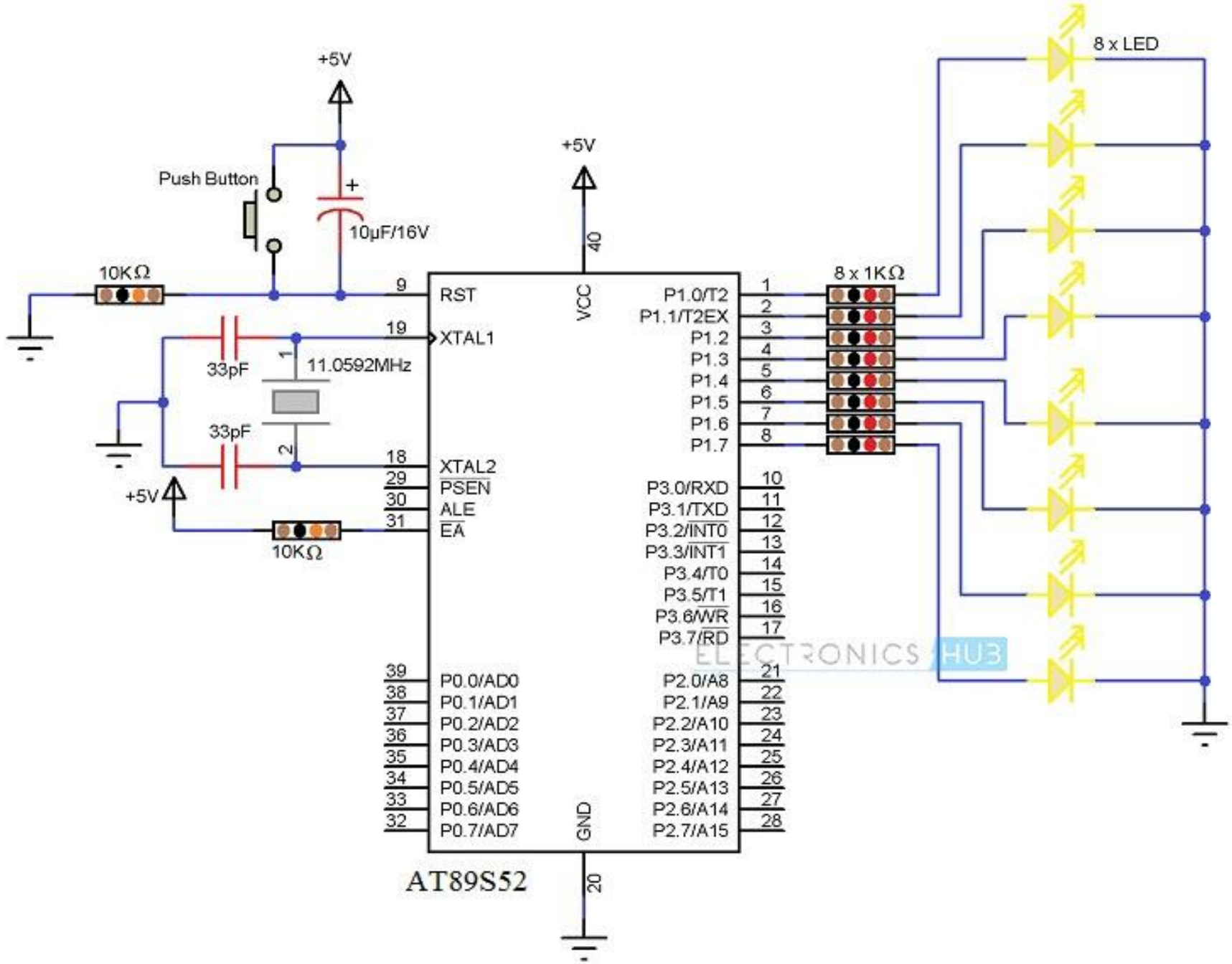
In case of 8051, Keil Compiler has the file “reg51.h”, which must be written at the beginning of every Embedded C Program.

- **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.

Example of Embedded C Program

- The image shows the circuit diagram that 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).




+5V
Push Button
10µF/16V
10KΩ
9
RST
19
XTAL1
33pF
11.0592MHz
33pF
18
XTAL2
29
PSEN
30
ALE
31
EA
+5V
10KΩ
+5V
40
VCC
GND
20
AT89S52
P1.0/T2
P1.1/T2EX
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.0/RXD
P3.1/TXD
P3.2/INT0
P3.3/INT1
P3.4/T0
P3.5/T1
P3.6/WR
P3.7/RD
P0.0/AD0
P0.1/AD1
P0.2/AD2
P0.3/AD3
P0.4/AD4
P0.5/AD5
P0.6/AD6
P0.7/AD7
P2.0/A8
P2.1/A9
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
8 x 1KΩ
8 x LED
ELECTRONICS HUB

```
#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 – -);
}
}
```

Embedded V General Computing Software

- Typically an Embedded System program will continue running (in a loop) and not reset.
- This can cause some subtle problems.

Example:

```
1   int a = 2;
2   void foo(int b, int* c) {
3     ...
4   }
5   int main(void) {
6     int d;
7     int* e;
8     d = ...;                  // Assign some value to d.
9     e = malloc(sizeInBytes);  // Allocate memory for e.
10    *e = ...;                 // Assign some value to e.
11    foo(d, e);
12    ...
13  }
```

a is a global variable assigned a fixed memory location
b and ***c*** are parameters which are given positions in the stack when foo is called
d and ***e*** are local variables declared in main and given postions in the stack.
When foo is called the stack location for ***b*** will acquire a copy of the ***d*** (pass by value). The data referred to by the pointer **e** is stored in memory allocated on the heap and passed by reference. The address is stored in the stack location for c.

Pitfall: The memory location for ***a*** is allocated when the program is loaded. Since in an embedded program we do not reload then if ***a*** is overwritten it will contain the wrong value the next time the program runs.

In Embedded C (or any other language) Global variables should be initialized in main.

(In you lab the compiler would not allow you to declarer a global variable outside of main)
[Hint: Revise, remember and fully explain this example !!]