Professional Elective-II: Embedded Systems (PECCSE601B)

Module - III

Embedded firmware design and development

Prepared by – Dr. Susmita Biswas
Associate Professor
University of Engineering and Management, Kolkata

Sub topic

- Embedded firmware design
- Embedded firmware development languages
- Programming in embedded C

Embedded Firmware Design

Hardware: Hardware refers to the physical components of a computer system. We can touch physically. Monitor, printer, canner, projector, keyboard, mouse, motherboard, CPU, RAM, ROM.

Software: Software is a collection of instructions, data, or computer programs that enable a computer to perform specific tasks. We can not touch physically. Word, excel, PowerPoint, Photoshop. You can access your software and install or uninstall it, modify it.

Firmware: Firmware is a specific class of computer software that provides the **low-level control for a device's specific hardware**. This is developed by manufacturer for specific application. These are specially used in embedded systems. You can not modify any firmware by your own.

Embedded Firmware Design

- **Impart intelligence:** We teach the baby how to walk, how to speak.

- **Adaptive:** When a baby eats with left hand, we can teach him what is the difference between left hand and right hand and we should eat with our right hand. Then baby understand and learn to eat with right hand. He used to adapt this.

- Firmware is imparting intelligence to the hardware but it is not adaptive. It can not be modified by the user.

Embedded Firmware Design

- **Super-loop based approach**
- **Embedded operating system based approach**

Embedded Firmware Design

• **Super-loop based approach:** Conventional procedural programming approach (serial).

```
Void main ()
{config()
  initialization ()
While(1)
{task 1;
  task 2;
  .
  .
  .
Task n;}
```

Infinite number of time

If you need to break the loop: reset, interrupt

Task 1

Task 2

Task 3

Embedded Firmware Design

- **Super-loop based approach:**

Advantages:

1. No time criticality
2. Response time is not important (Example: toy car)
3. No operating system is required because priority of each task is fixed, no task scheduling is required.
4. Low cost

Disadvantages:

1. If it hangs (Watch dog timer: it continuously follow-up the tasks, if something is wrong it resets the computing system)
2. Real time operation is not possible (to solve this use multiple processors)

Embedded Firmware Design

- **Embedded operating system based approach**

General-Purpose Operating System (GPOS)	Real-Time Operating System (RTOS)
is used for desktop PC and laptop Example: Windows, linux	Is used in embedded system Kernel: task scheduling, memory management, input/output controlling

Embedded Firmware Development Languages

Considerations during embedded system programming:

- Embedded devices have resource constraints (limited memory, less processing power)

- Smaller and less power consuming components

- Embedded systems are more tied to the hardware

Embedded Firmware Development Languages

Choice of programming languages (code speed, code size & code Portability)

- Machine language (not user friendly)

- Assembly language (inefficiency in terms of size and speed, assembly codes need high software development cost, codes are hardware specific (code portability is not there))

- High level language (C, C++, Java) (object oriented language such as C++ is not suitable for resource constraint environment in embedded system)

- Compare to other high level languages C offers more flexibility because it is a small structured language and it supports low level bit wise data manipulations.

Programming in Embedded C

• Embedded C is an extension of C language (structured programming language that uses main function, data type declaration, defining variables, loops, functions, statements). The extension of embedded C from standard C language is that input/output hardware addressing, fixed point arithmetic operations, address spaces.

• Advantages of using C language in embedded system:

1. Small and reasonably simpler to learn
2. C compilers are available in almost all embedded devices
3. Processor independent language (portable)
4. C combines functionalities of assembly language and features of high level languages.

Difference between C and Embedded C

Parameter	C	Embedded C
Definition	C is a versatile programming language that supports structured programming	Embedded C is a set of language extensions for the C programming language designed to program microcontrollers
Development	C is developed by Dennis M. Ritchie	Embedded C is developed by C Standards Committee
Hardware dependency	C language is hardware independent	Embedded C is hardware dependent language
Compiler Execution	A standard compiler facilitates the compilation and execution of a program.	Compilers that are capable of generating microcontroller based output needs to be used to execute Embedded C code

Difference between C and Embedded C

Parameter	C	Embedded C
Functionality	C language generates operating system dependent executable files	Embedded C generates hardware dependent files
Applications	Network drivers, interpreters, compilers, operating system and text editors are some of the applications	Robots, Vehicle tracking systems, smart monitoring systems are some of the applications.

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 . . . . .
….
….
}

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 /*….*/.

Different Components of an Embedded C Program

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.

Different Components of an 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.

Programming in Embedded C

1. Write a Program to read the number 1 from port 1, number 2 from port 2 , then add them ,store the result ,send it to Port 3.

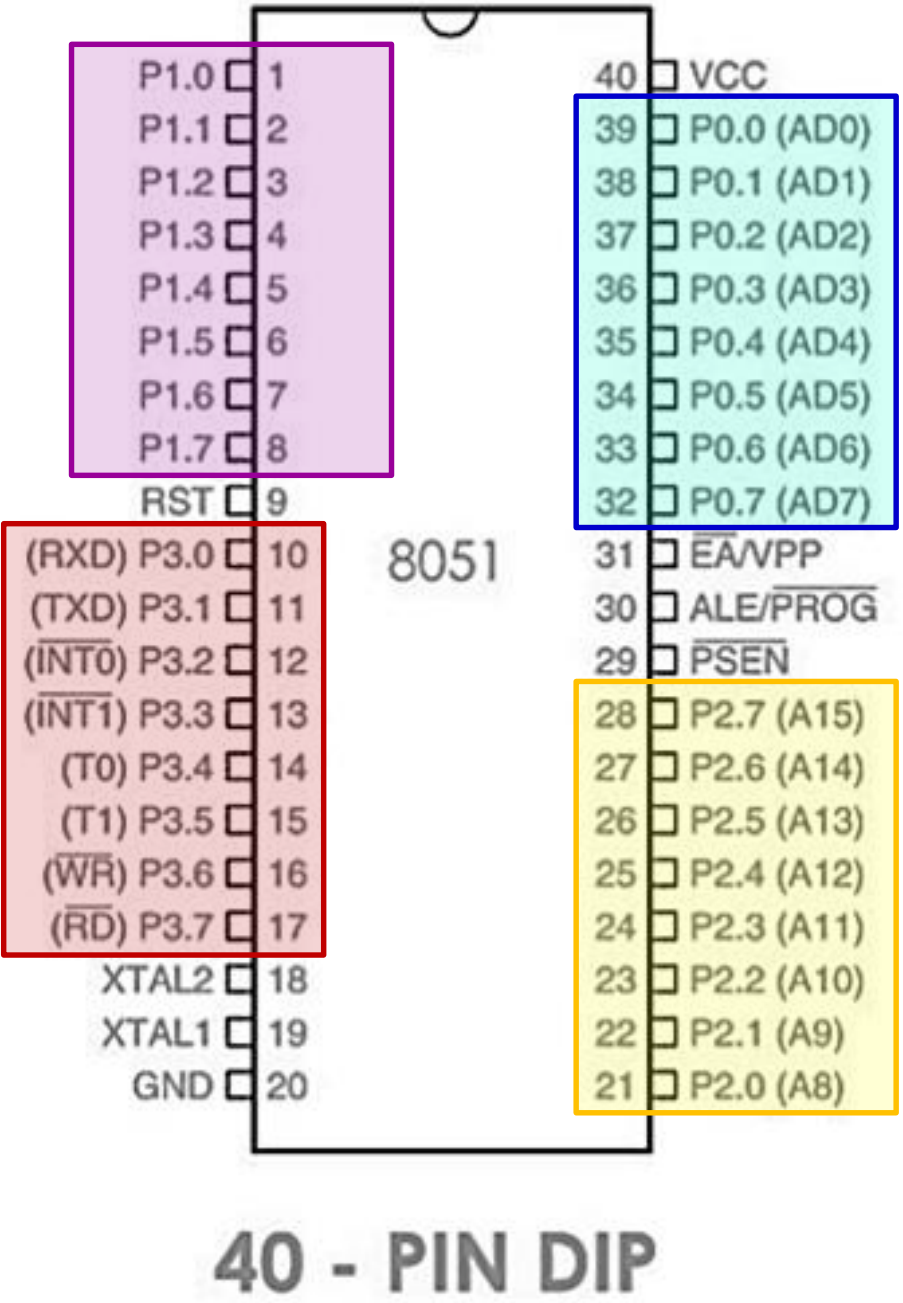
```
#include<reg.51.h>
void main()
{
unsigned char a,b,c ;
P1 = 0XFF ;     //make port 1 as input port
P2 = 0XFF ;     //make port 2 as input port
a=P1;
b=P2;
c= a+b ;
P3= c;
}
```

Programming in Embedded C

2. Write a program to transfer the data from port P0 to port P1.

```
#include<reg51.h>
void main (void )
{
unsigned char X;
P0=0XFF;    // P0 as input port
P1=0X00;   //  P1 as output port
while(1)
{
X = P0;    //  read port0
P1 = X;   // output data to port1
}
```

Programming in Embedded C

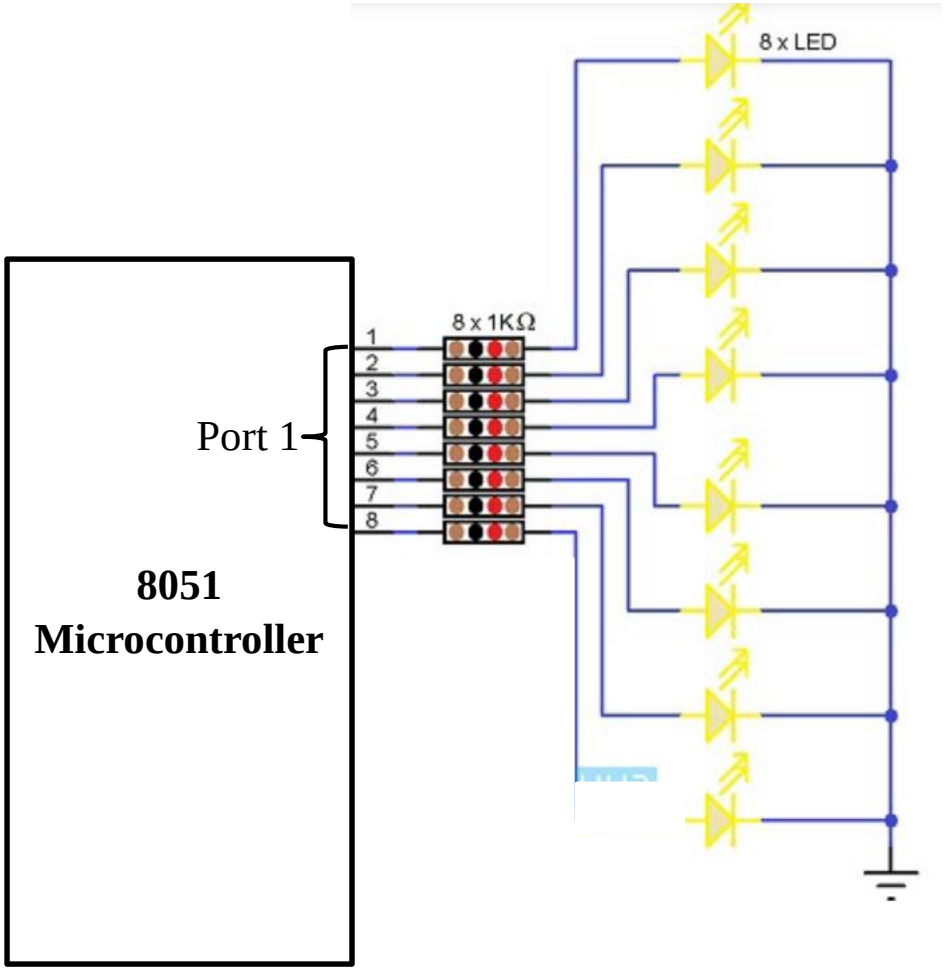


We are writing '0XFF' to turn on all the pins of a particular port of a microcontroller. In each port we have 8 pins that's why we are writing 'FF' which stands for '11111111'.

- For program 1 we have to take inputs from port1 and port 2. That's why we have to make these two ports on by utilizing '0XFF'.
- For program 1 we have to take inputs from port0. In this time output port (port1) should be in 'off' state. We can make it by utilizing '0X00'.

Programming in Embedded C

3. Write a program to Turn on and off (blink) LEDs connected to PORT1 of a 8051 microcontroller with some delay.



Programming in Embedded C

```
#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 */
```

Programming in Embedded C

```
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 – -);
}
}
```