Introduction to 8051 microcontrollers

Posted on May 7, 2008, by Ibrahim KAMAL, in Micro-controllers, tagged

This tutorial is specially tailored to electronics and robotics hobbyists that have already realized some simple electronics projects and want to go a step further and start using microcontrollers in their projects, more precisely the 89S52 microcontroller.

This first part introduce the main aspects and characteristics of the 89S52, providing to the absolute beginners a base of knowledge, which will help them to understand more advanced issues in the next part of the tutorial.

Introduction to micro-controllers

A micro-controller can be compared to a small stand alone computer, it is a very powerful device, which is capable of executing a series of pre-programmed tasks and interacting with other hardware devices. Being packed in a tiny integrated circuit (IC) whose size and weight is usually negligible, it is becoming the perfect controller for robots or any machines requiring some kind of intelligent automation. A single microcontroller can be sufficient to control a small mobile robot, an automatic washer machine or a security system. Any microcontroller contains a memory to store the program to be executed, and a number of input/output lines that can be used to interact with other devices, like reading the state of a sensor or controlling a motor.

Nowadays, microcontrollers are so cheap and easily available that it is common to use them instead of simple logic circuits like counters for the sole purpose of gaining some design flexibility and saving some space. Some machines and robots will even rely on a multitude of microcontrollers, each one dedicated to a certain task. Most recent microcontrollers are 'In System Programmable', meaning that you can modify the program being executed, without removing the microcontroller from its place.

Today, microcontrollers are an indispensable tool for the robotics hobbyist as well as for the engineer. Starting in this field can be a little difficult, because you usually can't understand how everything works inside that integrated circuit, so you have to study the system gradually, a small part at a time, until you can figure out the whole image and understand how the system works.

The 8051 micro-controller architecture

The 8051 is the name of a big family of microcontrollers. The device which we are going to use along this tutorial is the '*AT89S52*' which is a typical 8051 microcontroller manufactured by Atmel™. Note that this part doesn't aim to explain the functioning of the different components of a 89S52 microcontroller, but rather to give you a general idea of the organization of the chip and the available features, which shall be explained in detail along this tutorial.

The block diagram provided by Atmel™ in their datasheet showing the architecture the 89S52 device can seem very complicated, and since we are going to use the C high level language to program it, a simpler architecture can be represented as the ***figure 1.2.A***.

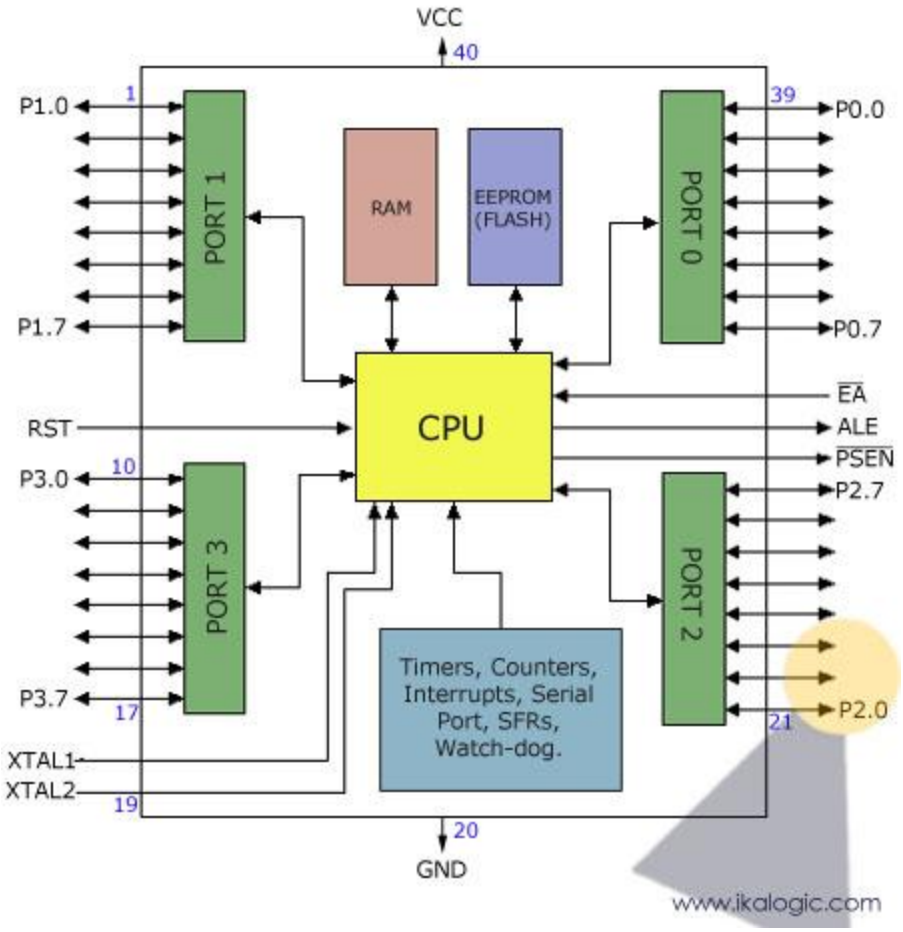

figure 1.2.A

This figures shows the main features and components that the designer can interact with. You can notice that the 89S52 has four different ports, each one having eight Input/output lines providing a total of 32 I/O lines. Those ports can be used to output DATA and orders do other devices, or to read the state of a sensor, or a switch. Most of the ports of the 89S52 have 'dual function' meaning that they can be used for two different functions: the fist one is to perform input/output operations and the second one is used to implement special features of the microcontroller like counting external pulses, interrupting the execution of the program according to external events, performing serial data transfer or connecting the chip to a computer to update the software.

Each port has eight pins, and will be treated from the software point of view as an 8-bit variable called 'register', each bit being connected to a different Input/Output pin.

You can also notice two different memory types: RAM and EEPROM. Shortly, RAM is used to store variable during program execution, while the EEPROM memory is used to store the program itself, that's why it is often referred to as the 'program memory'. The memory organization will be discussed in detail later.

The special features of the 89S52 microcontroller are grouped in the blue box at the bottom of ***figure 1.2.A***. At this stage of the tutorial, it is just important to note that the 89S52 incorporates hardware circuits that can be used to prevent the processor from executing various repetitive tasks and save processing power for more complex calculations. Those simple tasks can be counting the number of external pulses on a pin, or generating precise timing sequences.

It is clear that the CPU (Central Processing Unit) is the heart of the microcontrollers, It is the CPU that will Read the program from the FLASH memory and execute it by interacting with the different peripherals discussed above.

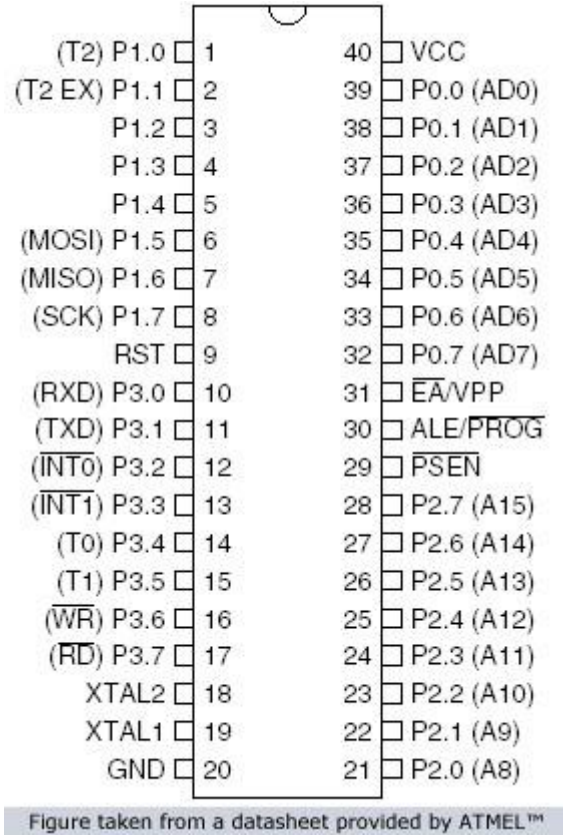

Figure taken from a datasheet provided by ATMEL™

figure 1.2.B

Figure 1.2.B shows the pin configuration of the 89S52, where the function of each pin is written next to it, and, if it exists, the dual function is written between brackets. The pins are written in the same order as in the block diagram of ***figure 1.2.A***, except for the VCC and GND pins which I usually note at the top and the bottom of any device.

Note that the pin that have dual functions, can still be used normally as an input/output pin. Unless you program uses their dual functions, All the 32 I/O pins of the microcontroller are configured as input/output pins.

Most of the function of the pins of the 89S52 microcontroller will be discussed in detail, except for the pins required to control an external memory, which are the pins number 29, 30 and 31. Since we are not going to use any external memory, pins 29 and 30 will be ignored through all the tutorial, and pin 31 (EA) always connected to VCC (5 Volts) to enable the micro-controller to use the internal on chip memory rather than an external one (connecting the pin 31 to ground would indicate to the microcontroller that an external memory is to be used instead of the internal one).

Memory organization

A RAM stands for Random Access Memory, it has basically the same purpose of the RAM in a desktop computer, which is to store some data required during the execution time of different programs. While an EEPROM, also called FLASH memory is a more elaborated ROM (Read Only Memory) which is the memory where the program being executed is stored. Even if that's not exactly true, you can compare an EEPROM to the Hard-Disk of a desktop computer from a general point of view. The EEPROM term stands for Electronically Erasable and Programmable Read Only Memory.

In microcontrollers, like in any digital system, memory is organized in ***Registers***, Which is the basic unit of construction of a memory. Each register is composed of a number of bits (usually eight) where the data can be stored. In the 8051 family of microcontrollers for example, most registers are 8-bit register, capable of storing values ranging from 0 to 255. In order to use bigger values, various register can be used simultaneously. ***Figure 1.3.A*** shows a typical 8-bit registers, where the notation D0 to D7 stands for the 8 DATA bits of the register.

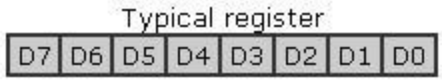

figure 1.3.A

As you shall see, the RAM memory of the 89S52, which contains 256 registers, is divided into to main parts, the GPR part, and the SFR part. GPR stands for 'General Purpose Register' and are the registers that you can use to store any data during the execution of your program. SFRs (Special function Register) are registers used to control the functioning of the microcontroller and to assist the processor through the various operations being executed. For example, SFRs can be used to control Input/Output lines, to retrieve data transmitted through the serial port of a desktop computer, or to configure one of the on-chip counters and timers.

In a memory each register has a specific address which is used by the processor to read and write from specific memory location. ***Figure 1.3.B*** shows the memory organization of the 256 registers of the RAM of the 89S52 microcontroller. The address is noted in Hexadecimal format as this notation simplifies

digital logic calculations for the designers, 00 corresponds to the first location and FF which is equal to 256 corresponds to the last location.

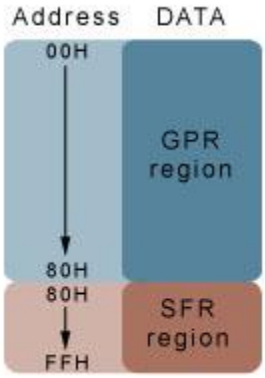

figure 1.3.B

A programmer that would use the assembly language, have to take this memory organization into consideration while choosing the locations where his variables are stored, as writing general purpose data into special function registers could prevent the microcontroller from working correctly, but since we will use the C language using the KEIL IDE (integrated development environment), this part will be totally handled by the compiler.

Clock concept

The clock concept is found in all modern digital electronics, it is a simple circuit that will generate pulses of electricity at a very specific frequency. Those pulses will cadence all the events happening inside a microcontroller, those pulses will also assure the synchronization of the events between various components inside the microcontroller. For example, if the CPU is waiting for some result of mathematical operation from the ALU (Arithmetic and Logic Unit), it will be known – according to very specific protocol – when and where the resulting data will be delivered to the CPU. The synchronization of those two devices is maintained because they share the same clock.

The clock has another very important role which is to enable the microcontroller to count timing. without a precise clock, it would be impossible to build a '*Real Time System*', or any other device that relies on time measurements. It can be deduced that the precision of the timing of a microcontroller depends on the frequency of its clock.

In the 89S52 microcontroller, the clock can be fixed to different value by connecting a crystal to the pins 18 and 19. Those crystals are sold with the frequency written on them in Mega Hertz. The maximum operating frequency of the AT89S52 is 33 Mhz, however other manufacturers like philips built similar 8051 microcontrollers that can run at frequencies up to 120 Mhz.

Life cycle of a micro-controller project

Before passing to the next part of the tutorial, is important to have a general idea of the steps that are followed to realize a project, from the very beginning when you get an idea to the very end when you finalize your project.

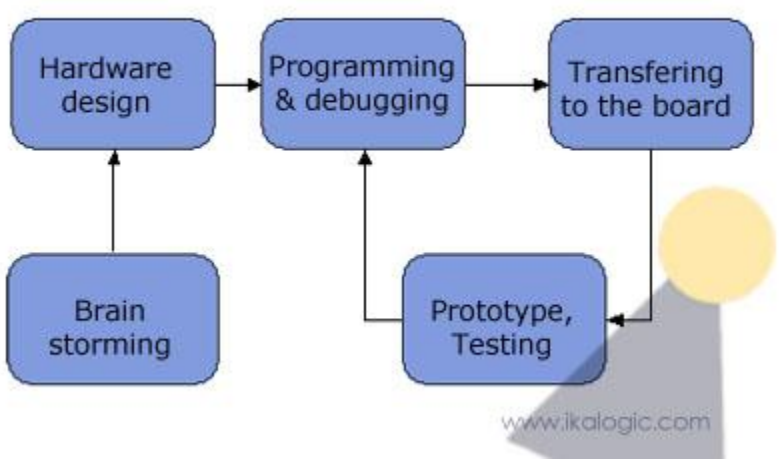

figure 1.5.A

As you can see in ***figure 1.5.A***, after you settle on the choice of your project in the 'brain storming' part, its imperative to imagine how it can be implemented from the hardware point of view, before passing to the programming phase, because programming is much more flexible than the hardware design. In other words, you start by designing the hardware, then you work on the programming while taking in consideration the eventual constraints imposed by the hardware design. The hardware design includes all the aspects of the electronic connections between other devices, like the compatibility of the voltage levels, or the required number of pins, etc…

After you're done with a first version of your program, you can transfer it to the microcontroller mounted on the board that you realized already, resulting in a first prototype. The transfer of the code is done using a special device called 'burner' or 'programmer' that connect to the computer, reads the HEX file generated by the compiler, and sends it to the 'program memory' of the microcontroller. The prototype will be used to test your project, correct eventual errors and enhance its performance, tacking in account the famous rule that states that any project never works the first time, at least it does not work as you expected!

Your project will always stay in the prototyping cycle, even if you decide that it is functioning correctly, simply because perfect machines or inventions do not exist, so there is always some room for little changes and updates.

Source: http://www.ikalogic.com/part-1-introduction-to-8051-microcontrollers/