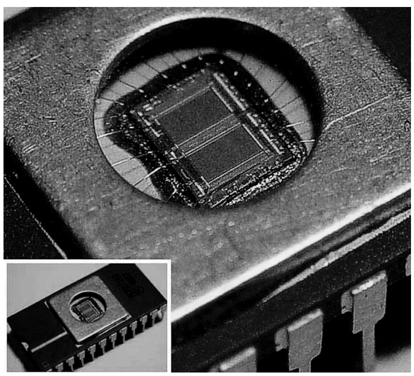
One Time Programmable ROM (OTP ROM)

One time programmable ROM enables you to download a program into it, but, as its name states, one time only. If an error is detected after downloading, the only thing you can do is to download the correct program to another chip.

UV Erasable Programmable ROM (UV EPROM)



Both the manufacturing process and characteristics of this memory are completely identical to OTP ROM. However, the package of the microcontroller with this memory has a recognizable “window” on its top side. It enables data to be erased under strong ultraviolet light. After a few minutes it is possible to download a new program into it.

Installation of this window is very complicated, which normally affects the price. From our point of view, unfortunately- negative...

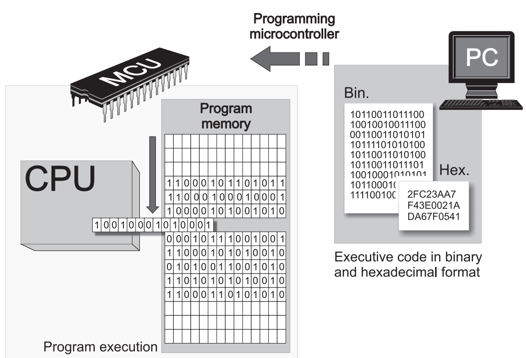
Flash Memory

This type of memory was invented in the 80s in the laboratories of INTEL and was represented as the successor to the UV EPROM. Since the contents of this memory can be written and cleared practically an unlimited number of times, microcontrollers with Flash ROM are ideal for learning, experimentation and small-scale production. Because of its great popularity, most microcontrollers are manufactured in flash technology today. So, if you are going to buy a microcontroller, the type to look for is definitely Flash!

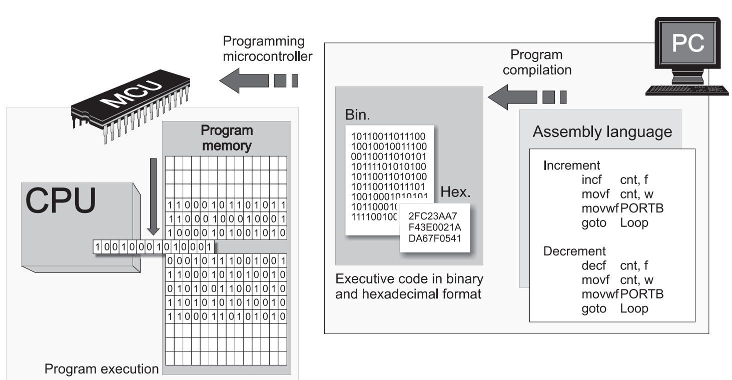
You certainly know that it is not enough just to connect the microcontroller to other components and turn the power supply on to make it work, don't you? There is something else that must be done. The microcontroller needs to be programmed to be capable of performing anything useful. If you think that it is complicated, then you are mistaken. The whole procedure is very simple. Just read the following text and you will change your mind.

2.1 PROGRAMMING LANGUAGES

The microcontroller executes the program loaded in its *Flash* memory. This is the so called executable code comprised of seemingly meaningless sequence of zeros and ones. It is organized in 12-, 14- or 16-bit wide words, depending on the microcontroller's architecture. Every word is considered by the CPU as a command being executed during the operation of the microcontroller. For practical reasons, as it is much easier for us to deal with hexadecimal number system, the executable code is often represented as a sequence of hexadecimal numbers called a *Hex code*. It used to be written by the programmer. All instructions that the microcontroller can recognize are together called the Instruction set. As for PIC microcontrollers the programming words of which are comprised of 14 bits, the instruction set has 35 different instructions in total.



As the process of writing executable code was endlessly tiring, the first "higher" programming language called assembly language was created. The truth is that it made the process of programming more complicated, but on the other hand the process of writing program stopped being a nightmare. Instructions in assembly language are represented in the form of meaningful abbreviations, and the process of their compiling into executable code is left over to a special program on a PC called compiler. The main advantage of this programming language is its simplicity, i.e. each program instruction corresponds to one memory location in the microcontroller. It enables a complete control of what is going on within the chip, thus making this language commonly used today.

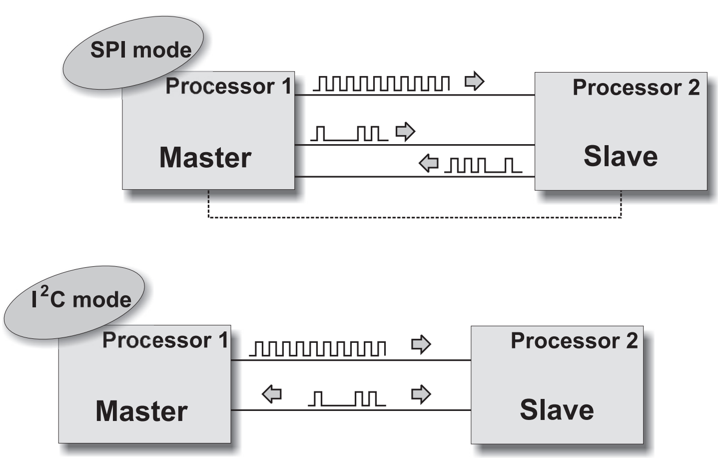


MASTER SYNCHRONOUS SERIAL PORT MODULE

MSSP module (*Master Synchronous Serial Port*) is a very useful, but at the same time one of the most complex circuits within the microcontroller. It enables high speed communication between the microcontroller and other peripherals or other microcontrollers by using few input/output lines (maximum two or three). Therefore, it is commonly used to connect the microcontroller to LCD displays, A/D converters, serial EEPROMs, shift registers etc. The main feature of this type of communication is that it is synchronous and suitable for use in systems with a single master and one or more slaves. A master device contains a circuit for baud rate generation and supplies all devices in the system with the clock. Slave devices may in this way eliminate the internal clock generation circuit. The MSSP module can operate in one out of two modes:

- SPI mode (*Serial Peripheral Interface*); and
- I^2C mode (*Inter-Integrated Circuit*).

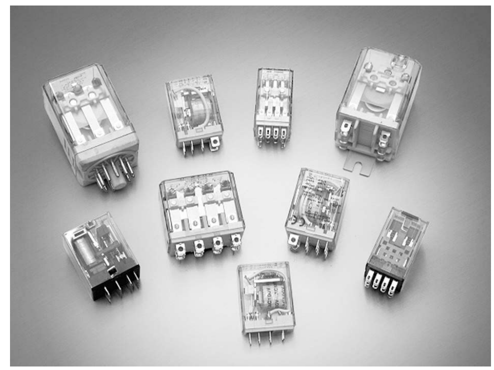
As seen in figure below, one MSSP module represents only a half of the hardware needed to establish serial communication, while the other half is stored in the device it exchanges data with. Even though the modules on both ends of the line are the same, their modes are essentially different depending on whether they operate as a *Master* or a *Slave*:
If the microcontroller to be programmed controls another device or circuit (peripherals), it should operate as a *master* device. It will generate clock when needed, i.e. only when data reception and transmission are required by the software. Obviously, connection establishment depends exclusively on the master device.



Otherwise, if the microcontroller to be programmed is integrated into a more complex device (for example, a PC) then it should operate as a *slave* device. As such, it always has to wait for data transmission request to be sent by the master device.

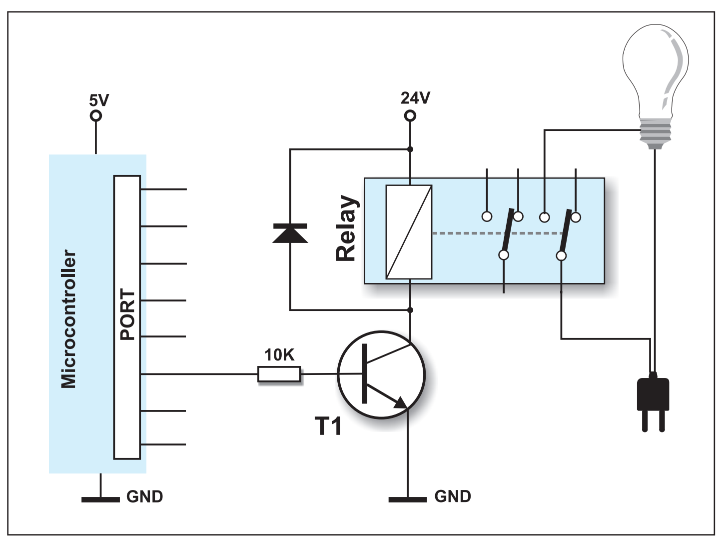
RELAY

A relay is an electrical switch that opens and closes under the control of another electrical circuit. It is therefore connected to output pins of the microcontroller and used to turn on/off high-power devices such as motors, transformers, heaters, bulbs, etc. These devices are almost always placed away from the board's sensitive components. There are various types of relays, but all of them operate in the same way. When current flows through the coil, the relay is operated by an electromagnet to open or close one or many sets of contacts. Similar to optocouplers, there is no galvanic connection (electrical contact) between input and output circuits. Relays usually demand both higher voltage and higher current to start operation, but there are also miniature ones that can be activated by low current directly obtained from a microcontroller pin.



This figure shows the most commonly used solution.

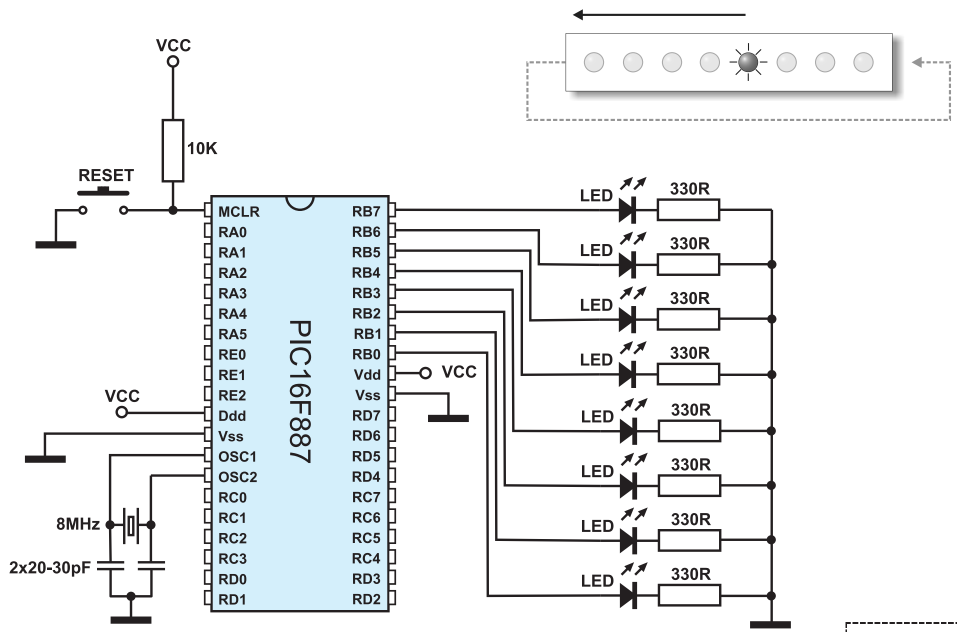
In order to prevent the appearance of high voltage self-induction, caused by a sudden stop of the current flow through the coil, an inverted polarized diode is connected in parallel to the coil. The purpose of this diode is to "cut off" the voltage peak.



EXAMPLE 5

Using watch-dog timer

This example illustrates how the watch-dog timer should not be used. A command used for resetting this timer is intentionally left out in the main program loop, thus enabling it to win the time battle and cause the microcontroller to be reset. As a result, the microcontroller will be reset all the time, which is reflected as PORTB LED blinking.



Example 5

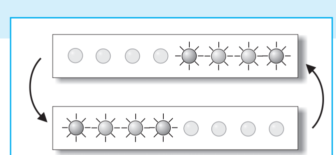
```
/*Header*****************************************************/

void main() {

  OPTION_REG = 0x0E;  // Prescaler is assigned to timer WDT  (1:64)
  asm CLRWDT;         // Assembly command to reset WDT timer
  PORTB = 0x0F;       // Initial value of the PORTB register
  TRISB = 0;          // All port B pins are configured as outputs
  Delay_ms(300);      // 30mS delay
  PORTB = 0xF0;       // Porta B value different from initial

  while (1);          // Endless loop. Program remains here until WDT
                      // timer resets the microcontroller
}
```

In order to make this example work properly, it is necessary to enable the watchdog timer by selecting the ***Watchdog Timer - Enabled*** *option in* ***mE programmer.***

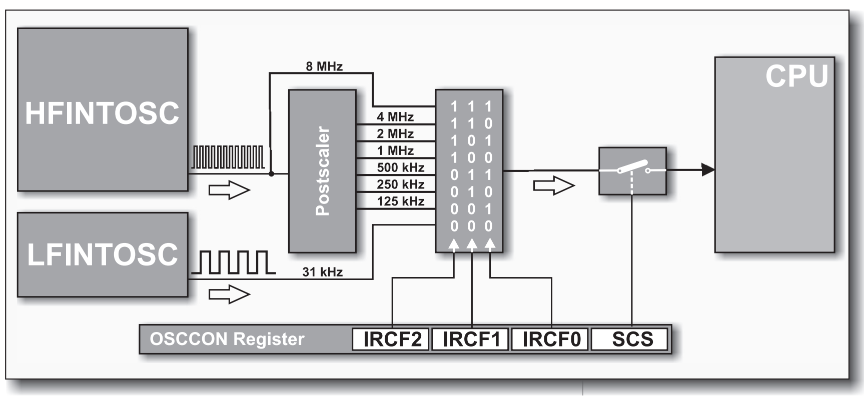


INTERNAL OSCILLATOR SETTINGS

The internal oscillator consists of two separate circuits.

1. The high-frequency internal oscillator HFINTOSC is connected to the postscaler (frequency divider). It is factory calibrated and operates at 8MHz. By using postscaler, this oscillator can output clock sources at one out of seven frequencies. The frequency selection is performed within software using the IRCF2, IRCF1 and IRCF0 pins of the OSCCON register.

The HFINTOSC is enabled by selecting one out of seven frequencies (between 8 MHz and 125 kHz) and setting the System Clock Source (SCS) bit of the OSCCON register. As seen in figure below, everything is performed by using bits of the OSCCON register.



2. The low-frequency oscillator LFINTOSC is uncalibrated and operates at 31 kHz. It is enabled by selecting this frequency (bits of the OSCCON register) and setting the SCS bit of the same register.