AVR Microcontroller

- AVR is referred as Advances Virtual RISC which was produced by Atmel in 1966.
- It supports Harvard Architecture in which program and data is stored in different spaces of microcontroller and can easily be accessed.
- It is considered as earlier types of controllers in which on-chip flash is used for storing program.

AVR Architechture

- » AVR architecture was produced by Vegard Wollan and Alf-Egil Bogen.
- » The AT90S8515 was the first controller that was based on AVR architechture.
- » However, AT90S1200 was the first AVR microcontroller that was commercially available in 1997.
- » The flash, EEPROM and SRAM all are integrated on a single chip, which removes the possibility of joining any external memory with the controller.
- » This controller has a watchdog timer and many power saving sleep modes that make this controller reliable and user-friendly.

Applications

» Peripheral controller of a PC

» Robotics and Embedded systems

» Bio-medical equipment

» Communication and power systems

» Automobiles and security systems

» Implanted medical equipment

» Fire detection devices

» Temperature and light sensing devices

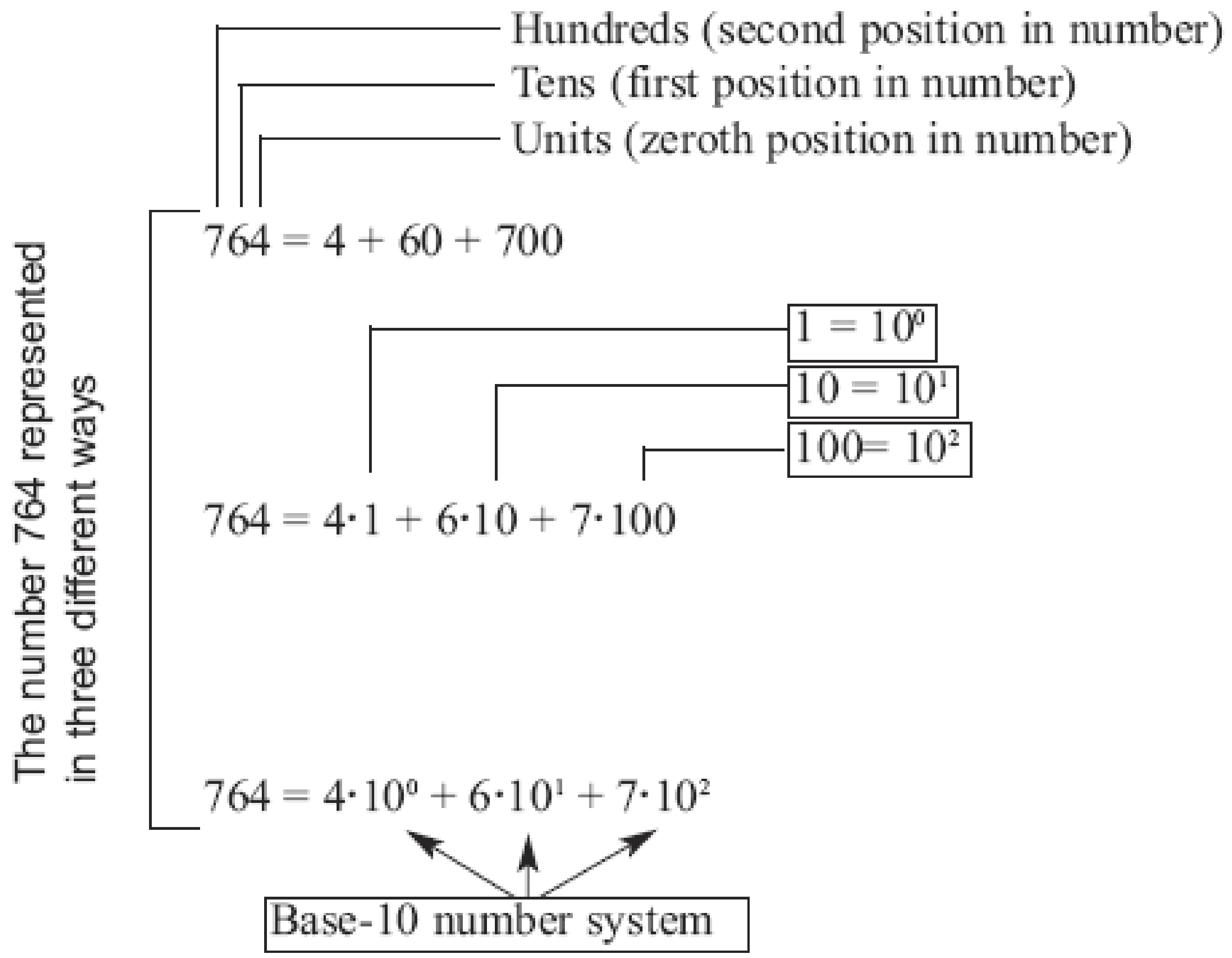
» Industrial automation devices

» Process control devices

» Measuring and Controlling revolving objects

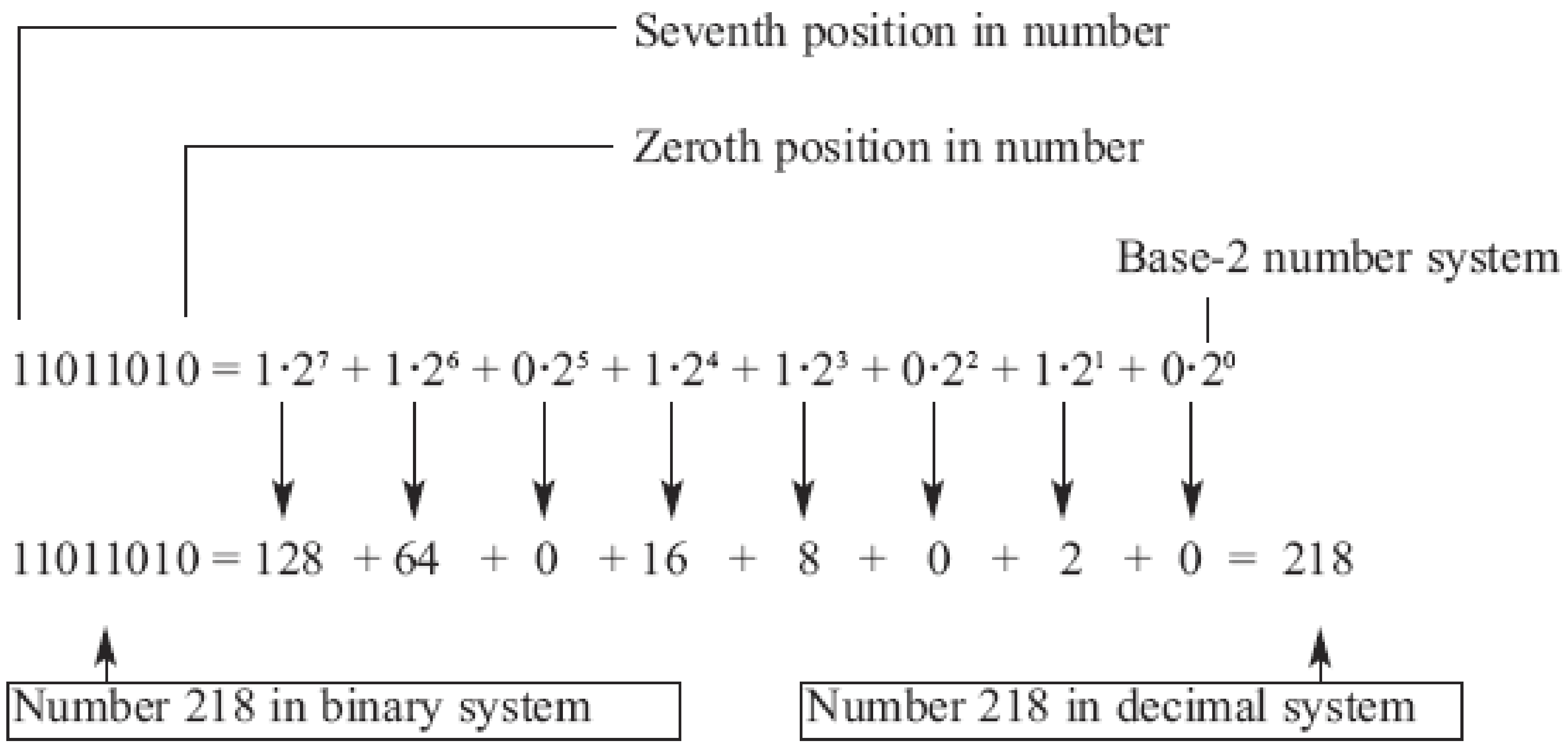
Microcontrollers

Decimal number system.



Hundreds (second position in number)
Tens (first position in number)
Units (zeroth position in number)
764 = 4 + 60 + 700
1 = 10⁰
10 = 10¹
100= 10²
764 = 4·1 + 6·10 + 7·100
764 = 4·10⁰ + 6·10¹ + 7·10²
Base-10 number system
The number 764 represented in three different ways

Binary number system


Seventh position in number
Zeroth position in number
Base-2 number system
$11011010 = 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$
$11011010 = 128 + 64 + 0 + 16 + 8 + 0 + 2 + 0 = 218$
Number 218 in binary system
Number 218 in decimal system

Hexadecimal number system

A hexadecimal number system consisting of digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F has been established.

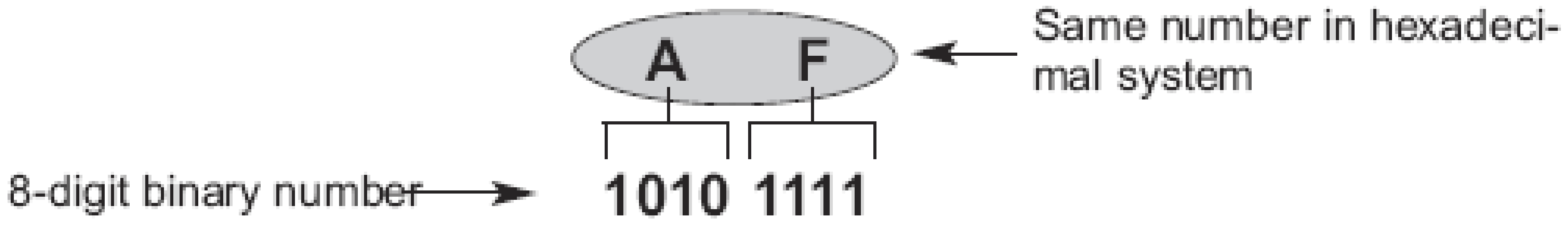


Fig. 0-4 Binary and Hexadecimal number

BCD code

BCD code is actually a binary code for decimal numbers only. It is used to enable electronic circuits to communicate in decimal number system with peripherals and in binary system within “their own world”.

It consists of 4-digit binary numbers which represent the first ten digits (0, 1, 2, 3 ... 8, 9).

Simply, even though four digits can give total of 16 possible combinations, only first ten are used.

Number system conversion

Binary number system is the most commonly used, decimal system is the most understandable while hexadecimal system is somewhere between them.

Therefore, it is very important to learn how to convert numbers from one number system to another, i.e. how to turn series of zeros and units into values understandable for us.

Binary to decimal number conversion

Digits in a binary number have different values depending on their position in that number. Additionally, each position can contain either 1 or 0 and its value may be easily determined by its position from the right. To make the conversion of a binary number to decimal it is necessary to multiply values with the corresponding digits (0 or1) and add all the results. The magic of binary to decimal number conversion works...You doubt? Look at the example:

110 = 1*2^2 + 1*2^1 + 0*2^0 = 6

Hexadecimal to decimal number conversion

In order to make conversion of a hexadecimal number to decimal, each hexadecimal digit should be multiplied with the number 16 raised by it’s position value. For example:

A37E (number in hexadecimal system)

$$\begin{aligned}
14\cdot 16^0 &= 14\cdot 1 &&= 14\\
7\cdot 16^1 &= 7\cdot 16 &&= 112\\
3\cdot 16^2 &= 3\cdot 256 &&= 768\\
10\cdot 16^3 &= 10\cdot 4096 &&= \underline{40960}
\end{aligned}$$

41854 (same number in decimal system)

Hexadecimal to binary number conversion

It is not necessary to perform any calculation in order to convert hexadecimal number to binary number system. Hexadecimal digits are simply replaced by the appropriate four binary digits.

Since the maximal hexadecimal digit is equivalent to decimal number 15, it is needed to use four binary digits to represent one hexadecimal digit. For example:

$$E4 = \underbrace{1110}_{E}\underbrace{0100}_{4}$$

Fig. 0-6 Hexadecimal to binary number conversion

DEC.	BINARY							HEX.
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	1
2	0	0	0	0	0	0	1	0	2
3	0	0	0	0	0	0	1	1	3
4	0	0	0	0	0	1	0	0	4
5	0	0	0	0	0	1	0	1	5
6	0	0	0	0	0	1	1	0	6
7	0	0	0	0	0	1	1	1	7
8	0	0	0	0	1	0	0	0	8
9	0	0	0	0	1	0	0	1	9
10	0	0	0	0	1	0	1	0	A
11	0	0	0	0	1	0	1	1	B
12	0	0	0	0	1	1	0	0	C
13	0	0	0	0	1	1	0	1	D
14	0	0	0	0	1	1	1	0	E
15	0	0	0	0	1	1	1	1	F
16	0	0	0	1	0	0	0	0	10
17	0	0	0	1	0	0	0	1	11

......

......

......

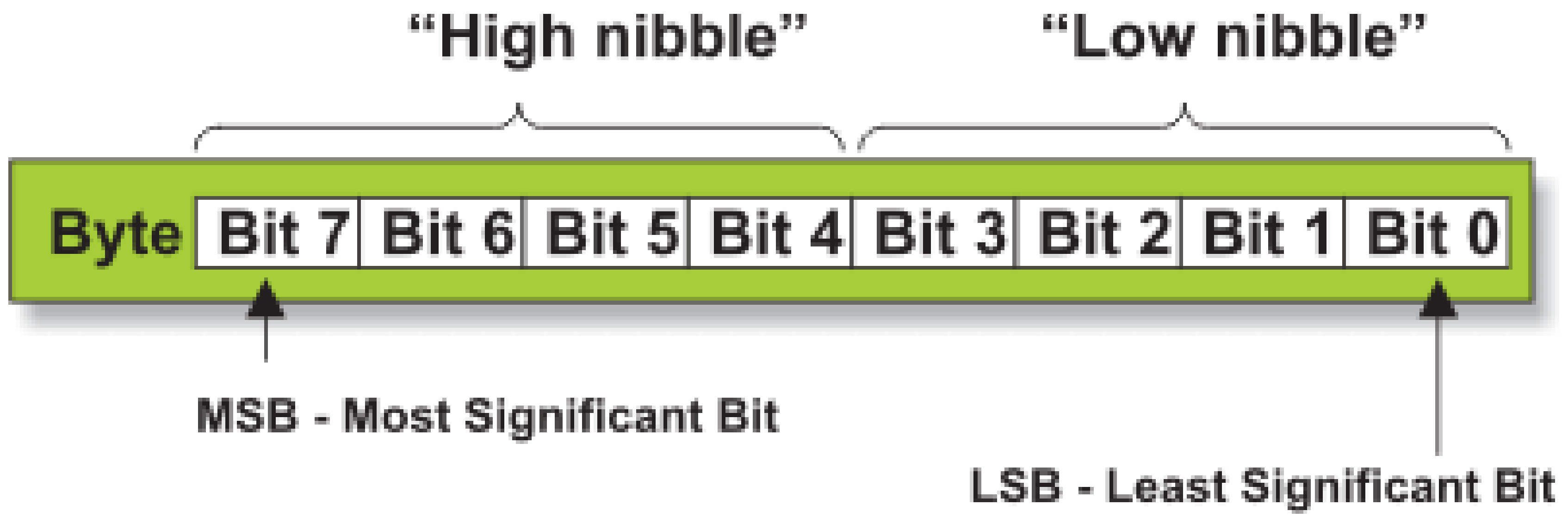
253	1	1	1	1	1	1	0	1	FD
254	1	1	1	1	1	1	1	0	FE
255	1	1	1	1	1	1	1	1	FF

Marking Numbers

The hexadecimal numbering system is along with binary and decimal number systems considered to be the most important for us. It is easy to make conversion of any hexadecimal number to binary and it is also easy to remember it. However, these conversions may cause confusion. For example, what does the statement “It is necessary to count up 110 products on assembly line” actually mean? Depending on whether it is about binary, decimal or hexadecimal, the result could be 6, 110 or 272 products, respectively! Accordingly, in order to avoid misunderstanding, different prefixes and suffixes are directly added to the numbers. The prefix $ or 0x as well as the suffix h marks the numbers in hexadecimal system. For example, hexadecimal number 10AF may look as follows $10AF, 0x10AF or 10AFh. Similarly, binary numbers usually get the suffix % or 0b, whereas decimal numbers get the suffix D.

Bit

Byte



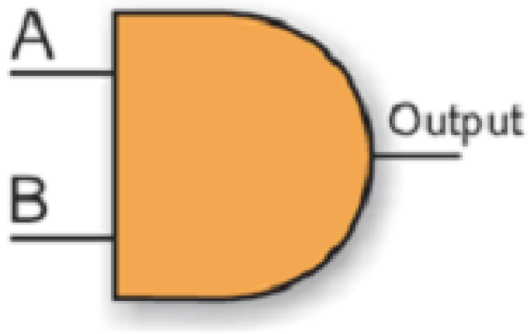
"High nibble"
"Low nibble"
Byte
Bit 7
Bit 6
Bit 5
Bit 4
Bit 3
Bit 2
Bit 1
Bit 0
MSB - Most Significant Bit
LSB - Least Significant Bit

AND gate

A logic gate "AND" has two or more inputs and one output. Let us presume that the gate used in this case has only two inputs. A logic one (1) will appear on its output only in case both inputs (A AND B) are driven to logic one (1). That's all!

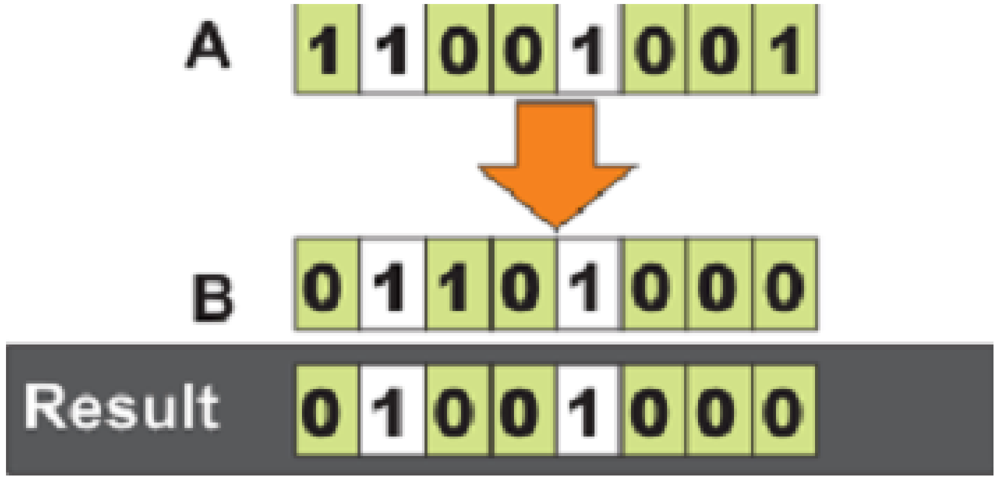
Schematic symbol of AND gate is shown in the figure on the right.

Additionally, the table shows mutual dependence between inputs and output.



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

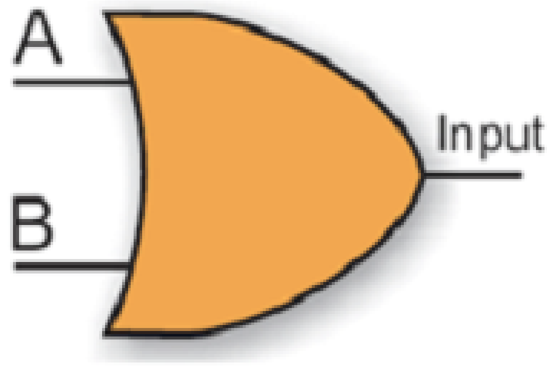
In case the gate has more than two inputs, the principle of operation is the same: a logic one (1) will appear on its output only in case all inputs are driven to logic one (1). Any other combination of input voltages will result in logic zero (0) on its output.



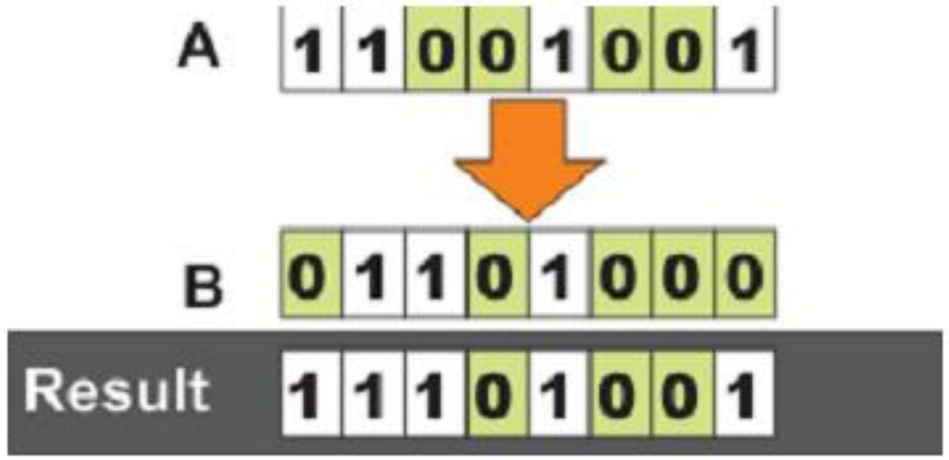
When used in a program, logic AND operation is performed by the program instruction, which will be discussed later. For the time being, it is enough to remember that logic AND in a program refers to the corresponding bits of two registers.

OR gate

Similar to the previous case, OR gate also has two or more inputs and one output. The gate with only two inputs will be considered in this case as well. A logic one (1) will appear on its output in case either one or another output (A OR B) is driven to logic one (1). In case the OR gate has more than two inputs, the following applies: a logic one (1) appears on its output in case at least one input is driven to logic one (1). In case all inputs are driven to logic zero (0), the output will be driven to logic zero (0).



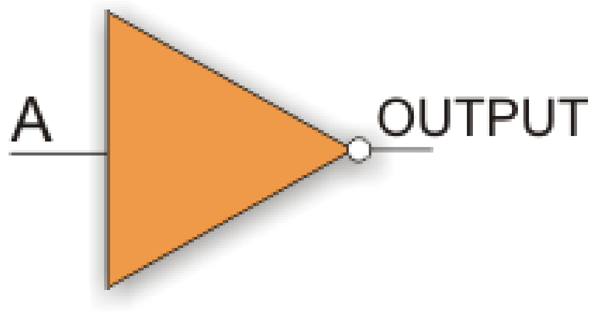
A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



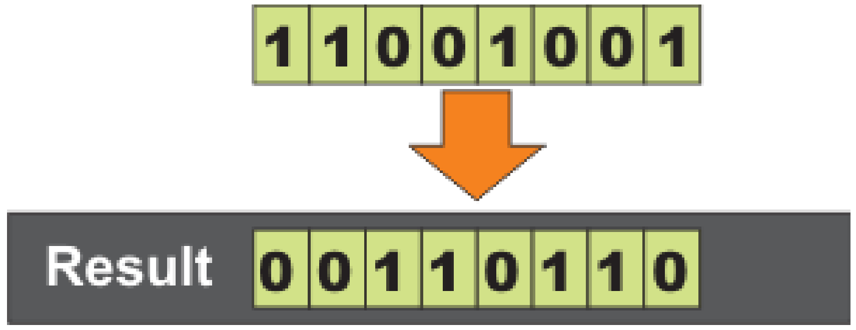
In a program, logic OR operation is performed between the corresponding registers' bits- the same as in logic AND operation.

NOT Gate

This logic gate has only one input and only one output. It operates in an extremely simple way. When logic zero (0) appears on its input, a logic one (1) appears on its output and vice versa. This means that this gate inverts the signal by itself. It is sometimes called inverter.



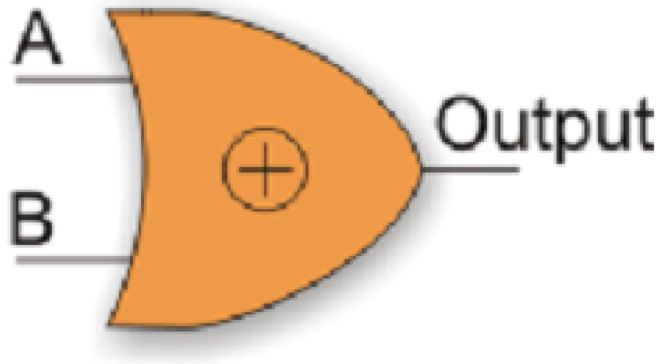
A	Output
0	1
1	0

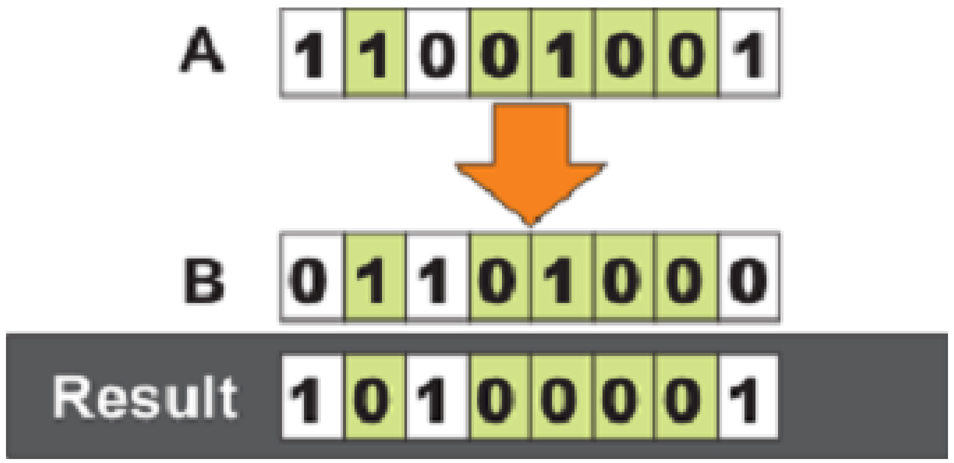


If a program, logic NOT operation is performed on one byte. The result is a byte with inverted bits. If byte is considered to be a number, the inverted value is actually a complement of that number, i.e. the complement of a number is what is needed to add to it to make it reach the maximal 8 bit value (255).

EXCLUSIVE OR gate

This gate is a bit complicated comparing to other gates. It represents combination of all previously described gates. It is not simple to define mutual dependence of input and output, but we will anyway try to do it. A logic one (1) appears on its output only in case the inputs have different logic states.

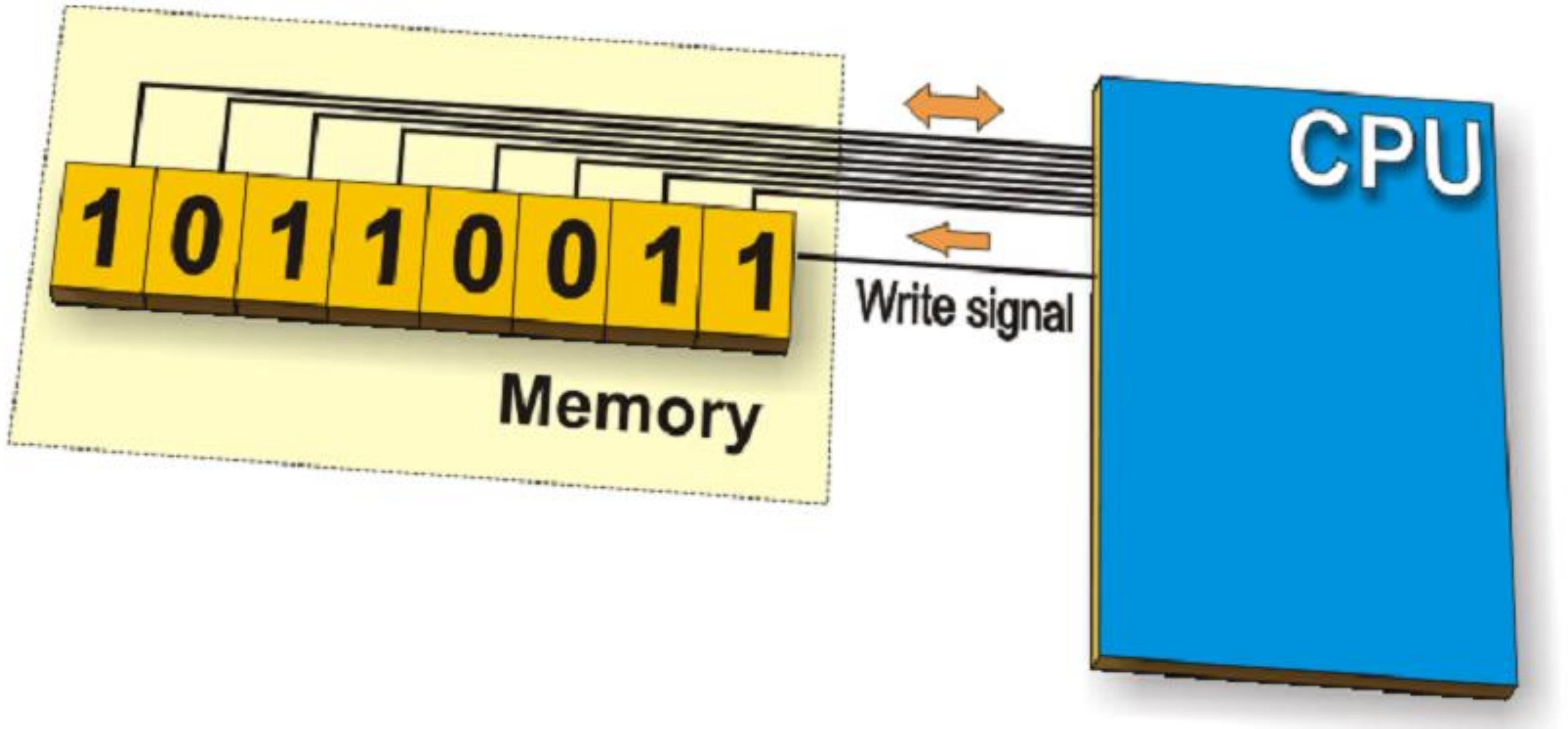




In a program, this operation is commonly used to compare two bytes. Subtraction may be used for the same purpose (if the result is 0, bytes are equal). The advantage of this logic operation is that there is no danger to subtract larger number from smaller one.

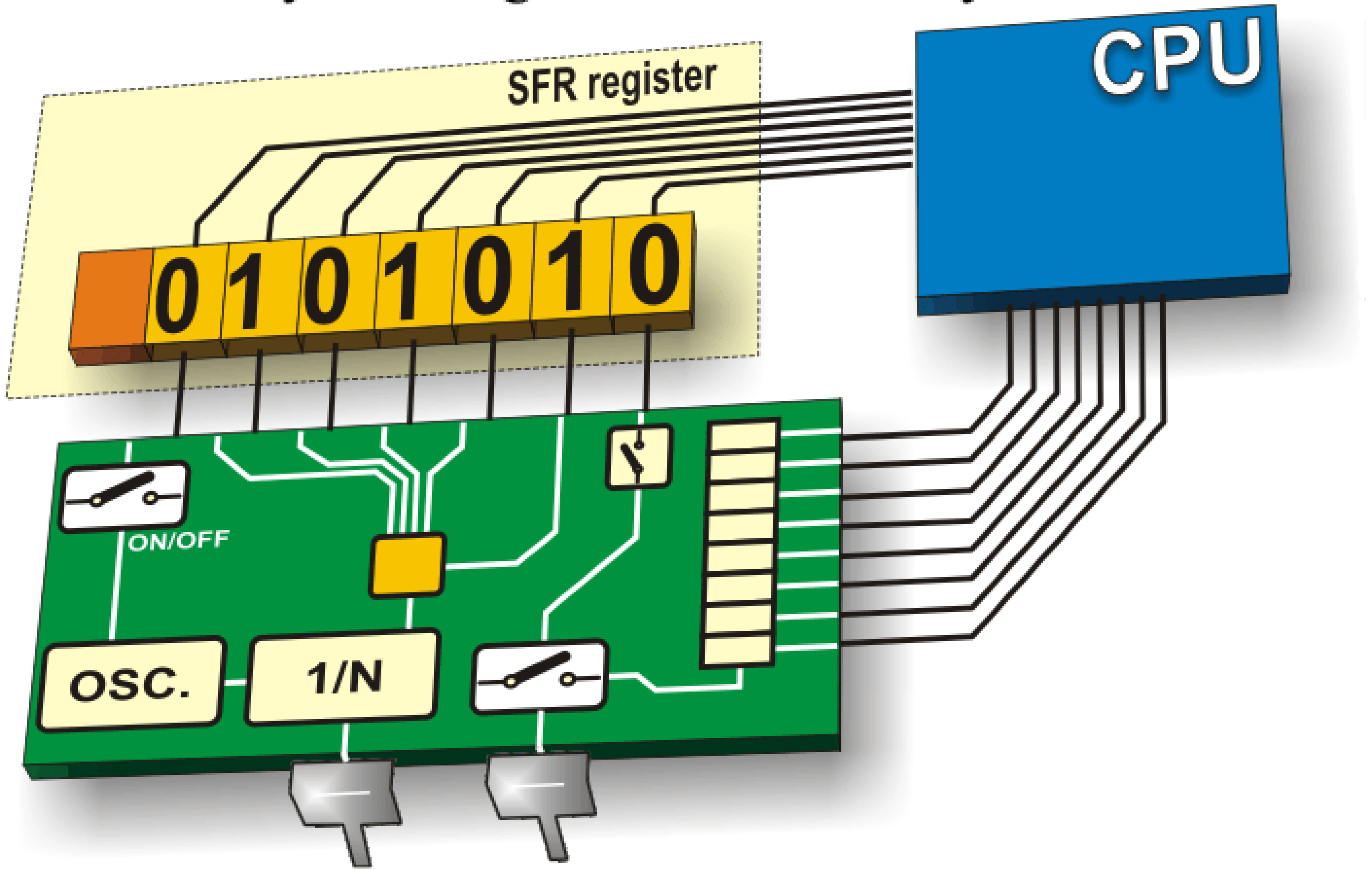
Register

A register or a memory cell is an electronic circuit which can memorize the state of one byte. In other words, what is a byte theoretically, it is a register practically.



SFR registers

In addition to the registers which do not have any special and predetermined function, every microcontroller has also a number of registers whose function is predetermined by the manufacturer. Their bits are connected (literally) to internal circuits such as timers, A/D converter, oscillators and others, which means that they are directly in command of the operation of the microcontroller. If you imagine that as eight switches which are in command of some smaller circuit within the microcontroller- you are right! SFRs do exactly that!



Input / Output ports

In order that the microcontroller is of any use, it has to be connected to additional electronics, i.e. peripherals. For that reason, each microcontroller has one or more registers (called “port” in this case) connected to the microcontroller pins. Why input/output? Becuse you can change the pin’s function as you wish. For example, suppose you want your device to turn on and off three signal LEDs and simultaneously monitor logic state of five sensors or push buttons. In accordance with that, some of ports should be configured so that there are three outputs (connected to LEDs) and five inputs (connected to sensors). It is simply performed by software, which means that pin’s function can be changed during operation.

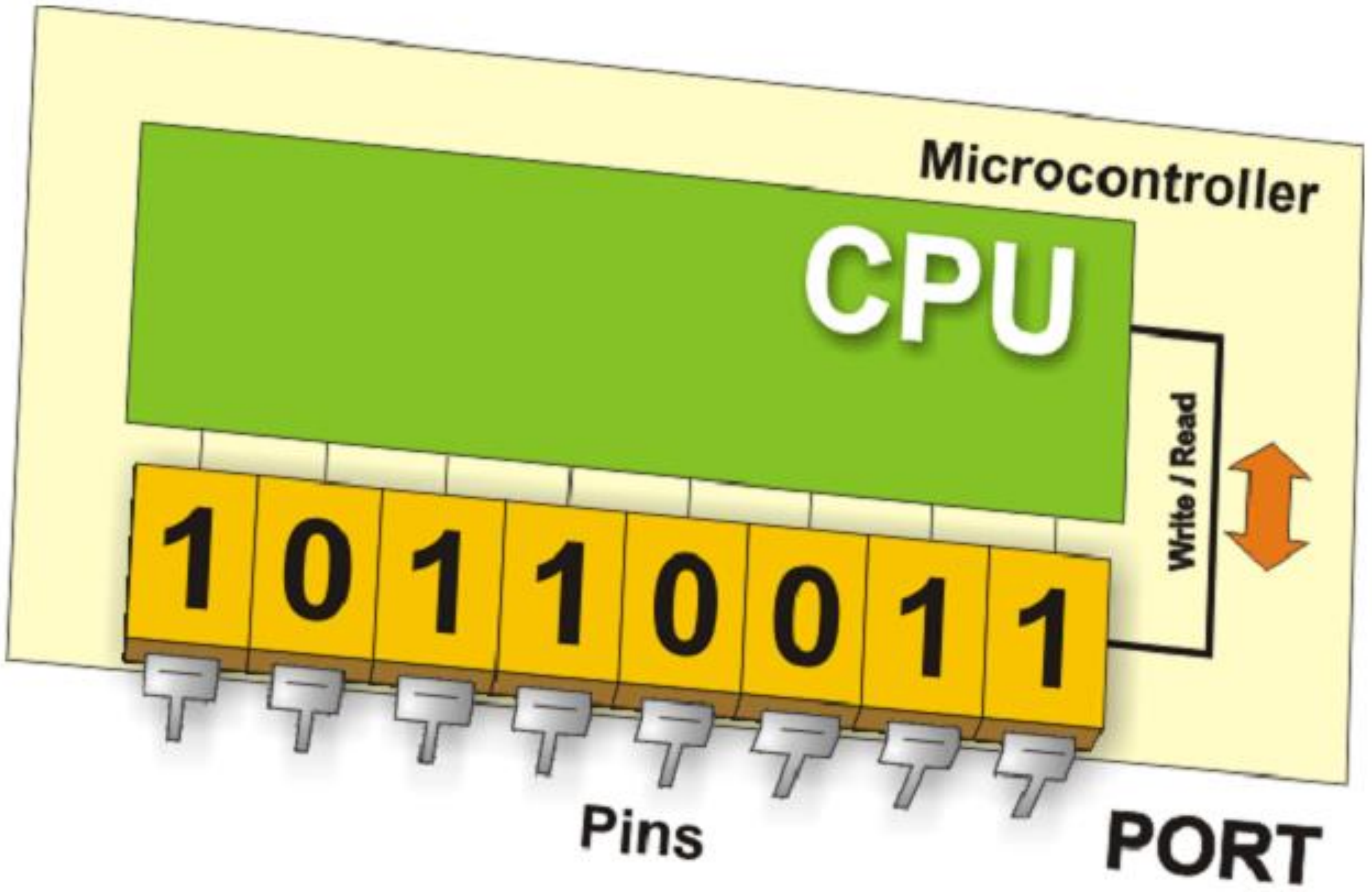


Fig. 0-19 Input / Output ports

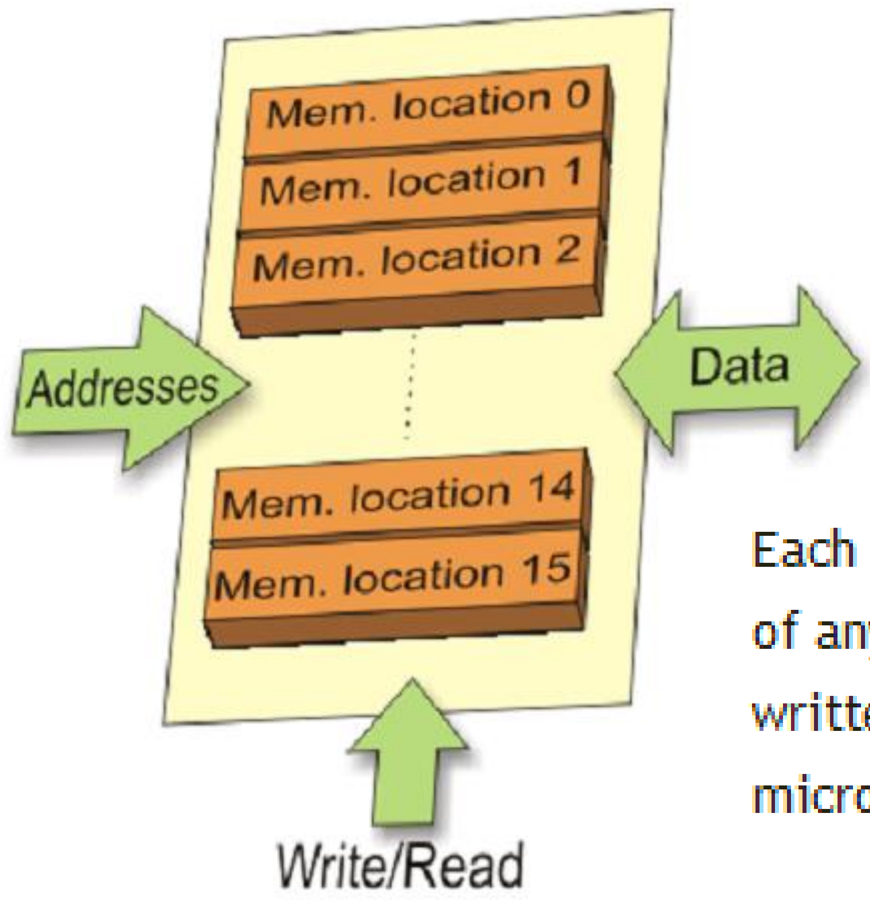
One of more important feature of I/O pins is maximal current they can give/get. For the most microcontrollers, current obtained from one pin is sufficient to activate a LED or other similar low-current consumer (10-20 mA). If the microcontroller has many I/O pins, then maximal current of one pin is lower. Simply, you cannot expect all pins to give maximal current if there are more than 80 of them on one microcontroller.

Another important pin feature is to (not) have pull-up resistors. These resistors connect pin to positive power supply voltage and their effect is visible when the pin is configured as input connected to mechanical switch or push button. The later versions of the microcontrollers have pull-up resistors connected to and disconnected from the pins by software.

Usually, each I/O port is under control of another SFR, which means that each bit of that register determines state of the corresponding microcontroller pin. For example, by writing logic one (1) to one bit of that control register SFR, the appropriate port pin is automatically configured as intput. It means that voltage brought to that pin can be read as logic 0 or 1. Otherwise, by writing zero to the SFR, the appropriate port pin is configured as output. Its voltage (0V or 5V) corresponds to the state of the appropriate bit of the port register.

Memory unit

Memory is part of the microcontroller used for data storage. The easiest way to explain it is to compare it with a big closet with many drawers. Suppose, the drawers are clearly marked so that it is easy to access any of them. It is enough to know the drawer's mark to find out its contents.



Each memory address corresponds to one memory location. The content of any location becomes known by its addressing. Memory can either be written to or read from. There are several types of memory within the microcontroller.

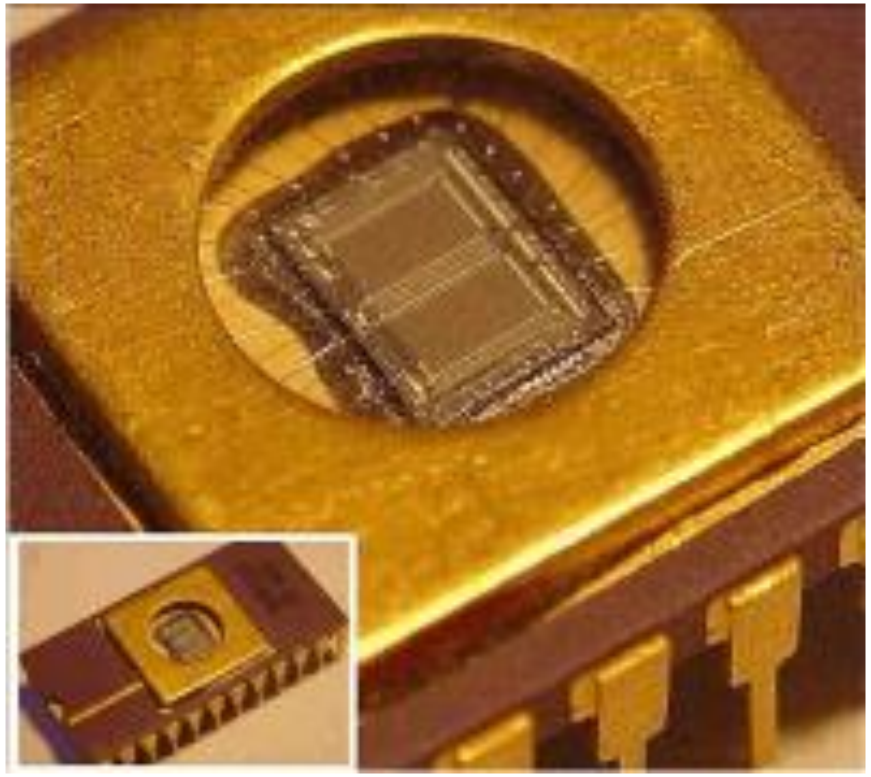
Read Only Memory (ROM)

ROM (Read Only Memory) is used to permanently save the program being executed. The size of a program that can be written depends on the size of this memory. Today's microcontrollers commonly use 16-bit addressing, which means that they are able to address up to 64 Kb of memory, i.e. 65535 locations. As a novice, your program will rarely exceed the limit of several hundred instructions. There are several types of ROM.

Masked ROM. Microcontrollers containing this ROM are reserved for the great manufacturers. Program is loaded into the chip by the manufacturer. In case of large scale manufacture, the price is very low. Forget it...

One Time Programmable ROM (OTP ROM). If the microcontroller contains this memory, you can download a program into this memory, but the process of program downloading is a "one-way ticket", meaning that it can be done only once. If an error is detected after downloading, the only thing you can do is to download the corrected 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 this microcontroller has a recognizable "window" on the upper side. It enables the surface of the silicon chip inside to be lit by an UV lamp, which effectively erases and program from the ROM.
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 were represented as the successor to th UV EPROM. Since the contents of this memory can be written and cleared practically an unlimited number of times, the microcontrolle with Flash ROM are ideal for learning, experimentation and small-scale manufacture. Because of its popularity, the most microcontrollers are manufactured in flash versions today. So, if you are going to buy a microcontroller, the type to look for is definite Flash!

Random Access Memory (RAM)

Once the power supply is off the contents of RAM (Random Access Memory) is cleared. It is used for temporary storing data and intermediate results created and used during the operation of the microcontroller. For example, if the program performs an addition (of whatever), it is necessary to have a register representing what in everyday life is called the "sum". For that purpose, one of the registers in RAM is called the "sum" and used for storing results of addition.

Electrically Erasable Programmable ROM (EEPROM)

The contents of the EEPROM may be changed during operation (similar to RAM), but remains permanently saved even upon the power supply goes off (similar to ROM). Accordingly, an EEPROM is often used to store values, created during operation, which must be permanently saved. For example, if you design an electronic lock or an alarm, it would be great to enable the user to create and enter a password, but useless if it is lost every time the power supply goes off. The ideal solution is the microcontroller with an embedded EEPROM.

Interrupt

The most programs use interrupts in regular program execution. The purpose of the microcontroller is mainly to react on changes in its surrounding. In other words, when some event takes place, the microcontroller does something... For example, when you push a button on a remote controller, the microcontroller will register it and respond to the order by changing a channel, turn the volume up or down etc. If the microcontroller spent most of its time endlessly a few buttons for hours or days... It would not be practical.

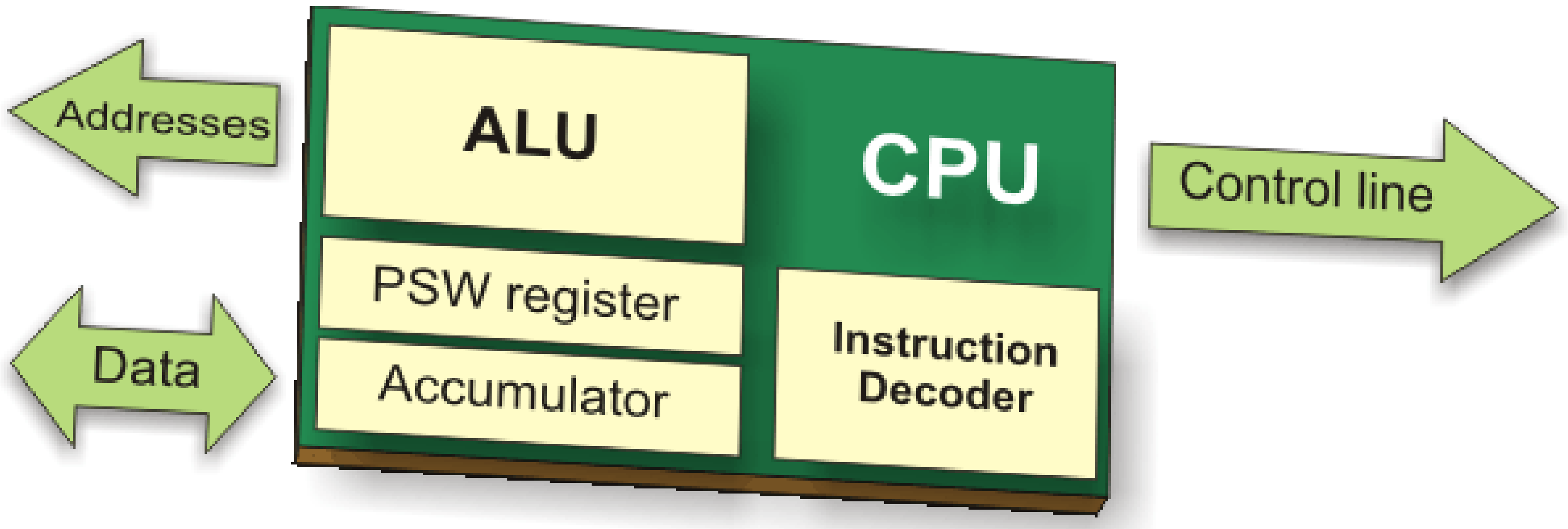
The microcontroller has learnt during its evolution a trick. Instead of checking each pin or bit constantly, the microcontroller delegates the “wait issue” to the “specialist” which will react only when something attention worthy happens.

The signal which informs the central processor about such an event is called an INTERRUPT.

Central Processor Unit (CPU)

As its name suggests, this is a unit which monitors and controls all processes inside the microcontroller. It consists of several smaller subunits, of which the most important are:

- **Instruction Decoder** is a part of the electronics which recognizes program instructions and runs other circuits on the basis of that. The "instruction set" which is different for each microcontroller family expresses the abilities of this circuit.
- **Arithmetical Logical Unit (ALU)** performs all mathematical and logical operations upon data.
- **Accumulator** is a SFR closely related to the operation of the ALU. It is a kind of working desk used for storing all data upon which some operation should be performed (addition, shift/move etc.). It also stores the results ready for use in further processing. One of the SFRs, called a Status Register (PSW), is closely related to the accumulator. It shows at any given moment the "status" of a number stored in the accumulator (number is greater or less than zero etc.).



Bus

Physically, the bus consists of 8, 16 or more wires. There are two types of buses: the address bus and the data bus. The address bus consists of as many lines as necessary for memory addressing. It is used to transmit the address from the CPU to the memory. The data bus is as wide as the data, in our case it is 8 bits or wires wide. It is used to connect all circuits inside the microcontroller.

Serial Communication

Parallel connections between the microcontroller and peripherals via input/output ports is the ideal solution for shorter distances- up to several meters. However, in other cases - when it is necessary to establish communication between two devices on longer distances it is not possible to use a parallel connection - such a simple solution is out of question. In these situations, serial communication is the best solution.

Today, most microcontrollers have built in several different systems for serial communication as a standard equipment. Which of these systems will be used depends on many factors of which the most important are:

- How many devices the microcontroller has to exchange data with?
- How fast the data exchange has to be?
- What is the distance between devices?
- Is it necessary to send and receive data simultaneously?

One of the most important things concerning serial communication is the *Protocol* which

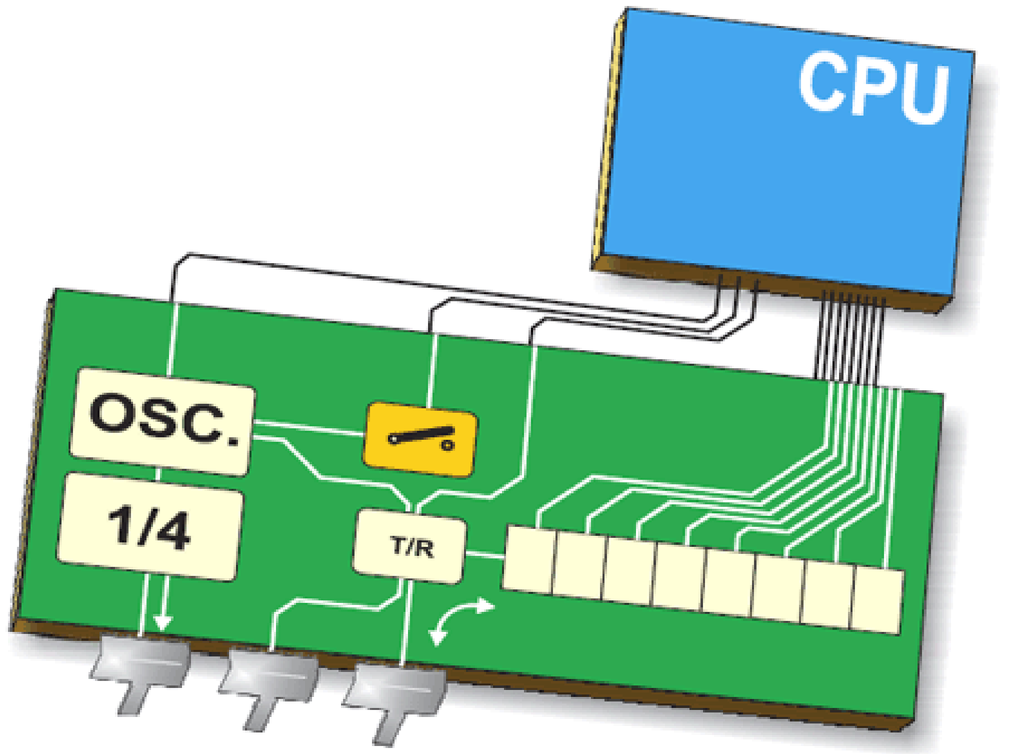


Fig. 0-23 Serial communication

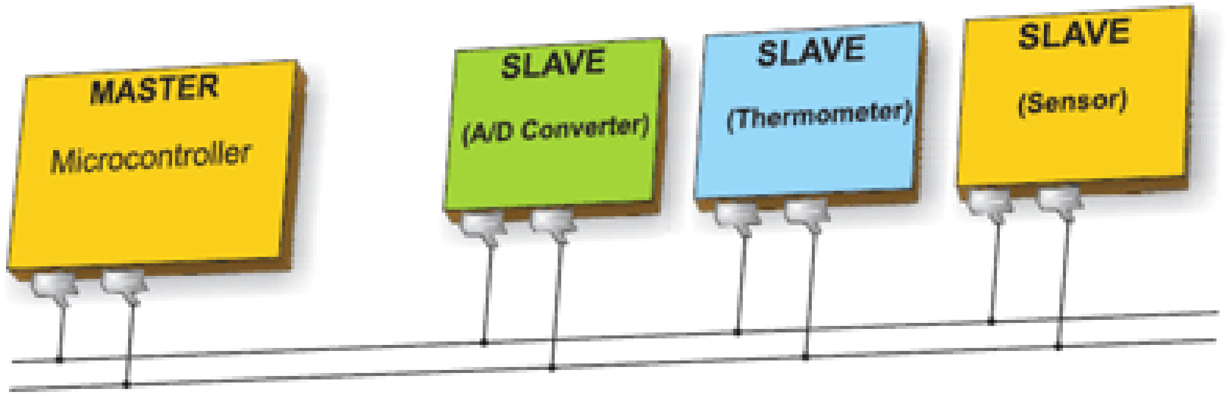
should be strictly observed. It is a set of rules which must be applied in order that the devices can correctly interpret data they mutually exchange. Fortunately, the microcontrollers automatically take care of this, so the work of the programmer/user is reduced to simple write (data to be sent) and read (received data).

Baud Rate

The term *Baud rate* is commonly used to denote the number of bits transferred per second [bps].

It should be noted that it refers to bits, not bytes! It is usually required by the protocol that each byte is transferred along with several control bits. It means that one byte in serial data stream may consist of 11 bits. For example, if the baud rate is 300 bps then maximum 37 and minimum 27 bytes may be transferred per second, which depends on type of connection and protocol in use.

The most commonly used serial communication systems are:



I2C (Inter Integrated Circuit) is a system used when the distance between the microcontrollers is short and specialized integrated circuits of of a new generation (receiver and transmitter are usually on the same printed circuit board). Connection is established via two conductors- one is used for data transfer whereas another is used for synchronization (clock signal). As seen in figure, one device is always the master. It performs addressing of one slave chip (subordinated) before communication starts. In this way one microcontroller can communicate with 112 different devices. Baud rate is usually 100 Kb/sec (standard mode) or 10 Kb/sec (slow baud rate mode). Systems with the baud rate of 3.4 Mb/sec have recently appeared. The distance between devices which communicate via an inter-integrated circuit bus is limited to several meters.