**Microcontroller and applications**

**Unit-1**

**Over view of architecture and microcontroller resources**

**Introduction of microprocessor:** A microprocessor is a central processing unit (CPU) that contains the entire processing system of a computer on a single integrated circuit (IC) or chip. It is the brain of a computer and performs all the calculations and executes instructions that a computer receives**.**

**Types of Microprocessors:**

1. CISC (Complex Instruction Set Computing): CISC microprocessors use complex instructions that perform multiple operations in a single clock cycle.

2. RISC (Reduced Instruction Set Computing): RISC microprocessors use simple instructions that perform a single operation in a single clock cycle.

3. DSP (Digital Signal Processing): DSP microprocessors are specialized for processing digital signals and are used in applications such as audio and image processing.

4. Embedded Microprocessors: Embedded microprocessors are used in embedded systems and are designed for specific applications such as automotive control systems and medical devices.

**Microcontroller:** converting microprocessor to a microcontroller involves adding additional components and peripherals to the microprocessor to make it a set. contain system

1. **Adding Memory**: A microcontroller typically includes on-chip memory, such as ROM, RAM, and EEPROM. This memory is used to store the program, data, and configuration settings.

2. **Adding Input/Output (I/O) Peripherals**: Microcontrollers often include a range of I/O peripherals, such as GPIO (General Purpose Input/Output) ports, UARTs (Universal Asynchronous Receiver-Transmitters), SPI (Serial Peripheral Interface), and I2C (Inter-Integrated Circuit) interfaces. These peripherals allow the microcontroller to interact with external devices and sensors.

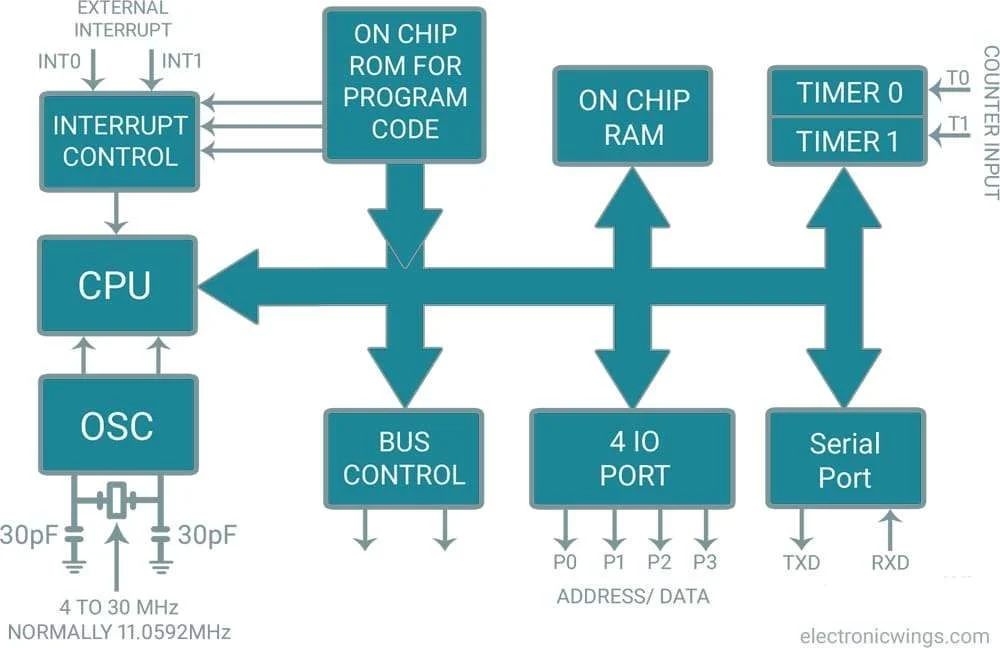
3. **Adding Timers and Counters:** Microcontrollers often include timers and counters that can be used to generate timing signals, count events, and perform other timing-related tasks.

4**. Adding Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs**): Many microcontrollers include ADCs and DACs that allow them to interface with analog sensors and actuators.

**5. Adding Communication Interfaces:** Microcontrollers may include communication interfaces, such as USB, Ethernet, or wireless interfaces, that allow them to communicate with other devices and systems.

6. **Adding a Clock Generator**: Microcontrollers often include a clock generator that provides a stable clock signal to the system.

7. **Adding Power Management**: Microcontrollers may include power management features, such as voltage regulators, power-on reset, and sleep modes, that allow them to manage power consumption and extend battery life.

**Architecture of microcontroller:** 

**CPU**- Central Processing Unit comprising of ALU and Control units

ALU-Arithmetic and Logic Unit performing the arithmetic and logical operations. These operations are addition, subtraction, multiplication, logical AND, OR etc. To do these operations one operand should be in Accumulator, another may in B register or in general purpose register. Mostly the result of the ALU operations is in A register. Some results are in B register also

**OSC**

Oscillator provides clock for controller operation. Crystal oscillator provides stability and perfect clock. So crystal oscillator is used in this microcontroller. The crystal connected to the pins are intended for this purpose.

**INTERRUPT CONTROLLER**

Some interrupts are needed for microcontroller operation. Five interrupts are used. The controller controls the operation interrupts. ie some interrupts may be allowed, some others are disabled and priority assigned and changed.

**BUS CONTROL**

In 8051, Data Bus has a width of 8 bits and Address Bus has a width of 16 bits. Lower byte address bus is used for both Address and data. The bus usage is controlled by BUS control. There are 3 control signals, EA, PSEN and ALE. These signals known as External Access (EA), Program Store Enable (PSEN), and Address Latch Enable (ALE) are used for external memory interfacing.

**ON CHIP RAM**

The 8051 has 4 kilobytes of inbuilt ROM. It is otherwise called program memory. Usually program-code is stored in ROM. To store program into ROM, programmer is needed. If more area is required in ROM, an external ROM may be connected. Maximum of 64kb ROM memory can be used.

**ON CHIP ROM**

The 8051 has 128 bytes of RAM as inbuilt. Some versions have 256 bytes also. It is used as data memory. If the system needs more memory, external RAM may be connected up to 64kb.In 128-byte RAM chip, 00h to 71Fh are the address range. In this range ,00H to 1FH are the general-purpose registers. 20H to 2F are the bit Addressable area and rest of this are byte addressable. This is used as general-purpose scratch pad. In 256-byte RAM chip, another 128 bytes are used for Special Function Registers

**I/O PORTS**

There are four IO ports in 8051.These are named as P0,P1 ,P2,P3.All are bidirectional. Each port have its address , latch, output driver and input buffer. Each port are output by default. To make the port as input, it should be initialized by sending ‘1’ level in each pin.

**SERIAL PORT**

TXD and RXD are used for serial port. These pins are available in Port 3. To transmit data serially TXD pin **is used. To receive data serially RXD pin is used .Each pin have separate buffer registers named SBUF.**

**TIMER/COUNTER**

There are two types counter/timer in 8051.These are named as Timer/Counter0 and Timer/Counter1. Each one may be used either as Timer or Counter. 16 bit timer register are used for counting in timer operation or counter operation. Clock pulses are counted in Timer operation and external events are counted in counter operation

**MICROCONTROLLER RESOURCES:**

**1. Hardware Resources**

Microcontroller Development Boards

* Arduino: Widely used for hobbyists and prototyping (e.g., Arduino Uno, Nano).
* Raspberry Pi Pico: Built with the RP2040 microcontroller.
* STM32 Boards: High-performance 32-bit ARM Cortex-M microcontrollers.
* ESP32/ESP8266: Affordable Wi-Fi-enabled microcontrollers for IoT projects.

**Peripherals and Accessories**

* **Sensors**: Temperature, pressure, humidity, motion, etc.
* **Actuators**: Motors, servos, LEDs, etc.
* **Communication Modules**: Wi-Fi, Bluetooth, Zigbee, LoRa, etc.
* **Power Supplies**: Batteries, power management ICs, and regulators.

**Debugging Tools**

* **Logic Analyzers**: Debugging communication protocols (I2C, SPI, UART).
* **Oscilloscopes**: For analyzing waveforms.

**2. Software Resources**

**Integrated Development Environments (IDEs)**

* **Arduino IDE**: Simple, beginner-friendly.
* **STM32CubeIDE**: For STM32 family boards.
* **Keil µVision**: Professional ARM development.
* **MPLAB X IDE**: For Microchip PIC microcontrollers.

**Programming Languages**

* **C/C++**: Most commonly used.
* **Python**: For microcontrollers like MicroPython or CircuitPython.
* **Assembly**: For low-level programming.
  + STM32 Community
  + Microchip Developer Forums
  + Stack Overflow (Embedded Systems Tag)

**3. Simulation and Design Tools**

* **Proteus**: Simulation for embedded systems.
* **TinkerCAD**: Beginner-friendly circuit design tool.
* **KiCAD/Eagle**: PCB design tools.

**Advanced Embedded system Recourses**

**1. Advanced Hardware Features**

**High-Performance Microcontrollers**

* ARM Cortex-M and Cortex-A Series, RISC-V MCUs, Hybrid Processors

**Integrated Connectivity**

* Wireless Modules. Cellular Connectivity:

**AI and ML Accelerators**

* Tensor Processing DSP Co-Processors

**Energy Efficiency**

* Low-Power Modes Energy Harvesting Support

**Advanced Interfaces**

* High-Speed Communication Graphics Support:

**2. Specialized Software Tools**

**Development Environments**

* NXP MCUXpresso: Espressif ESP-IDF: Renesas e² Studio:

**AI/ML Frameworks**

* TensorFlow Lite for Microcontrollers: Edge Impulse: STMicroelectronics NanoEdge AI Studio:

**RTOS and Middleware**

* Zephyr RTOS: FreeRTOS+IoT: ThreadX:

**Cloud Integration**

* AWS IoT Core Azure Sphere Google Cloud IoT Core

**3. Design and Simulation Tools**

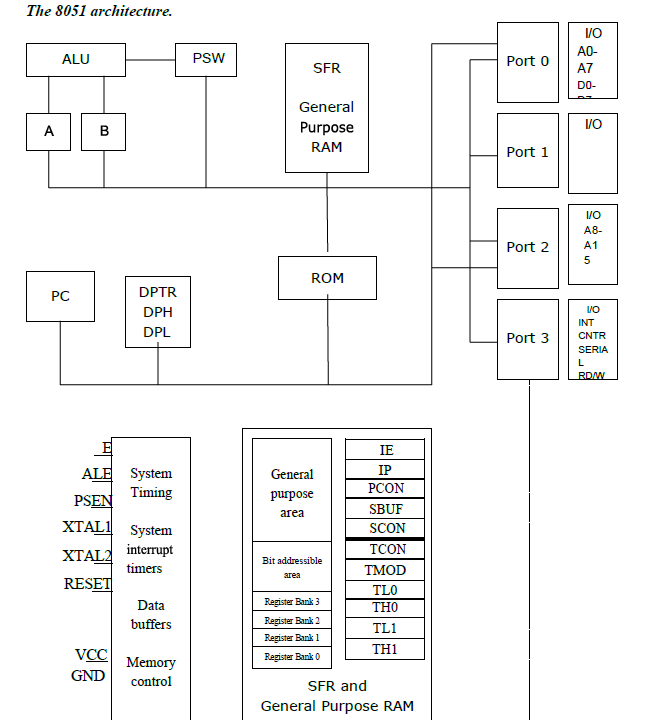
* MATLAB/Simulink: Proteus VSM Advanced: Altair Embed:

**4. Security and Cyber-Physical Systems**

* Secure Boot: TrustZone for Arm Cortex-M: Hardware Cryptographic Engines: Root of Trust: PSA Certified: OpenSSL Integration:

**5. Cutting-Edge Use Cases**

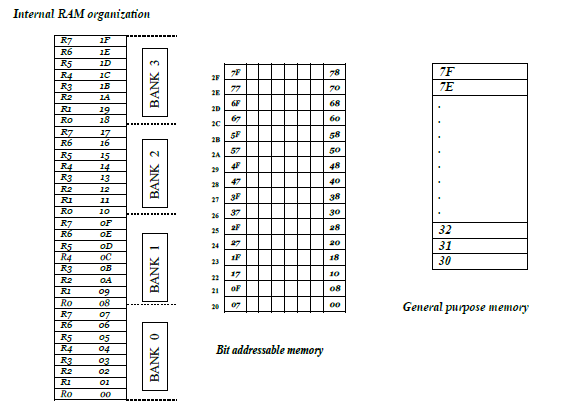
* Autonomous Systems: Edge AI: Industrial IoT: Wearables and Health:



* 8051 has 4 K Bytes of internal ROM. The address space is from 0000 to 0FFFh. If the program size is more than 4 K Bytes 8051 will fetch the code automatically from external memory.
* Accumulator is an 8 bit register widely used for all arithmetic and logical operations. Accumulator is also used to transfer data between external memory. B register is used along with Accumulator for multiplication and division. A and B registers together is also called MATH registers.
* PSW (Program Status Word). This is an 8 bit register which contains the arithmetic status of ALU
* and the bank select bits of register banks.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| CY | AC | F0 | RS1 | RS0 | OV | P |

* CY - carry flag
* AC - auxiliary carry flag
* F0 - available to the user for general purpose
* RS1, RS0 - register bank select bits
* OV - overflow
* P - parity
* Stack Pointer (SP) – it contains the address of the data item on the top of the stack. Stack may reside
* anywhere on the internal RAM. On reset, SP is initialized to 07 so that the default stack will start
* from address 08 onwards.
* Data Pointer (DPTR) – DPH (Data pointer higher byte), DPL (Data pointer lower byte). This is a
* 16 bit register which is used to furnish address information for internal and external program
* memory and for external data memory.
* • Program Counter (PC) – 16 bit PC contains the address of next instruction to be executed. On reset
* PC will set to 0000. After fetching every instruction PC will increment by one.

**MEMORY ORGANIZATION**

*Working Registers*

**Register Banks**: 00h to 1Fh. The 8051 uses 8 general-purpose registers R0 through R7 (R0, R1, R2, R3, R4, R5, R6, and R7). There are four such register banks. Selection of register bank can be done through

RS1, RS0 bits of PSW. On reset, the default Register Bank 0 will be selected.

**Bit Addressable RAM: 20h to 2Fh** . The 8051 supports a special feature which allows access to bit variables. This is where individual memory bits in Internal RAM can be set or cleared. In all there are 128

bits numbered 00h to 7Fh. Being bit variables any one variable can have a value 0 or 1. A bit variable can be set with a command such as SETB and cleared with a command such as CLR. Example instructions

are:

*SETB 25h ; sets the bit 25h (becomes 1)*

*CLR 25h ; clears bit 25h (becomes 0)*

*Note, bit 25h is actually bit 5 of Internal RAM location 24h.*

The Bit Addressable area of the RAM is just 16 bytes of Internal RAM located between 20h and 2Fh.

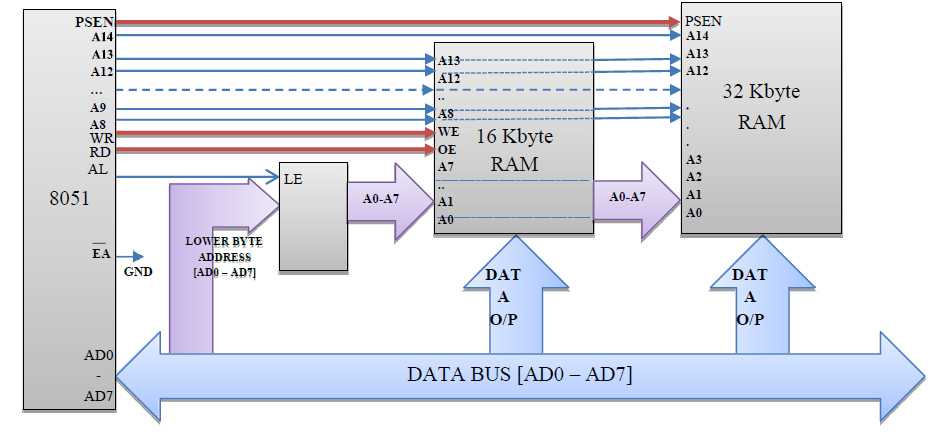
**General Purpose RAM: 30h to 7Fh.** Even if 80 bytes of Internal RAM memory are available forgeneral-purpose data storage, user should take care while using the memory location from 00 -2Fhsince these locations are also the default register space, stack space, and bit addressable space. It is a good practice to use general purpose memory from 30 – 7Fh. The general purpose RAM can be accessed usingdirect or indirect addressing modes.

**EXTERNAL MEMORY INTERFACING**

**Eg. Interfacing of 16 K Byte of RAM and 32 K Byte of EPROM to 8051**

Number of address lines required for ***16 Kbyte memory is 14 lines*** and that ***of 32Kbytes of memory is 15lines.***

The connections of external memory is shown below.



The lower order address and data bus are multiplexed. De-multiplexing is done by the latch. Initially the address will appear in the bus and this latched at the output of latch using ALE signal. The output of the

latch is directly connected to the lower byte address lines of the memory. Later data will be available in this bus. Still the latch output is address it self. The higher byte of address bus is directly connected to the

memory. The number of lines connected depends on the memory size.

The RD and WR (both active low) signals are connected to RAM for reading and writing the data. PSEN of microcontroller is connected to the output enable of the ROM to read the data from the memory.

EA (active low) pin is always grounded if we use only external memory. Otherwise, once the program size exceeds internal memory the microcontroller will automatically switch to external memory.

**Counters and Timers:**

A **timer** is a specialized type of clock which is used to measure time intervals. A timer that counts from zero upwards for measuring time elapsed is often called a **stopwatch**. It is a device that counts down from

a specified time interval and used to generate a time delay, for example, an hourglass is a timer.

A **counter** is a device that stores (and sometimes displays) the number of times a particular event or process occurred, with respect to a clock signal. It is used to count the events happening outside the microcontroller. In electronics, counters can be implemented quite easily using register-type circuits such as a flip-flop.

Difference between a Timer and a Counter

The points that differentiate a timer from a counter are as follows –

|  |  |
| --- | --- |
| **Timer** | **Counter** |
| The register incremented for  every machine cycle | The register is incremented considering 1 to  0 transition at its corresponding to an  external input pin (T0, T1). |
| Maximum count rate is 1/12 of | Maximum count rate is 1/24 of the oscillator |
| the oscillator frequency. | frequency. |
| A timer uses the frequency of  the internal clock, and  generates delay. | A counter uses an external signal to count  pulses. |

**Timers of 8051 and their Associated Registers**

The 8051 has two timers, Timer 0 and Timer 1. They can be used as timers or as event counters. Both Timer 0 and Timer 1 are 16-bit wide. Since the 8051 follows an 8-bit architecture, each 16 bit is accessed

as two separate registers of low-byte and high-byte.

**Timer 0 Register**

The 16-bit register of Timer 0 is accessed as low- and high-byte. The low-byte register is called TL0 (Timer 0 low byte) and the high-byte register is called TH0 (Timer 0 high byte). These registers can be

accessed like any other register. For example, the instruction **MOV TL0, #4H** moves the value into the low-byte of Timer #0.



**Timer 1 Register**

The 16-bit register of Timer 1 is accessed as low- and high-byte. The low-byte register is called TL1 (Timer 1 low byte) and the high-byte register is called TH1 (Timer 1 high byte). These registers can be

accessed like any other register. For example, the instruction **MOV TL1, #4H** moves the value into the low-byte of Timer



**TMOD (Timer Mode) Register**

Both Timer 0 and Timer 1 use the same register to set the various timer operation modes. It is an 8-bit register in which the lower 4 bits are set aside for Timer 0 and the upper four bits for Timers. In each case,

the lower 2 bits are used to set the timer mode in advance and the upper 2 bits are used to specify the location.



**Gate** − When set, the timer only runs while INT(0,1) is high.

**C/T** − Counter/Timer select bit.

**M1** − Mode bit 1.

**M0** − Mode bit 0.

**GATE**

Every timer has a means of starting and stopping. Some timers do this by software, some by hardware,and some have both software and hardware controls. 8051 timers have both software and hardware

controls. The start and stop of a timer is controlled by software using the instruction **SETB TR1** and **CLR TR1** for timer 1, and **SETB TR0** and **CLR TR0** for timer 0.

The SETB instruction is used to start it and it is stopped by the CLR instruction. These instructions start and stop the timers as long as GATE = 0 in the TMOD register. Timers can be started and stopped by an

external source by making GATE = 1 in the TMOD register.

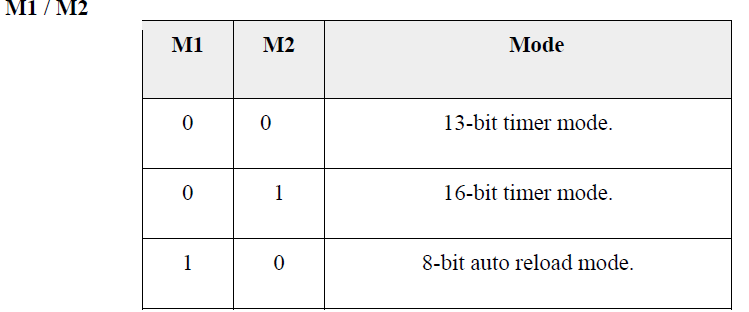
**C/T (CLOCK / TIMER)**

This bit in the TMOD register is used to decide whether a timer is used as a **delay generator** or an **event manager**. If C/T = 0, it is used as a timer for timer delay generation. The clock source to create the time

delay is the crystal frequency of the 8051. If C/T = 0, the crystal frequency attached to the 8051 also decides the speed at which the 8051 timer ticks at a regular interval. Timer frequency is always 1/12th of the frequency of the crystal attached to the 8051. Although various

8051 based systems have an XTAL frequency of 10 MHz to 40 MHz, we normally work with the XTAL frequency of 11.0592 MHz It is because the baud rate for serial communication of the 8051.XTAL =

11.0592 allows the 8051 system to communicate with the PC with no errors



**Interrupts:**

An interrupt is a signal to the processor emitted by hardware or software indicating an event that needs immediate attention. Whenever an interrupt occurs, the controller completes the execution of the current

instruction and starts the execution of an **Interrupt Service Routine** (ISR) or **Interrupt Handler**. ISR tells the processor or controller what to do when the interrupt occurs. The interrupts can be either hardware

interrupts or software interrupts.

**Hardware Interrupt**

A hardware interrupt is an electronic alerting signal sent to the processor from an external device, like a disk controller or an external peripheral. For example, when we press a key on the keyboard or move the

mouse, they trigger hardware interrupts which cause the processor to read the keystroke or mouse position.

**Software Interrupt**

A software interrupt is caused either by an exceptional condition or a special instruction in the instruction set which causes an interrupt when it is executed by the processor. For example, if the processor's

arithmetic logic unit runs a command to divide a number by zero, to cause a divide-by-zero exception, thus causing the computer to abandon the calculation or display an error message. Software interrupt

instructions work similar to subroutine calls.