

in [8051 PROJECTS](#)

Interfacing ADC to 8051

SEPTEMBER 6, 2012

ADC (Analog to digital converter) forms a very essential part in many embedded projects and this article is about interfacing an ADC to 8051 embedded controller. ADC 0804 is the ADC used here and before going through the interfacing procedure, we must neatly understand how the ADC 0804 works.

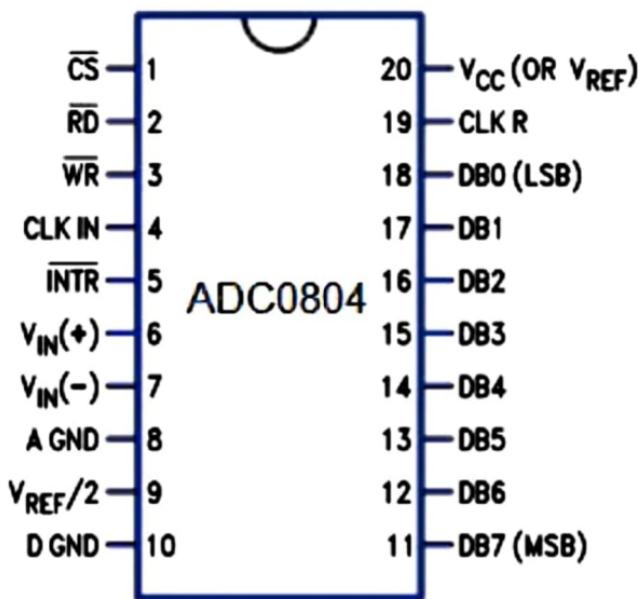
ADC 0804.

ADC0804 is an 8 bit successive approximation analogue to digital converter from National semiconductors. The features of ADC0804 are differential analogue voltage inputs, 0-5V input voltage range, no zero adjustment, built in clock generator, reference voltage can be

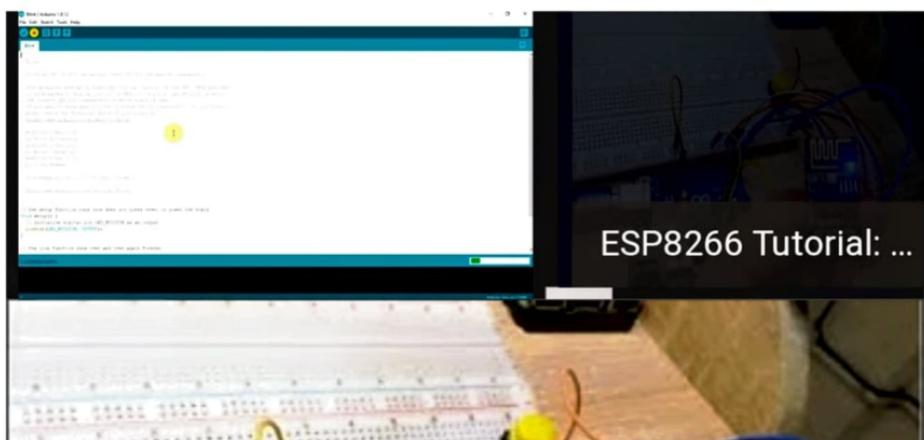


1. BASIC ELECTRICAL WIRING DIAGRAM >

2. WIRE SIZE AMPS >



ADC0804 pinout



X Interfacing ADC to 8051. ...
circuitstoday.com



| Vref/2 (pin9) (volts) | Input voltage span (volts) | Step size (mV) |
|-----------------------------|----------------------------------|-------------------|
|-----------------------------|----------------------------------|-------------------|

| | | |
|-----------|-------|--------------|
| Left open | 0 – 5 | 5/255 = 19.6 |
|-----------|-------|--------------|

| | | |
|---|-------|---------------|
| 2 | 0 – 4 | 4/255 = 15.69 |
|---|-------|---------------|

| | | |
|-----|-------|---------------|
| 1.5 | 0 – 3 | 3/255 = 11.76 |
|-----|-------|---------------|

| | | |
|------|----------|------------------|
| 1.28 | 0 – 2.56 | 2.56/255 = 10.04 |
|------|----------|------------------|

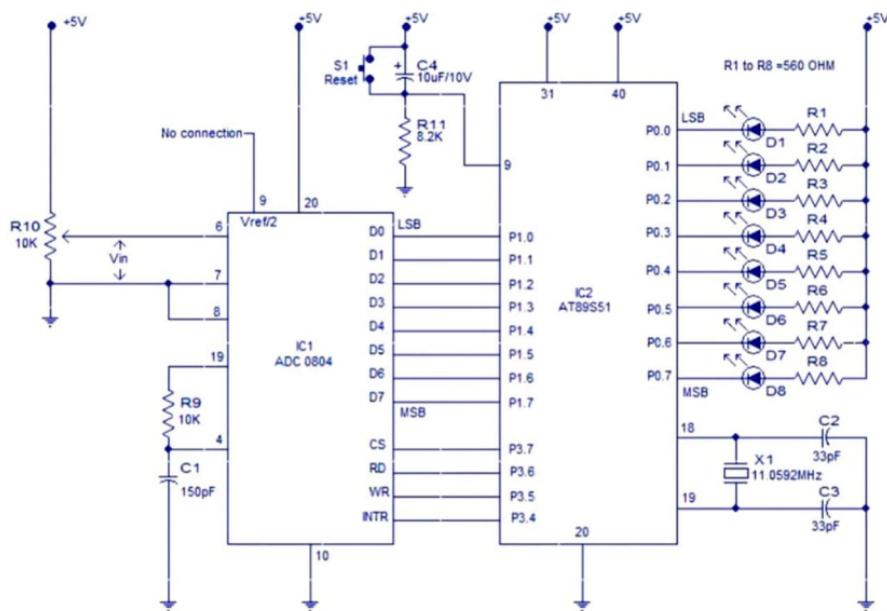
| | | |
|-----|-------|--------------|
| 1.0 | 0 – 2 | 2/255 = 7.84 |
|-----|-------|--------------|

| | | |
|-----|-------|--------------|
| 0.5 | 0 – 1 | 1/255 = 3.92 |
|-----|-------|--------------|



- If conversion is finished ($\text{INTR}=0$), go to the next step.
- Make $\text{CS}=0$ and send a high to low pulse to RD pin to read the data from the ADC.

Circuit diagram.



Interfacing ADC to 8051

The figure above shows the schematic for interfacing ADC0804 to 8051. The circuit initiates the ADC to convert a given analogue input, then accepts the corresponding digital data and displays it on the LED array connected at P0. For example, if the analogue input

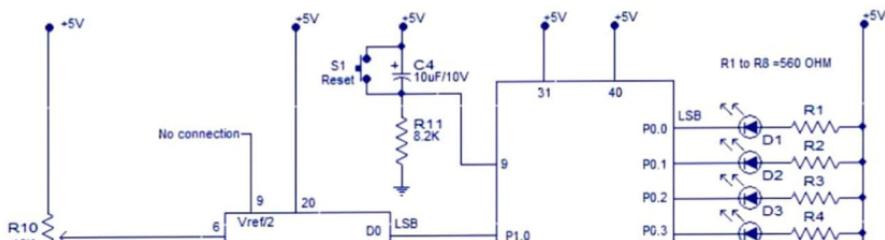
X Interfacing ADC to 8051. ... circuitstoday.com



Steps for converting the analogue input and reading the output from ADC0804.

- Make CS=0 and send a low to high pulse to WR pin to start the conversion.
- Now keep checking the INTR pin. INTR will be 1 if conversion is not finished and INTR will be 0 if conversion is finished.
- If conversion is not finished (INTR=1) , poll until it is finished.
- If conversion is finished (INTR=0), go to the next step.
- Make CS=0 and send a high to low pulse to RD pin to read the data from the ADC.

Circuit diagram.



signal output is identical to the analog signal input.

Binary digital impulses, all by themselves, appear as long strings of ones and zeroes, and have no apparent meaning to a human observer. But when a DAC is used to decode the binary digital signals, meaningful output appears. This might be a voice, a picture, a musical tune, or mechanical motion. Both the DAC and the ADC are of significance in some applications of digital signal processing. The intelligibility or fidelity of an analog signal can often be improved by converting the analog input to digital form using an ADC, then clarifying the digital signal, and finally converting the "cleaned-up" digital impulses back to analog form using a DAC.

A common use of digital-to-analog converters is generation of audio signals from digital information in music players. Digital video signals are converted to analog in televisions and cell phones to display colors and shades. Digital-to-analog conversion can degrade a signal, so conversion details are normally chosen so that the errors are negligible. Due to cost and the need for matched components DACs are almost exclusively manufactured on integrated circuits (ICs). There are many DAC architectures which have different advantages and disadvantages. The suitability of a particular DAC for an application is determined by a variety of measurements including speed and resolution.

2. THEORY

The digital to analog converter is a device widely used to convert digital pulses to analog signals. The two methods of creating DAC are binary weighted and R-2R ladder. DAC 0808 uses the R-2R method since it can achieve a high degree of precision. The first criterion for judging a DAC is its resolution, which is the function of the number of binary inputs. The common ones are 8, 10 and 12 bits. The number of data bit inputs decides the resolution of the DAC since the number of analog output levels is equal to 2^n , where n is the number of data inputs. DAC 0808 provides 256 discrete voltage or current levels of output. In DAC 0808, the digital inputs are converted into current I_{out} and by connecting a resistor to I_{out} pin, we convert the result to voltage. The total current provided by I_{out} pin is a function of binary numbers at the D0-D7 pins inputs to DAC 0808 and reference current (I_{ref}) is as follows:

$$I_{out} = I_{ref} (D7/2 + D6/4 + D5/8 + D4/16 + D3/32 \dots + D0/256)$$

Where D0 is the LSB, D7 is the MSB for the inputs and I_{ref} is the input current that must be applied.

3. IMPLEMENTATION

In case of matrix Keypad both the ends of switches are connected to the port Pin. Over here a 4x4 matrix keypad is considered i.e. four rows and four columns. The keypad is connected to port zero of the microcontrollers. When one of the Column Pins is pulled low & the row pins are checked and if any of the Pin is

low then it is observable that which switch is pressed. Suppose column 1 pin is made low and while checking the rows if Row 3 is low then it will show that switch 7 has been pressed.

LCD's are very simple to interface with the controller as well as are cost effective. The ALPHANUMERIC display used here in Figure1 below is 4x20 (four lines & Twenty characters per line) shown in figure. It is connected to port two. The LCD requires 3 control lines (RS, R/W & EN) & 8 (or 4) data lines. The number of data lines depends on the mode of operation. If operated in 8-bit mode then 8 data lines + 3 control lines i.e. total 11 lines are required. And if operated in 4-bit mode then 4 data lines + 3 control lines i.e. 7 lines are required. How can one decide which mode to use? It's simple if sufficient data lines are available then go for 8 bit mode & if there is a time constrain i.e. display should be faster then it becomes mandatory to use 8-bit mode because basically 4-bit mode takes twice as more time as compared to 8-bit mode.

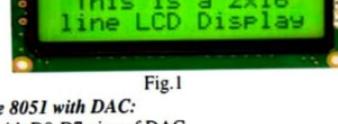


Fig.1

3.1 Algorithm for interface 8051 with DAC:

- Step1: Connect the P1 of 8051 with D0-D7 pins of DAC
- Step2: Give +5v to VCC & Vref of DAC
- Step3: Connect -12v to VEE of DAC
- Step4: Connect OPAMP to OUT pin of the DAC With 5K resistor
- Step5: Connect the oscilloscope to the OPAMP to View the output

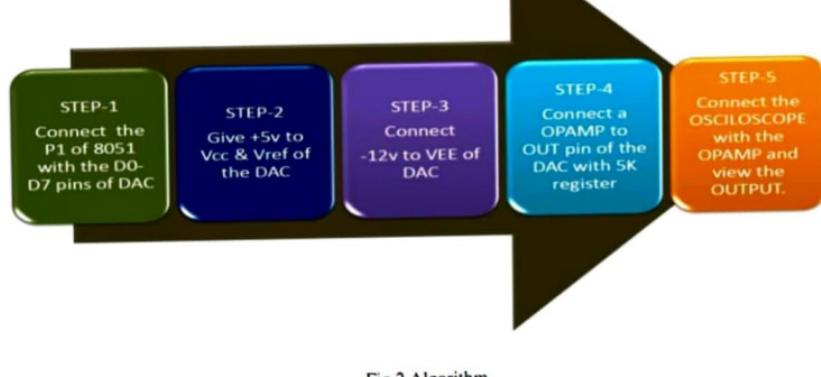


Fig.2 Algorithm

Digital to Analog converters are required when a digital code must be converted to analog signal. It has eight digital input lines and an output line for analog signal. The number of data bits reduces resolution of DAC. Outputting digital data 00 to FF at regular intervals to DAC, results in generation of different waveforms namely square wave, triangular wave, sine wave etc.

Procedure for interfacing Microcontroller unit with DAC-0808:

3.2 Block Diagram

digital input lines and an output line for analog signal. The number of data bits reduces resolution of DAC. Outputting digital data 00 to FF at regular intervals to DAC, results in generation of different waveforms namely square wave, triangular wave, sine wave etc.

Procedure for interfacing Microcontroller unit with DAC-0808:

3.2 Block Diagram

125

International Journal of Research in Advent Technology, Vol.2, No.2, February 2014

E-ISSN: 2321-9637

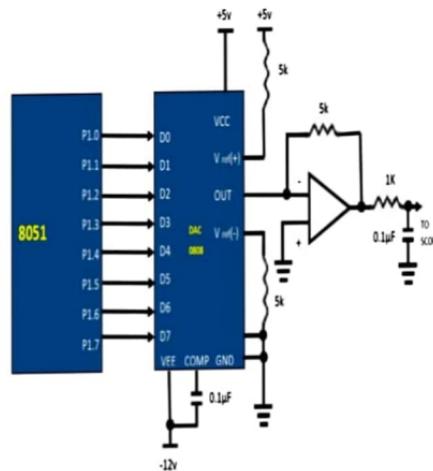


Fig.3 Circuitry of DAC

4. RESULTS

Thus Microcontroller 8051, when interfaced with DAC-0808 and at the output we get Square wave and Triangular wave respectively. It is shown in the figure below.

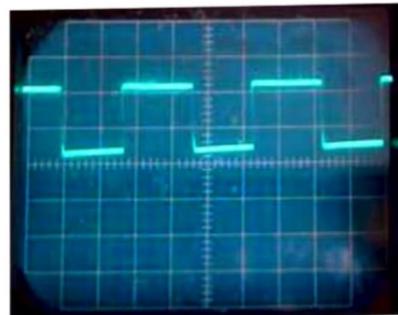
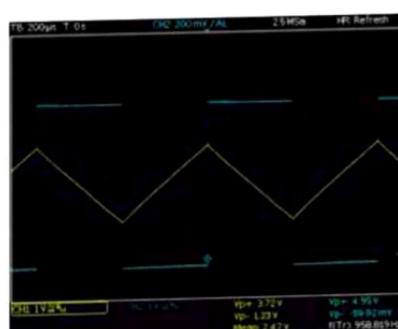


Fig.4 Square Wave



126

Timer & Counter of 8051 µC

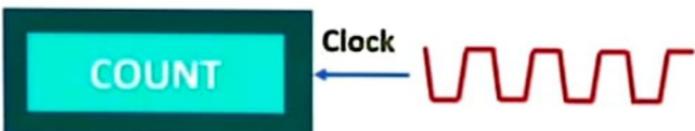
- 8051 has Two 16 bits Timers T0 & T1, working as up counters.
T0 and T1 is further divided into 8bits of registers TH0-TL0 and TH1-TL1.



- How to Load Count?
- This Timers are Up counter.
- So, on given clock it will increment by 1.
- When it reaches to FFFFH, it will rolls back to 0000H and during that it will generates Timer Overflow interrupt.

$$\text{Count} = \text{FFFFH} - \text{Value} + 1$$

- So, if you wants to count 9 then Count = FFFF – 9 + 1 = FFF7H
- ```
MOV TH0, #FFH
MOV TL0, #F7H
```



- This count loaded in T0 or T1 will increment after every clock.

## Timer or Counter?

If clock to the count is given by internal clock of 8051 then it will be timer and if clock is given by external clock on then it will counter.

- That is to be configured by TMOD register of 8051.
- T/C bit will decide timer or counter configuration of 8051.
- How Timer / Counter works?

## Load Count in T0 or T1

Count will increase after every clock

Make TF0 or TF1 bit to 0 before it jumps to ISR Address

When Count rolls from FFFFH to 0000H, it will make TF0 or TF1 bit to 1.  
(It is interrupt to 8051)



...

- ISR Address of Timer 0 and Timer 1

- Timer 0 ISR Address is 000BH and Timer 1 ISR Address is 001BH

# TCON & TMOD of 8051 µC

- ❑ 8051 has Two 16 bits Timers T0 & T1, working as up counters.  
T0 and T1 is further divided into 8bits of registers TH0-TL0 and TH1-TL1.
- ❑ If T0 & T1 counts internal clock pulses, then it is timer.
- ❑ If T0 & T1 counts External clock pulses, then it is Counter.
- ❑ Timer action is controlled by TCON and TMOD registers.
- ✓ TCON register – {Bit Address TCON.7 to TCON.0}

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
|-----|-----|-----|-----|-----|-----|-----|-----|

- ❑ TF1 and TF0 – Timer Overflow Flag
- ❑ SET 1 = When timer 1 and timer 0 overflows, when timer roll overs to all 0's.
- ❑ Clear 0 = When processor executes ISR after overflow. {For Timer 1 ISR address is 001BH and Timer 0 ISR address is 000BH}
- ❑ TR1 and TR0 – Timer Run Control Bit
- ❑ SET 1 = Start Counting Timer.
- ❑ Clear 0 = Halts Timer.
- ❑ IE1 and IE0 – External Interrupt bit
- ❑ SET 1 = when 8051 receives interrupt on  $\overline{INT1}$  and  $\overline{INT0}$ .
- ❑ Clear 0 = when ISR executed. {For  $\overline{INT1}$  ISR address is 0013H and  $\overline{INT0}$  ISR address is 0003H}

- ❑ IT1 and IT0 – External Interrupt Type bit
- ❑ SET 1 =  $\overline{INT1}$  and  $\overline{INT0}$  must be –ve edge trigger.
- ❑ Clear 0 =  $\overline{INT1}$  and  $\overline{INT0}$  must be low level trigger.

## ✓ TMOD register – Timer Mode Control register



- ❑  $C/T$  – Counter / Timer Type bit
- ❑ SET 1 = Acts as Counter. {External Frequency on T1 & T0}
- ❑ Clear 0 = Acts as Timer. {Internal Frequency Fosc/12}
- ❑ GATE – Gate Enable Control bit
- ❑ SET 1 = Timer Controlled by Hardware. {INTX Signal}
- ❑ Clear 0 = Timer independent on INTX signal.

## ❑ M1 & M0 – Mode Control bits

| M1 | M0 | Timer Mode   |
|----|----|--------------|
| 0  | 0  | Timer Mode 0 |
| 0  | 1  | Timer Mode 1 |
| 1  | 0  | Timer Mode 2 |
| 1  | 1  | Timer Mode 3 |

# Working of Timer/Counter in 8051 $\mu$ C

Step-1 - Load Count in TH0-TLO or TH1-TL1.

Step-2 - Select timer/counter by C/T bit of TMOD

Step-3 Start Timer by TCON reg.

✓ TMOD register – Timer Mode Control register

|         |     |    |    |         |     |    |    |
|---------|-----|----|----|---------|-----|----|----|
| GATE    | C/T | M1 | M0 | GATE    | C/T | M1 | M0 |
| Timer 1 |     |    |    | Timer 0 |     |    |    |

✓ TCON register – {Bit Address TCON.7 to TCON.0}

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
|-----|-----|-----|-----|-----|-----|-----|-----|



# Working of Timer/Counter in 8051 µC

- ✓ TCON register – {Bit Address TCON.7 to TCON.0}

|     |     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|-----|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
|-----|-----|-----|-----|-----|-----|-----|-----|

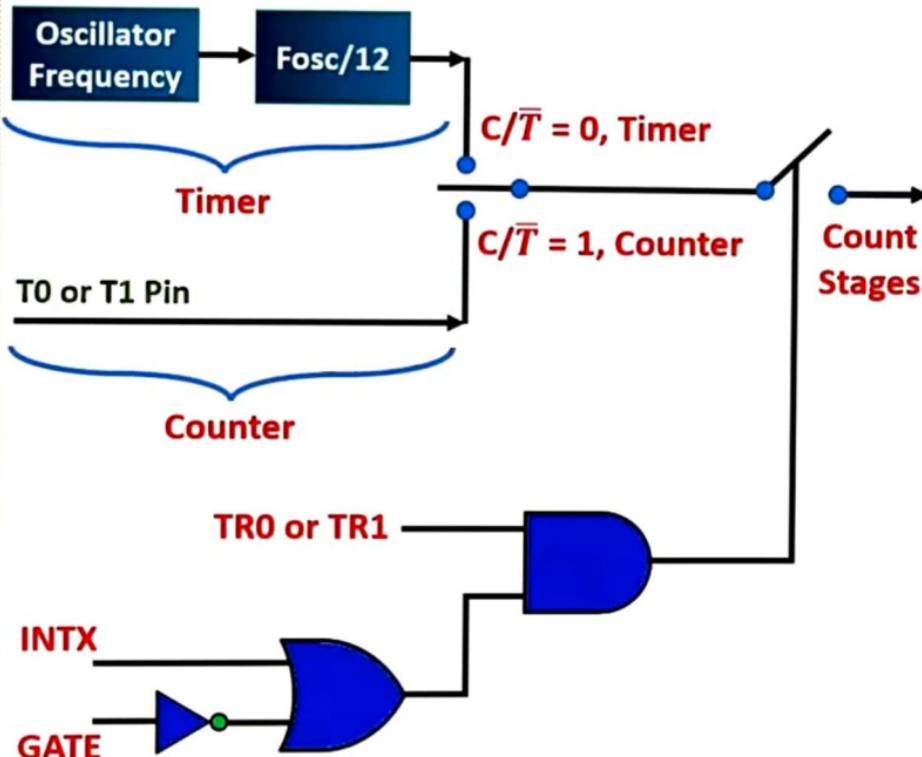
- ✓ TMOD register – Timer Mode Control register

|      |     |    |    |      |     |    |    |
|------|-----|----|----|------|-----|----|----|
| GATE | C/T | M1 | M0 | GATE | C/T | M1 | M0 |
|------|-----|----|----|------|-----|----|----|

Timer 1

Timer 0

- 8051 has Two 16 bits Timers T0 & T1, working as up counters.
- By C/T bit we can select timer and counter.
- In Timer, clock will be given by internal clock.
- In counter, clock will be given by T0 or T1 Pin of 8051.
- To have running counter, TR bit of TCON register must be 1.
- If Timer/Counter is triggered by external signal then GATE = 1 of TMOD register, which means Timer/ Counter operation will get trigger by INTX {INT0 or INT1}.
- If Timer 0 is configured then with GATE bit we use INT0 hardware interrupt to trigger Timer/Counter.
- If Timer 1 is configured then with GATE bit we use INT1 hardware interrupt to trigger Timer/Counter.
- If GATE bit is logic 1, then INTX pin will used for timer/Counter only.



# Modes of Timer/Counter in 8051 $\mu$ C

## ✓ Timer Mode 0 {13 bits Timer/ Counter}



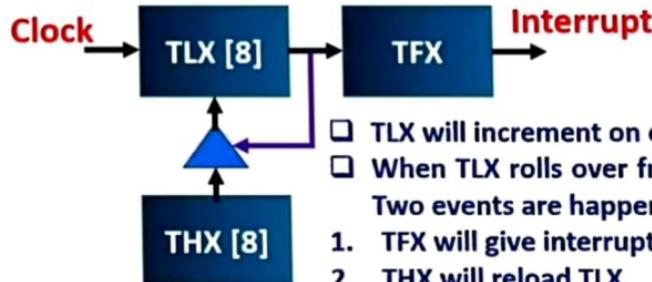
- ❑ TLX has 5 bits for count and THX has 8 bits for count. So in total, 13 bits of count is available in this mode 0.
- ❑ After 32 counts TLX rolls over and it will increment THX.
- ❑ So TLX will divide the frequency by 32.
- ❑ By this mode, total maximum count can be  $2^{13} = 8K$ .
- ❑ So maximum delay = 8192 (12/Fosc)

## ✓ Timer Mode 1 {16 bits Timer/ Counter}



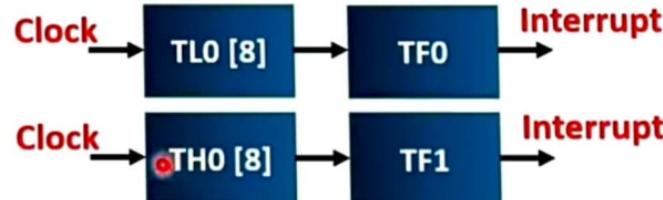
- ❑ TLX and THX used completely here with Mode 1.
- ❑ On each clock 16 bits will increment by 1.
- ❑ TFX will set to 1, when all 16 bits rolls from FFFFH to 0000H.
- ❑ By this mode, total maximum count can be  $2^{16} = 64K$ .
- ❑ So maximum delay = 65536 (12/Fosc)

## ✓ Timer Mode 2 {8 bits Auto reload TL from TH}



- ❑ TLX will increment on every count.
- ❑ When TLX rolls over from FFH to 00H, two events are happening.
  1. TFX will give interrupt
  2. THX will reload TLX
- ❑ Maximum count =  $2^8 = 256$
- ❑ Maximum delay = 256 (12/Fosc)

## ✓ Timer Mode 3 {Two 8 bits timer by Timer 0}



- ❑ TL0 and TH0 used with two separate timers.
- ❑ TL0 will give interrupt to TFO flag bit of Timer 0.
- ❑ TL0 can be used as Timer and Counter.
- ❑ TH0 will give interrupt to TF1 flag bit of Timer 1.
- ❑ TH0 can only be used as Timer.



```
ORG 0x00 ; Set the origin address

MOV PCON, #0x80 ; Configure SMO and serial port control
MOV SCON, #0x50 ; Set serial control register
MOV TMOD, #0x20 ; Set Timer 1 in 8-bit auto-reload mode

MOV TH1, #0xFD ; Load TH1 to achieve desired value
MOV TL1, #0xFD ; Adjust these values

SETB TR1 ; Start Timer 1

MAIN: ; Main loop
 MOV SBUF, #'Y' ; Load 'Y' into serial buffer
 CALL DELAY ; Add a delay to control transmission
 MOV SBUF, #'E' ; Load 'E' into serial buffer
 CALL DELAY
 MOV SBUF, #'S' ; Load 'S' into serial buffer
 CALL DELAY
 SJMP MAIN ; Repeat the process

DELAY: ; Delay subroutine
 MOV R2, #0x0F ; Initialize outer loop counter

OUTER_LOOP:
 MOV R1, #0xFF ; Initialize inner loop counter

INNER_LOOP:
 DJNZ R1, INNER_LOOP ; Decrement inner loop counter
 DJNZ R2, OUTER_LOOP ; Decrement outer loop counter
```

# ← Interrupts.pdf



first, PC-H after – because the stack grows upwards in 8051), and PC will be loaded with 000B H. The interrupt handler for T0 should be placed here, and it should end with the instruction:

RETI

## 4 Interrupt Priorities

8051 has two levels of interrupt priorities: high or low. By assigning priorities, we can control the order in which multiple interrupts will be serviced. Priorities are set by bits in a special function register called IP, which is at the byte address B8H. This register is also bit addressable. The assembler defines special names for bits of this register.

SFR IP at byte address B8H

| Bit No.  | 7  | 6  | 5  | 4  | 3   | 2   | 1   | 0   |
|----------|----|----|----|----|-----|-----|-----|-----|
| Bit Addr | BF | BE | BD | BC | BB  | BA  | B9  | B8  |
| Bit Name | U  | U  | U  | PS | PT1 | PX1 | PT0 | PX0 |

3

Notice that the bits are in the polling order of interrupts. A 1 in a bit position assigns a high priority to the corresponding source of interrupts – a 0 gives it a low priority. In case of multiple interrupts, the following rules apply:

- While a low priority interrupt handler is running, if a high priority interrupt arrives, the handler will be interrupted and the high priority handler will run. When the high priority handler does 'RETI', the low priority handler will resume. When this handler does 'RETI', control is passed back to the main program.
- If a high priority interrupt is running, it cannot be interrupted by any other source – even if it is a high priority interrupt which is higher in polling order.
- A low-priority interrupt handler will be invoked only if no other interrupt is already executing. Again, the low priority interrupt cannot preempt another low priority interrupt, even if the later one is higher in polling order.
- If two interrupts occur at the same time, the interrupt with higher priority will execute first. If both interrupts are of the same priority, the interrupt which is higher in polling sequence will be executed first. This is the only context in which the polling sequence matters.

## 5 Serial Interrupts

Serial interrupts are handled somewhat differently from the timers. There are independent interrupt flags for reception and transmission of serial data. RI and TI. RI indicates that a byte has been received and is available for reading in the input buffer. TI indicates that the previous byte has been sent serially and a new byte can be written to the serial port. A serial interrupt occurs if either of these flags is set. However, the interrupt must be enabled for it to be triggered.





Search tutorials, courses and ebooks...



## IP (Interrupt Priority) Register

We can change the priority levels of the interrupts by changing the corresponding bit in the Interrupt Priority (IP) register as shown in the following figure.

- A low priority interrupt can only be interrupted by the high priority interrupt, but not interrupted by another low priority interrupt.
- If two interrupts of different priority levels are received simultaneously, the request of higher priority level is served.
- If the requests of the same priority levels are received simultaneously, then the internal polling sequence determines which request is to be serviced.

| -    | -    | PT2  | PS   | PT1  | PX1  | PT0  | PX0  |
|------|------|------|------|------|------|------|------|
| bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |

- IP.6 Reserved for future use.

- IP.5 Reserved for future use.

PS IP.4 It defines the serial port interrupt priority level.

PT1 IP.3 It defines the timer interrupt of 1 priority.



Search tutorials, courses and ebooks...



received simultaneously, then the internal polling sequence determines which request is to be serviced.

| -    | -    | PT2  | PS   | PT1  | PX1  | PT0  | PX0  |
|------|------|------|------|------|------|------|------|
| bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |

- IP.6 Reserved for future use.
- IP.5 Reserved for future use.
- PS IP.4 It defines the serial port interrupt priority level.
- PT1 IP.3 It defines the timer interrupt of 1 priority.
- PX1 IP.2 It defines the external interrupt priority level.
- PT0 IP.1 It defines the timer0 interrupt priority level.
- PX0 IP.0 It defines the external interrupt of 0 priority level.

## TCON Register

TCON register specifies the type of external interrupt to the microcontroller.

## ChatGPT 3.5



With specific requirements like baud rate and data format, can be quite involved. Here's a simplified example to get you started. Note that the exact details may vary based on your specific 8051 variation and the assembler you are using.

assembly

Copy code

```
ORG 0H ; Start address

MOV SCON, #50H ; Set UART mode 1, 8-bit
MOV TMOD, #20H ; Set Timer 1 in mode 2

SETB TR1 ; Start Timer 1

MAIN:
 MOV SBUF, #'A' ; Load 'A' into serial port
 CALL DELAY ; Delay for a short period
 SJMP MAIN

DELAY:
 MOV R2, #0FFH
DELAY_LOOP:
 DJNZ R2, DELAY_LOOP
 RET
```



```
ORG 0H ; Start address
```

```
MOV DPTR, #HEX_NUMBER ; Load address of
```

```
MOV R0, #04H ; Set loop counter for 4
```

#### CONVERT\_LOOP:

```
 MOV A, #00H ; Clear accumulator for
```

```
 MOVX A, @DPTR ; Read next nibble from
```

```
 CALL HEX_TO_ASCII ; Convert nibble to
```

```
 ; Now, 'A' register contains the ASCII
```

```
 ; Your code to send or display the ASC
```

```
 ; (For example, use serial communicati
```

```
 INC DPTR ; Move to the next nibbl
```

```
 DJNZ R0, CONVERT_LOOP ; Repeat for al
```

```
END_PROGRAM: NOP ; End of program
```

```
HEX_TO_ASCII: ; Subroutine to convert a
```

```
 ADD A, #30H ; Convert to ASCII character
```

```
 MOV R7, A ; Save ASCII character in
```

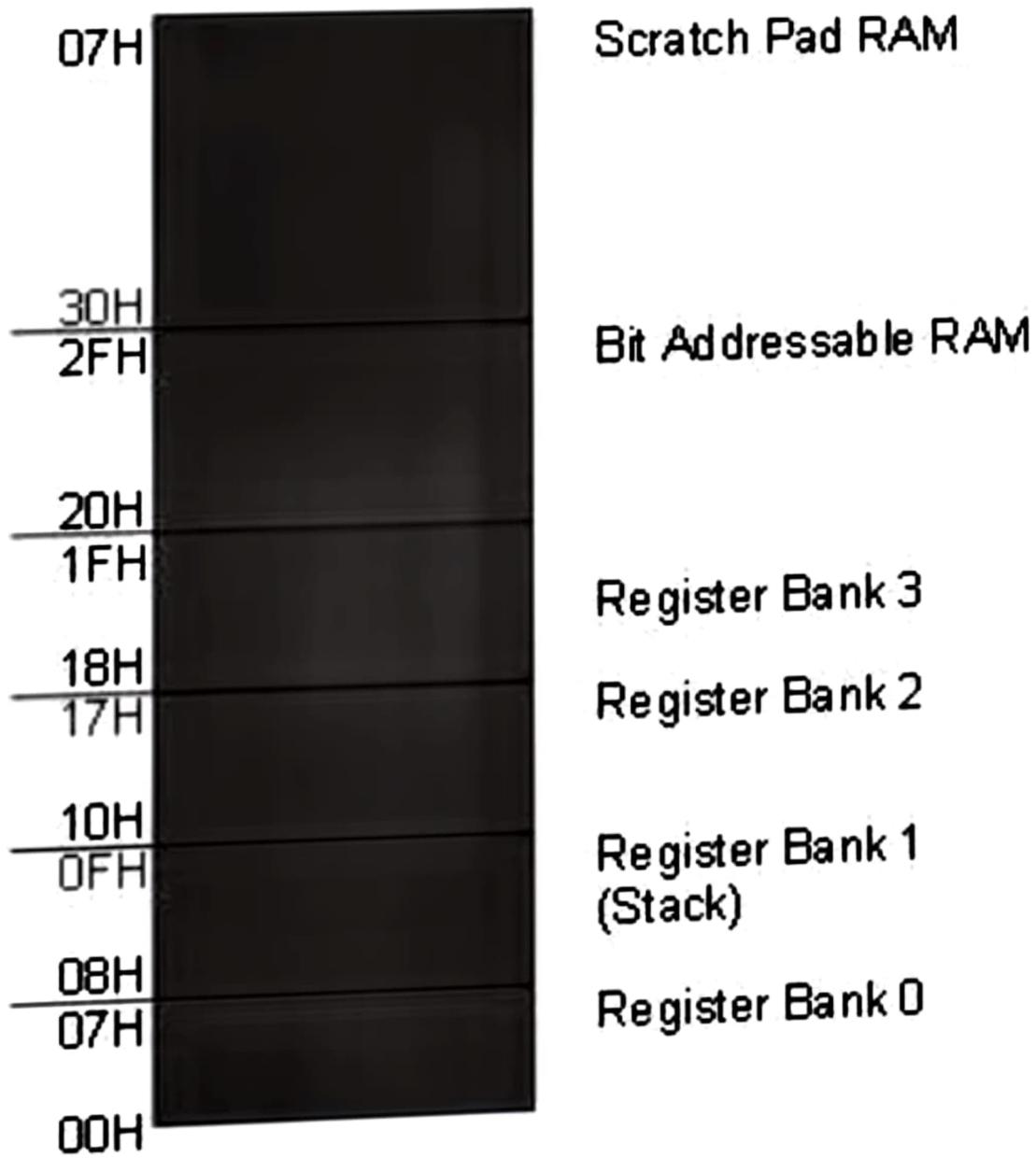
```
 MOV A, #00H ; Clear accumulator
```

```
 MOV A, R7 ; Move ASCII character ba
```

```
 RET
```



- 32 bytes from 00H to 1FH locations are set aside for register banks and the stack.
- 16 bytes from 20H to 2FH locations are set aside for bit-addressable read/write memory.
- 80 bytes from 30H to 7FH locations are used for read and write storage; it is called as **scratch pad**. These 80 locations RAM are widely used for the purpose of storing data and parameters by 8051 programmers.





Search tutorials, courses and ebooks...



Registers banks, which register bank 0 is R0-R7 do we have access to when the 8051 is powered up? The answer is register bank 0; that is, RAM locations from 0 to 7 are accessed with the names R0 to R7 when programming the 8051. Because it is much easier to refer these RAM locations by names such as R0 to R7, rather than by their memory locations.

## How to Switch Register Banks

Register bank 0 is the default when the 8051 is powered up. We can switch to the other banks using PSW register. D4 and D3 bits of the PSW are used to select the desired register bank, since they can be accessed by the bit addressable instructions SETB and CLR. For example, "SETB PSW.3" will set PSW.3 = 1 and select the bank register 1.

| RS1 | RS2 | Bank Selected |
|-----|-----|---------------|
| 0   | 0   | Bank0         |
| 0   | 1   | Bank1         |
| 1   | 0   | Bank2         |
| 1   | 1   | Bank3         |



GEEKSFORGEEKS

## Architecture of 8086

### Introduction :

The 8086 microprocessor is an 8-bit/16-bit microprocessor designed by Intel in the late 1970s. It is the first member of the x86 family of microprocessors, which includes many popular CPUs used in personal computers.

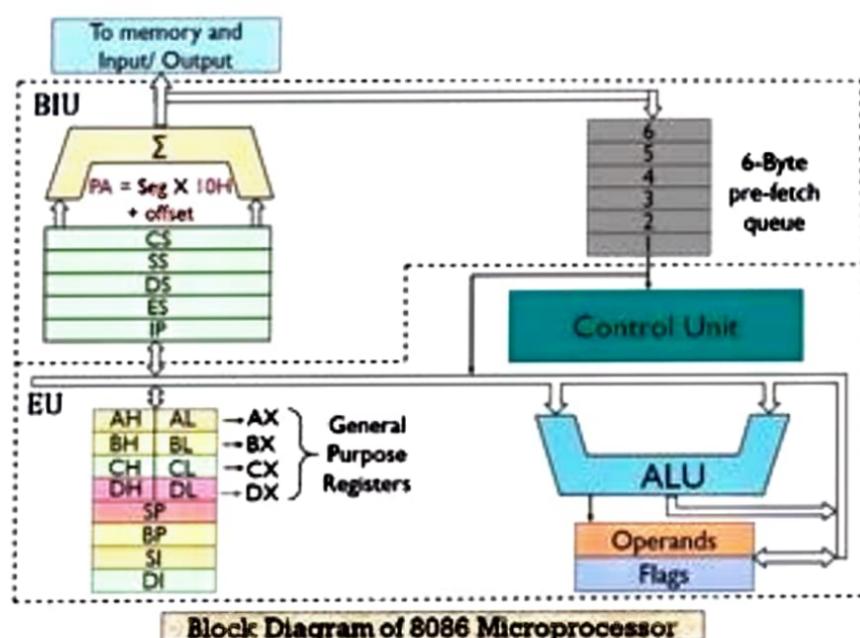
The architecture of the 8086 microprocessor is based on a complex instruction set computer (CISC) architecture, which means that it supports a wide range of instructions, many of which can perform multiple operations in a single instruction. The 8086 microprocessor has a 20-bit address bus, which can address up to 1 MB of memory, and a 16-bit data bus, which can transfer data between the microprocessor and memory or I/O devices.

The 8086 microprocessor has a segmented memory architecture, which means that memory is divided into segments that are addressed using both a segment register and an offset. The segment register points to the start of a segment, while the offset specifies the location of a specific byte within the segment. This allows the 8086 microprocessor to access large amounts of memory while still using

[Open In App](#)

bus.

8086 provides the programmer with 14 internal registers, each of 16 bits or 2 bytes wide. **The main advantage of the 8086 microprocessor is that it supports Pipelining.**



## 8086 Architecture

### Memory segmentation:

- In order to increase execution speed and fetching speed, 8086 segments the memory.
- Its 20-bit address bus can address 1MB of memory, it segments it into 16 64kB segments.
- 8086 works only with four 64KB segments within the whole 1MB memory.

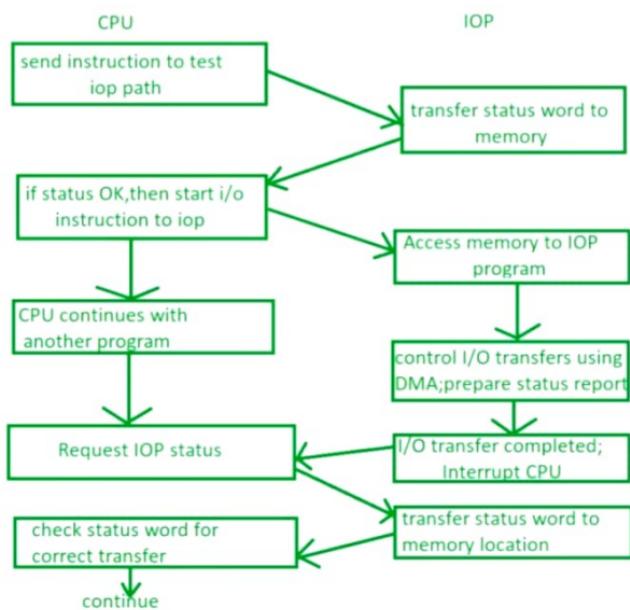
The internal architecture of Intel 8086 is divided into



GEEKSFORGEEKS

## Communication channel between CPU and IOP

There is a communication channel between IOP and CPU to perform task which come under computer architecture. This channel explains the commands executed by IOP and CPU while performing some programs. The CPU do not executes the instructions but it assigns the task of initiating operations, the instructions are executed by IOP. I/O transfer is instructed by CPU. The IOP asks for CPU through interrupt. This channel starts by CPU, by giving “test IOP path” instruction to IOP and then the communication begins as shown in diagram:



**Figure –** Communication channel between IOP and

CPU Whenever CPU gets interrupt from IOP to access  
**Open In App**

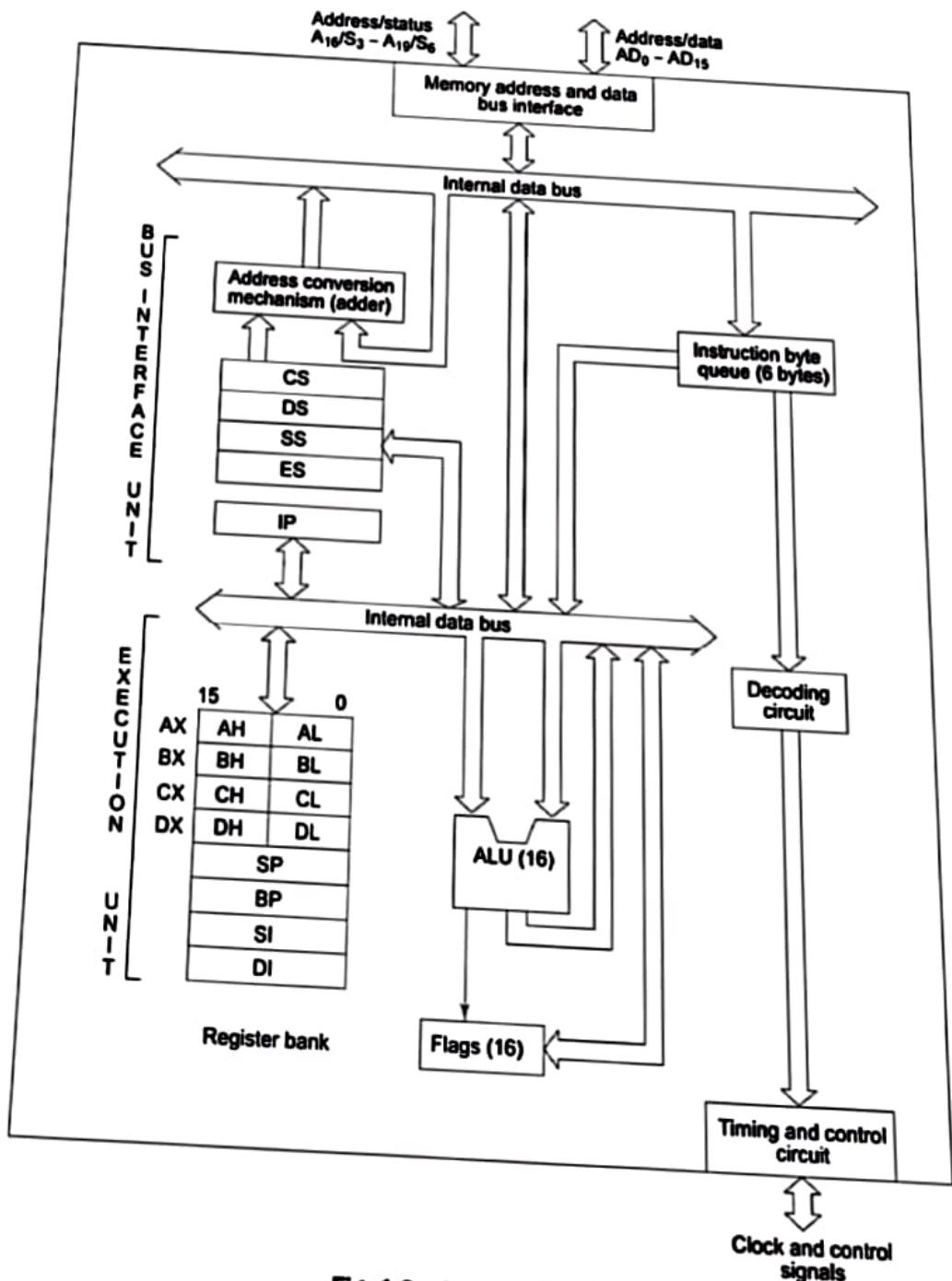


Fig. 1.2 8086 Architecture

X  Physical Memory Organis...  
From geeksforgeeks.org – delivered by



GEEKSFORGEEKS

## Physical Memory Organisation of 8086

Intel introduced its first 4-bit microprocessor in 1971 and 8-bit microprocessor in 8008 in 1972. However, this microprocessor could not survive due to its design and performance limitations. Later, the launch of the first 8-bit general purpose registers 8080 in 1974 by Intel is considered to be the first stepping stone towards the development of advanced microprocessors. The microprocessor 8085 was followed by 8080, with a few more added features to its architecture, which resulted in a functionally complete microprocessor.

## Some Basic Terminologies in 8086 Physical Memory Organization

Below are some basic terminologies that are used in 8086 Physical Memory Organization

### Memory Segment

It is a portion of memory that is used to address the data and instructions.

Offset Address

Open In App

## 8051 Memory Organization

download.mikroe.com



8051 Memory Organization  
8051 Specifics >

< >

# 8051 Memory Organization

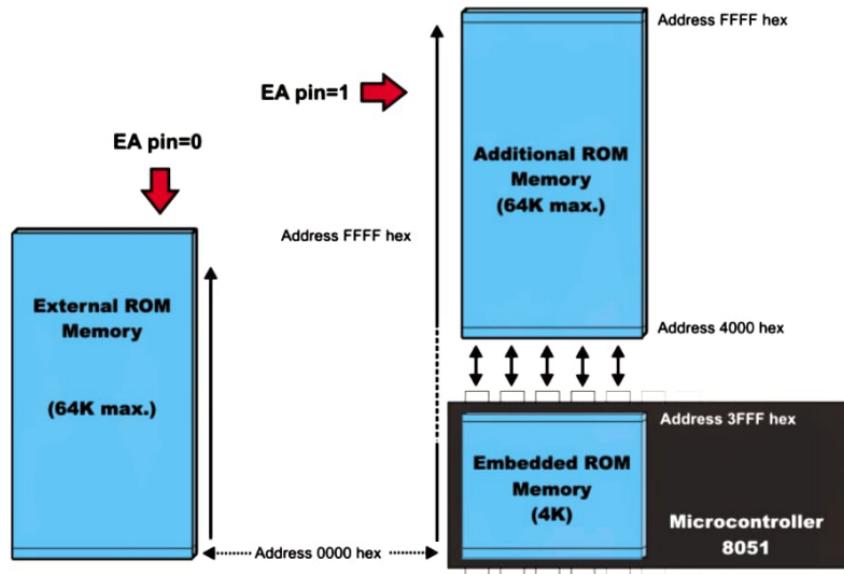
The 8051 microcontroller's memory is divided into Program Memory and Data Memory. Program Memory (ROM) is used for permanent saving program being executed, while Data Memory (RAM) is used for temporarily storing and keeping intermediate results and variables.

## Program Memory (ROM)

Program Memory (ROM) is used for permanent saving program (CODE) being executed. The memory is read only. Depending on the settings made in compiler, program memory may also be used to store constant variables. The 8051 executes programs stored in program memory only. `code` memory type specifier is used to refer to program memory.

8051 memory organization allows external program memory to be added.

How does the microcontroller handle external memory depends on the pin EA logical state.



## Internal Data Memory

Up to 256 bytes of internal data memory are available depending on the 8051 derivative. Locations available to the user occupy addressing space from 0 to 7Fh, i.e. first 128 registers and this part of RAM is divided in several blocks. The first 128 bytes of internal data memory are both directly and indirectly addressable. The upper 128 bytes of data memory (from 0x80 to 0xFF) can be addressed only indirectly.

Since internal data memory is used for CALL stack also and there is only 256 bytes split over few different memory areas



# Microcontroller

All About 8051 Microcontroller. Learn All Basics  
Of Microcontrollers And Embedded Systems.

[Home](#)[Service](#)[About](#)[Advertise](#)[Contact](#)

## Some Salient Features Of The 8051 Microcontroller

By [s](#) On [23:40](#)

- **On-chip RAM:** Random access memory of 128 byte is used for data storage in 8051. RAM as a non-volatile memory consists of register banks, stacks for temporary data storage and some special function registers.
- **On-chip ROM:** 8051 consists of 4KB ROM for program storage. ROM as a volatile memory helps in permanent data storage.
- **Timers and Counters:** Timer helps in providing delay between the events. In 8051, there are two timer pins T0, T1. If these pins are used in the counter mode, we can count the external pulses. In T0, it is possible to store 16 bit data. This is done by storing the lower 8 bit in TL0 and the upper 8 bit in TH0. Similarly, we can store 16 bit data in T1 also. TMOD and TCON helps in the timer operation.

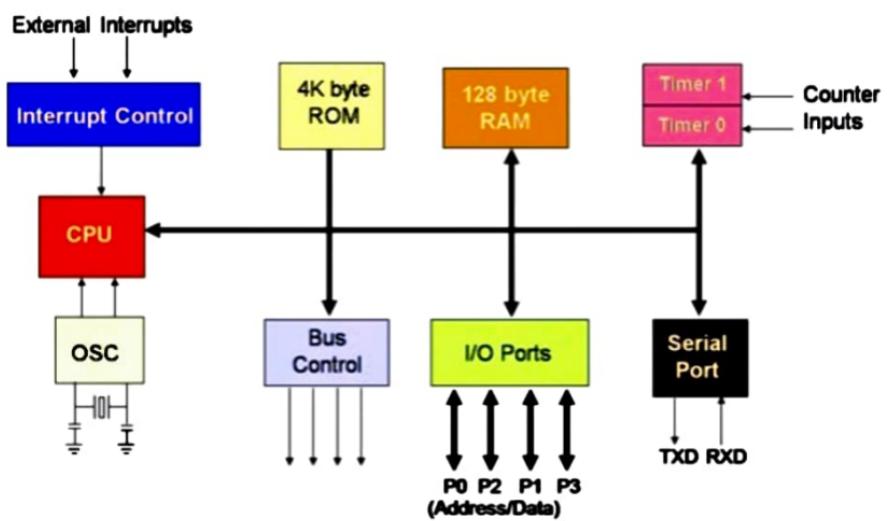
[Like 0](#)[Post](#)

[← Prev](#)[Next →](#)

# 8051

## Microcontroller Architecture

Let's see the internal architecture of 8051 Microcontroller represented in form of block diagram as shown below:



Basic components present internally inside 8051 Microcontroller architecture are:

ADVERTISEMENT

# Difference Between Microprocessor and Microcontroller

Here is the difference between Microprocessor vs. Microcontroller



## Microprocessor

Microprocessor is the heart of Computer system.

It is only a processor, so memory and I/O components need to be connected externally

Memory and I/O has to be connected externally, so the

## Microcontroller

Micro Controller is the heart of an embedded system.

Micro Controller has a processor along with internal memory and I/O components.

Memory and I/O are already present, and the



= ChatGPT 3.5



TE

**You**

(a) Write a program to copy the value 5511 into RAM memory locations 40H to 41H using in 8051 assembly language

1) direct addressing mode

2) register indirect addressing mode without a loop

3) with a loop

**ChatGPT**

Certainly! Here are three examples in 8051 assembly language for copying the value 5511 into RAM memory locations 40H to 41H using different addressing modes:

## 1) Direct Addressing Mode:

assembly

Copy code

```
MOV DPTR, #40H ; Set DPTR to the sta
```

Message ChatGPT...