

Topics for Final (Tentative)

| 1. | Assembly Language: Arithmetic + Some Opcode | Section | Date | Class |
|--------------------------|--|---------|------|-------|
| 2. | Assembly Language: Interrupt Support (Hard Copy) | | | 01 |
| 3. | How to Mask Interrupts/ Masking of Interrupts | | | 01 |
| 4. | SIM Instruction (Set Interrupt Mask) | | | |
| 5. | RIM Instruction (Read Interrupt Mask) | | | |
| 6. | SIM vs RIM | | | |
| 7. | Interfacing Memory And I/O Devices With 8085 | | | |
| 8. | Interfacing Memory Chips With 8085 | | | 04 |
| 9. | Specification of IC 2764 | | | 05-06 |
| 10. | Interfacing IC 2764 With The 8085 | | | 05-06 |
| 11. | Circuit used to generate MEMR and MEMW signals | | | 04 |
| 12. | Specification of IC 6264 | | | 07-08 |
| 13. | Interfacing 6264 IC With The 8085 | | | 07-08 |
| 14. | Specification of IC 6116 | | | 09-10 |
| 15. | Interfacing Two 6116 RAM Chips Using 74LS138 Decoder | | | 09-10 |
| 16. | Peripheral Mapped I/O Interfacing | | | 09-10 |
| 17. | Interfacing of 8-bit DIP switch with 8085 | | | 09-10 |
| 18. | INTEL 8255: (Programmable Peripheral Interface) | | | 11-16 |
| 19. | INTEL 8255: (Programmable Peripheral Interface): Pinout | | | 11-16 |
| 20. | INTEL 8255: (Programmable Peripheral Interface): Blocks | | | 11-16 |
| 21. | INTEL 8255: (Programmable Peripheral Interface): Ports | | | 11-16 |
| 22. | INTEL 8255: (Programmable Peripheral Interface): Modes | | | 11-16 |
| 23. | INTEL 8259A Programmable Interrupt Controller: Description | | | 11-16 |
| 24. | INTEL 8259A Programmable Interrupt Controller: Pinout | | | 11-16 |
| 25. | INTEL 8259A Programmable Interrupt Controller: Blocks | | | 11-16 |
| 26. | INTEL 8259A Programmable Interrupt Controller: IRR | | | 11-16 |
| 27. | INTEL 8259A Programmable Interrupt Controller: IMR | | | 11-16 |
| 28. | INTEL 8259A Programmable Interrupt Controller: ISR | | | 11-16 |
| 29. | 8257 Direct Memory Access Controller: Description | | | 11-16 |
| 30. | 8257 Direct Memory Access Controller: Pinout | | | 11-16 |
| 31. | 8257 Direct Memory Access Controller: Internal Architecture | | | 11-16 |
| 32. | 8257 Direct Memory Access Controller: Address Register | | | 11-16 |
| 33. | 8257 Direct Memory Access Controller: Mode | | | 11-16 |
| 34. | 8257 Direct Memory Access Controller: Status | | | 11-16 |
| 35. | 8257 Direct Memory Access Controller: Signal Description | | | 11-16 |
| 36. | Interfacing 8257 with 8086 | | | 11-16 |
| 37. | Minimum and Maximum Modes of 8086 | | | 11-16 |
| 38. | Intel 8251: Description | | | 11-16 |
| 39. | Intel 8251: Pinout | | | 11-16 |
| 40. | Intel 8251: Blocks | | | 11-16 |
| 41. | Intel 8251: Interfacing | | | 11-16 |
| 42. | Programmable Interval Timer | | | 11-16 |
| 43. | Difference between 8253 and 8254 | | | 11-16 |
| 44. | The INTEL 8279 is a Keyboard/Display Controller | | | 11-16 |
| 45. | The INTEL 8279 is a Keyboard/Display Controller: Sections | | | 11-16 |
| 46. | The INTEL 8279 is a Keyboard/Display Controller: Interfacing | | | 11-16 |
| 47. | ADC0808 8 bit A/D Convertor | | | 11-16 |
| 48. | ADC0808 8 bit A/D Convertor: Features | | | 11-16 |
| 49. | ADC0808 8 bit A/D Convertor: Pin Layout | | | 11-16 |
| 50. | ADC0808 Interfacing With 8051 | | | 11-16 |
| 51. | ADC0808 Interfacing With 8051 for Temperature Sensor | | | 11-16 |
| 52. | 8051 microcontrollers: Memory organization | | | 11-16 |
| 53. | 8051 microcontrollers: Basic 8051 Architecture | | | 11-16 |
| 54. | 8051 microcontrollers: features | | | 11-16 |
| 55. | What is 7-Segment Display | | | |
| 56. | 7-Segment Display: Interfacing with 8051 | | | |
| More topic will be added | | | | |

| | |
|---|------------------------------|
| Class 01 Topics: 1. Assembly Language: Arithmetic + Some Opcode 2. Assembly Language: Interrupt Support (Hard Copy) | DATE: © www.imanulhuq.com |
|---|------------------------------|

Assembly Language: Arithmetic + Some Opcode

Instructions (Reference):

| Opcode | Operand | Meaning | Explanation |
|--------|-------------------|--|---|
| ADD | R M | Add register or memory, to the accumulator | The contents of the register or memory are added to the contents of the accumulator and the result is stored in the accumulator. Example – ADD K. |
| ADC | R M | Add register to the accumulator with carry | The contents of the register or memory & M the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. |
| ADI | 8bD | Add the immediate to the accumulator | The 8-bit data is added to the contents of the accumulator and the result is stored in the accumulator. |
| ACI | 8bD | Add the immediate to the accumulator with carry | The 8-bit data and the Carry flag are added to the contents of the accumulator and the result is stored in the accumulator. |
| LXI | Reg. pair 16bD | Load the register pair immediate | The instruction stores 16-bit data into the register pair designated in the operand. |
| DAD | Reg. pair | Add the register pair to H and L registers | The 16-bit data of the specified register pair are added to the contents of the HL register. |
| SUB | R M | Subtract the register or the memory from the accumulator | The contents of the register or the memory are subtracted from the contents of the accumulator, and the result is stored in the accumulator. |
| SBB | R M | Subtract the source and borrow from the accumulator | The contents of the register or the memory & M the Borrow flag are subtracted from the contents of the accumulator and the result is placed in the accumulator. |
| SUI | 8bD | Subtract the immediate from the accumulator | The 8-bit data is subtracted from the contents of the accumulator & the result is stored in the accumulator. |
| SBI | 8bD | Subtract the immediate from the accumulator with borrow | The contents of register H are exchanged with the contents of register D, and the contents of register L are exchanged with the contents of register E. |
| INR | R M | Increment the register or the memory by 1 | The contents of the designated register or the memory are incremented by 1 and their result is stored at the same place. |
| INX | R | Increment register pair by 1 | The contents of the designated register pair are incremented by 1 and their result is stored at the same place. |
| DCR | R M | Decrement the register or the memory by 1 | The contents of the designated register or memory are decremented by 1 and their result is stored at the same place. |
| DCX | R | Decrement the register pair by 1 | The contents of the designated register pair are decremented by 1 and their result is stored at the same place. |

| Instruction | Operation | Instruction | Operation |
|-------------|-----------------------|-------------|-------------------------|
| ADD R | $A+R \rightarrow A$ | ADC R | $A+R+C \rightarrow A$ |
| ADD M | $A+M \rightarrow A$ | ADC M | $A+M+C \rightarrow A$ |
| ADI 8bD | $A+8bD \rightarrow A$ | ADC 8bD | $A+8bD+C \rightarrow A$ |
| SUB R | $A-R \rightarrow A$ | SBB R | $A-R-C \rightarrow A$ |
| SUB M | $A-M \rightarrow A$ | SBB M | $A-M-C \rightarrow A$ |

Class 02 Topics:

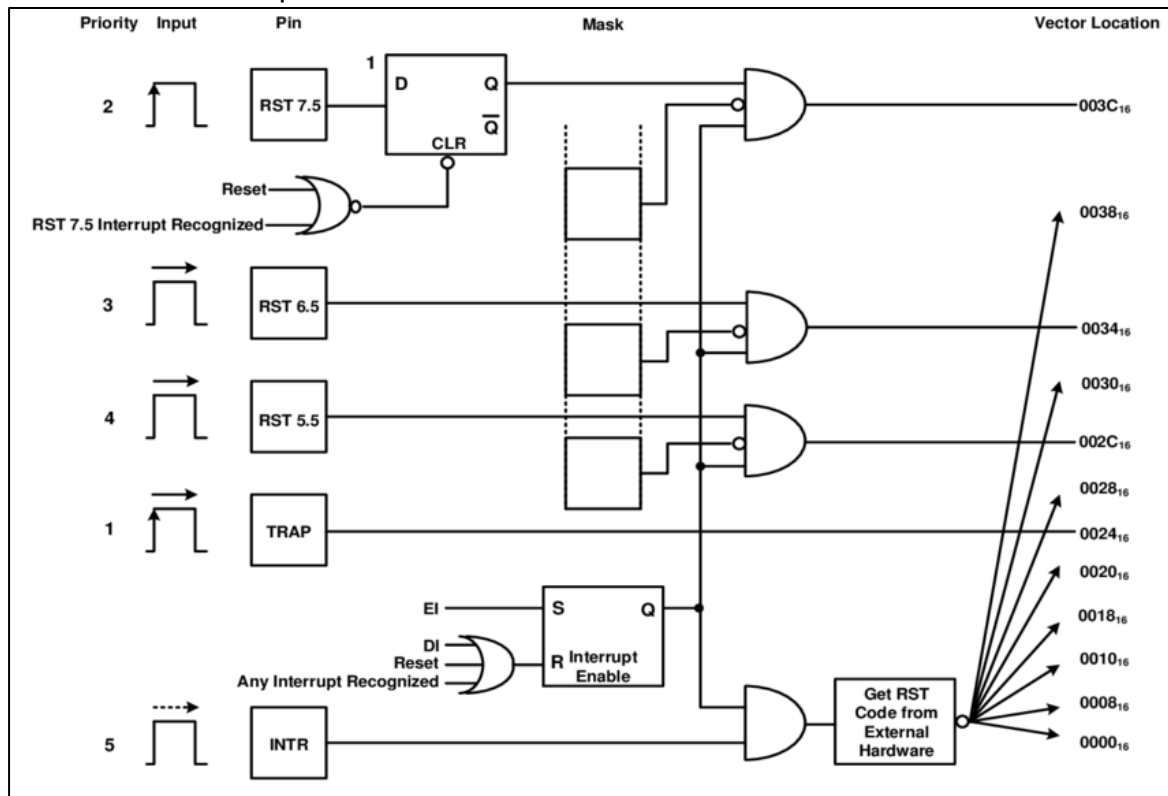
1. How to Mask Interrupts/ Masking of Interrupts
2. SIM Instruction (Set Interrupt Mask)
3. RIM Instruction (Read Interrupt Mask)

DATE:

© www.imanulhuq.com

How to Mask Interrupts/ Masking of Interrupts

- ✓ Masking can be done for 4 hardware interrupts INTR, RST 5.5, RST 6.5 & RST 7.5
- ✓ The masking of 8085 interrupts is done at different levels
- ✓ 8085 Interrupts and Vector Locations



✓ Explanation of Figure:

- i. The maskable interrupts are by default masked by the Reset signal
- ii. So, no interrupt is recognized by the hardware reset
- iii. The interrupts can be enabled by the EI instruction
- iv. The three RST interrupts can be selectively masked by loading the appropriate word in the accumulator and executing SIM instruction. This is called software masking.
- v. All maskable interrupts are disabled whenever an interrupt is recognized.
- vi. All maskable interrupts can be disabled by executing the DI instruction.

1. SIM Instruction (Set Interrupt Mask)
2. RIM Instruction (Read Interrupt Mask)

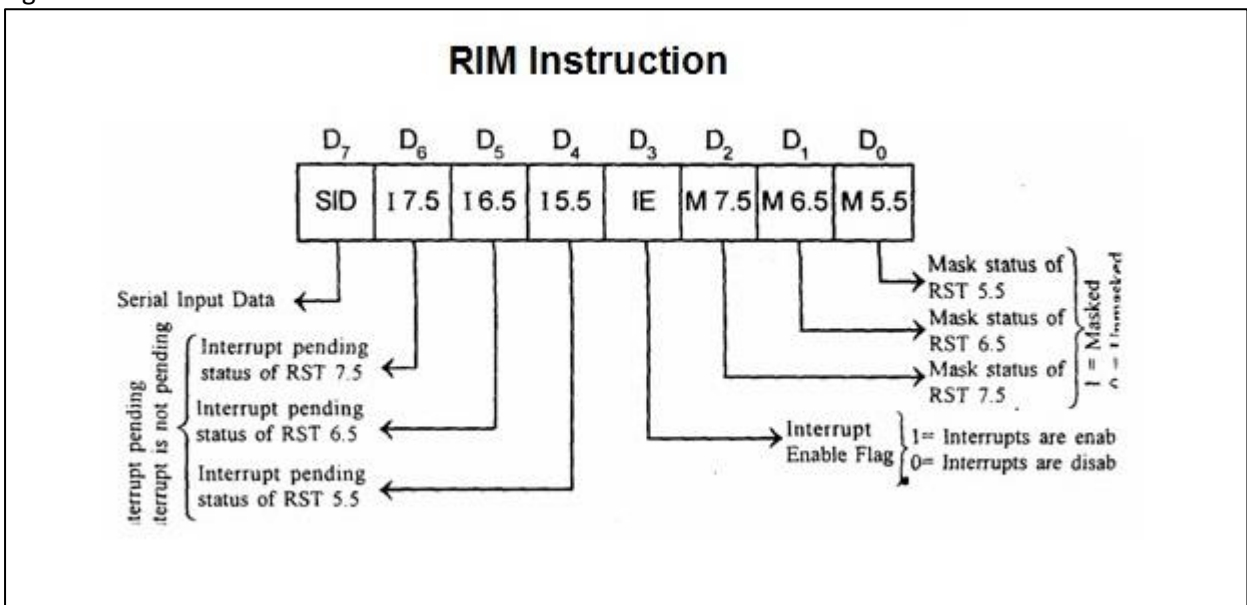
SIM Instruction (Set Interrupt Mask)

- ✓ The SIM instruction is used to mask or unmask RST hardware interrupts
- ✓ When executed, the SIM instruction reads the content of accumulator
- ✓ And accordingly, mask or unmask the interrupts.
- ✓ The format of control word to be stored in the accumulator before executing SIM instruction
- ✓ Figure:

| Bit position | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|--------------|------------------------|--|----------|-------------------------|--|--------------------------|--------------------------|--------------------------|
| Name | SOD | SDE | X | R7.5 | MSE | M7.5 | M6.5 | M5.5 |
| Explanation | Serial data to be sent | Serial data enable— set to 1 for sending | Not used | Reset RST 7.5 flip-flop | Mask set enable— Set to 1 to mask interrupts | Set to 1 to mask RST 7.5 | Set to 1 to mask RST 6.5 | Set to 1 to mask RST 5.5 |

RIM Instruction (Read Interrupt Mask)

- ✓ RIM instruction is used to read the status of the interrupt mask bits
- ✓ When RIM instruction is executed, the accumulator is loaded with the current status of the interrupt masks
- ✓ And the pending interrupts
- ✓ The format and the meaning of the data stored in the accumulator after execution of RIM instruction
- ✓ In addition, RIM instruction is also used to read the serial data on the SID pin of the processor
- ✓ The data on the SID pin is stored in the MSB of the accumulator after the execution of the RIM instruction
- ✓ Figure:



Class 04 Topics:

1. SIM vs RIM
2. Interfacing Memory And I/O Devices With 8085
3. Interfacing Memory Chips With 8085
4. Circuit used to generate MEMR and MEMW signals

DATE:

© www.imanulhuq.com

Difference SIM and RIM

Home Work

Date of Submission:

Interfacing Memory And I/O Devices With 8085

The programs and data that are executed by the microprocessor have to be stored in ROM/EPROM and RAM, which are basically semiconductor memory chips.

The programs and data that are stored in ROM/EPROM are not erased even when power supply to the chip is removed. Hence, they are called non-volatile memory. They can be used to store permanent programs.

In a RAM, stored programs and data are erased when the power supply to the chip is removed. Hence, RAM is called volatile memory.

RAM can be used to store programs and data that include, programs written during software development for a microprocessor-based system, program written when one is learning assembly language programming and data enter while testing these programs.

Input and output devices, which are interfaced with 8085, are essential in any microprocessor-based system.

They can be interfaced using two schemes: I/O mapped I/O and memory-mapped I/O. In the I/O mapped I/O scheme, the I/O devices are treated differently from memory. In the memory-mapped I/O scheme, each I/O device is assumed to be a memory location.

Interfacing Memory Chips With 8085

8085 has 16 address lines (A0 - A15), hence a maximum of 64 KB (= 2¹⁶ bytes) of memory locations can be interfaced with it. The memory address space of the 8085 takes values from 0000H to FFFFH.

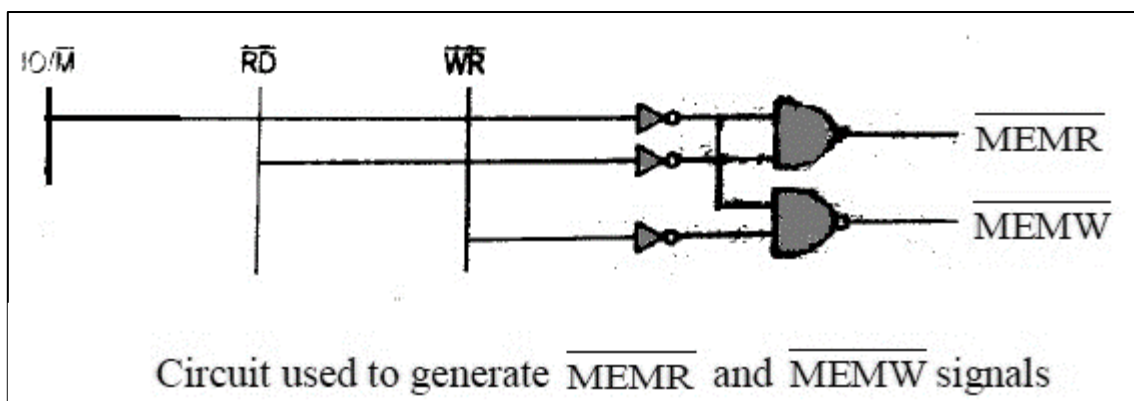
The 8085 initiates set of signals such as IO/M, RD and WR when it wants to read from and write into memory.

Similarly, each memory chip has signals such as CE or CS (chip enable or chip select), OE or RD (output enable or read) and WE or WR (write enable or write) associated with it.

Generation of Control Signals for Memory:

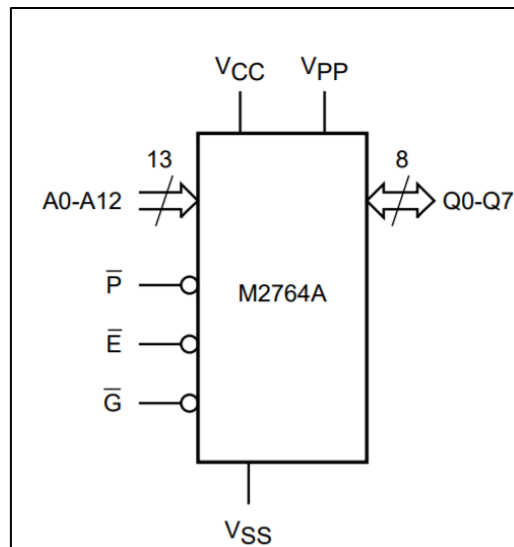
| $\overline{\text{IO/M}}$ | $\overline{\text{RD}}$ | $\overline{\text{WR}}$ | Operation |
|--------------------------|------------------------|------------------------|------------------------------|
| 0 | 0 | 1 | 8085 reads data from memory |
| 0 | 1 | 0 | 8085 writes data into memory |

Circuit used to generate MEMR and MEMW signals



Specification of IC 2764

- ✓ NMOS
- ✓ 64 Kbit (8Kb x 8) = 8 KB (8 x 2¹⁰ byte)
- ✓ UV EPROM
- ✓ 13 address lines (2¹³ bytes = 8 KB)
- ✓ Logic Block of M2764A:



- ✓ Pinout M2764A:

Read Mode

The M2764A has two control functions, both of which must be logically satisfied in order to obtain data at the outputs. Chip Enable (\bar{E}) is the power control and should be used for device selection. Output Enable (\bar{G}) is the output control and should be used to gate data to the output pins, independent of device selection.

Assuming that the addresses are stable, address access time (t_{AVQV}) is equal to the delay from \bar{E} to output (t_{ELQV}). Data is available at the outputs after the falling edge of \bar{G} , assuming that \bar{E} has been low and the addresses have been stable for at least $t_{AVQV} - t_{GLQV}$.

Standby Mode

The M2764A has a standby mode which reduces the maximum active power current from 75mA to 35mA. The M2764A is placed in the standby mode by applying a TTL high signal to the \bar{E} input. When in the standby mode, the outputs are in a high impedance state, independent of the \bar{G} input.

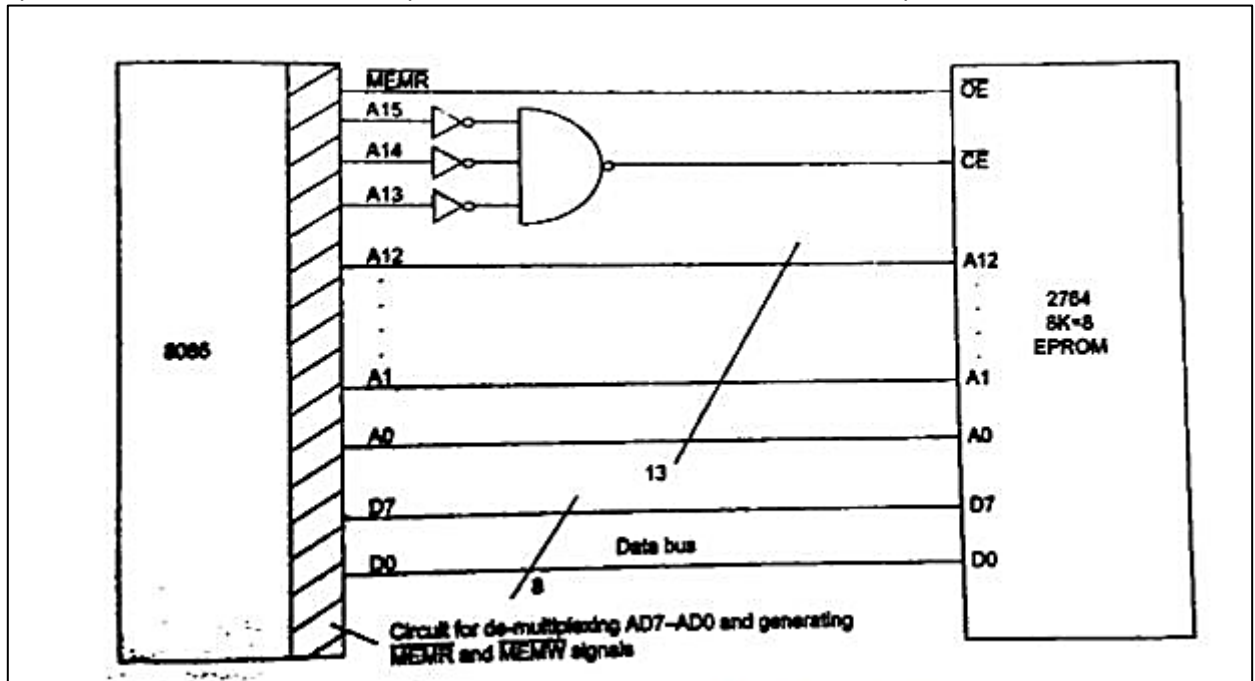
| | | | | |
|-----|----|--------|----|-----------|
| VPP | 1 | | 28 | VCC |
| A12 | 2 | | 27 | \bar{P} |
| A7 | 3 | | 26 | NC |
| A6 | 4 | | 25 | A8 |
| A5 | 5 | | 24 | A9 |
| A4 | 6 | | 23 | A11 |
| A3 | 7 | M2764A | 22 | \bar{G} |
| A2 | 8 | | 21 | A10 |
| A1 | 9 | | 20 | \bar{E} |
| A0 | 10 | | 19 | Q7 |
| Q0 | 11 | | 18 | Q6 |
| Q1 | 12 | | 17 | Q5 |
| Q2 | 13 | | 16 | Q4 |
| VSS | 14 | | 15 | Q3 |

Interfacing IC 2764 With The 8085

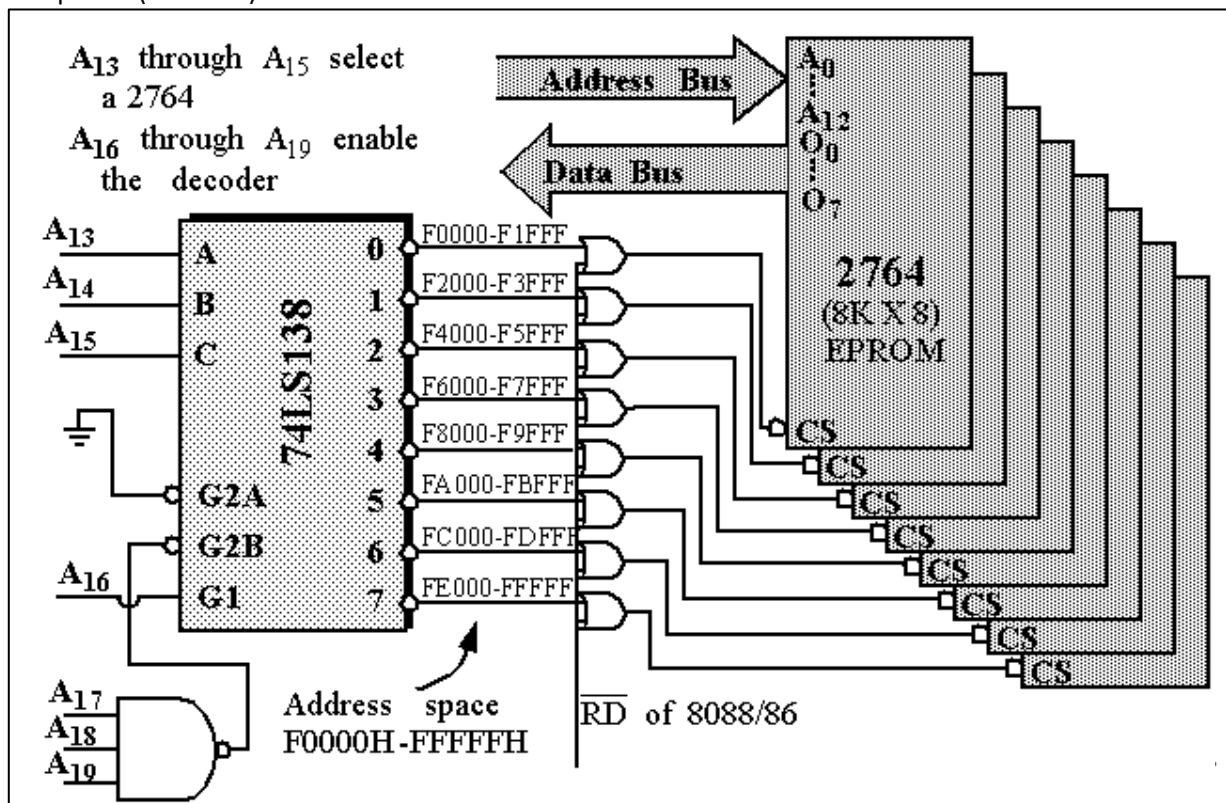
- ✓ 13 address lines of IC are connected to the corresponding address lines of 8085
- ✓ Remaining address lines of 8085 are connected to address decoder formed using logic gates, the output of which is connected to the CE pin of IC
- ✓ Address range allocated to the chip is shown in Table

| A ₁₅ | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ADD |
|-----------------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|-------|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8000H |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 87FFH |

- ✓ Chip is enabled whenever the 8085 places an address allocated to EPROM chip in the address bus



Using Multiplexer (Advance)

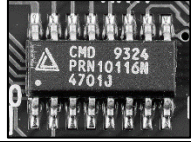


Specification of IC 6264

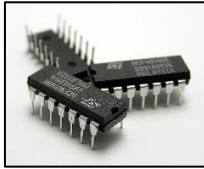
- ✓ The 6264 is a JEDEC-standard static RAM integrated circuit. It has a capacity of 64 Kbit (8 KB). It is produced by a wide variety of different vendors, including Hitachi, Hynix, and Cypress Semiconductor.
- ✓ It is available in a variety of different configurations, such as DIP, SPDIP, and SOIC

Note:

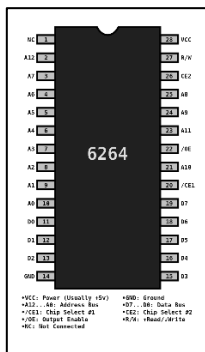
- small outline integrated circuit (SOIC)



- dual in-line package (DIP or DIL)

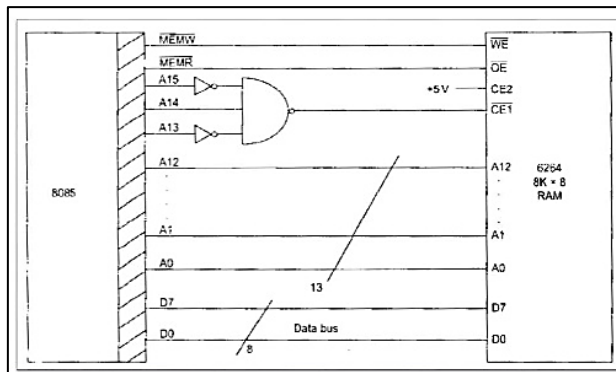


- ✓ Some versions of the 6264 can run in ultra-low-power mode and retain memory when not in use, thus making them suitable for battery backup applications
- ✓ Pinout:

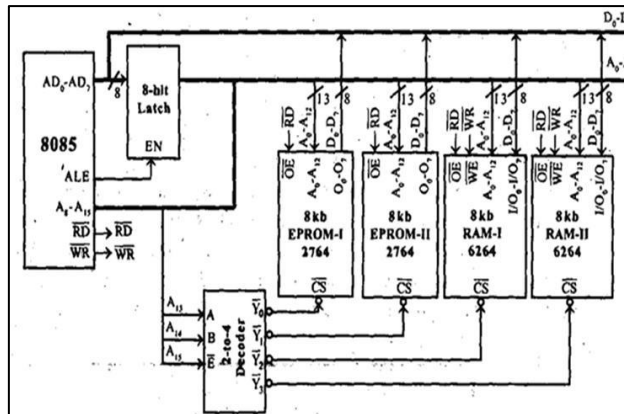


- ✓ 8K x 8 RAM
- ✓ 8 KB = 213 bytes
- ✓ 13 address lines

Interfacing 6264 IC With The 8085



Using Multiplexer (Advance)

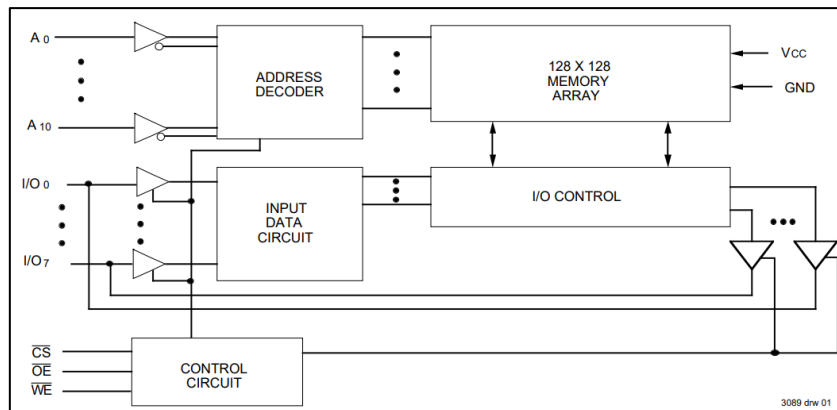


Specification of IC 6116

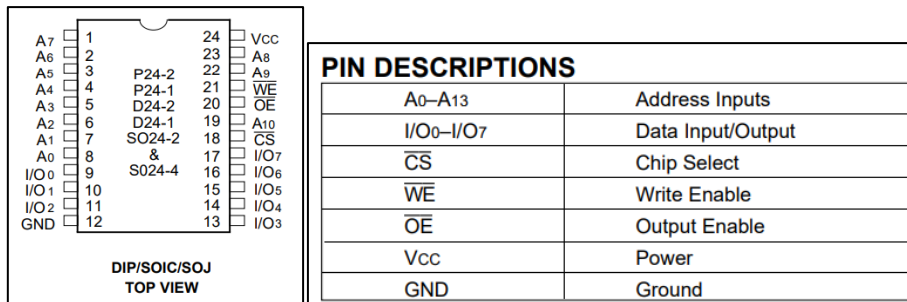
Features:

- ✓ High-speed access and chip select times
 - Military: 20/25/35/45/55/70/90/120/150ns (max.)
 - Commercial: 15/20/25/35/45ns (max.)
- ✓ Low-power consumption
- ✓ Battery backup operation
 - 2V data retention voltage (LA version only)
- ✓ Produced with advanced CMOS high-performance technology
- ✓ CMOS process virtually eliminates alpha particle soft-error rates
- ✓ Input and output directly TTL-compatible
- ✓ Static operation: no clocks or refresh required
- ✓ Available in ceramic and plastic 24-pin DIP, 24-pin Thin Dip and 24-pin SOIC and 24-pin SOJ
- ✓ Military product compliant to MIL-STD-883, Class B

Functional Block Diagram



Pinout:



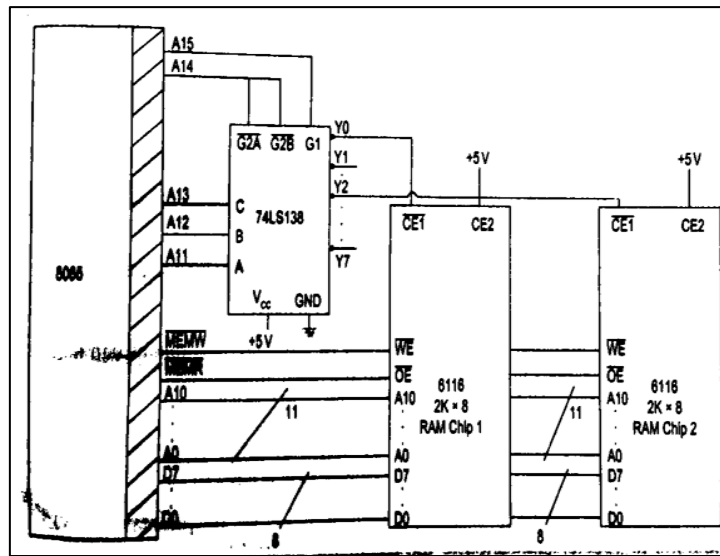
Interfacing Two 6116 RAM Chips Using 74LS138 Decoder

- ✓ A0 – A10 lines of 8085 are connected to 11 address lines of the RAM chips.
- ✓ Three address lines of 8085 having specific value for a particular RAM are connected to the three select inputs (C, B and A) of 74LS138 decoder.
- ✓ Table shows that A13=A12=A11=0 for the address assigned to RAM 1 and A13=0, A12=1 and A11=0 for the address assigned to RAM 2

| A15 | A14 | A13 | A12 | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | Address |
|-----|-----|-----|-----|-----|-----|----|----|----|----|----|----|----|----|----|----|-----------------------|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 8000H |
| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 87FFH (RAM chip 1) |
| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9000H |
| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 97FFH (RAM chip 2) |

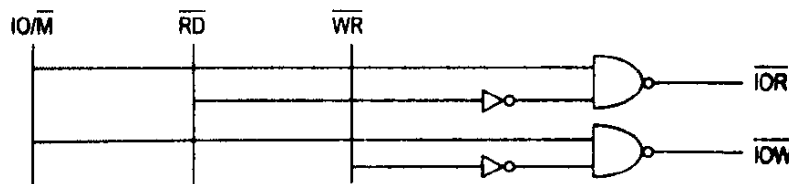
- ✓ Remaining lines of 8085 which are constant for the address range assigned to the two RAM are connected to the enable inputs of decoder

- ✓ When 8085 places any address between 8000H and 87FFH in the address bus, the select inputs C, B and A of the decoder are all 0. The Y0 output of the decoder is also 0, selecting RAM 1. When 8085 places any address between 9000H and 97FFH in the address bus, the select inputs C, B and A of the decoder are 0, 1 and 0. The Y2 output of the decoder is also 0, selecting RAM 2
- ✓ The Y2 output of the decoder is also 0, selecting RAM 2
- ✓ Figure:



Peripheral Mapped I/O Interfacing

In this method, the I/O devices are treated differently from memory chips. The control signals I/O read (\overline{IOR}) and I/O write (\overline{IOW}), which are derived from the $\overline{IO/\overline{M}}$, \overline{RD} and \overline{WR} signals of the 8085, are used to activate input and output devices, respectively. Generation of these control signals is shown in Fig. 20. Table 11 shows the status of $\overline{IO/\overline{M}}$, \overline{RD} and \overline{WR} signals during I/O read and I/O write operation.



Generation of \overline{IOR} and \overline{IOW} signals

IN instruction is used to access input device and OUT instruction is used to access output device. Each I/O device is identified by a unique 8-bit address assigned to it. Since the control signals used to access input and output devices are different, and all I/O device use 8-bit address, a maximum of 256 (2^8) input devices and 256 output devices can be interfaced with 8085.

Status of \overline{IOR} and \overline{IOW} signals in 8085.

| $\overline{IO/\overline{M}}$ | \overline{RD} | \overline{WR} | \overline{IOR} | \overline{IOW} | Operation |
|------------------------------|-----------------|-----------------|------------------|------------------|--------------------------------|
| 1 | 0 | 1 | 0 | 1 | I/O read operation |
| 1 | 1 | 0 | 1 | 0 | I/O write operation |
| 0 | X | X | 1 | 1 | Memory read or write operation |

Class 11-16 Topics:

INTEL 8255: Programmable Peripheral Interface
INTEL 8259A: Programmable Interrupt Controller
8257 Direct Memory Access Controller
Intel 8251
Programmable interval timer
INTEL 8279
ADC0808
8051 microcontrollers

DATE:

© www.imanulhuq.com

INTEL 8255: Programmable Peripheral Interface

Intro

The Intel 8255 (or i8255) Programmable Peripheral Interface (PPI) chip was developed and manufactured by Intel in the first half of the 1970s for the Intel 8080 microprocessor. The 8255 provides 24 parallel input/output lines with a variety of programmable operating modes.

The 8255 is a member of the MCS-85 Family of chips, designed by Intel for use with their 8085 and 8086 microprocessors and their descendants. It was first available in a 40-pin DIP and later a 44-pin PLCC packages. It found wide applicability in digital processing systems and was later cloned by other manufacturers. The 82C55 is a CMOS version for higher speed and lower current consumption.

The functionality of the 8255 is now mostly embedded in larger VLSI processing chips as a sub-function. A CMOS version of the 8255 is still being made by Renesas but mostly used to expand the I/O of microcontrollers.

Function

The 8255 gives a CPU or digital system access to programmable parallel I/O. The 8255 has 24 input/output pins. These are divided into three 8-bit ports (A, B, C). Port A and port B can be used as 8-bit input/output ports. Port C can be used as an 8-bit input/output port or as two 4-bit input/output ports or to produce handshake signals for ports A and B.

The three ports are further grouped as follows:

1. Group A consisting of port A and upper part of port C.
2. Group B consisting of port B and lower part of port C.

Eight data lines (D0–D7) are available (with an 8-bit data buffer) to read/write data into the ports or control register. Address lines A₁ and A₀ allow to access a data register for each port or a control register, as listed below:

| A ₁ | A ₀ | Port selected |
|----------------|----------------|------------------|
| 0 | 0 | port A |
| 0 | 1 | port B |
| 1 | 0 | port C |
| 1 | 1 | control register |

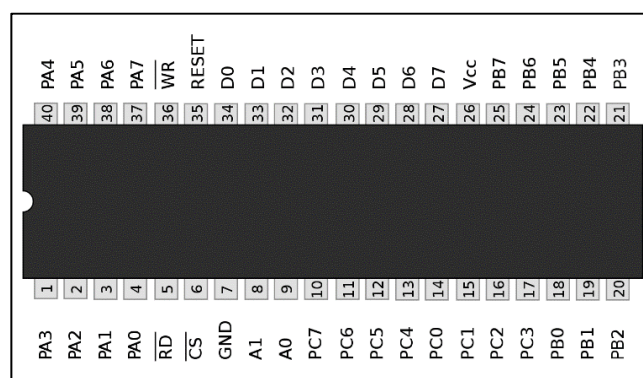
Operational modes of 8255

There are two basic operational modes of 8255:

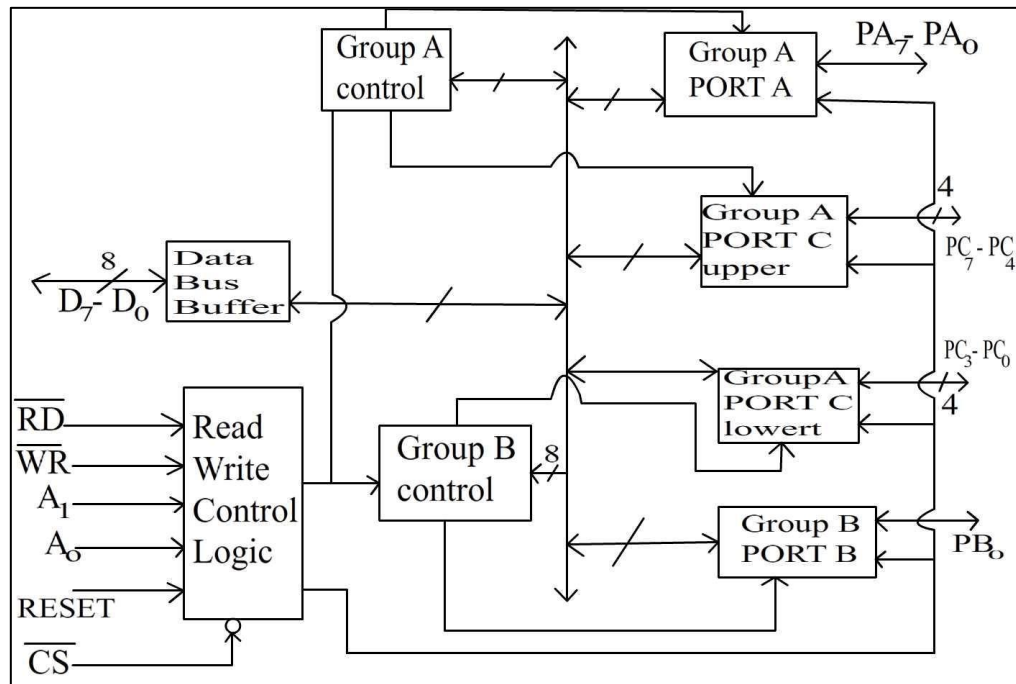
1. Bit Set/Reset mode (BSR mode).
2. Input/output mode (I/O mode).

The two modes are selected on the basis of the value present at the D7 bit of the control word register. When D7 = 1, 8255 operates in I/O mode, and when D7 = 0, it operates in the BSR mode.

Pinout



Block



Modes

There are three basic modes of operation that can be selected by the system software.

Mode 0: Basic Input/output

Mode 1: Strobes Input/output

Mode 2: Bi-direction bus.

INTEL 8259A: Programmable Interrupt Controller

Intro

The Intel 8259 is a Programmable Interrupt Controller (PIC) designed for the Intel 8085 and Intel 8086 microprocessors. The initial part was 8259, a later A suffix version was upward compatible and usable with the 8086 or 8088 processor. The 8259 combines multiple interrupt input sources into a single interrupt output to the host microprocessor, extending the interrupt levels available in a system beyond the one or two levels found on the processor chip. The 8259A was the interrupt controller for the ISA bus in the original IBM PC and IBM PC AT.

Functional description

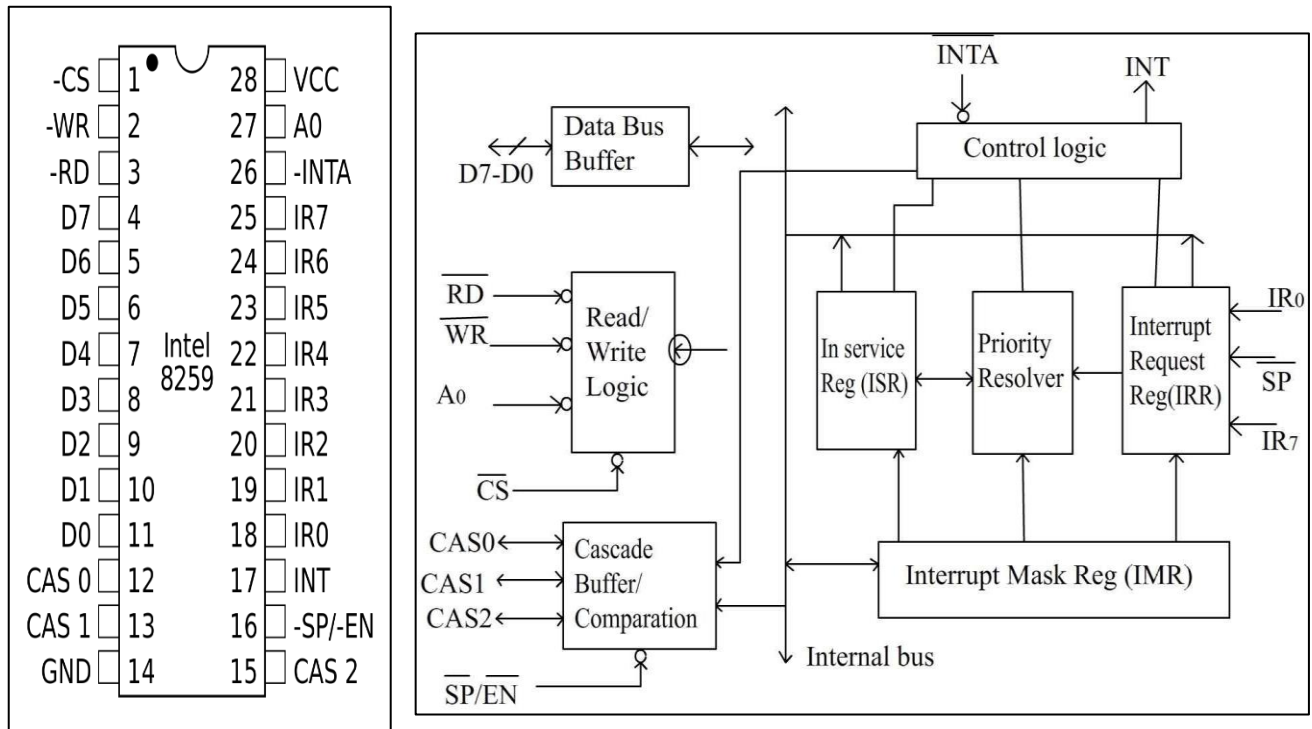
The main signal pins on an 8259 are as follows: eight interrupt input request lines named IRQ₀ through IRQ₇, an interrupt request output line named INTR, interrupt acknowledgment line named INTA, D₀ through D₇ for communicating the interrupt level or vector offset. Other connections include CAS₀ through CAS₂ for cascading between 8259s.

Up to eight slave 8259s may be cascaded to a master 8259 to provide up to 64 IRQs. 8259s are cascaded by connecting the INT line of one slave 8259 to the IRQ line of one master 8259.

There are three registers, an Interrupt Mask Register (IMR), an Interrupt Request Register (IRR), and an In-Service Register (ISR). The IRR maintains a mask of the current interrupts that are pending acknowledgement, the ISR maintains a mask of the interrupts that are pending an EOI, and the IMR maintains a mask of interrupts that should not be sent an acknowledgement.

End of Interrupt (EOI) operations support specific EOI, non-specific EOI, and auto-EOI. A specific EOI specifies the IRQ level it is acknowledging in the ISR. A non-specific EOI resets the IRQ level in the ISR. Auto-EOI resets the IRQ level in the ISR immediately after the interrupt is acknowledged.

Edge and level interrupt trigger modes are supported by the 8259A. Fixed priority and rotating priority modes are supported.



Data bus buffer: This 3- state, bidirectional 8-bit buffer is used to interface the 8259A to the system data bus. Control words and status information are transferred through the data bus buffer.

Read/Write & control logic: The function of this block is to accept OUTPUT commands from the CPU. It contains the initialization command word (ICW) register and operation command word (OCW) register which store the various control formats for device operation. This function block also allows the status of 8159A to be transferred to the data bus.

Interrupt request register (IRR): IRR stores all the interrupt inputs that are requesting service. Basically, it keeps track of which interrupt inputs are asking for service. If an interrupt input is unmasked, and has an interrupt signal on it, then the corresponding bit in the IRR will be set.

Interrupt mask register (IMR): The IMR is used to disable (Mask) or enable (Unmask) individual interrupt inputs. Each bit in this register corresponds to the interrupt input with the same number. The IMR operation on the IRR. Masking of higher priority input will not affect the interrupt request lines of lower priority. To unmask any interrupt the corresponding bit is set '0'.

In service register (ISR): The in-service registers keep tracks of which interrupt inputs are currently being serviced. For each input that is currently being serviced the corresponding bit will be set in the in-service register. Each of these 3-reg can be read as status reg.

Priority Resolver: This logic block determines the priorities of the set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during pulse.

8257 Direct Memory Access Controller

Intro

DMA stands for Direct Memory Access. It is designed by Intel to transfer data at the fastest rate. It allows the device to transfer the data directly to/from memory without any interference of the CPU.

Using a DMA controller, the device requests the CPU to hold its data, address and control bus, so the device is free to transfer data directly to/from the memory. The DMA data transfer is initiated only after receiving HLDA signal from the CPU.

How DMA Operations are Performed

Following is the sequence of operations performed by a DMA –

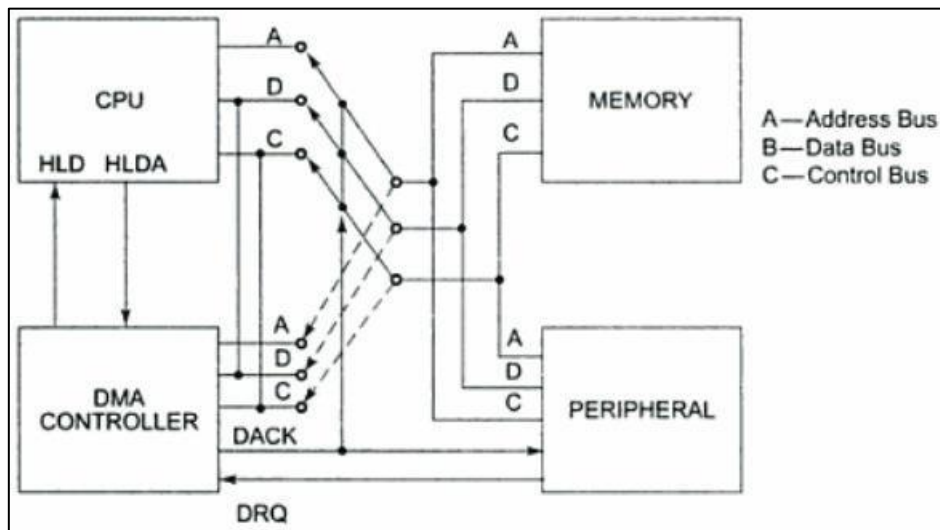
1. Initially, when any device has to send data between the device and the memory, the device has to send DMA request (DRQ) to DMA controller.
2. The DMA controller sends Hold request (HRQ) to the CPU and waits for the CPU to assert the HLDA.
3. Then the microprocessor tri-states all the data bus, address bus, and control bus. The CPU leaves the control over bus and acknowledges the HOLD request through HLDA signal.
4. Now the CPU is in HOLD state and the DMA controller has to manage the operations over buses between the CPU, memory, and I/O devices.

Features of 8257

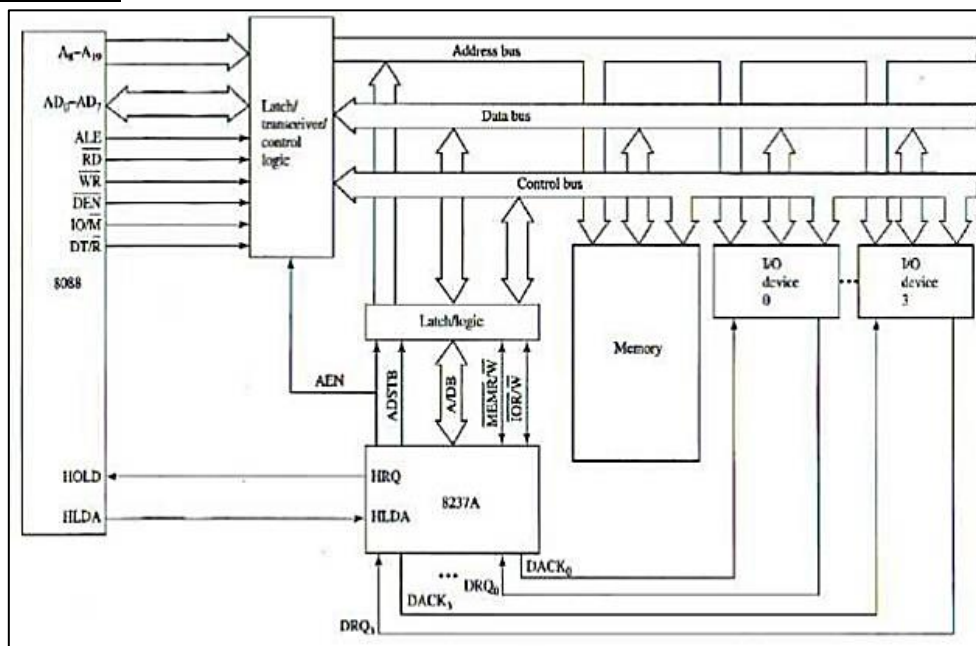
Here is a list of some of the prominent features of 8257 –

- It has four channels which can be used over four I/O devices.
- Each channel has 16-bit address and 14-bit counter.
- Each channel can transfer data up to 64kb.
- Each channel can be programmed independently.
- Each channel can perform read transfer, write transfer and verify transfer operations.
- It generates MARK signal to the peripheral device that 128 bytes have been transferred.
- It requires a single-phase clock.
- Its frequency ranges from 250Hz to 3MHz.
- It operates in 2 modes, i.e., Master mode and Slave mode

A conceptual implementation of the



Interfacing 8257 with 8086



INTEL 8251

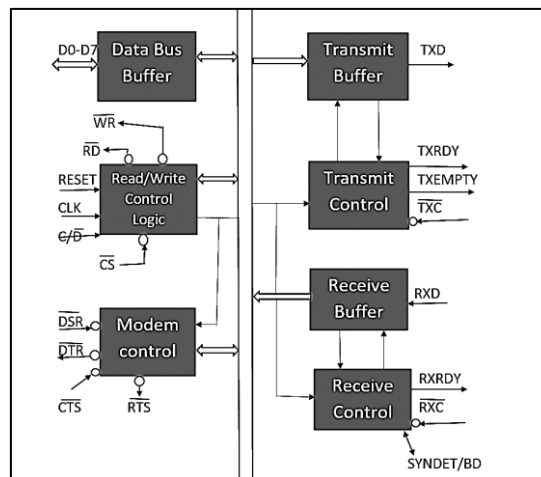
The 8251 is a Universal Synchronous/Asynchronous Receiver/Transmitter packaged in a 28-pin DIP made by Intel. It is typically used for serial communication and was rated for 19.2 kilobits per second signaling rate.

It is commonly confused with the much more common 8250 UART that was made popular as the serial port in the IBM Personal Computer.

It includes 5 sections

1. read/write control logic
2. transmitter
3. receiver
4. data bus system
5. modem control

Block Diagram of 8251 USART



Programmable interval timer

In computing and in embedded systems, a programmable interval timer (PIT) is a counter that generates an output signal when it reaches a programmed count. The output signal may trigger an interrupt.

PITs may be one-shot or periodic. One-shot timers will signal only once and then stop counting. Periodic timers signal every time they reach a specific value and then restart, thus producing a signal at periodic intervals. Periodic timers are typically used to invoke activities that must be performed at regular intervals.

Counters are usually programmed with fixed intervals that determine how long the counter will count before it signals. The interval determines how long the counter will count before it will output a signal.

The Intel 8253 and 8254 are Programmable Interval Timers (PITs), which perform timing and counting functions using three 16-bit counters.

The 825x family was primarily designed for the Intel 8080/8085-processors, but later used in x86 compatible systems. The 825x chips or equivalent circuit embedded in a larger chip are found in all IBM PC compatibles.

Difference between 8253 and 8254

The following table differentiates the features of 8253 and 8254 –

| 8253 | 8254 |
|---|--|
| Its operating frequency is 0 - 2.6 MHz | Its operating frequency is 0 - 10 MHz |
| It uses N-MOS technology | It uses H-MOS technology |
| Read-Back command is not available | Read-Back command is available |
| Reads and writes of the same counter cannot be interleaved. | Reads and writes of the same counter can be interleaved. |

INTEL 8279

The INTEL 8279 is a Keyboard/Display Controller specially developed for interfacing keyboard and display devices to Intel 8085, 8086 and 8088 microprocessors.

Its important features are:

- ✓ Simultaneous keyboard and display operations.
- ✓ Scanned keyboard mode.
- ✓ Scanned sensor mode.
- ✓ 8-character keyboard FIFO.
- ✓ Right or left entry 16-byte display RAM.
- ✓ Programmable scan timing.
- ✓ Used for Interaction between keyboard and different microprocessor.

Keyboard section:

- ✓ The keyboard section consists of eight return lines RLO - RL7 that can be used to form the columns of a keyboard matrix.
- ✓ It has two additional inputs : shift and control/strobe. The keys are automatically debounced.
- ✓ The two operating modes of keyboard section are 2-key lockout and N-key rollover.
- ✓ In the 2-key lockout mode, if two keys are pressed simultaneously, only the first key is recognized.
- ✓ In the N-key rollover mode, simultaneous keys are recognized and their codes are stored in FIFO.
- ✓ The keyboard section also has an 8 x 8 FIFO (First In First Out) RAM.
- ✓ The FIFO can store eight key codes in the scan keyboard mode. The status of the shift key and control key are also stored along with key code. The 8279 generates an interrupt signal when there is an entry in FIFO.
- ✓ In sensor matrix mode, the state (i.e., open/close status) of 64 switches is stored in FIFO RAM. If the state of any of the switches changes, then the 8279 asserts IRQ as high to interrupt the processor.

Display section:

- ✓ The display section has eight output lines divided into two groups A0-A3 and B0-B3.
- ✓ The output lines can be used either as a single group of eight lines or as two groups of four lines, in conjunction with the scan lines for a multiplexed display.
- ✓ The output lines are connected to the anodes through driver transistors in case of common cathode 7-segment LEDs.
- ✓ The cathodes are connected to scan lines through driver transistors.
- ✓ The display can be blanked by BD (low) line.
- ✓ The display section consists of 16 x 8 display RAM. The CPU can read from or write into any location of the display RAM.

Scan section:

- ✓ The scan section has a scan counter and four scan lines, SLO to SL3.
- ✓ In decoded scan mode, the output of the scan lines will be similar to a 2-to-4 decoder.
- ✓ In encoded scan mode, the output of scan lines will be binary count, and an external decoder should be used to convert the binary count to decoded output.
- ✓ The scan lines are common for keyboard and display.
- ✓ The scan lines are used to form the rows of a matrix keyboard and also connected to digit drivers of a multiplexed display, to turn ON/OFF.

CPU interface section:

- ✓ The CPU interface section takes care of data transfer between the 8279 and the processor.
- ✓ This section has eight bidirectional data lines DB0 to DB7 for data transfer between 8279 and CPU.
- ✓ It has a single address line to select the data buffer (A0=0) or the control register (A0=1) of the 8279.
- ✓ The control signals WR (active low), RD (active low), CS (active low) and A0 are used for read/write to 8279.
- ✓ It has an interrupt request line IRQ, for interrupt driven data transfer with processor.
- ✓ The internal clock frequency of the 8279 is nominally 100 kHz. An internal prescaler divides an externally applied clock by an integer between 2 and 31 (both inclusive) to produce the internal clock.
- ✓ The RESET signal sets the 8279 in 16-character display with two -key lockout keyboard modes.

Interfacing of 8279 with 8085

- ✓ Keyboard scanning
- ✓ Key debouncing
- ✓ Key code generation
- ✓ Sending display code to LED
- ✓ Display refreshing

ADC0808

One of the most commonly used ADC is ADC0808. ADC 0808 is a Successive approximation type with 8 channels i.e. it can directly access 8 single ended analog signals. The difference between ADC0804 and ADC0808 is number of analog input signals. In ADC0804 we can give only one analog input. But in ADC0808 we can give 8 analog inputs. So if you want to interface more number of analog sensors, please select this ADC0808.

ADC0808 is an 8 bit analog to digital converter with eight input analog channels, i.e., it can take eight different analog inputs. The input which is to be converted to digital form can be selected by using three address lines. The voltage reference can be set using the Vref+ and Vref- pins. The step size is decided based on set reference value. Step size is the change in analog input to cause a unit change in the output of ADC. The default step size is 19.53mV corresponding to 5V reference voltage. ADC0808 needs an external clock to operate unlike ADC0804 which has an internal clock. The ADC needs some specific control signals for its operations like start conversion and bring data to output pins. When the conversion is complete the EOC pins goes low to indicate the end of conversion and data ready to be picked up.

Features of ADC 0808

- ✓ Easy interface to all microprocessors
- ✓ Operates ratio metrically or with 5 V DC or analog span adjusted voltage reference
- ✓ No zero or full-scale adjust required
- ✓ 8-channel multiplexer with address logic
- ✓ 0V to 5V input range with single 5V power supply
- ✓ Outputs meet TTL voltage level specifications
- ✓ Standard hermetic or molded 28-pin DIP package
- ✓ 28-pin molded chip carrier package

How to use the ADC0808

Start

Select the channel.

A Low – High transition on ALE to latch in the address.

A Low – High transition on Start to reset the ADC's SAR.

A High – Low transition on ALE.

A High – Low transition on start to start the conversion.

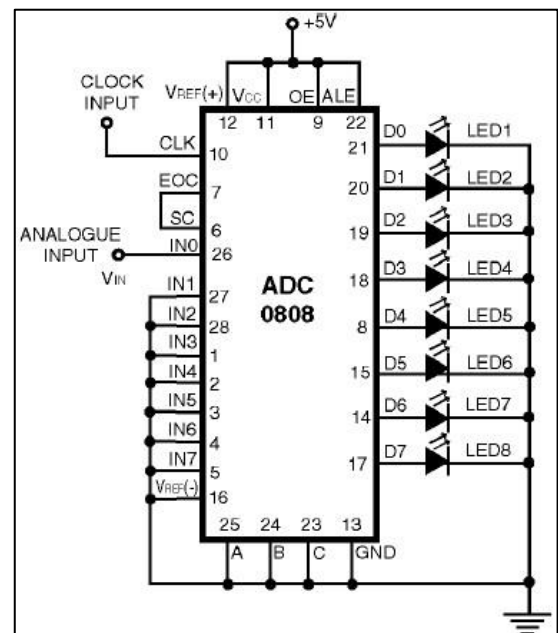
Wait for End of cycle (EOC) pin to become high.

Make Output Enable pin High.

Take Data from the ADC's output

Make Output Enable pin Low.

Stop



8051 microcontrollers

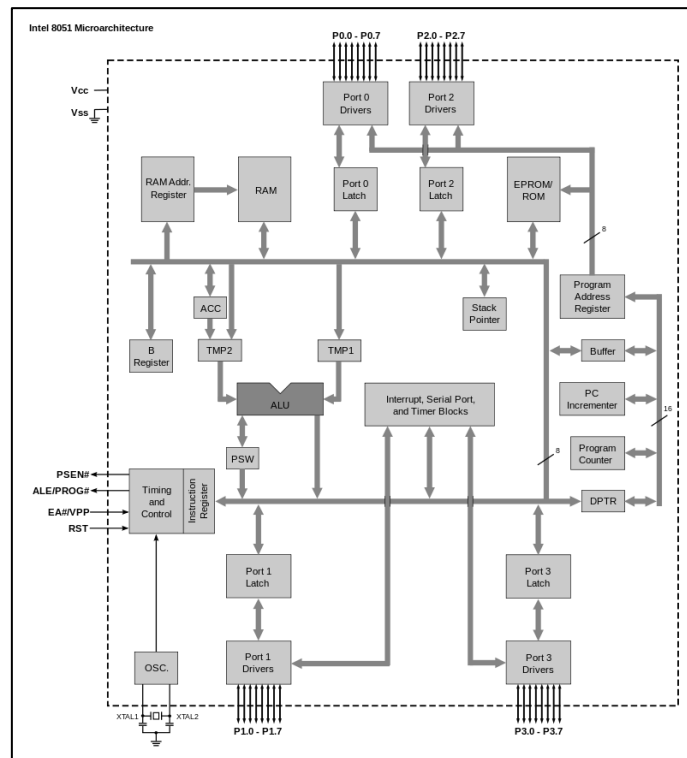
The Intel MCS-51 (commonly termed 8051) is a single chip microcontroller (MCU) series developed by Intel in 1980 for use in embedded systems. The architect of the instruction set of the Intel MCS-51 was John H. Wharton. Intel's original versions were popular in the 1980s and early 1990s and enhanced binary compatible derivatives remain popular today. It is an example of a complex instruction set computer, and has separate memory spaces for program instructions and data.

Intel's original MCS-51 family was developed using N-type metal-oxide-semiconductor (NMOS) technology like its predecessor Intel MCS-48, but later versions, identified by a letter C in their name (e.g., 80C51) used complementary metal-oxide-semiconductor (CMOS) technology and consume less power than their NMOS predecessors. This made them more suitable for battery-powered devices.

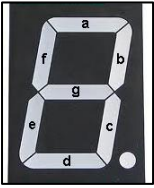
Important features and applications of 8051 microcontrollers

1. 8-bit arithmetic logic unit (ALU) and accumulator, 8-bit registers (one 16-bit register with special move instructions), 8-bit data bus and 2x16-bit address bus, program counter, data pointer, and related 8/11/16-bit operations; hence it is mainly an 8-bit microcontroller
2. Boolean processor with 17 instructions, 1-bit accumulator, 32 registers (4 bit-addressable 8-bit) and up to 144 special 1 bit-addressable RAM variables (18 bit-addressable 8-bit)
3. Multiply, divide and compare instructions
4. 4 fast switchable register banks with 8 registers each (memory mapped)
5. Fast interrupt with optional register bank switching
6. Interrupts and threads with selectable priority
7. 128 or 256 bytes of on-chip RAM (IRAM)
8. Dual 16-bit address bus; it can access 2x2¹⁶ memory locations: 64 KB (65,536 locations) each of ROM (PMEM) and external RAM (XRAM)
9. On-chip ROM (not included on 803x variants)
10. Four 8-bit bi-directional input/output ports, bit addressable
11. UART (serial port)
12. Two 16-bit counter/timers
13. Power saving mode (on some derivatives)

Architecture of 8051 microcontrollers



7-Segment Display



A 7 Segment LED Display is an electronic device that contains an array of 8 individual LEDs.

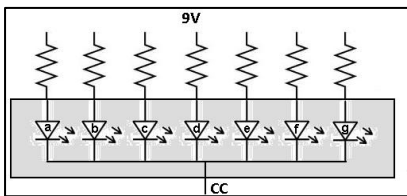
Each of the 8 segments that make up the display can be either be on the on state or the off state. Depending on which LEDs are lit determines the character which is displayed on the LED. The LED can be made to show a wide range of characters, including all numerical digits from 0-9.

The versatility of a 7-segment display lies in the fact that it can be a numerical value indicator. LED displays are used in all types of products, including alarm clocks, scoreboards, and all other signs showing character output.

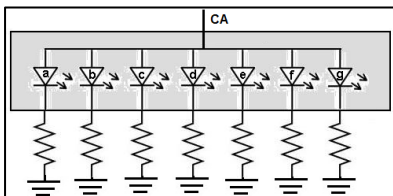
Types of 7 Segment Displays

There are all types of LED Displays, but as far as wiring and connecting purposes are concerned, there are two main types:

- ✓ common cathode LED displays

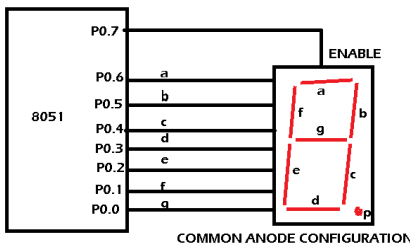


- ✓ common Anode LED Displays



7-Segment Display: Interfacing with 8051

Let's Create a Counter Circuit



Code: (Please follow explanation of class)

```
org 0000h
;pin connections:
;pin 0.6=a
;pin 0.5=b
;pin 0.4=c
;pin 0.3=d
;pin 0.2=e
;pin 0.1=f
;pin 0.0=g
repeat:
mov p0,#10000001b ; displaying 0
acall delay
mov p0,#11001111b ; displaying 1
acall delay
mov p0,#10010010b ; displaying 2
acall delay
mov p0,#10000110b ; displaying 3
acall delay
mov p0,#11001100b ; displaying 4
acall delay
mov p0,#10100100b ; displaying 5
acall delay
mov p0,#10100000b ; displaying 6
acall delay
mov p0,#10001111b ; displaying 7
acall delay
mov p0,#10000000b ; displaying 8
acall delay
mov p0,#10000100b ; displaying 9
acall delay
sjmp repeat
delay:
mov r3,#010h
l3:mov r2,#0ffh
l2:mov r1,#0ffh
l1:djnz r1,l1
djnz r2,l2
djnz r3,l3
ret
```

Courtesy: <http://amitbiswal.blogspot.com.org>

Pinout of 8051:

