# Laboratory Manual

For

# Microprocessor Architecture, Programming and Interfacing

# (IT 403)

B.Tech

SEM IV (IT)

June 2010

Faculty of Technology
Dharamsinh Desai University
Nadiad.
www.ddu.ac.in

# Table of Contents

**EXPERIMENT-1**

**EXPERIMENT-2**

**EXPERIMENT-3**

**EXPERIMENT-4**

**EXPERIMENT-5**

**EXPERIMENT-6**

**EXPERIMENT-7**

**EXPERIMENT-8**

**EXPERIMENT-9**

**EXPERIMENT-10**

**EXPERIMENT-11**

**LABWORK BEYOND CURRICULA**

**EXPERIMENT-12**

**EXPERIMENT-13**

**In all Experimental Laboratory, student will follow following procedural steps :**

**Step 1: Analyzing/Defining the Problem:**

The first step in writing a program is to think very carefully about the problem that you want the program to solve. In other words, ask yourself many times, "what do I really want this program to do ? " It is good idea to write down exactly what you want the program to do and the order in which you want the program to do it. At this point student should not write down program statements, should write the operations in general terms.

**Step 2: Designing the solution/Representing Program Operations:**

The formula or sequence of operations used to solve a programming problem is often called the algorithm of the program. Draw flowchart or use pseudo code to represent program which you want to write to solve your problem. In EXPERIMENT it is better to use flowchart.

**Step 3: Implementing the Solution**

**3.1 Define your Constant, Variables & Pointers for the program.**

**3.2 Finding the right instruction:**

After student prepare flowchart of a program, the next step is to determine the instruction statements required to do each part of the program. Student has to remember the complete instruction set for 8085.Each statement in flow chart could be implemented using one or more instructions.

**Standard program format for assembly language program :**

Student should finally write the program in following format :

| Memory Address | Label | Menmoincs | Label | Hex Code | Comments |
|---|---|---|---|---|---|
| 2100 | start: | MVI A, 55 | | 3E,55H | ;Acc=55 |

**3.3 Constructing the machine codes from 8085 instructions:**

Student will refer the table and manually convert its assembly language program into machine code.

**Step 4: Loading/Running the solution:**

Student will use trainer kit to load his machine code program in RAM and run his program. Student will use debugging tools availed in EXPERIMENT kit to find out the software bugs in the program.

**Step 5: Testing the Solution**

Test the solution by observing the content of various registers and memory addressess.

This is the standard procedure for all EXPERIMENT experiments, hence not required to write separate procedure for all experiments. The list of experiments is attached.

**Example**: Move a block of 8-bit numbers from one place to other place.

**Data(H):** 37, A2, F2, 82, 57, 5A, 7F, DA, E5, 8B, A7, C2, B8, 10, 19, 98

**Step 1:** By analyazing the problem statement, you require following things.

i) Block size (How many number of 8-bit numbers you want to move)
ii) Source Memory Pointer
iii) Destination Memory Pointer

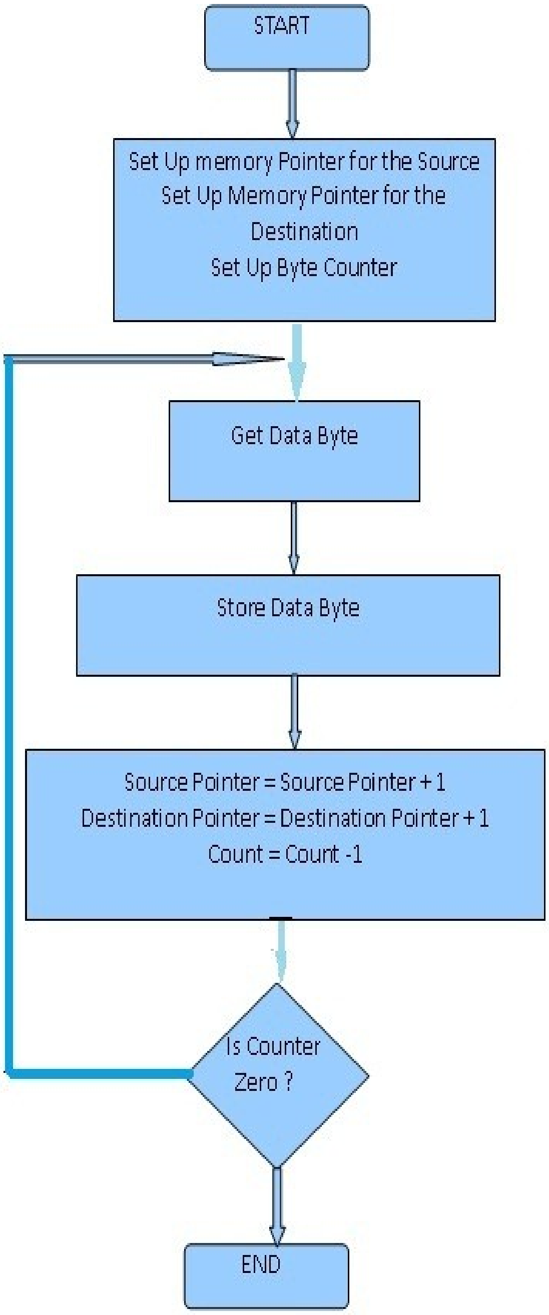
**Step 2:** Desiging the solution/Representing Program Operations.

(Flow Chart for block transfer)

**Step 3: Implementing the Solution**

**3.1 Define your Constant, Variables & Pointers**

**3.2 Finding the right instruction:**


START
Set Up memory Pointer for the Source
Set Up Memory Pointer for the Destination
Set Up Byte Counter
Get Data Byte
Store Data Byte
Source Pointer = Source Pointer + 1
Destination Pointer = Destination Pointer + 1
Count = Count -1
Is Counter Zero ?
END

| | |
|---|---|
| LXI H, XX50 | Set HL pointer  for source memory |
| LXI D, XX70 | Set DE pointer for destination memory |
| MVI B,10 | Set B as a byte counter |
| **NEXT**:<br>MOV A,M | Get  data from source memory |
| STAX D | Store data in destination memory |
| INX    H<br>INX D<br>DCR B | Get ready for next byte |
| JNZ NEXT | Go back to next byte if counter is not equal to 0 |

**Step 3: Implementing the Solution**
**3.3 Constructing the machine codes from 8085 instructions:**

| Memory Address | Label | Mnemonics | Label | Hex code | Comments |
|---|---|---|---|---|---|
| XX00 | START | LXI H, XX50H | | 21, 50, XX | Set HL as Source Pointer |
| XX03 | | LXI D, XX70H | | 11, 70, XX | Set DE as Destination Pointer |
| XX06 | | MVI B, 10H | | 06, 10 | Set B to Count 16 Bytes |
| XX08 | NEXT | MOV A,M | | 7E | Get Data byte from source memory |
| XX09 | | STAX D | | 12 | Store data byte at destination |
| XX0A | | INX H | | 23 | Point HL to next source locatoin |
| XX0B | | INX D | | 13 | Point DE to next destination |
| XX0C | | DCR B | | 05 | One transfer is complete, decrement count |
| XX0D | | JNZ | NEXT | C2, 08, XX | If counter is not 0, go back to transfer next byte |
| XX10 | | HLT | | 76 | End of Program. |

**Step 4: Loading/Running the solution:**

Data: XX50 37
↓ ↓
XX5F 98

**Step 5: Testing the Solution**

A.E (After Execution) XX70 37
↓ ↓
XX7F 98

## EXPERIMENT- 1

**Aim:** a) Introduction to Microprocessor Trainer Kit
b) Addition of two 8-bit numbers.

**Tools / Apparatus:** 8085 microprocessor trainer kit, oscilloscope, voltmeter, tools etc.

### a) Introduction to Microprocessor Trainer kit:

- **Study of HEX Keypad:**

**Study the functions of the following Keys:**

LOAD
SAVE
CODE
STEP
VI
RUN
EXEC
REG

RES
SET
INC
DEC
SPH
SPL
PCH
PCL

0
1
2
3
4
5
6
7

8
9
A
B
C
D
E
F

H
L

**LOAD:** This command is opposite to save command. The contents of audio cassette block is loaded (retrieved back) in the system RAM from a given DS (file name)

**SAVE:** This command is used to save the contents of specified block on to a audio cassette for permanent storage.

**CODE:** When this command key is pressed the address field remains blank, Data field shows a dot indicating that it expects a code.
User is provided with a table of codes, indicating the meaning and pre-requisites of each code. User loads the appropriate code and executes it by pressing EXEC.
The monitor branches to the appropriate sub-routines pointed by the code.

**STEP:** Mere running a program with RUN is done whenever the program development is complete i.e. to *run* a final working program. During the program development stage some sort of aid to execute the part of program at a time and then test its success is required

The STEP command helps you to do the above.
There are two ways of stepping the program.
SINGLE STEPPING: to execute single instruction at a time, using STEP command. The STEP command requires a start address, break address and the no.of times the br should occur.
BREAK POINT: set a software breakpoint *RST1*. This software breakpoint can be done using the *RUN* command. It requires *RST1 (CFH)* to be inserted to a location where you want to break.
The disadvantage of this method is that you have to insert and remove *'CF'* and you have to operate in the *RAM* area only.

**VI:** This key causes immediate recoginition of *RST 7.5* interrupt and control passess to location 003C in the monitor. This location has a jump to location 20CE in user *RAM*. You can put any instruction or jump in20CE to 20D0

- Interrupts must be enabled (EI) instruction.
- RST 7.5 must be unmasked (mask reset by SIM instruction)

**RUN:** This command is used to execute your programs or subroutines from a specified address.
To execute program from 2100:

| | | Address | Data |
|---|---|---|---|
| Press Key | RUN | PPP | XX |
| | 2000 | 2000. | |
| | EXEC | E | |

**EXEC:** Pressing EXEC will place the data field contents into the named register and terminate the command.

**REG:** This command allows you to display and optionally modify the contents of 8085 CPU registers. The various registers are A, B, C, D, E, F, I, H, L, SPH, SPL, PCH, PCL. (H – higher byte, L – lower byte)

**RES:** On RES, the display shows *MP – 85* as a sign on message, indicating that the monitor is ready to accept a command. Pressing any non-command key generates “Err” message. After “-Err” user can immediately give a valid comand.

**SET:** You can use this command to SET the address of a required memory location. A dot in the address field of display indicated that the entry will be displayed in the address field.

**INC:** Pressing INC, first time will shift the dot to the data field of display. Data field will show the contents of the memory location pointed by the address set. You can modify or retain the data field to any value.\

**DEC:** DEC acts as similar to INC, except the address field is decremented, pointing to previous memory locations.

| | |
|---|---|
| **SPH:** | Stack pointer Register (Higher byte) |
| **SPL:** | Stack pointer Register (Lower byte) |
| **PCH:** | Program Counter Register (Higher byte) |
| **PCL:** | Program Counter Register (Lower byte) |
| **0 – F:** | Hex Keypad |
| **H,L:** | Registers H & L |

- **Study of following Devices:**
  1. IC – 8251 (Programmable Synchronous and asynchronous serial data transmiter)
  2. IC – 8253 (Programmable interval Timer /Counter)
  3. IC – 8255 (Programmable Parallel IO Device)
  4. IC – 8279 (Keyboard Display Interface)
  5. IC – 6264 (RAM)
  6. IC – 2764 (EPROM)
  7. ADC
  8. DAC
  9. IC – 8085 (Microprocessor)
- **Study of Memory Address Space:**

  **ROM : 0000 – 1FFF**<br>
  **RAM : 2000 – 3FFF**

  **Memory Address Space Used by Firmware Program: 2000 – 20FF**

  Students should not use this address range for their program or do not modify the content of these locations

  Memofy is expandable to 64K Bytes by interfacing appropriate RAM IC in the empty sockets.
- **Crystal Frequency:**

  Crystal Frequency = 6.144 MHz
- **Study of Onboard Interfaces:**

  The kit has following onboard Interfaces:
  - Parallel I/O using 8255
  - Serial I/O using 8251/8253
  - Keyboard/Display using 8279
  - ADC/DAC using 8255 / Latch -373
- **Study of Interrupts:**

  The Kit uses following interrupts

| | | |
|---|---|---|
| – RST 7.5 | - | VI |
| – RST 5.5 | - | 8279 |
| – NMI | - | Counter 0 output |
| – RST 6.5 | - | Used to implement Single Step |
| – INTR | - | Used to implement Single Step |

**b) Addition of two 8-bit numbers.**

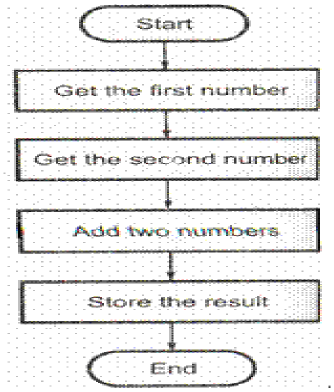
**STANDARD PROCEDURE:**

**1. Analyzing the Problem**:

Addition of two 8-bit numbers stored at memory location 2200 & 2201. And store the result at 2202.

Basic instructions that we are going to use are LDA, MOV, ADD, STA.

**2. Representing ProgramOperations:**



**3. Implementing the Solution**

| Address | Lable | Mnemonics | Machine Code | Remark |
|---|---|---|---|---|
| 2100 | START: | LDA 2200 | 3A,00,22 | Load register A with the content Of memory address 2200. |
| 2103 | | MOV B,A | 47 | Move content of register A to Register B. |
| 2104 | | LDA 2201 | 3A,01,22 | Load register B with the content Of memory address 2201. |
| 2107 | | ADD B | 80 | Add content of register B to Register A. |
| 2108 | | STA 2202 | 32,02,22 | Store the content of register A to memory address 2202. |
| 210B | | RST 1 | CF | |

**4. Loading /Running and the Solution.**

- **Steps to Enter the program in Kit:**
  - Press RESET Key
  - Press SET Key
  - Enter Starting Program Address
  - Press INC key
  - Enter the Hex Code & Press INC Key

- **Steps to run the program:**
  - Press RESET Key
  - Press RUN Key
  - Enter Starting Program Address
  - Press EXEC Key
- **Steps to check registers content after running the code:**
  - Press RESET Key
  - Press REG Key
  - Press the A Key to check accumulator
  - Press INC Key or DEC key to check all other register’s content.

## 5. Testing the Solution

After running the program content of the memory addresses:

**Observation Table:**

| Address | Content |
|---|---|
| 2200 | 1C |
| 2201 | 12 |
| 2202 | |
| Register A | |
| Register B | |
| Flag F | |

**Conclusions**:

Using 8085 microprocessor trainer kit we can add two 8-bit numbers stored at memory location 2200 & 2201. And store the result at 2202.

- **Pin Diagram Of 8085:**

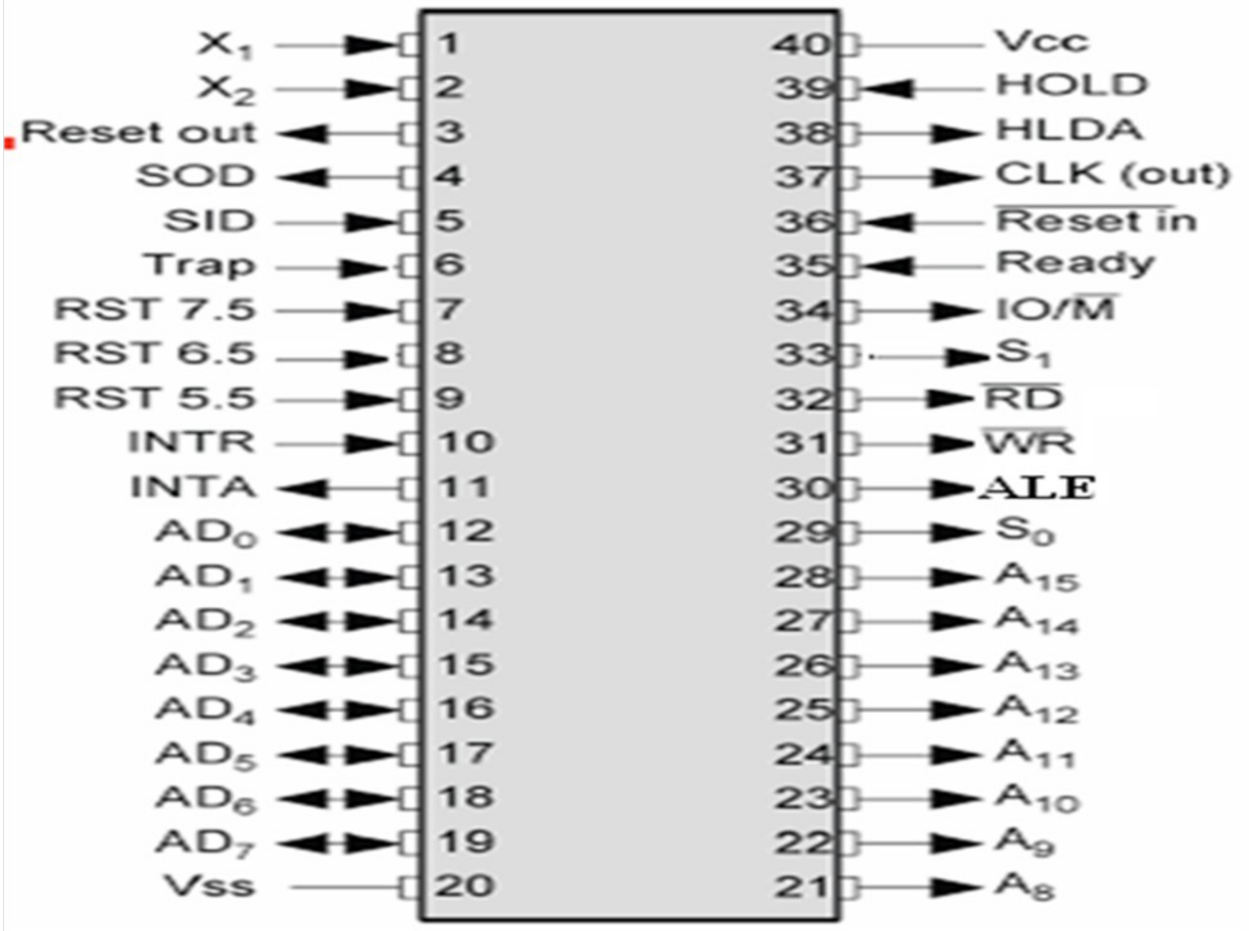


Fig 1.1 Pin Diagram of 8085

## EXPERIMENT-2

**Aim:** Addition of ten 8-bit numbers stored in memory.

**Tools / Apparatus:** 8085 microprocessor trainer kit, oscilloscope, voltmeter, tools etc.

Instructions useful to implement the program.

| | |
|---|---|
| LXI Rp | Register Pair Initialization |
| MOV R,M | Memory Read |
| MOV M,R | Memory Write |
| INX Rp | Pointer Increment |
| ADD M | Addition with Memory Contents |
| MVI R, 8 bit data | Register initialization |
| JNC 16-bit address | Branching Instruction |
| DCR R | Decrement Register Contents |

Study the above instructions from the book. As per the common procedure specified in this manual, perform the experiments.

**Testing:** Test the program with worst case values.

**Assignments:**

1. Subtraction of ten 8-bit numbers
2. Addition of two 16-bit numbers

## EXPERIMENT-3

**Aim:** Find no. of negative elements in a block of data

**Tools / Apparatus:** 8085 microprocessor trainer kit, oscilloscope, voltmeter, tools etc.

**Instruction Used:**

| **INSTUCTION** | | **DESCRIPTION** |
|---|---|---|
| CMP R<br>CMP M | Compare register or memory with accumulator | The contents of the operand (register or memory) are compared with the contents of the accumulator.<br>Both contents are preserved.The result of the comparison is shown by setting the flags of the PSW as follows:<br>if (A) < (reg/mem): carry flag is set<br>if (A) = (reg/mem): zero flag is set<br>if (A) > (reg/mem): carry and zero flags are reset<br>Example: CMP B or CMP M |
| ANI 8-bit data | Logical AND immediate with accumulator | The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.<br>Example: ANI 86H |

**Assignment:**

1. Find Larger of two Numbers stored in memory
2. Find the largest element from a block of data

## EXPERIMENT-4

**Aim:** Observing T-States, Machine cycles and instruction cycle on oscilloscope.

**Tools / Apparatus:** oscilloscope, voltmeter, tools etc.8085 microprocessor trainer kit.

**Description:**

– **T- State:** One subdivision of an operation. A T-state lasts for one clock period.

  - An instruction’s execution length is usually measured in a number of T-states. (clock cycles).

– **Machine Cycle:** The time required to complete one operation of accessing memory, I/O, or acknowledging an external request.

  - This cycle may consist of 3 to 6 T-states.

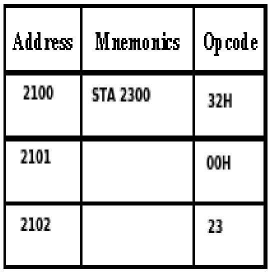
– **Instruction Cycle:** The time required to complete the execution of an instruction.

  - In the 8085, an instruction cycle may consist of 1 to 6 machine cycles.

To Observe Timing diagram for **STA 2300H**.

Function of the STA Instruction:

- STA means Store Accumulator -The content of the accumulator is stored in the specified address (2300).
- The opcode of the STA instruction is said to be 32H. It is fetched from the memory 2100H
- Then the lower order memory address is read (00). - Memory Read Machine Cycle
- Read the higher order memory address (23).- Memory Read Machine Cycle
- The combination of both the addresses is considered and the content from accumulator is written in 2300. - Memory Write Machine Cycle
- Assume the memory address for the instruction and let the content of accumulator is 55H. So, 55H from accumulator is now stored in 2300
- The Instruction Timing Diagram will look like:



| Address | Mnemonics | Opcode |
|---|---|---|
| 2100 | STA 2300 | 32H |
| 2101 | | 00H |
| 2102 | | 23 |

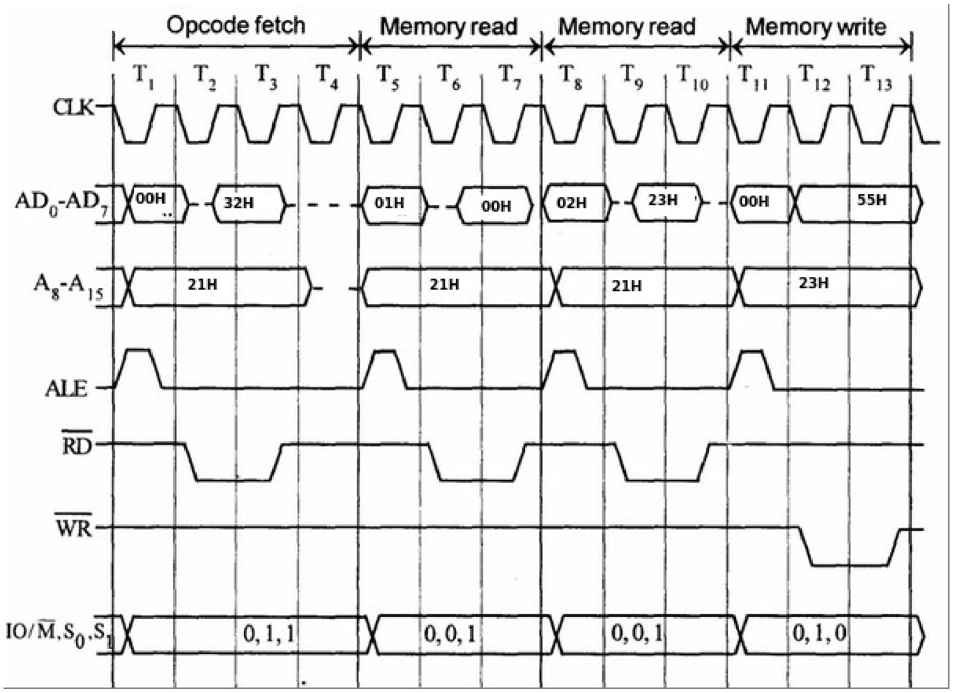


Fig 4.1 Timing diagram for **STA 2300H**

**Importance of the Experiment:**

System bus content changes dynamically and hence difficult to observe on oscilloscope. One can see steady state wave form if the oscilloscope is trigerred with a signal which is

1. Repetitive in nature
2. Occuring at equal intervals
3. Occuring once only in a loop

To observe the bus activity for the instruction one has to write a program which will give the trigger signal who will satisfy the above requirements.

Now if you would like to observe the waveform on oscilloscope when this instruction is executed, you have to trigger the oscilloscope with proper signal. Trigger signal will be generated from the following program.

**Program:**

```
2100        MVI A, 55H
2102        STA 2300
2105        JMP    2100
```

One can observe that WR will be generated only once in a loop, hence can be used as a trigger pulse & reference to this trigger pulse all T-states can be observed steadily on oscilloscope.

**Observation:** Draw the timing diagram for this loop.

## EXPERIMENT-5

**Aim:** Sorting of numbers - Ascending/Descending

**Tools / Apparatus**: 8085 microprocessor trainer kit, oscilloscope, voltmeter, tools etc.

**Instruction Used:**

| **INSTUCTION** | | **DESCRIPTION** |
|---|---|---|
| LXI Reg. pair, 16-bit data | Load register pair immediate | The instruction loads 16-bit data in the register pair designated in the operand.<br>Example: LXI H, 2034H or LXI H, XYZ |
| DCX R | Decrement register pair by 1 | The contents of the designated register pair are decremented by 1 and the result is stored in the same place.<br>Example: DCX H |
| INX R | Increment register pair by 1 | The contents of the designated register pair are incremented by 1 and the result is stored in the same place.<br>Example: INX H |
| CMP R<br>CMP M | Compare register or memory with accumulator | The contents of the operand (register or memory) are compared with the contents of the accumulator.<br>Both contents are preserved.The result of the comparison is shown by setting the flags of the PSW as follows:<br>if (A) < (reg/mem): carry flag is set<br>if (A) = (reg/mem): zero flag is set<br>if (A) > (reg/mem): carry and zero flags are reset<br>Example: CMP B or CMP M |
| XRA R<br>XRA M | Exclusive OR register or memory with accumulator | The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.<br>Example: XRA B or XRA M |

## EXPERIMENT-6

**Aim:** Convert Binary number to BCD

**Tools / Apparatus:** 8085 microprocessor trainer kit.

**Binary to BCD**

**Instruction Used:**

| INSTUCTION | | DESCRIPTION |
|---|---|---|
| CALL 16-bit address | Unconditional subroutine call | The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Before the transfer, the address of the next instruction after CALL (the contents of the program counter) is pushed onto the stack.<br>Example: CALL 2034H or CALL XYZ |
| ANI 8-bit data | Logical AND immediate with accumulator | The contents of the accumulator are logically ANDed with the 8-bit data (operand) and the result is placed in the accumulator. S, Z, P are modified to reflect the result of the operation. CY is reset. AC is set.<br>Example: ANI 86H |
| STAX Reg. pair | Store accumulator indirect | The contents of the accumulator are copied into the memory location specified by the contents of the operand (register pair). The contents of the accumulator are not altered.<br>Example: STAX B |
| PUSH Reg. pair | Push register pair onto stack | The contents of the register pair designated in the operand are copied onto the stack in the following sequence. The stack pointer register is decremented and the contents of the high order register (B, D, H, A) are copied into that location. The stack pointer register is decremented again and the contents of the low-order register (C, E, L, flags) are copied to that location.<br>Example: PUSH B or PUSH A |
| RRC | Rotate accumulator right | Each binary bit of the accumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. CY |

| | | |
|---|---|---|
| | | is modified according to bit D0. S, Z, P, AC are not affected. Example: RRC |
| POP Reg. pair | Pop off stack to register pair | The contents of the memory location pointed out by the stack pointer register are copied to the low-order register (C, E, L, status flags) of the operand. The stack pointer is incremented by 1 and the contents of that memory location are copied to the high-order register (B, D, H, A) of the operand. The stack pointer register is again incremented by 1.<br>Example: POP H or POP A |
| RET | Return from subroutine unconditionally | The program sequence is transferred from the subroutine to the calling program. The two bytes from the top of the stack are copied into the program counter, and program execution begins at the new address.<br>Example: RET |
| XRA R<br>XRA M | Exclusive OR register or memory with accumulator | The contents of the accumulator are Exclusive ORed with the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset.<br>Example: XRA B or XRA M |

- **Assignment:**

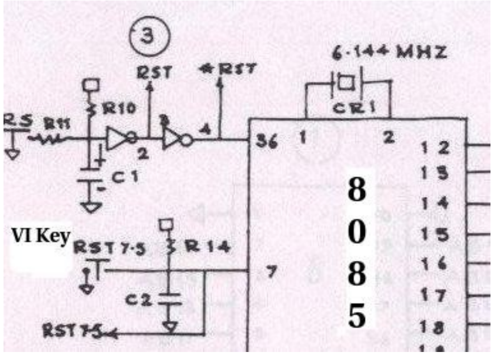
i) BCD to Binary
ii) Decimal to ASCII
iii) ASCII to Decimal

# EXPERIMENT-7

**Aim:** Write a program to develop an 8-bit Hex Counter counter and display on data field as foreground program. Write an Interrupt subroutine as a background program which wil increment a memory location every time a RST 7.5 interrupt arrives.

**Tools / Apparatus:** 8085 microprocessor trainer kit, oscilloscope, voltmeter, tools etc.

## Implementation of RST 7.5 on MP-85 Kit:



Whenever a 'VI' key is pressed a low to high going pulse will be generated on RST 7.5 ping generating +ve edge on RST 7.5 pin. SIM instruation is used to unmask RST 7.5 and also to clear RST 7.5 Flip Flop.

**Fig: RST 7.5 Implementation.**

When RST 7.5 arrives, the auto vector address is 003C and this location is a part of monitor program (non-volatile memory). At 003C, jumpt to 20CE is written

**003C JMP 20CE ;20CE is a RAM location**

So at 20CE write the following Instruction:

**20CE JMP 2300 (RAM Address Space, for subroutine program)**

So, now user can write his subroutine at 2300H.

**2300: Subroutine for incrementing a content of a memory location.**

**2500: Memory location whose content will be incremented in ISR.**

**Follow the following steps:**

**Step 1:** Enable Interrupt Process (EI)

**Step 2:** Set interrupt Mask (SIM), for allowing RST 7.5 Interrupt.

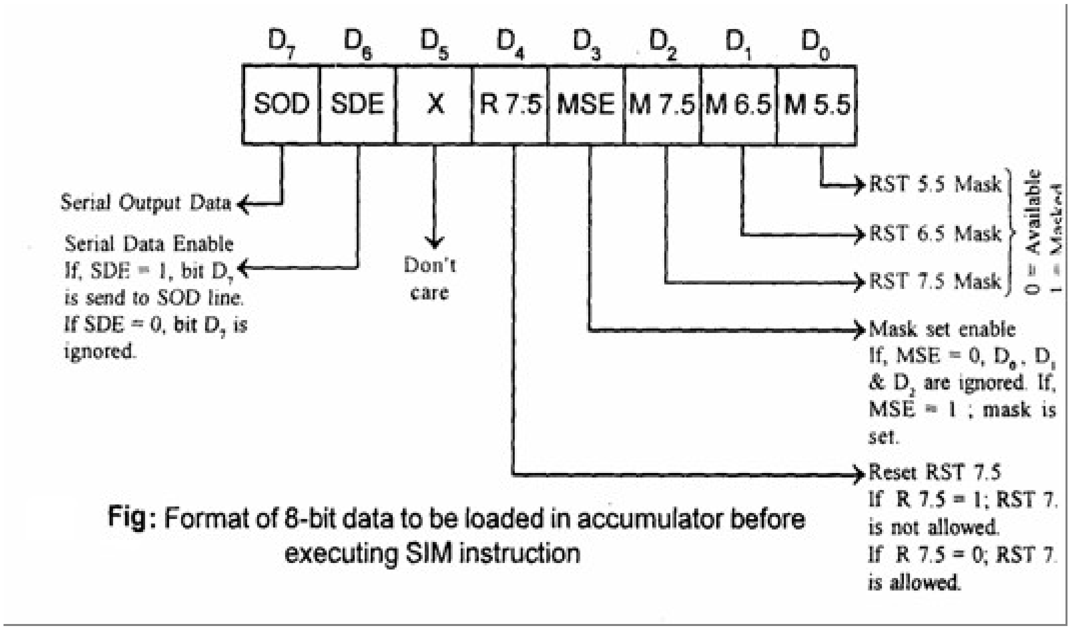

$D_7$
$D_6$
$D_5$
$D_4$
$D_3$
$D_2$
$D_1$
$D_0$
SOD
SDE
X
R 7.5
MSE
M 7.5
M 6.5
M 5.5
Serial Output Data
Serial Data Enable If, SDE = 1, bit $D_7$ is send to SOD line. If SDE = 0, bit $D_7$ is ignored.
Don't care
RST 5.5 Mask
RST 6.5 Mask
RST 7.5 Mask
0 = Available
1 = Masked
Mask set enable If, MSE = 0, $D_0$, $D_1$ & $D_2$ are ignored. If, MSE = 1 ; mask is set.
Reset RST 7.5 If R 7.5 = 1; RST 7. is not allowed. If R 7.5 = 0; RST 7. is allowed.

**Fig:** Format of 8-bit data to be loaded in accumulator before executing SIM instruction

**Step 3:** Initialize Hex count =00, Initialize Memory Pointer =2500

**Step 4:** Write Instructions to Display counter on data field using Routine 036E

**Step 5:** Increment counter and repeat step 4.

This is an end less loop and displa 8-bit hex counter continuously on data field.

**Step 6:** Write Subroutine program at address 2300 to count no.of occurances of 'VI' key. Store the count value at memory location 2500.

**Action:** Now every time 'VI' key is pressed, RST 7.5 will be generated.

**Observation:** Press 4 times 'VI' key and reset the kit. You are supposed to get number 4 in memory location 2500H

**Analyasing the observation & remedial action:** Verify the observation, analyase & modify the ISR to get the desired result.

**SUBROUTINE:** 036E, displays accumulator content on data filed.

**Function** **:** Output Char. To data field

**Parameter A** **:** 8 bit Word

**B** **:** 0 No Dot

**:** 1 Dot

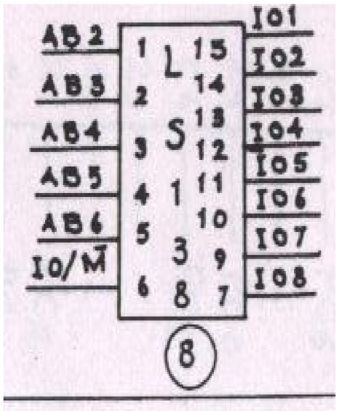
**Destroys** **:** Everything.

## EXPERIMENT- 8

**Aim:** Write a program to generate square wave of 1 Khz on PC4 on 8255#2 using BSR Mode.

**Tools / Apparatus: 8085 microprocessor trainer kit, oscilloscope, tools etc.**

**The following decoder circuit will give you port addressess for 8255 and other given IC's. This circuit will be used for finding the port addressess of IC's used in Later experiments.**



- **– 8255 -1 (U22) Port Address 00-03 // 80-84**
- **– 8279 KB/ Display Controller Port Address 04-05/06-07 // 84-85/86-87**
- **– 8255 -2 (U13) Port Address 08-0B // 88-8B**
- **– 8253 Timer/Counter Port Address 0C-0F // 8C-8F**
- **– 8251 UART Port Address 10-11/12-13 // 90-91/92-93**
- **– 8255 -3 (U23 – ADC Option) Port Address 14-17 // 94-97**
- **– DAC Option (L4LS 373 U34) Port Address 18 // 98**
- **– NC (Not Connected)**

**Fig: Decoder output connected to Peripheral Devices with Port Addressess**

| ADDRESS BUS | | | | | | IO LINE | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **AB7** | **AB6** | **AB5** | **AB4** | **AB3** | **AB2** | **IO1** | **IO2** | **IO3** | **IO4** | **IO5** | **IO6** | **IO7** | **IO8** | **ADDRESS** | **Device** |
| x | 0 | 0 | **0** | **0** | **0** | **0** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 00-03/80-84 | 8255 -1 |
| x | 0 | 0 | **0** | **0** | **1** | 1 | **0** | 1 | 1 | 1 | 1 | 1 | 1 | 04-05/06-07<br>84-85/86-87 | 8279 |
| x | 0 | 0 | **0** | **1** | **0** | 1 | 1 | **0** | 1 | 1 | 1 | 1 | 1 | 08-0B/88-8B | 8255 -2 |
| x | 0 | 0 | **0** | **1** | **1** | 1 | 1 | 1 | **0** | 1 | 1 | 1 | 1 | 0C-0F/8C-8F | 8253 |
| x | 0 | 0 | **1** | **0** | **0** | 1 | 1 | 1 | 1 | **0** | 1 | 1 | 1 | 10-11/12-13<br>90-91/92-93 | 8251 |
| x | 0 | 0 | **1** | **0** | **1** | 1 | 1 | 1 | 1 | 1 | **0** | 1 | 1 | 14-17/94-97 | 8255 -3 |
| x | 0 | 0 | **1** | **1** | **0** | 1 | 1 | 1 | 1 | 1 | 1 | **0** | 1 | 18/98 | DAC |
| x | 0 | 0 | **1** | **1** | **1** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | **0** | NC | NC |

**Fig: Table for I/O Select Lines**

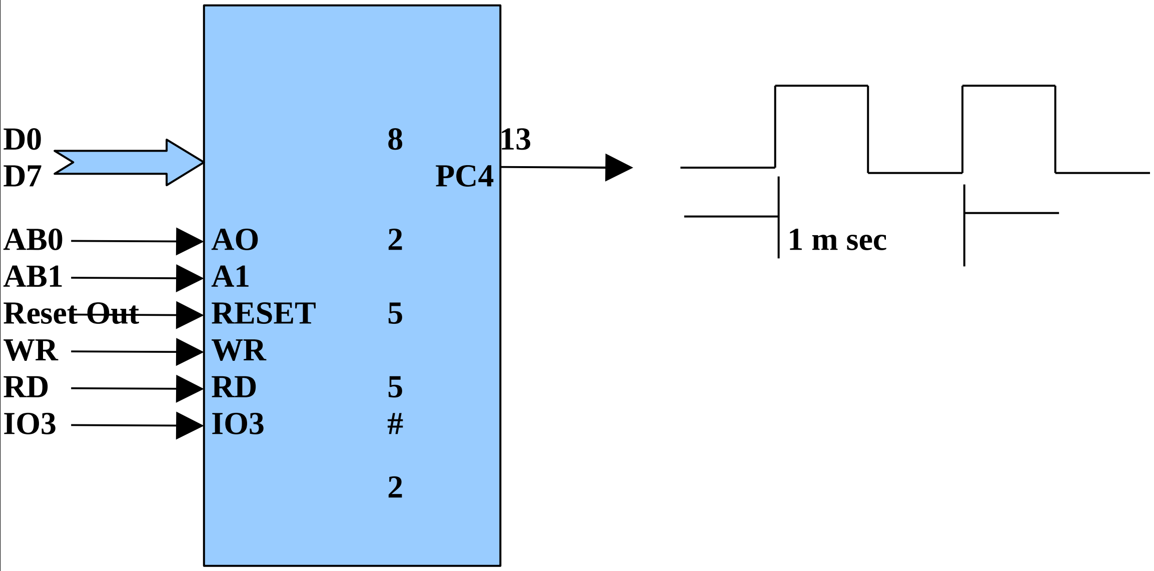


**Fig: 8255 # -2 Interfacing**

| *** Port Addressess** | **8255(A0, A1)** | **Address** | **Selection** |
|---|---|---|---|
| | | **08** | **Port A** |
| | | **09** | **Port B** |
| | | **0A** | **Port C** |
| | | **0B** | **Control Register** |

*** The Frequency of Processor Clock: 6.144/2 = 3.072MHz**

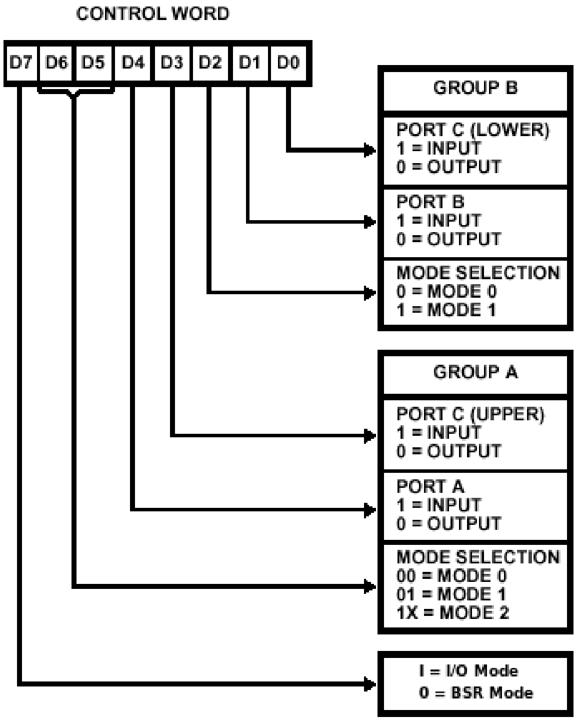


**Fig: Contol Word Format for 8255**

**Follow the following procedure.**

**Step 1:** Initialize 8255 to use Port C bits as an output port.

**Step 2:** Set PC4 bit

**Step 3:** Call Delay of 0.5 msec

**Step 4:** Reset PC4

**Step 5:** Call Delay of 0.5 msec

**Step 4:** Observe the o/p on CRO

**Observation:** Observe the waveform on PC4 pin and verify that it is 1KHz.

## EXPERIMENT-9

**Aim:** To Serially Transmit and receive data between two kits using 8251 (USART)

**Tools / Apparatus:** 8085 microprocessor trainer kits, RS -232 Serial Cable (Buid it using D-type male connectors and wires)

**Specifications given:**

| | |
|---|---|
| **Crystal Frequency of 8085** | **:** 6.144 Mhz |
| **Baud** | **:** 1200 |
| **Data bits** | **:** 8 |
| **Parity** | **:** No |
| **Stop bits** | **:** 2 |
| **Mode** | **:** Asynchronous X 16 (Baud Factor) |

**Note: Counter 1 of 8253 is used to generate necessary transmit/receive clock for 8251. The processor clock out is divided by 2 & given as clock input to counter 1 of 8253**

The 8251 IC is interfaced using 8253.

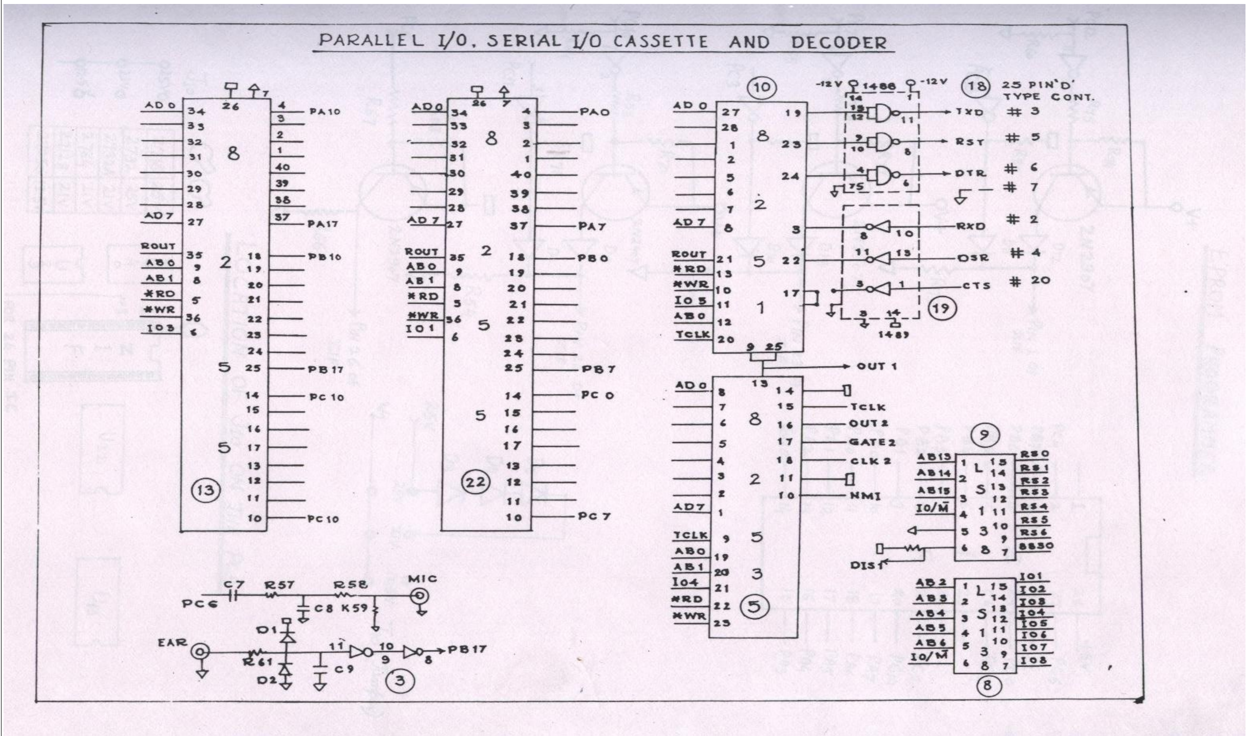


**Fig: 8251 Interfacing**

**From the diagram we see that the TxC and RxC are connected through OUT1 of 8253, hence we need to iniatialize 8253 first and then 8251**

**Lets us find the count required to be loaded in counter 1 of 8253**

**Kit Frequency** = Crystal Freq /2
= 6.144 /2 X $10^6$
= 3.072 X $10^6$

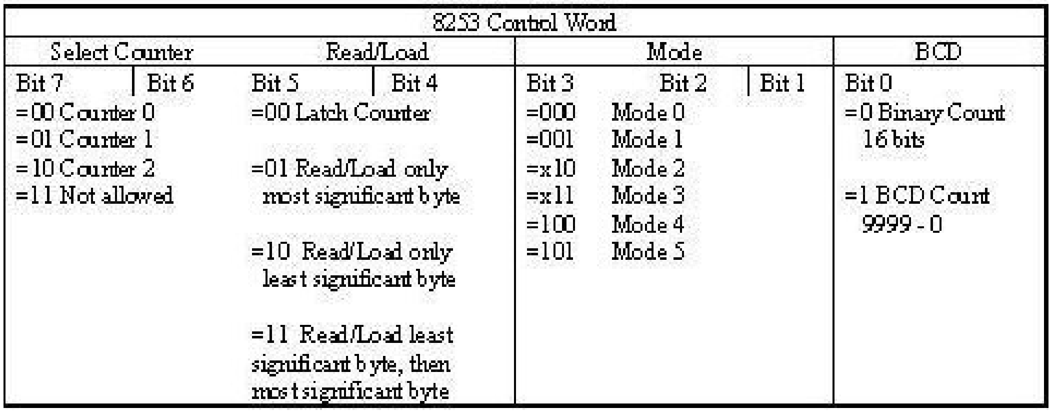
**Therefore CLK1** = 3.072 /2 X $10^6$
= 1.536 Mhz

**We need to find the TxC frequency. As Baud and Baud Factor are given**

**TxC** = 1200 X 16
**Therefore Count** = CLK1 / TxC
= 1536000 / 1200 x 16
= 80
= 50H

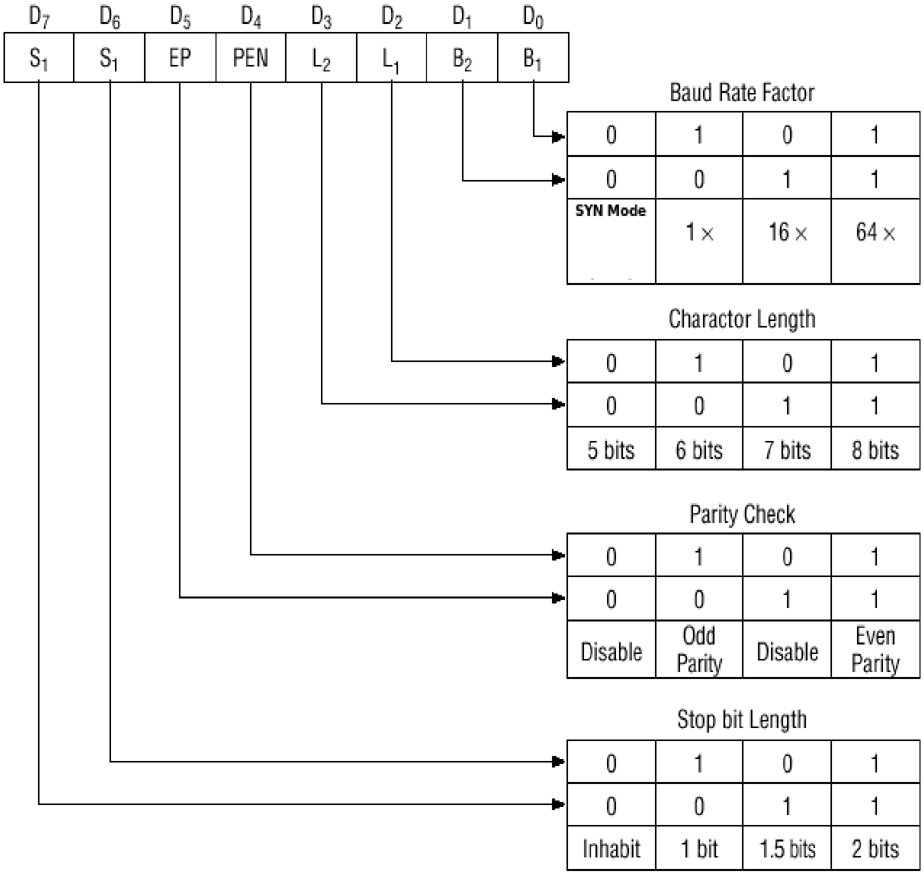
| **Port Address:8253(A0, A1)** | **Address** | **Selection** |
|---|---|---|
| | **0C** | **Counter 0** |
| | **0D** | **Counter 1** |
| | **0E** | **Counter 2** |
| | **0F** | **Control Register** |

| **Port Address:8251 (A0)** | **Address** | **Selection** |
|---|---|---|
| **C/D'** | **10** | **Data** |
| | **11** | **Control** |



| 8253 Control Word | | | | | | | |
|---|---|---|---|---|---|---|---|
| Select Counter | | Read/Load | | Mode | | | BCD |
| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| =00 Counter 0 | | =00 Latch Counter | | =000 | Mode 0 | | =0 Binary Count 16 bits |
| =01 Counter 1 | | | | =001 | Mode 1 | | |
| =10 Counter 2 | | =01 Read/Load only most significant byte | | =x10 | Mode 2 | | |
| =11 Not allowed | | | | =x11 | Mode 3 | | =1 BCD Count 9999 - 0 |
| | | | | =100 | Mode 4 | | |
| | | =10 Read/Load only least significant byte | | =101 | Mode 5 | | |
| | | =11 Read/Load least significant byte, then most significant byte | | | | | |

**Fig: 8253 Control Word**



| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| $S_1$ | $S_1$ | EP | PEN | $L_2$ | $L_1$ | $B_2$ | $B_1$ |

Baud Rate Factor

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| SYN Mode | 1 × | 16 × | 64 × |

Charactor Length

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 5 bits | 6 bits | 7 bits | 8 bits |

Parity Check

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| Disable | Odd Parity | Disable | Even Parity |

Stop bit Length

| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| Inhabit | 1 bit | 1.5 bits | 2 bits |

**Fig: Mode Word Format (8251)**

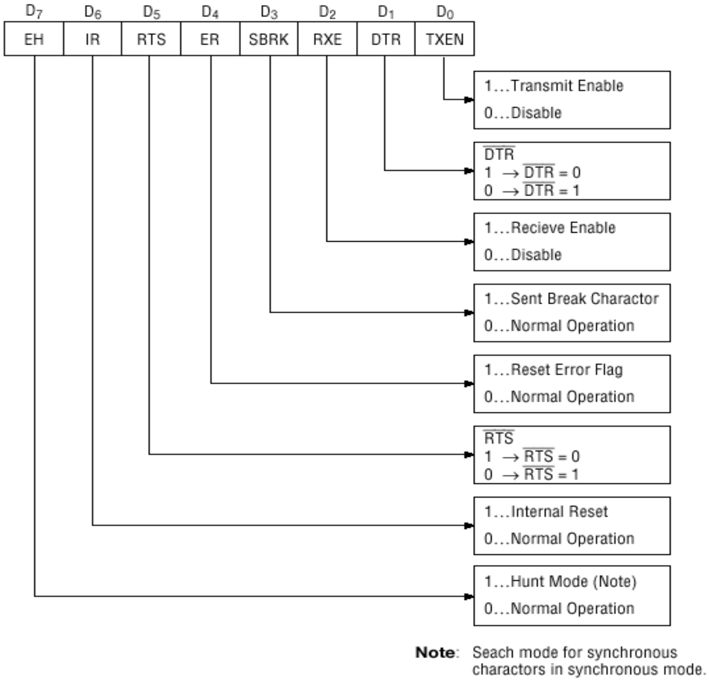


**Fig: Command Word Format (8251)**

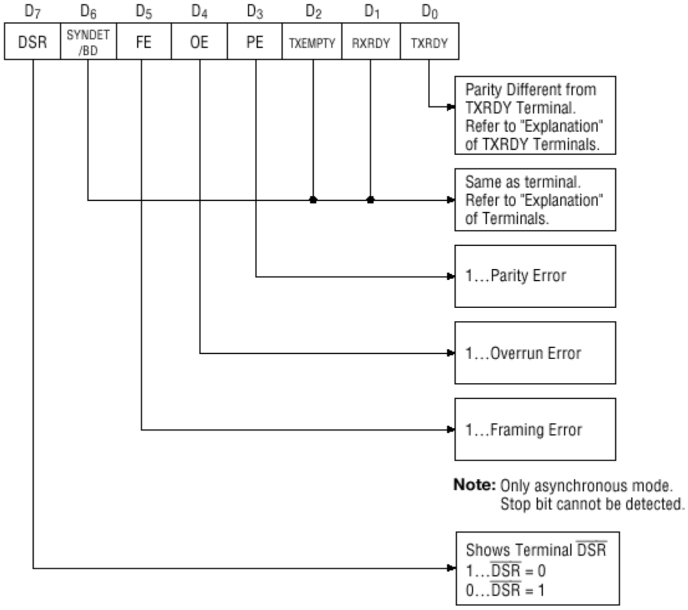
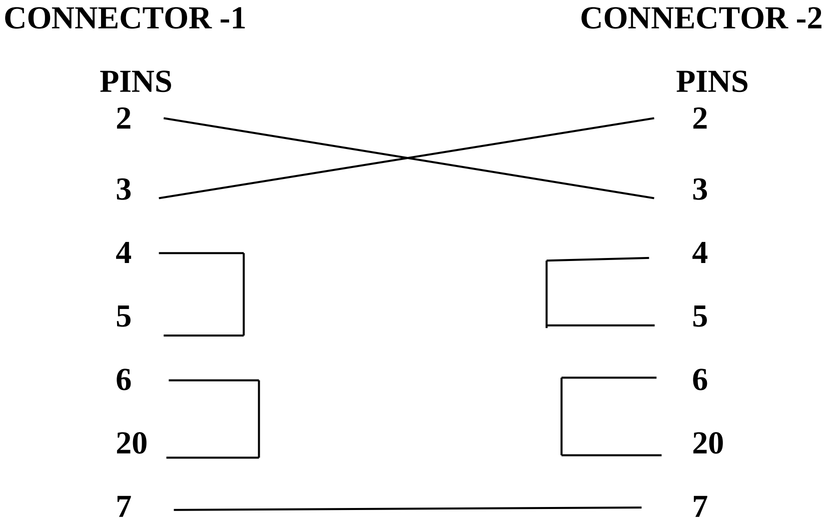

$D_7$ $D_6$ $D_5$ $D_4$ $D_3$ $D_2$ $D_1$ $D_0$
DSR
SYNDET /BD
FE
OE
PE
TXEMPTY
RXRDY
TXRDY
Parity Different from TXRDY Terminal. Refer to "Explanation" of TXRDY Terminals.
Same as terminal. Refer to "Explanation" of Terminals.
1…Parity Error
1…Overrun Error
1…Framing Error
Note: Only asynchronous mode. Stop bit cannot be detected.
Shows Terminal $\overline{DSR}$
1…$\overline{DSR}$ = 0
0…$\overline{DSR}$ = 1

**Fig: Status Word Format (8251)**

**You also require RS-232 serial cable. Use following connectors to build RS-232 Cable.**

**Fig:** D-type 25 pin Male Connector.

**Take two such connectors and build the cable as per following connections.**



**Follow the following Procedure to complete the experiment.**

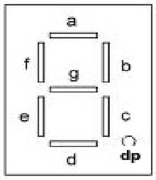
**Step 1:** Connect the two kits (One for Transmitter and other for Receiver) using RS-232 Cable build by you
**Step 2:** Initialize the Timer IC (8253) to generate the required baud.
**Step 3:** Initialize 8251 For the given specifications. (Both For Transmitter and Receiver Side)
**Step 4:** Enable Transmitter on Transmitter Side and Receiver on Receiver Side
**Step 5:** Check Transmitter Ready and Receiver Ready Status.
**Step 6:** Transmit and Receive the Character.

**Assignment:** 1. Repeat the same for String of Data Byte
2. Transmit & Receive Data Simultaneously.

# EXPERIMENT-10

**Aim:** i) To Display moving string on 7-segment display (8279)
ii) To Read Keyboard from Hex Keyboard using 8279

**Tools / Apparatus:** 8085 microprocessor trainer kit, oscilloscope, voltmeter, tools etc.



Seven-Segment Display

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| A3<br>d | A2<br>c | A1<br>b | A0<br>a | B3<br>Dp | B2<br>g | B1<br>f | B0<br>e |

**Fig: Bit Pattern to be used for character formation**

**Note:** A segment will glow if the bit position is 0 for that segment.

**e.g. If you want to display A (capital-a) then code to be written in Accumulatr will be as follows. As you need not to display segment d and dp, 1 is placed for that.**

**Code for “A” = 88**

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

**Fig: Code to display A**

| **Port Addressess:** | **8279(A0)** | **Address** | **Selection** |
|---|---|---|---|
| | | **05** | **Control Word** |
| | | **04** | **Data** |

*** Keyboard / Display Mode**

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | D | D | K | K | K |

**Fig: Keyboard / Display Mode Set**

| DD | |
|---|---|
| 0 0 | 8 8-bit character display – Left Entry |
| 0 1 | 16 8-bti character display – Left Entry |
| 1 0 | 8 8-bit character display – Right Entry |
| 1 1 | 16 8-bit character display – Right Entry |

| KKK | |
|---|---|
| 0 0 0 | Encoded Scan Keyboard – 2 Key Lockout |
| 0 0 1 | Decoded Scan Keyboard – 2 Key Lockout |
| 0 1 0 | Encoded Scan Keyboard – 2 Key Rollever |
| 0 1 1 | Decoded Scan Keyboard – 2 Key Rollever |
| 1 0 0 | Encoded Scan Sensor Matrix |
| 1 0 1 | Decoded Scan Sensor Matrix |
| 1 1 0 | Strobed Input, Encoded Display Scan |
| 1 1 1 | Strobed Input, Decoded Display Scan |

*** Clear Display RAM**

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | $C_D$ | $C_D$ | $C_D$ | $C_F$ | $C_A$ |

**Fig: Clear Display RAM**

| $C_D$ | $C_D$ | $C_D$ | |
|---|---|---|---|
| | 0 | X | - All Zeros ( X = Don't Care) |
| | 1 | 0 | - AB = Hex 20 (0010 0000) |
| | 1 | 1 | - All ones |
| 1 | - | - | - Enable clear Display Ram When = 1 |
| | | | - If $C_F$ = 1, FIFO Status is cleared |
| | | | - $C_A$ , Clear all bit, has combined effect of $C_D$ & $C_F$ |

*** Write Display RAM**

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | AI | A | A | A | A |

**Fig: Write Display RAM**

AAAA – Select one of the 16 rows
if AI =1, this row address will be incremented after each following read or write to the display RAM.

Find the code for the string you want to generate using Bit-pattern table

Follow the following Procedure

**Step 1:** Disable Interrupt
**Step 2:** Initialize 8279 in 8 8-bit character right entry.
**Step 3:** Find the control and data address for 8279 KB Controller (Decoder Dia.)
**Step 4:** Write Display RAM command
**Step 5:** Display the Character  you want to display, first by obtaining the code of it.
**Step 6:** Repeat step 5 until all charaters are displayed
**Step 7:** Repeat all steps to get scrolling display

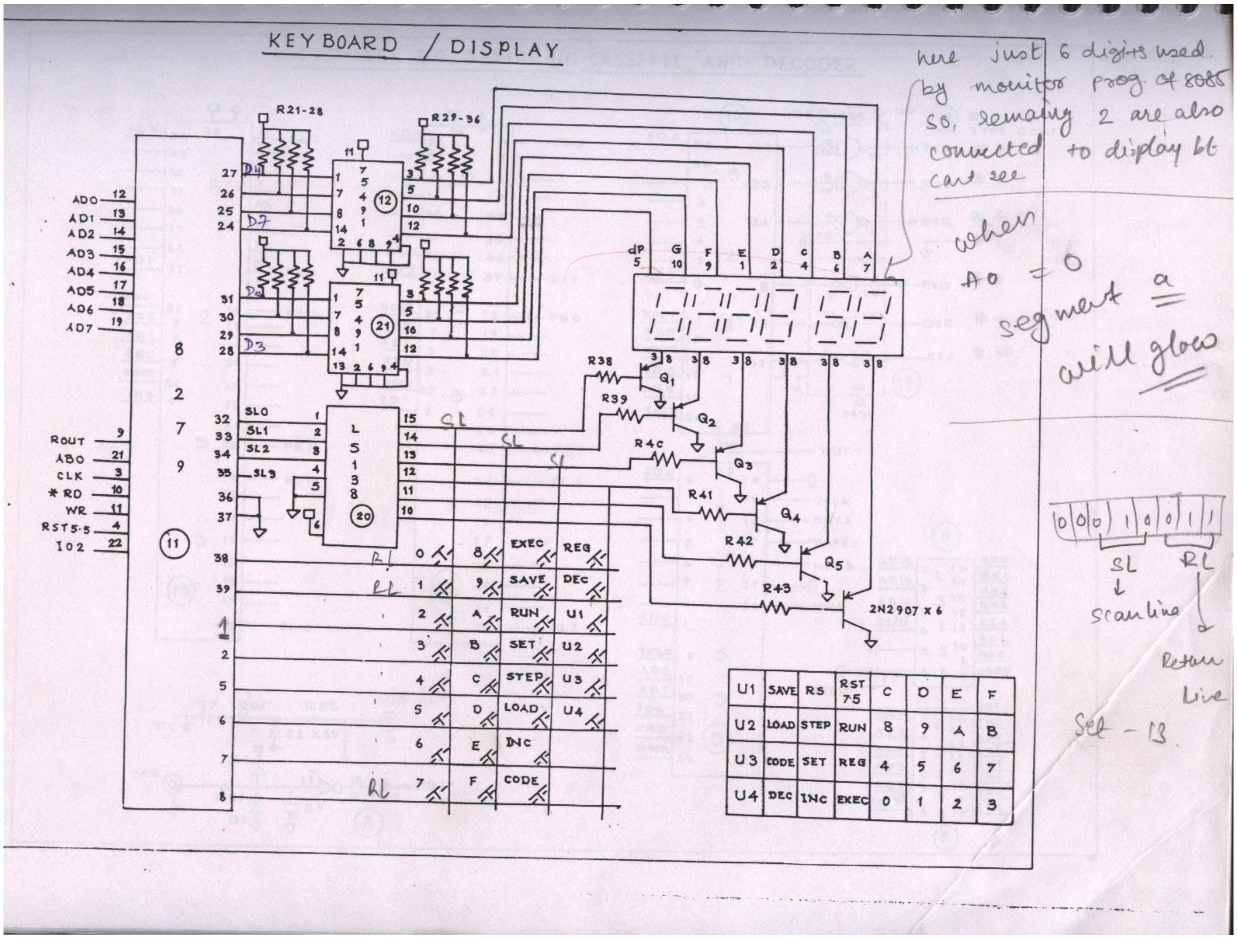


**Fig: KB/DISPLAY INTERFACE DIAGRAM**

### ii) To Read Keyboard from Hex Keyboard using 8279

### * Read FIFO/Sensor RAM

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | AI | X | A | A | A |

**Fig: Read FIFO/Sensor RAM Mode**

**x = don't care**

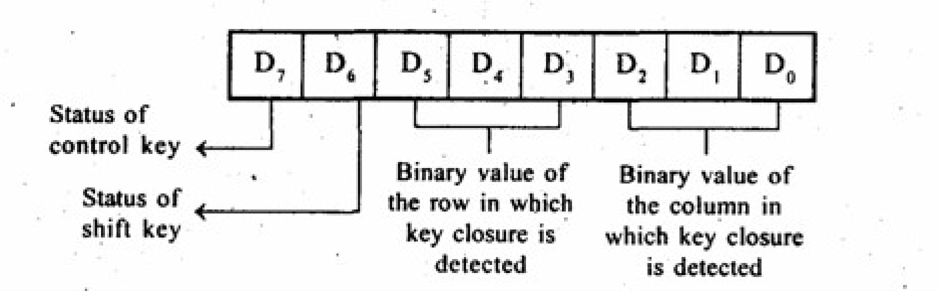
### * FIFO Status Word

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| $D_U$ | S/E | O | U | F | N | N | N |

**Fig: FIFO Status Word**

| | |
|---|---|
| N N N | - Numbers of Characters in FIFO |
| F | - FIFO Full |
| U | - Error – Underrum |
| O | - Error – Overrun |
| S/E | - Sensor Closure/Error Flag for Multiple Closures |
| $D_U$ | - Display Unavailable |

### * Scanned Keyboard Data Format


D7
D6
D5
D4
D3
D2
D1
D0
Status of control key
Status of shift key
Binary value of the row in which key closure is detected
Binary value of the column in which key closure is detected

D5, D4, D3 – Scan Lines
D2, D1, D0 – Return Lines

### Follow the following Procedure

**Step 1:** Disable Interrupt (Monitor Program has enabled RST 5.5 because 8279 is connected to RST 5.5 interrupt. So we can not use 8279 in interrupt mode, rather we have to use 8279 in polled mode.)
**Step 2:** Initialize 8279 for Encoded Scan Keyboard – 2 Key Lockout
**Step 3:** Write Read /FIFO Sensor RAM control Word
**Step 4:** Status Check if a key is pressed (Use FIFO Status Word)
**Step 5:** Read the Key Pressed.
**Step 6:** Verify the keycode with code generated from circuit diagram.

e.g. if key **set** is pressed, you will get the content of Acc as following

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |

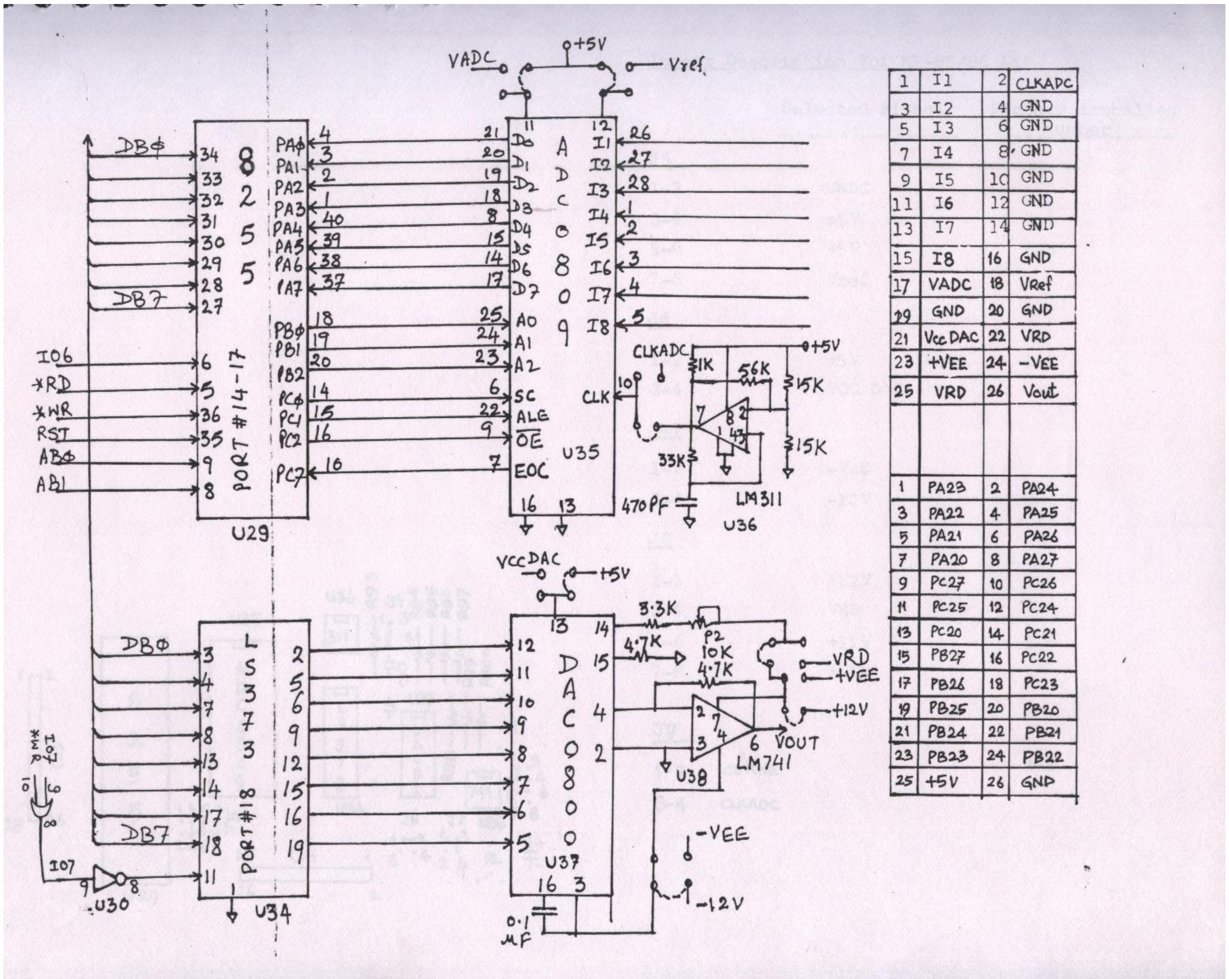
**= 13 H**

**Assignment:** Combine keyboard/display. Display the scan code on display field continuously.

## EXPERIMENT-11

**Aim:** i) Reading analog voltage through ADC 0809
ii) Generating different waveforms on DAC800 output

**Tools / Apparatus:** 8085 microprocessor trainer kit, oscilloscope, voltmeter, tools etc.



| | | | |
|---|---|---|---|
| 1 | I1 | 2 | CLKADC |
| 3 | I2 | 4 | GND |
| 5 | I3 | 6 | GND |
| 7 | I4 | 8 | GND |
| 9 | I5 | 10 | GND |
| 11 | I6 | 12 | GND |
| 13 | I7 | 14 | GND |
| 15 | I8 | 16 | GND |
| 17 | VADC | 18 | VRef |
| 19 | GND | 20 | GND |
| 21 | Vcc DAC | 22 | VRD |
| 23 | +VEE | 24 | -VEE |
| 25 | VRD | 26 | Vout |

| | | | |
|---|---|---|---|
| 1 | PA23 | 2 | PA24 |
| 3 | PA22 | 4 | PA25 |
| 5 | PA21 | 6 | PA26 |
| 7 | PA20 | 8 | PA27 |
| 9 | PC27 | 10 | PC26 |
| 11 | PC25 | 12 | PC24 |
| 13 | PC20 | 14 | PC21 |
| 15 | PB27 | 16 | PC22 |
| 17 | PB26 | 18 | PC23 |
| 19 | PB25 | 20 | PB20 |
| 21 | PB24 | 22 | PB21 |
| 23 | PB23 | 24 | PB22 |
| 25 | +5V | 26 | GND |

**Fig: ADC/DAC Interfacing Circuit Diagram with 8085**

### i) Reading analog voltage through ADC 0809

- ADC is interfaced to 8085 using 8255 # 3.
- The I1 to I8 Lines at the input of ADC are analog input channels.
- Port A of 8255 is used to read the converted digital data
- Port B is connected to A0, A1, A2 of ADC that is used for Analog channel selection.
- Port C Pins are connected to Control Signals of ADC
- Hence 8255 # 3 should be configured as

| | |
|---|---|
| **Mode** | **:** 0 |
| **Port A** | **:** Input |
| **Port B** | **:** Output |
| **Port $C_L$** | **:** Output |
| **Port $C_U$** | **:** Input |

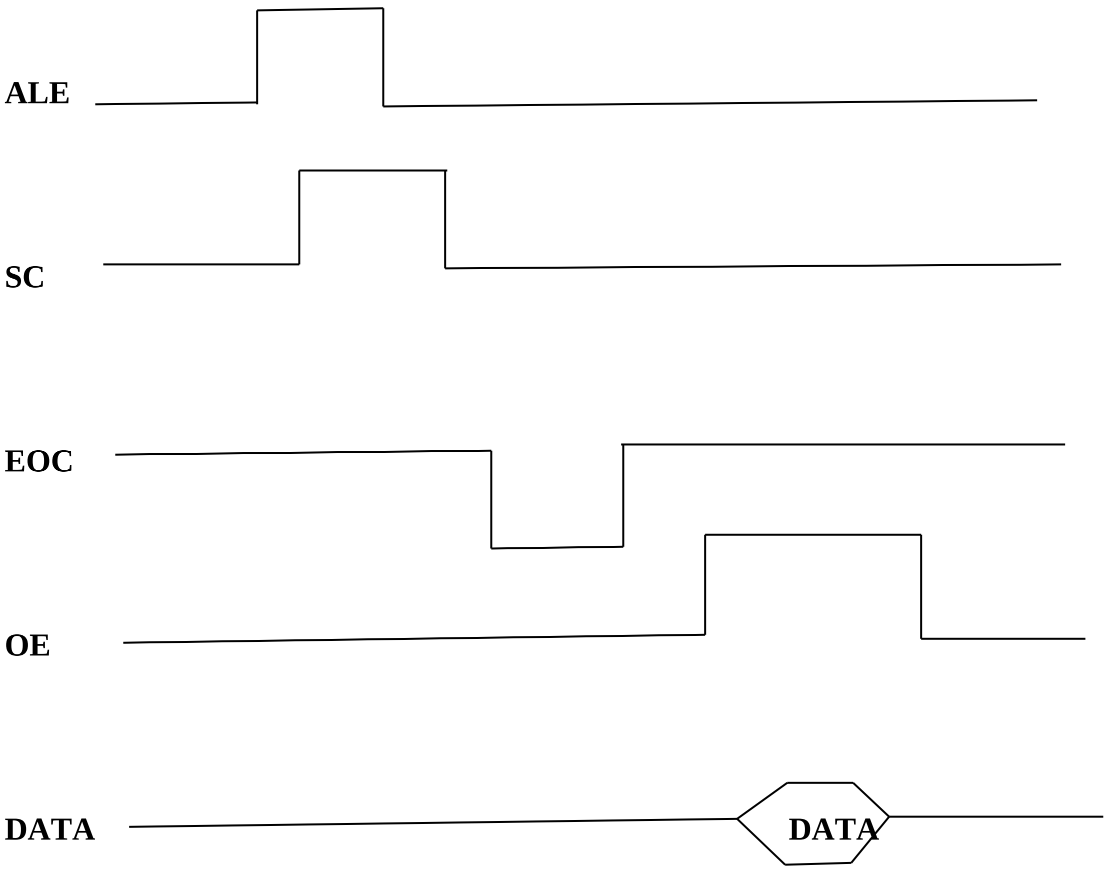
| **Port Addressess:** | **8255 # 3 (A0,A1)** | **Address** | **Selection** |
|---|---|---|---|
| | | **14** | **Port A** |
| | | **15** | **Port B** |
| | | **16** | **Port C** |
| | | **17** | **Control Register** |

**Control Word 98 (See Control Word Format)**

**Follow the following procedure**

**Step 1:** Set Channel 0 through PB (output port)
**Step 2:** Issue SC (Start Conversion) high
**Step 3:** Make ALE High
**Step 4:** Make ALE Low
**Step 5:** Make SC Low
**Step 6:** Status Check, EOC is high, if not, repeat till it goes high
**Step 7:** Make OE High
**Step 8:** Get input through PA and display it
**Step 9:** Make OE Low
**Step 10:** Go to Step 2

- **Control Signal should be given according to the following diagram**



### ii) Generating different waveforms on DAC800 output; DAC output is configured for 0 to 10V

**a) Write a program to generat ramp waveform through DAC**

**–** ADC is interfaced to 8085 using 74LS373
**–** Port Address of 373 is 18
– O/P of Latch is connected to DAC input
– DAC output is connected to op-Amp
– Observer waveform on output pin 6 of op-Amp

**Follow the following procedure**

**Step 1:** Store Digital input in Acc.
**Step 2:** Increment the Digital Value
**Step 3:** Output Digital input to LATCH
**Step 4:** Reapeat Steps 1 to 3
**Step 5:** Observer waveform on CRO.

**b) Write a program to generat square wave on OP-Amp output of 0-10 V (Also 0 to 5 V)**

**Follow the following procedure**

**Step 1:** Store High Digital Value in Acc. (FF)
**Step 2:** Output it to LATCH
**Step 3:** Give Appropriate delay
**Step 4:** Store Low Value in Acc. (00)
**Step 5:** Output it to LATCH
**Step 6:** Give Appropriate delay
**Step 7:** Repeat Steps 1 to 6
**Step 5:** Observe waveform on CRO.

**Assignment: Write a program to generat ramp triangular waveform through DAC**
**Write a program to generate staircase waveform through DAC**

## Experiment-12

**Aim:** Design a kit that can be used as software digital clock.

**Tools / Apparatus:** 8085 microprocessor trainer kit, oscilloscope, voltmeter, tools etc.

## Experiment-13

**Aim:** Design a kit that can be used as voltmeter to measure voltmeter to measure 0 to 5 volt.

**Tools / Apparatus:** 8085 microprocessor trainer kit, oscilloscope, voltmeter, tools etc.

**Procedure**:

Use ADC 0809.Channel 0 of CRO has to connect with it. Take one 26 pin FRC cable and connect it at potential meter. Now connect it at channel1 of CRO to see the out put.

## References

Reference Books

- Microprocessor Architecture, Programming and Applications with the 8085 by Ramesh Gaonkar
- 8085 Assembly Language Programming by Lance A. Leventhal

**WEBSITE:**

- www.8085projects.info