BVM Engineering College, VV Nagar

Gujarat Technological University

MICROPROCESSOR & INTERFACING

SEM 4
PRESENTATION

8255 Interfacing, DAC, ADC

Electronics & Communication Dept.

Presented By :

1. Dharati Bhimani (140080111011)
2. Abhishek Budhbhatti (140080111012)
3. Chaitanya Tejaswi (140080111013)
4. Hemi Chaudhary (140080111015)
5. Darshi Contractor (140080111016)
6. Sarju Dadhaniya (140080111017)

Guided By:

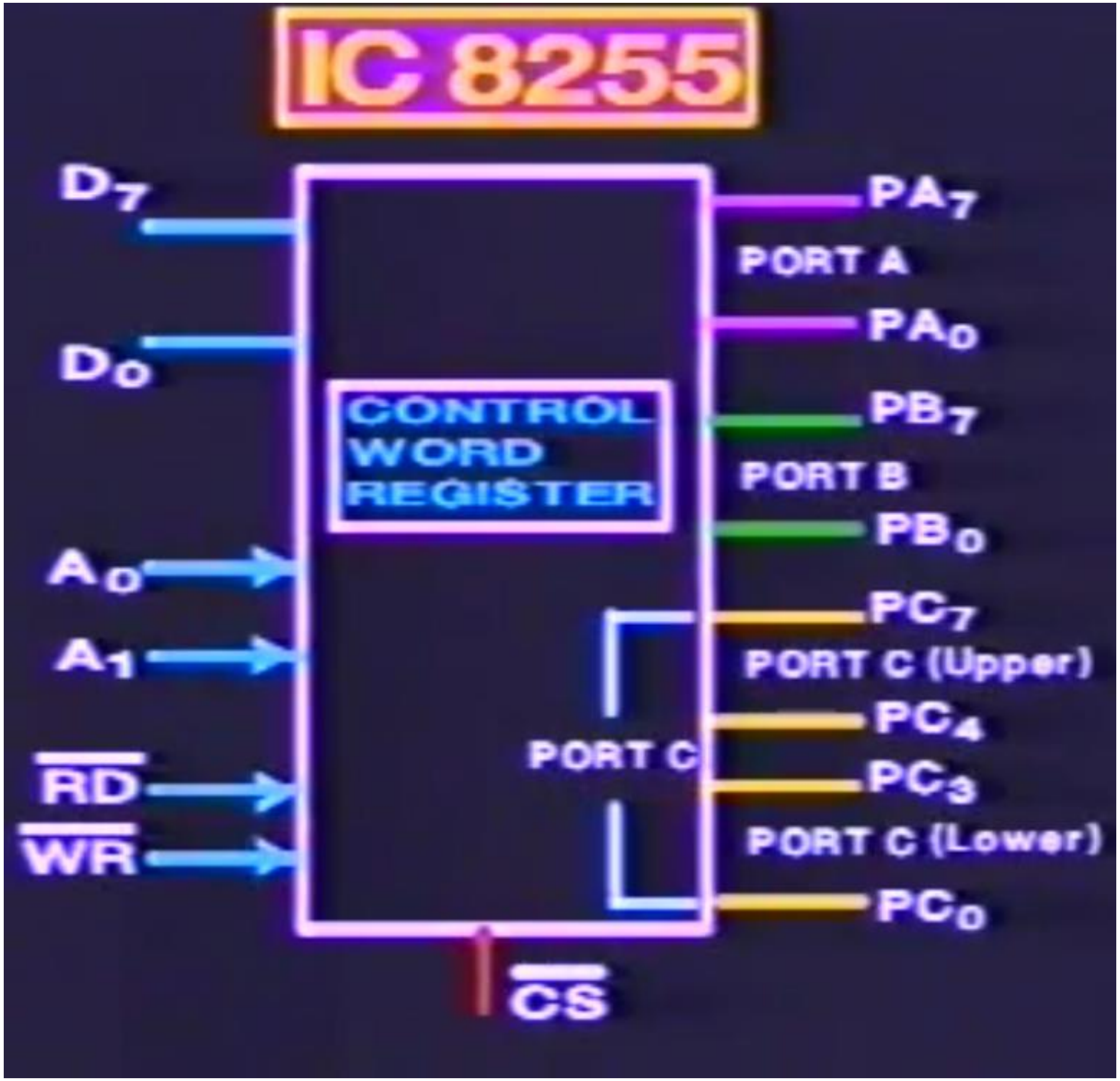
Dr. Bhargav C Goradiya

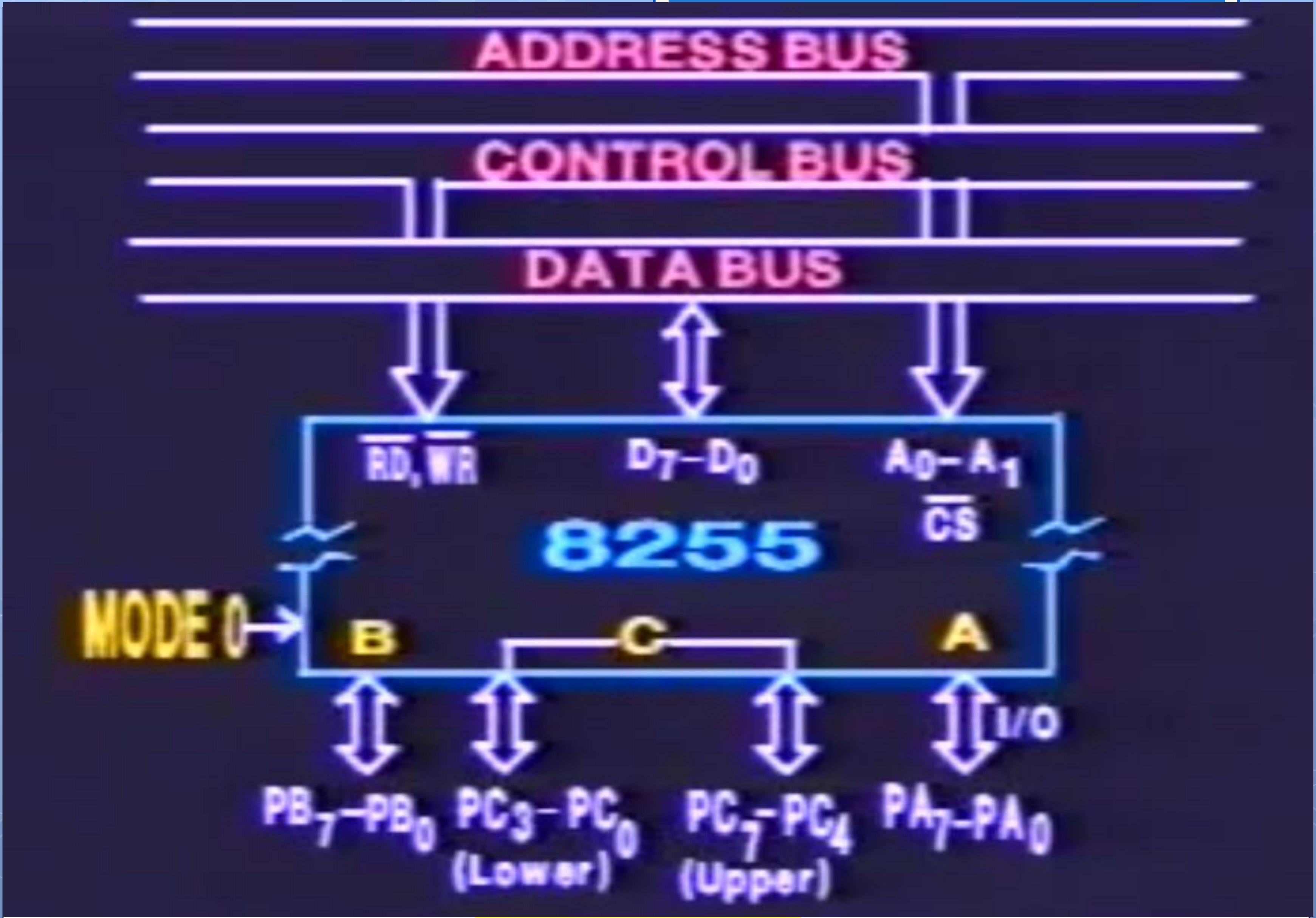
8255 PPI Interfacing: Pushbutton & Matrix Keyboard

DAC & ADC

Overview

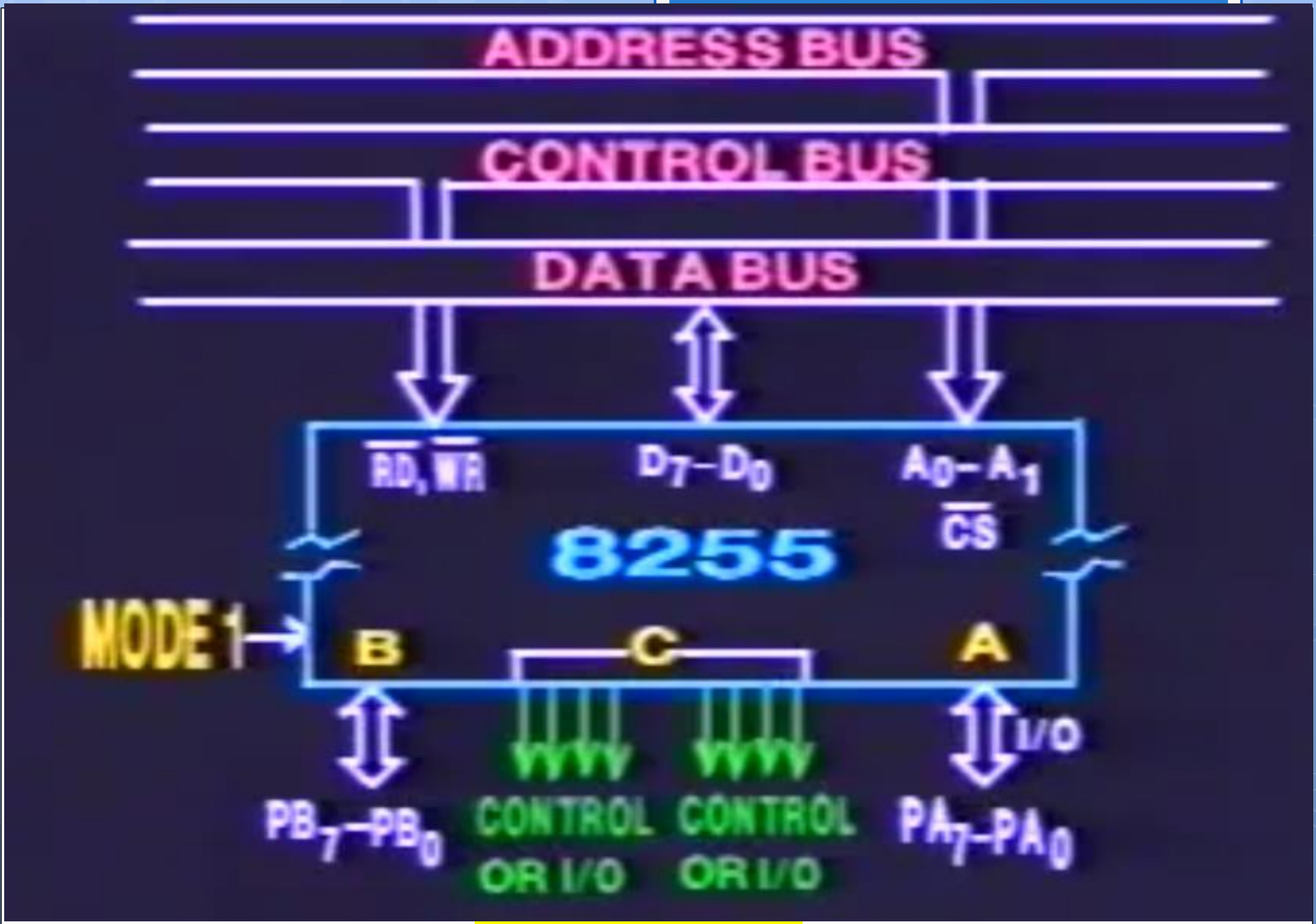
Block Diagram



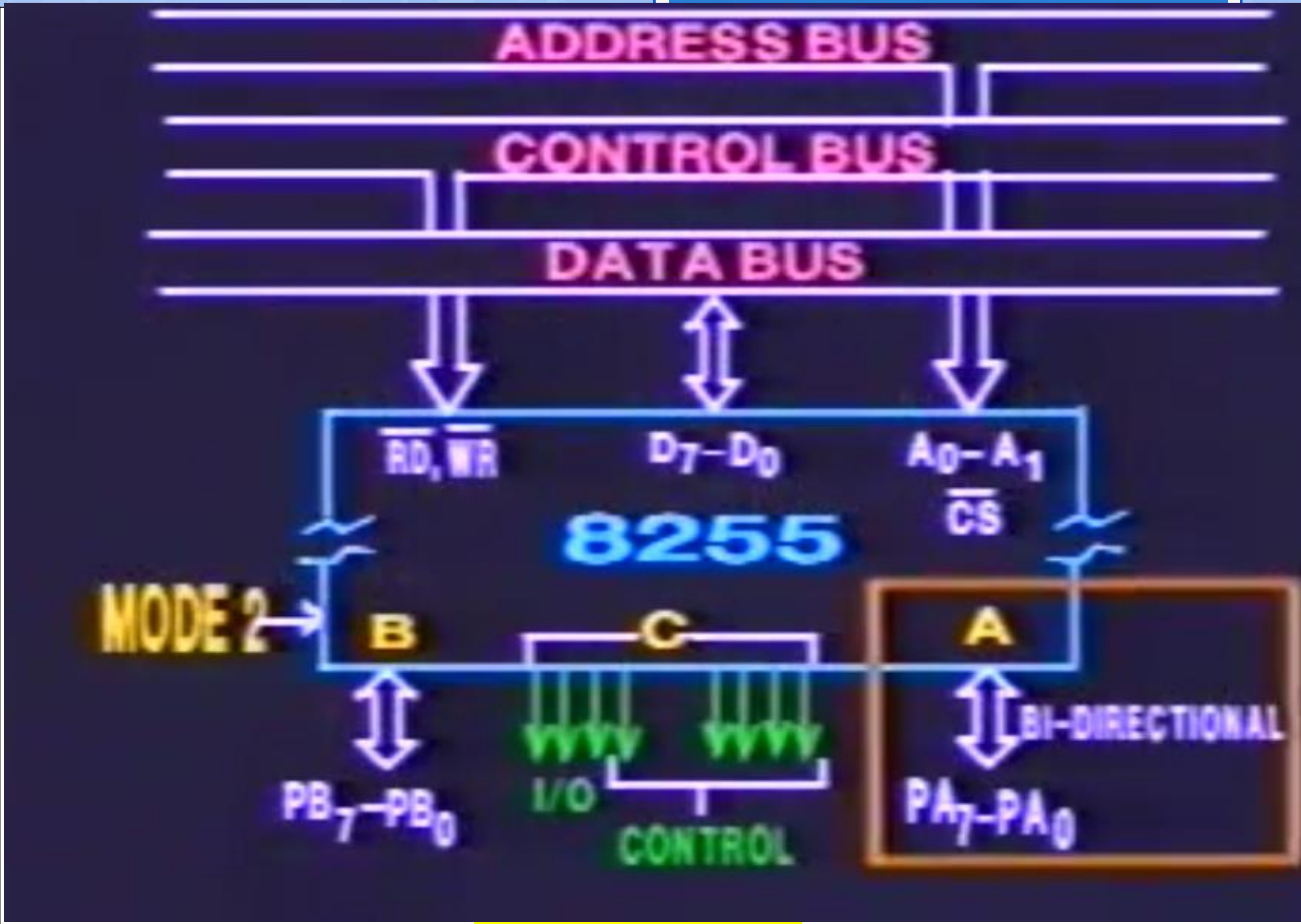


ADDRESS BUS
CONTROL BUS
DATA BUS
$\overline{RD}$, $\overline{WR}$
$D_7 - D_0$
$A_0 - A_1$
$\overline{CS}$
8255
MODE 0
B
C
A
I/O
$PB_7 - PB_0$
$PC_3 - PC_0$
(Lower)
$PC_7 - PC_4$
(Upper)
$PA_7 - PA_0$

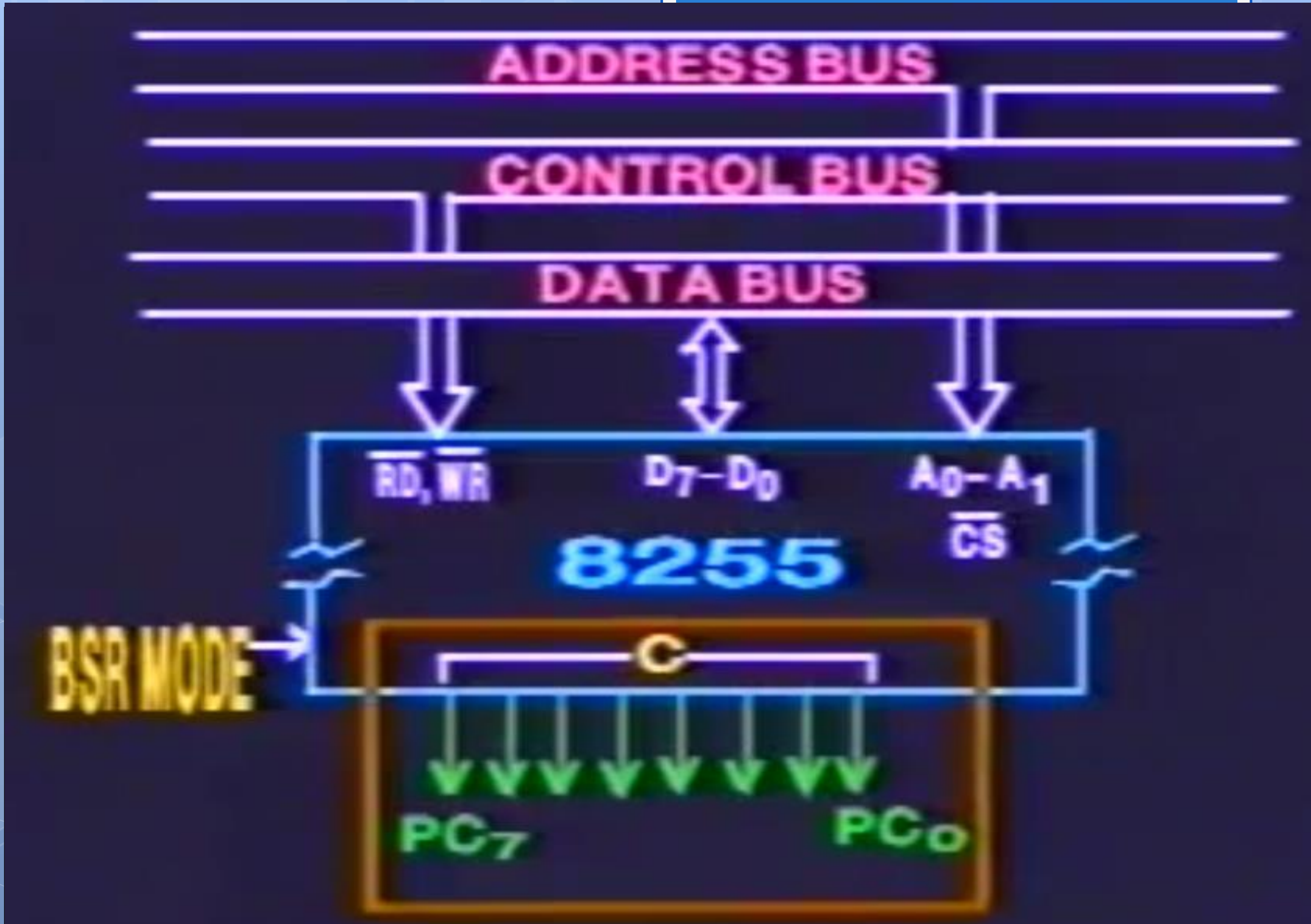
MODE:0


ADDRESS BUS
CONTROL BUS
DATA BUS
RD, WR
D7–D0
A0–A1
CS
8255
MODE 1
B
C
A
I/O
PB7–PB0
CONTROL OR I/O
CONTROL OR I/O
PA7–PA0

MODE:1


ADDRESS BUS
CONTROL BUS
DATA BUS
RD, WR
D7–D0
A0–A1
CS
8255
MODE 2
B
C
A
BI-DIRECTIONAL
PB7–PB0
I/O
CONTROL
PA7–PA0

MODE:2



ADDRESS BUS
CONTROL BUS
DATA BUS
$\overline{RD}$, $\overline{WR}$
D_7–D_0
A_0– A_1
$\overline{CS}$
8255
BSR MODE
C
PC_7
PC_0

BSR Mode

In Summary,

I/O MODE : Uses Port A,B,C

MODE:0

Port A & B Unidirectional (I/P or O/P)

Port C split into two 4-bit port, each can be used as I/P or O/P

MODE:1

Uses Port C for Control Signals

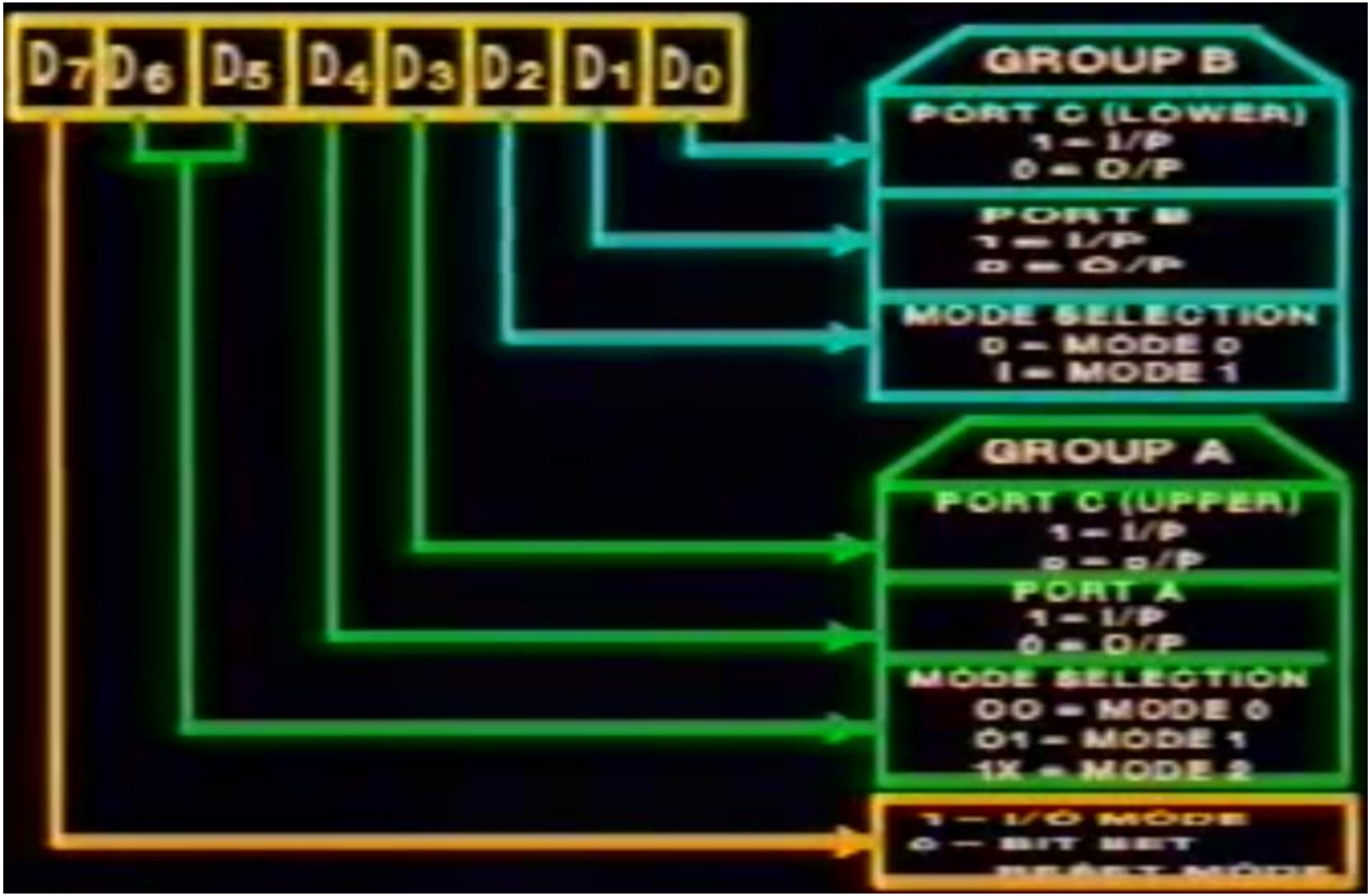
(Handshaking)

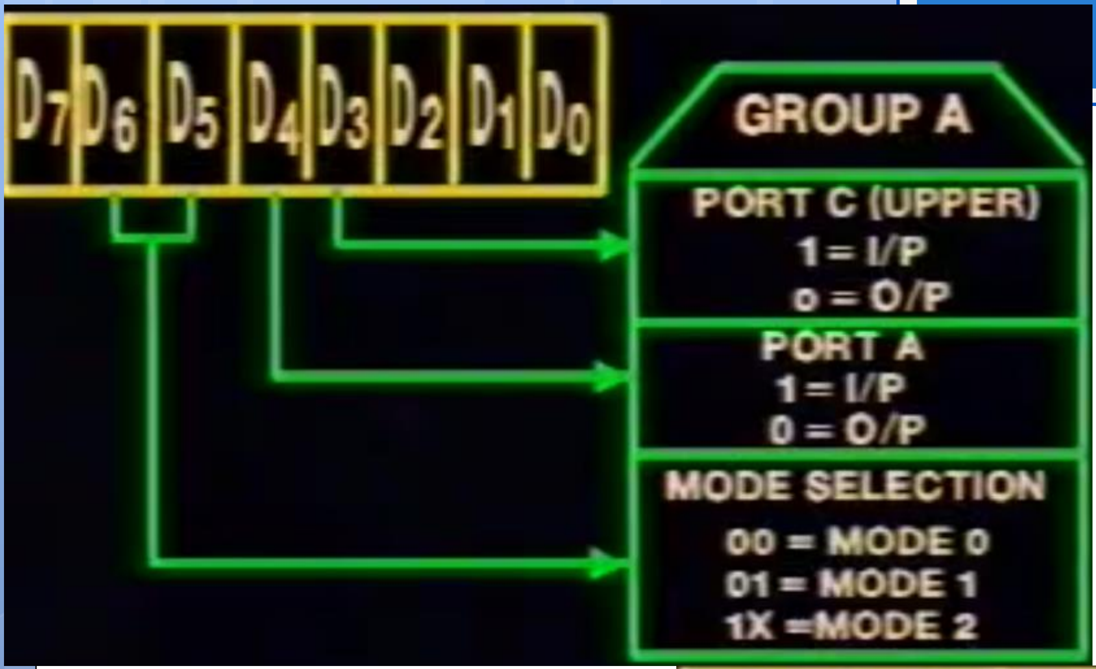
MODE:2

Port A Bi-directional

BST MODE : Uses ONLY Port C

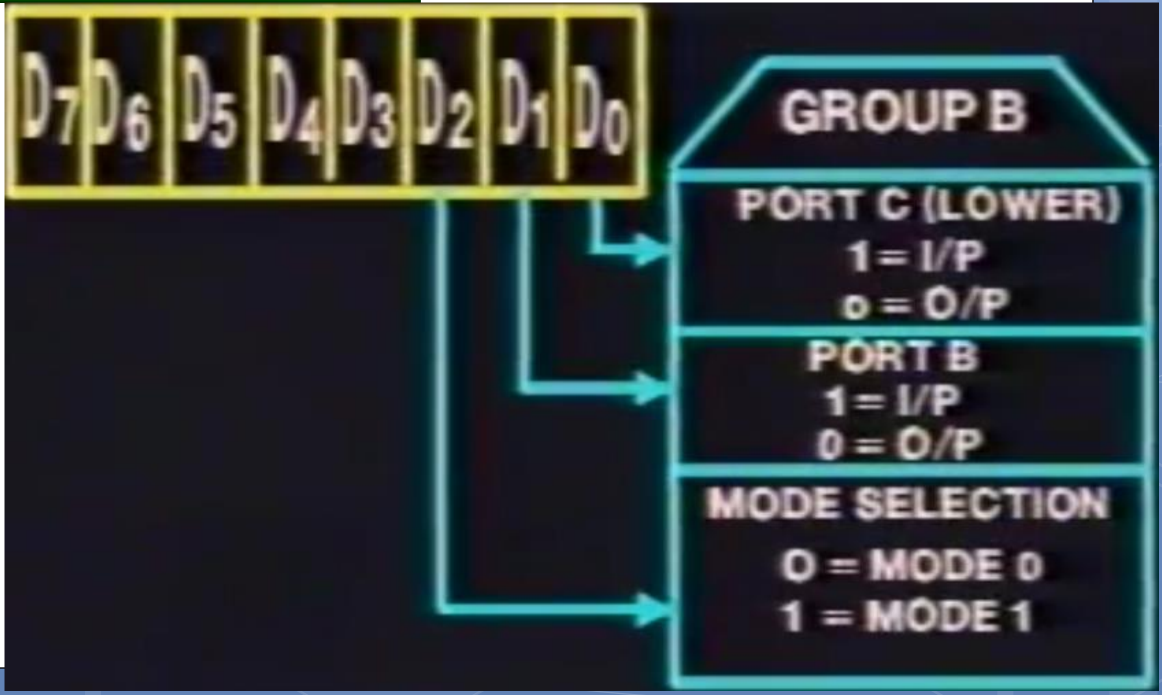
Control Word Format





Group A

Group B



In Summary,

D7 - *(IO-Mode/BST-Mode=1/0)*

Group A: **D(6,5,4,3)**

6-5 - Mode Select

4 - Port A (I/0=1/0)

3 - Port C-up (I/0=1/0)

Group B: **D(2,1,0)**

2 - Mode Select

1 - Port B (I/0=1/0)

0 - Port C-low (I/0=1/0)

I/O Address

Table 4-2: PC/XT I/O Address Map

Hex Range	Usage	Hex Range	Usage
000-00F	DMA chip 8237A-5	380-38C	SDLC communications
020-021	Interrupt 8259A	380-389	Binary synch comm. (secondary)
040-043	Timer 8253-5	390-393	Cluster
060-063	PPI 8255A-5	3A0-3A9	Binary synch comm. (primary)
080-083	DMA page registers	3B0-3BF	IBM monochrome display/printer
0AX	NMI mask register	3D0-3DF	Color/graphics
200-20F	Game control	3F0-3F7	Diskette
210-217	Expansion unit	3F8-3FF	Asynch communications (primary)
2F8-2FF	Asynch communications (secondary)	790-793	Cluster (adapter 1)
300-31F	Prototype card	B90-B93	Cluster (adapter 2)
320-32F	Fixed disk	1390-1393	Cluster (adapter 3)
378-37F	Printer	2390-2393	Cluster (adapter 4)

PC Enable Of 8255 Ports

Port A Address = 60H All input	PA0	Keyboard or data from pointing device
	PA1	
	PA2	
	PA3	
	PA4	
	PA5	
	PA6	
	PA7	

Port B Address = 61H All output	PB0	Timer 2 speaker enable
	PB1	Speaker data
	PB2	not used
	PB3	not used
	PB4	RAM parity check enable
	PB5	I/O channel check enable
	PB6	not used
	PB7	not used

Port C Address = 62H All input	PC0	not used
	PC1	Coprocessor installed if = 1
	PC2	Disk installed if = 0
	PC3	not used
	PC4	not used
	PC5	Timer channel 2 output
	PC6	I/O channel check
	PC7	RAM parity check

Keyboard & 7-Segment Display Interfacing

(Typical Example of Mode 0 Configuration)

Keyboard Interfacing

Problem Statement:

A pushbutton keyboard is connected to Port A & 7-segment display to port B. WAP to sense pressed key & display the key's number at the 7-segment LED.

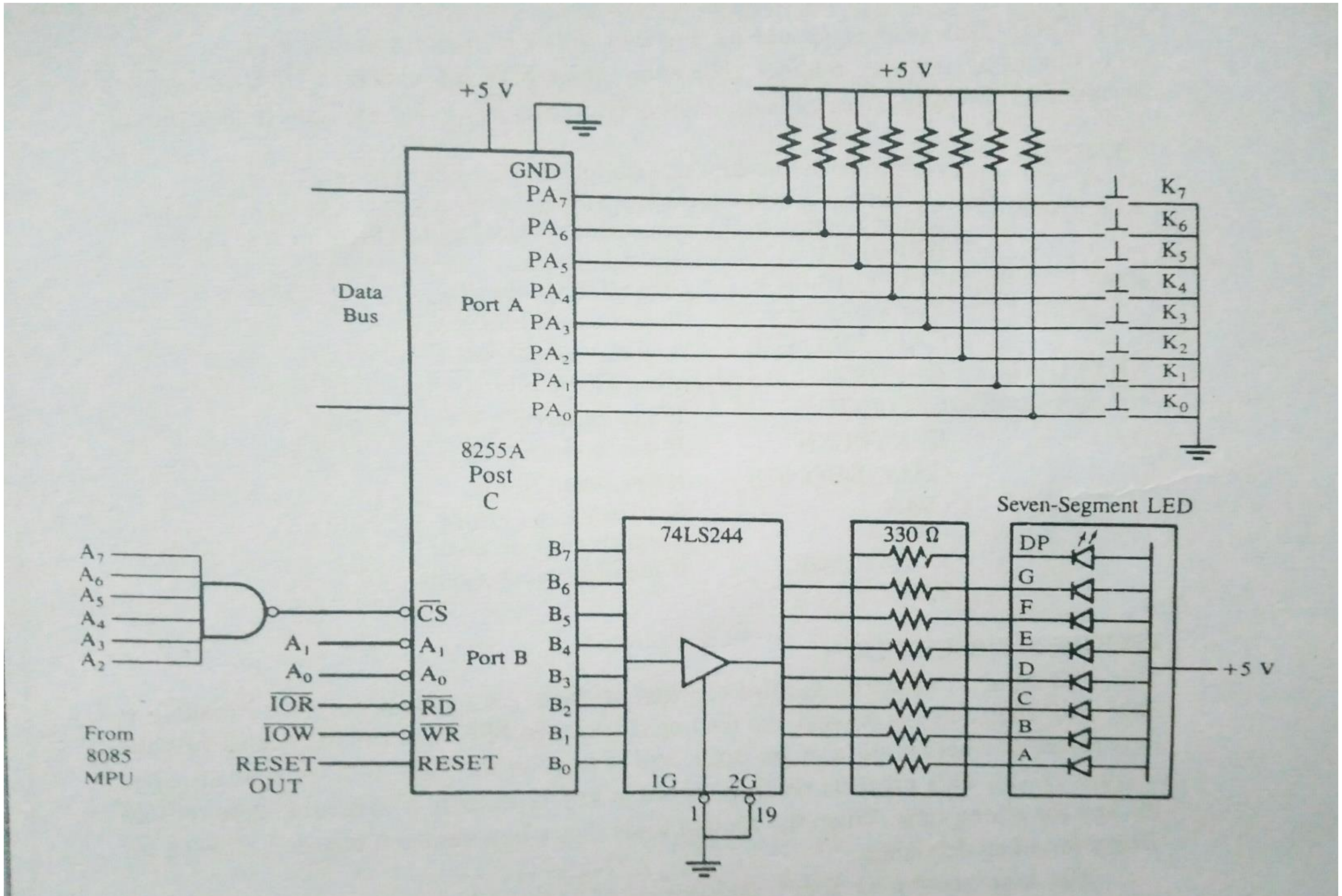


FIGURE 15.15
Interfacing a Keyboard and a Seven-Segment LED

Subroutines

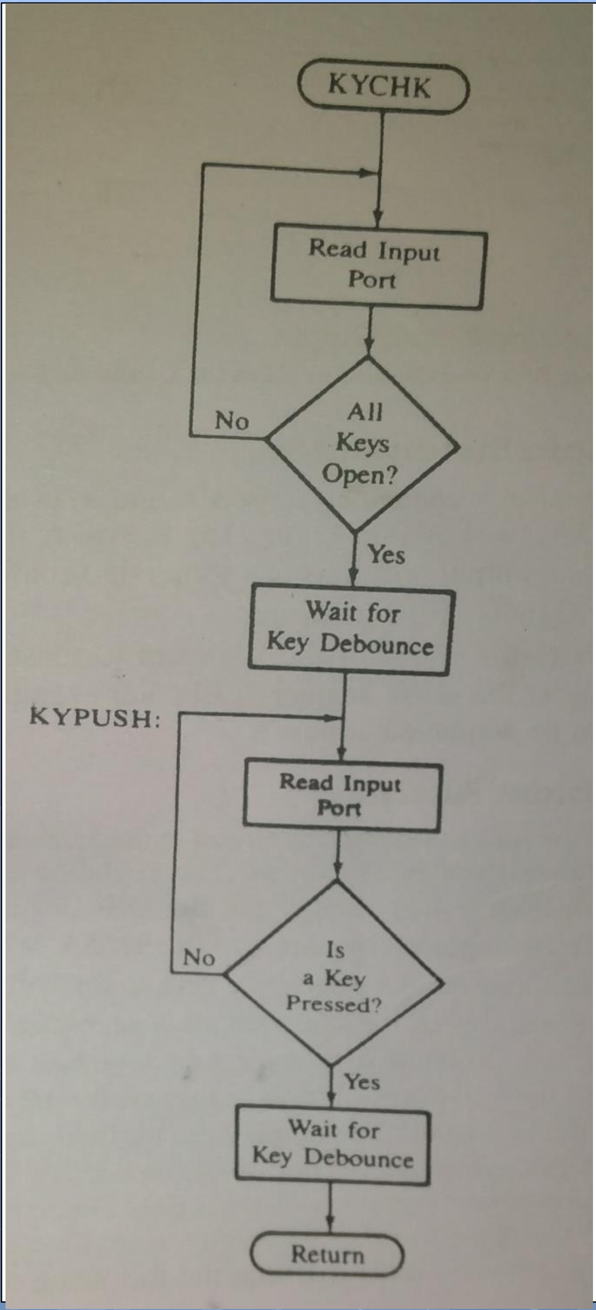
MAIN Program

15.2.3

Now to monitor the keyboard and display the key pressed, we need to initialize the 8255A ports and combine the software modules discussed below:

```
KYBORD: ;This program initializes the 8255A ports; port A and port B in Mode 0
        ;   and then calls the subroutine modules discussed
        ;   previously to monitor the keyboard
PORTA   EQU FCH          ;Port A address
PORTB   EQU FDH          ;Port B address
CNTRL   EQU FFH          ;Control register
CNWORD  EQU 90H          ;Mode 0 control word, port A input and port B output
STACK   EQU 20AFH        ;Beginning stack address
        LXI SP,STACK
PPI:    MVI A,CNWORD
        OUT CNTRL        ;Set up port A in Mode 1
NEXTKY: CALL KYCHK       ;Check if a key is pressed
        CALL KYCODE      ;Encode the key
        CALL DSPLAY      ;Display key pressed
        JMP NEXTKY       ;Check the next key pressed
```

Key Check




```
KYCHK:  ;This subroutine first checks whether all keys are open.
        ;  Then, it checks for a key closure, debounces the key, and places
        ;  the reading in the accumulator. See Figure 15.16 for flowchart.
        IN PORTA        ;Read keyboard
        CPI 0FFH        ;Are all keys open?
        JNZ KYCHK       ;If not, wait in loop
        CALL DBONCE     ;If yes, wait 20 ms
KYPUSH: IN PORTA        ;Read keyboard
        CPI 0FFH        ;Is key pressed?
        JZ KYPUSH       ;If not, wait in loop
        CALL DBONCE     ;If yes, wait 20 ms
        CMA             ;Set 1 for key closure
        ORA A           ;Set 0 flag for an error
        JZ KYPUSH       ;It is error, check again
        RET
```

```
KYCODE: ;This routine converts (encodes) the binary hardware reading of the key
        ;   pressed into appropriate binary format according to the number of the
        ;   key.
        MVI C,08H       ;Set code encounter
NEXT:   DCR C           ;Adjust key code
        RAL             ;Place MSB in CY
        JNC NEXT        ;If bit = 0, go back to check next bit
        MOV A,C         ;Place key code in the accumulator
        RET
```

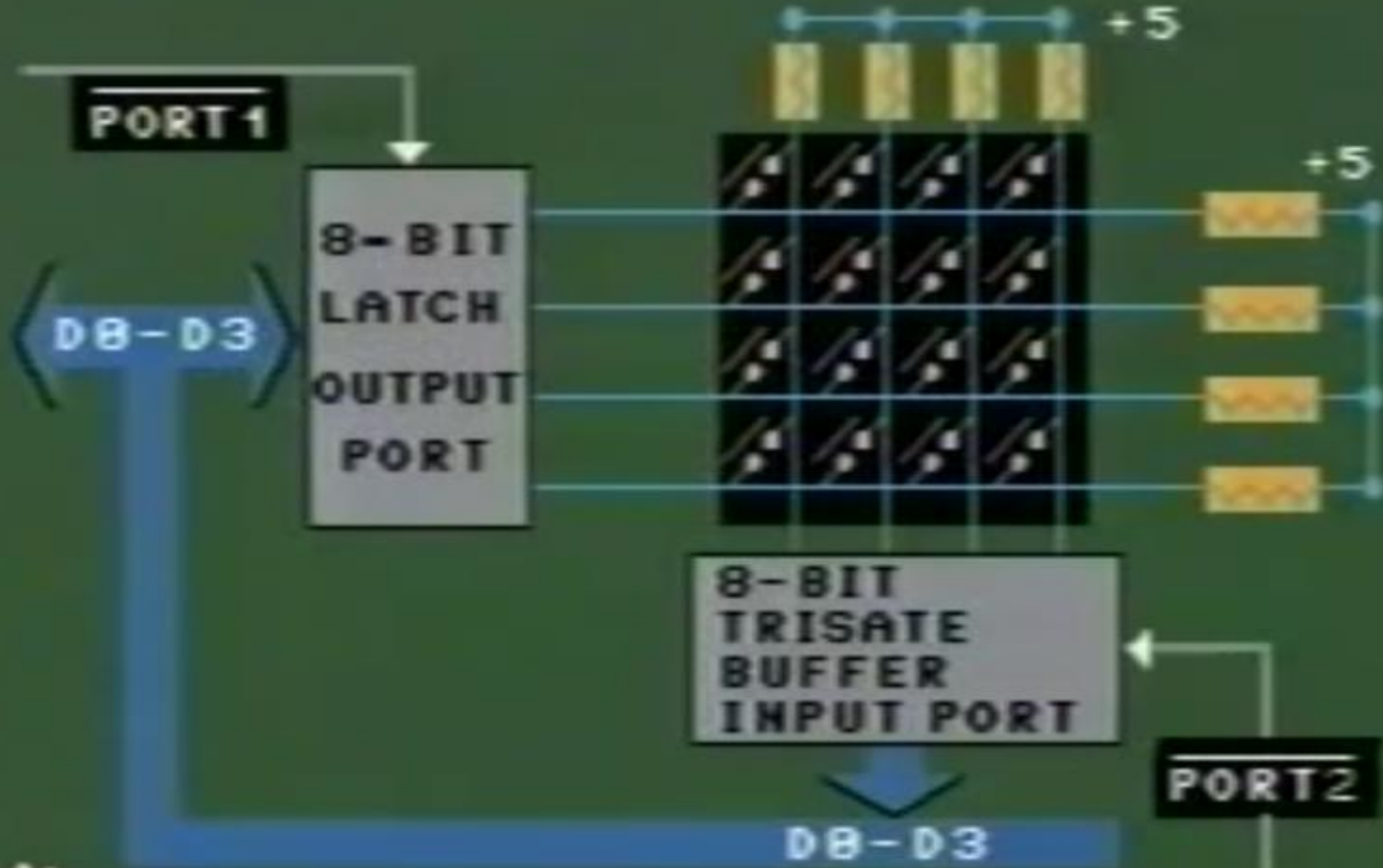
Encoding

```
DBONCE: ;This is a 20 ms delay routine
        ;The delay COUNT should be calculated based on system frequency
        ;This does not destroy any register contents
        ;Input and Output = None
        PUSH B          ;Save register contents
        PUSH PSW
        LXI B,COUNT     ;Load delay count
LOOP:   DCX B           ;Next count
        MOV A,C
        ORA B           ;Set Z flag if (BC) = 0
        JNZ LOOP
        POP PSW         ;Restore register contents
        POP BC
        RET
```

Debouncing

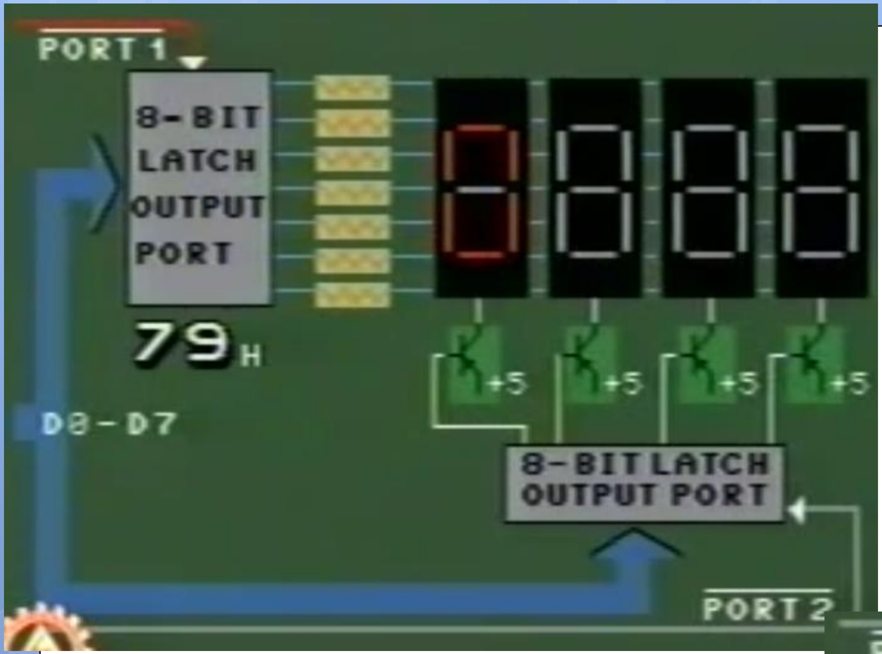
```
DSPLAY:  ;This routine takes the binary number and converts into its common-
         ;  anode seven-segment LED code. The codes are stored in memory
         ;  sequentially, starting from the address CACODE
         ;Input: Binary number in accumulator
         ;Output: None
         ;Modifies contents of HL and A
         LXI H,CACODE        ;Load starting address of code table in HL
         ADD L               ;Add digit to low-order address in L
         MOV L,A             ;Place code address in L
         MOV A,M             ;Get code from memory
         OUT PORTB           ;Send code to port B
         RET
CACODE:  ;Common-anode seven-segment codes are stored sequentially in memory
         DB 40H,79H,24H,30H,19H,12H  ;Codes for digits from 0 to 5
         DB 02H,78H,00H,18H,08H,03H  ;Codes for digits from 6 to B
         DB 46H,21H,06H,0EH          ;Codes from digits from C to F
```

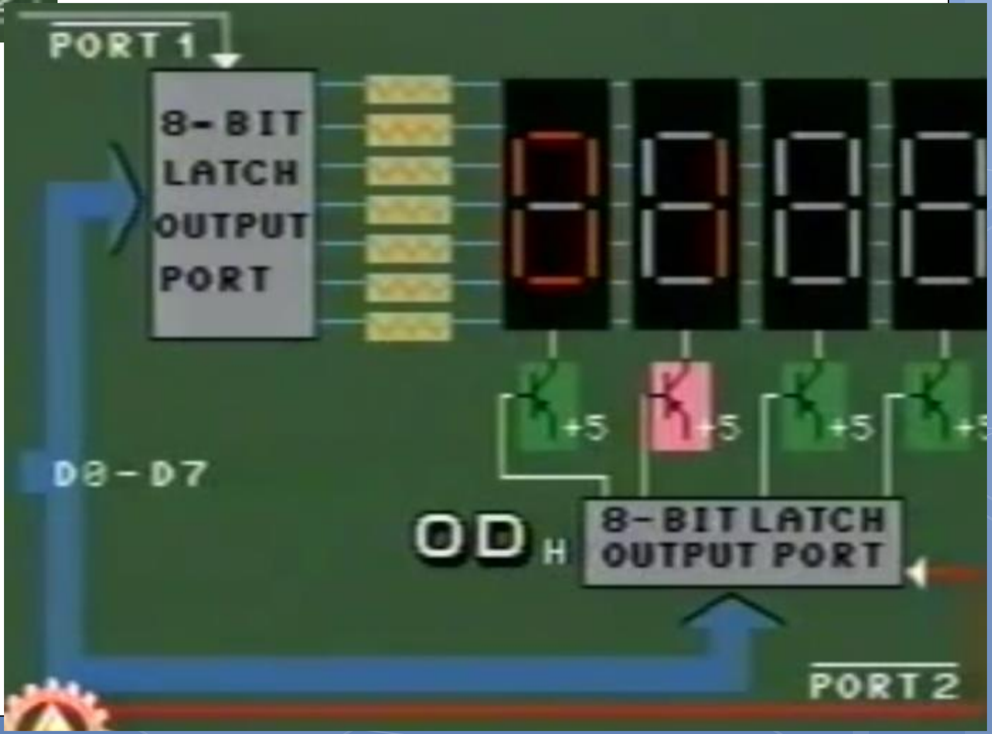
7-Segment Display



4X4 Keyboard Matrix

Multiplexed Display Using O/P Port





Converter Interfacing

Interfacing ADC_DAC with 8085.mp4

DAC Interfacing

D/A Converter

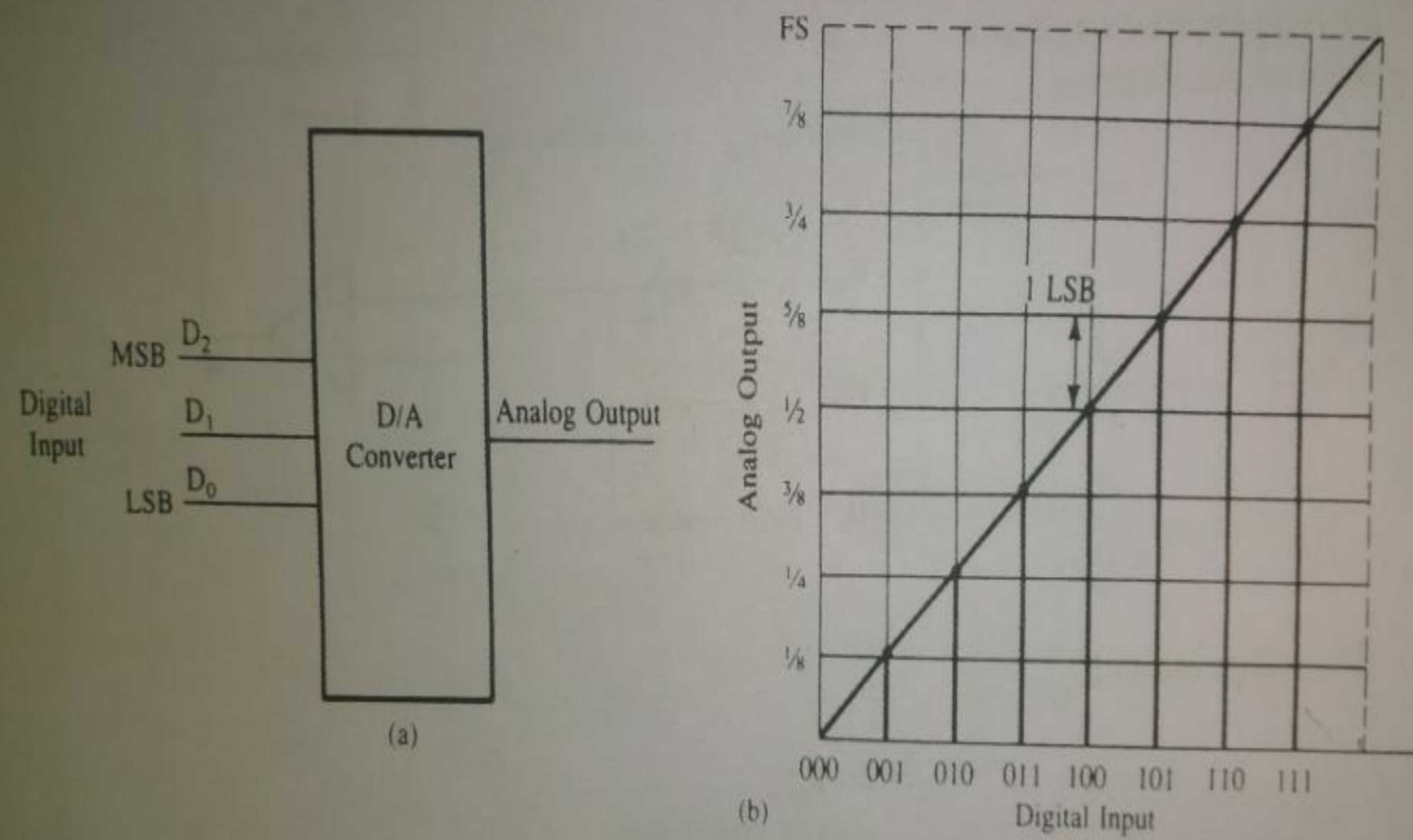
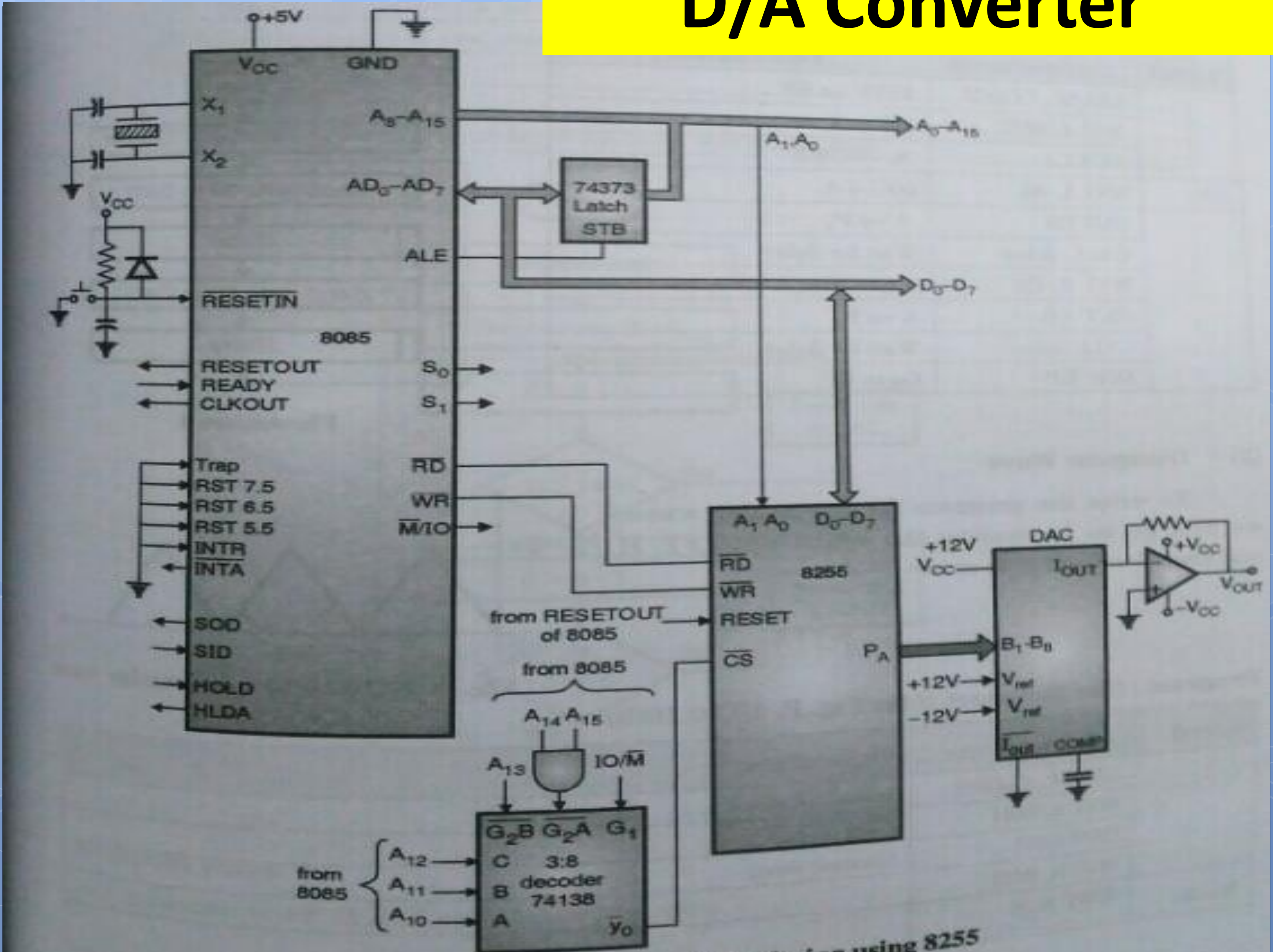


FIGURE 13.1
A 3-Bit D/A Converter: Block Diagram (a) and Digital Input vs. Analog Output (b)

D/A Converter



...facing using 8255

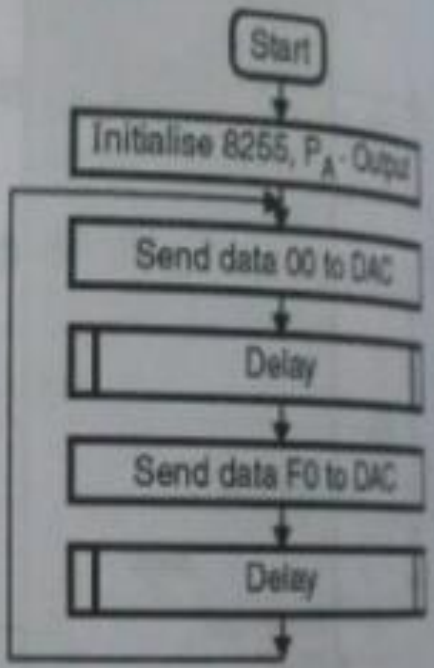
Subroutines

Square wave

(1) Square wave

To generate square wave we require two levels of inputs 00 H & F0 H. This is achieved by using output data 00 H wait for some time, then output data F0 H wait for some time. Program (For flowchart refer Flowchart 8).

Label	Instructions	Operation
	LXI SP, FFFFH	FFFF → SP
	MVI A, 80H	80 → A
	OUT C3	A → CWR
UP :	MVI A, 00	00 → A
	OUT C0	A → P_A
	CALL delay	Wait for delay
	MVI A, F0	F0 → A
	OUT C0	A → P_A
	CALL delay	Wait for delay
	JMP UP	Go to UP.



Flowchart 8

Triangular wave

(2) Triangular Wave

To write the program for triangular wave, we first go on increasing the count upto FF H and then decrease it to zero.

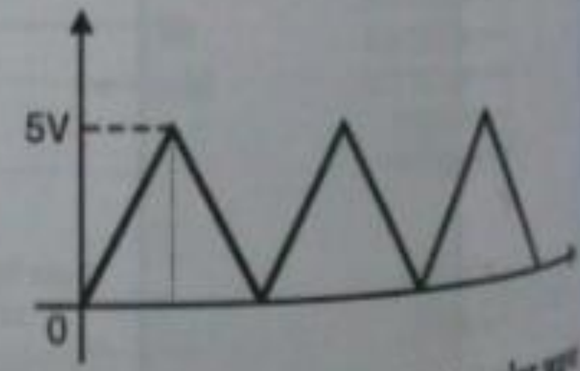
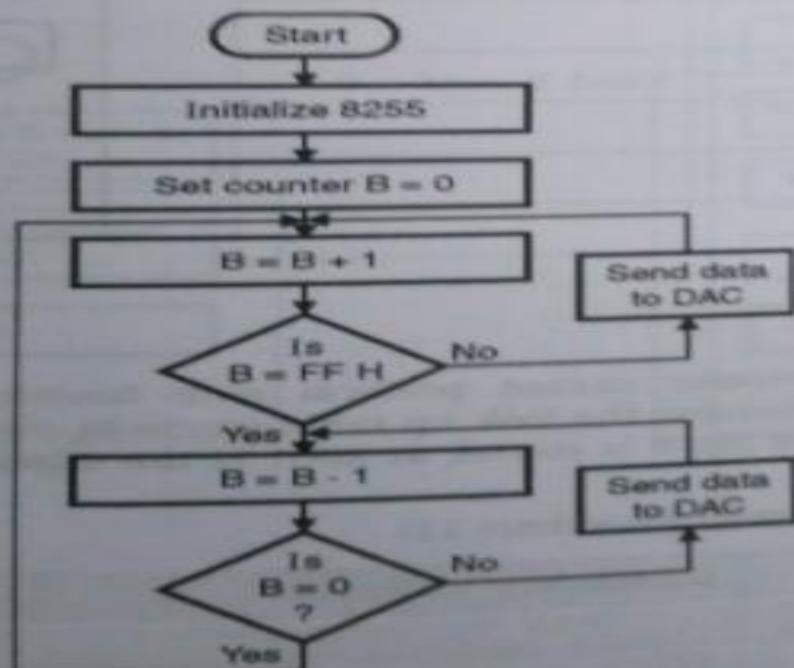


Fig. P. 10.10.10(b) : Triangular wave

Program : (For flowchart refer Fig. P. 10.10.10(b)

Label	Instructions	Operation
	LXI SP, C600H	Initialize stack
	MVI A, 80H	
	OUT C3H	Control word
	MVI B, 00H	
BACK:	MOV A, B	

Label	Instructions	Operation
UP:	OUT C1H	Send rising edge data
	CALL DELAY	If required give delay
	INR B	
	MOV A, B	Store new data/count
	CPI FFH	Check for 5 V.
	JNZ UP	
AGAIN:	DCR B	If 5V, decrement count
	MOV A, B	Save count
	CPI 00H	Check for 0V
	JZ BACK	If yes, start for rising edge count.
	OUT C1H	
	CALL DELAY	
	JMP AGAIN	Give delay & continue with falling edge



Sawtooth wave

Sawtooth Wave

We initialize to FF H and count down to zero.

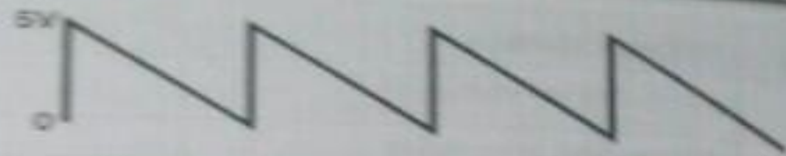
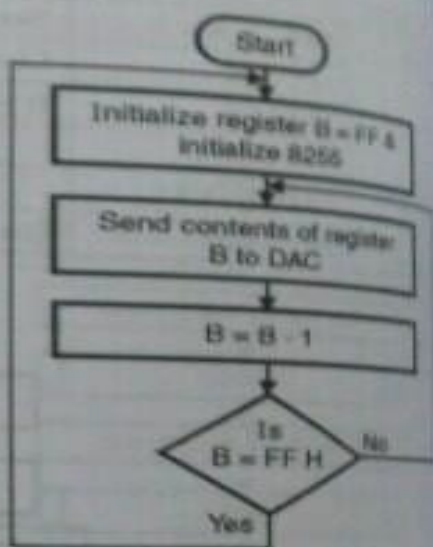


Fig. P. 10.10.10(c) : Sawtooth wave

Program : (For flowchart refer Flowchart 10)

Label	Instruction	Comment
	LXI SP, C600H	
	MVI A, 80H	Control word
	OUT C3H	
BACK:	MVI B, FFH	Count for 5V
UP:	OUT C1H	
	DCR B	Decrement count
	MOV A, B	
	CPI 00H	Check if count = 0.
	JZ BACK	
	CALL DELAY	
	JMP UP	



Sine wave

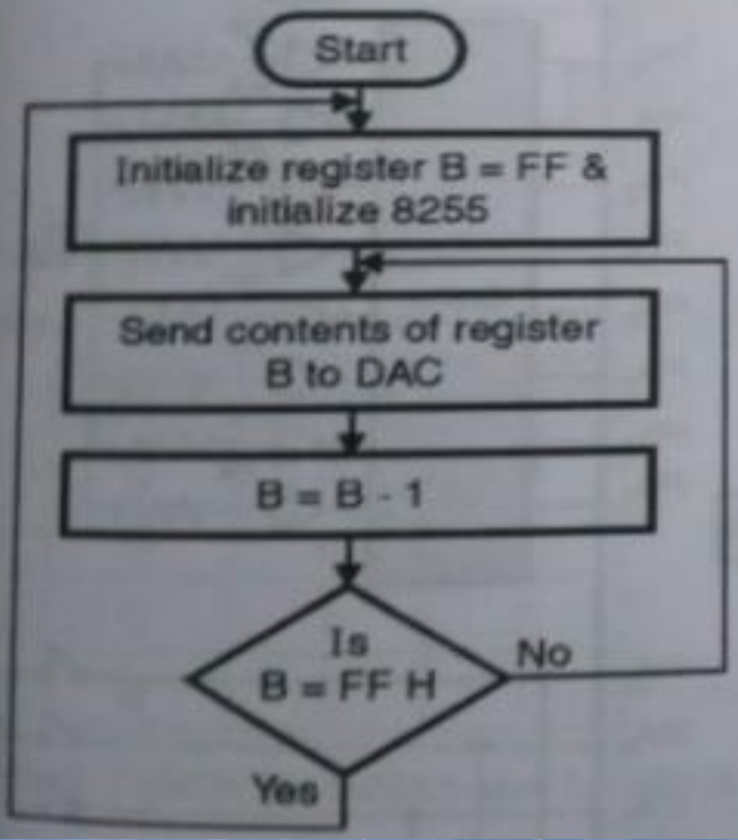
(4) Sine Wave

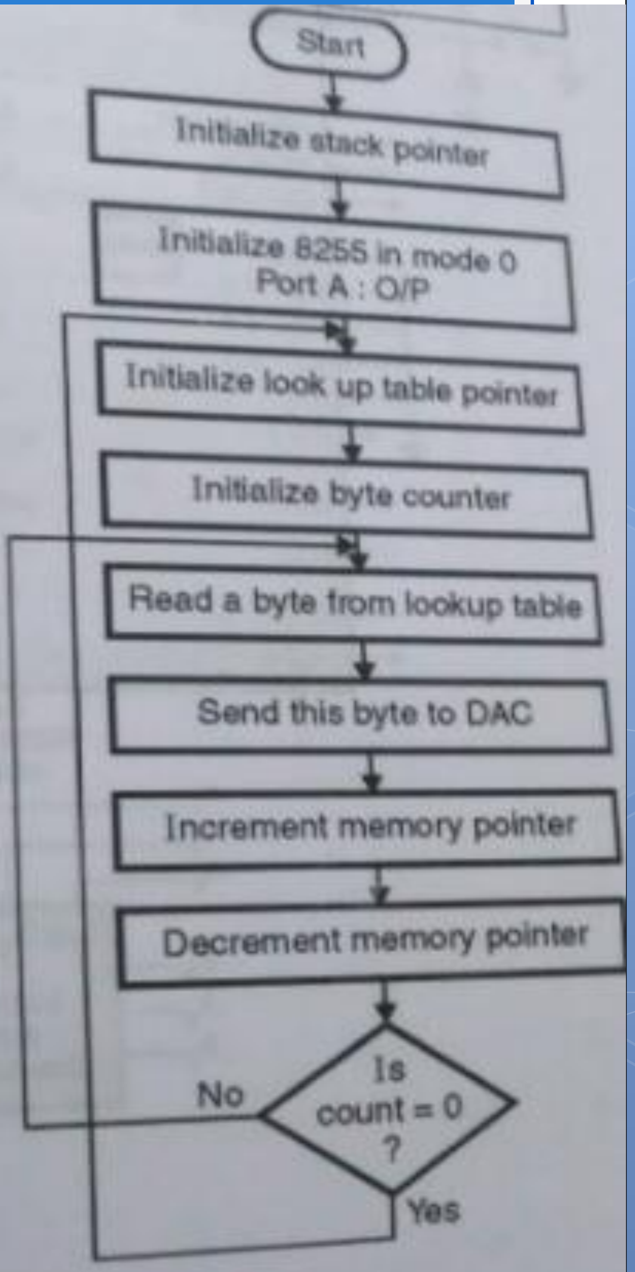
The 8085 microprocessor cannot perform sine function, hence look up table technique must be used. Assume the look up table starts at memory location 2000 H. At 2000 H the digital value of Sin 0 is stored, at 2001 H the digital value of Sin 1 is stored and so on.

Program : (For flowchart refer Flowchart 11)

Label	Instruction	Comment
	LXI SP, FFFFH	
	MVI D, 80H	8255 in mode 0
	OUT C3	Port A o/p
BACK:	LXI H, 2000H	Look up table pointer
	LXI D, 0168H	Byte counter (360 values)
UP:	MOV A, M	Send value to DAC
	OUT C0	

Label	Instruction	Comment
	INX H	Increment look up table pointer.
	DCX D	Decrement byte counter.
	MOV A, E	
	ORA D	
	JNZ UP	
	JMP BACK	


Start
Initialize register B = FF &
initialize 8255
Send contents of register
B to DAC
B = B - 1
Is
B = FF H
No
Yes

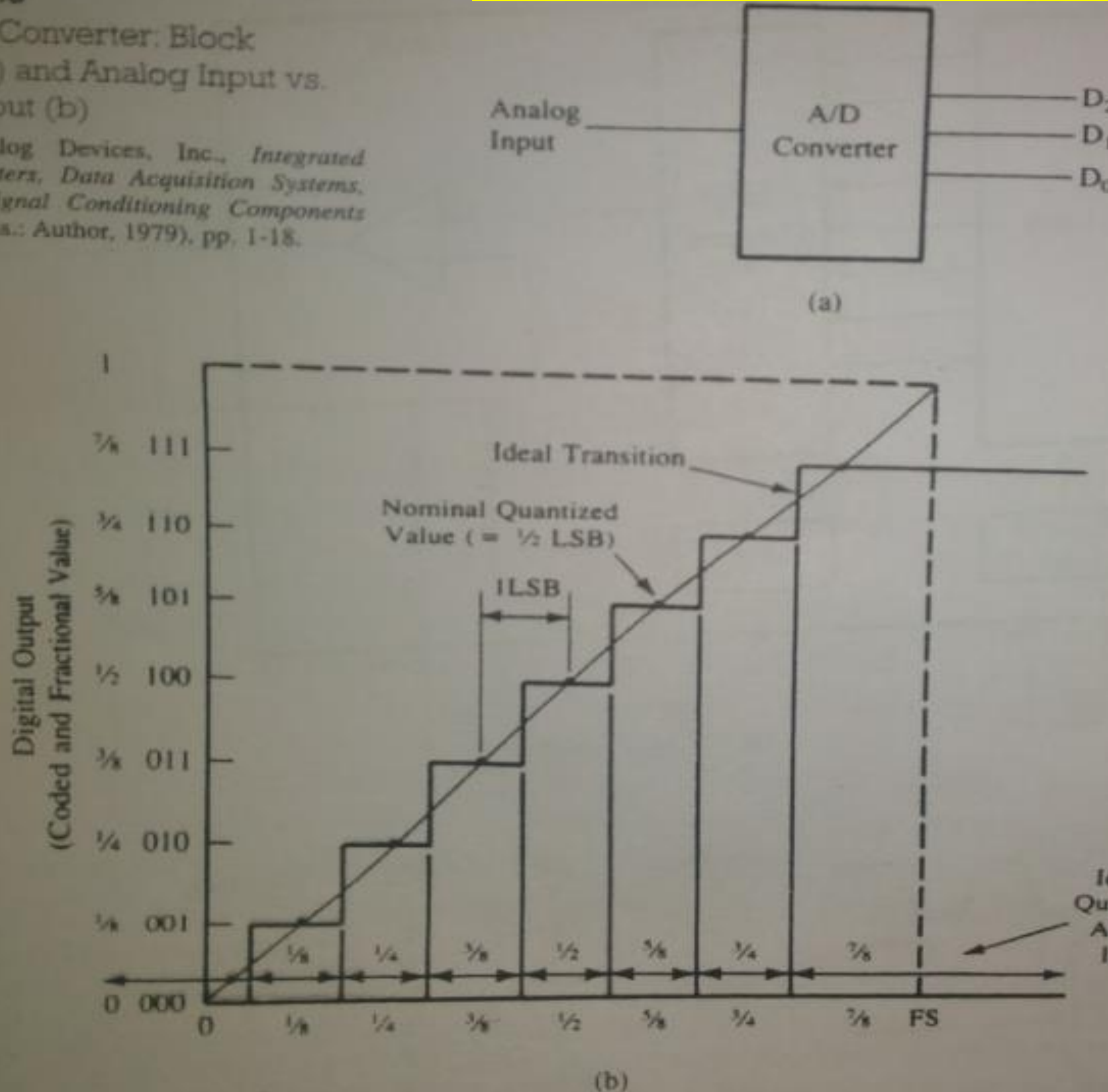

Start
Initialize stack pointer
Initialize 8255 in mode 0
Port A : O/P
Initialize look up table pointer
Initialize byte counter
Read a byte from lookup table
Send this byte to DAC
Increment memory pointer
Decrement memory pointer
Is
count = 0
?
No
Yes

ADC Interfacing

A/D Converter

FIGURE 13.8
A 3-Bit A/D Converter: Block Diagram (a) and Analog Input vs. Digital Output (b)

SOURCE: Analog Devices, Inc., *Integrated Circuit Converters, Data Acquisition Systems, and Analog Signal Conditioning Components* (Norwood, Mass.: Author, 1979), pp. 1-18.



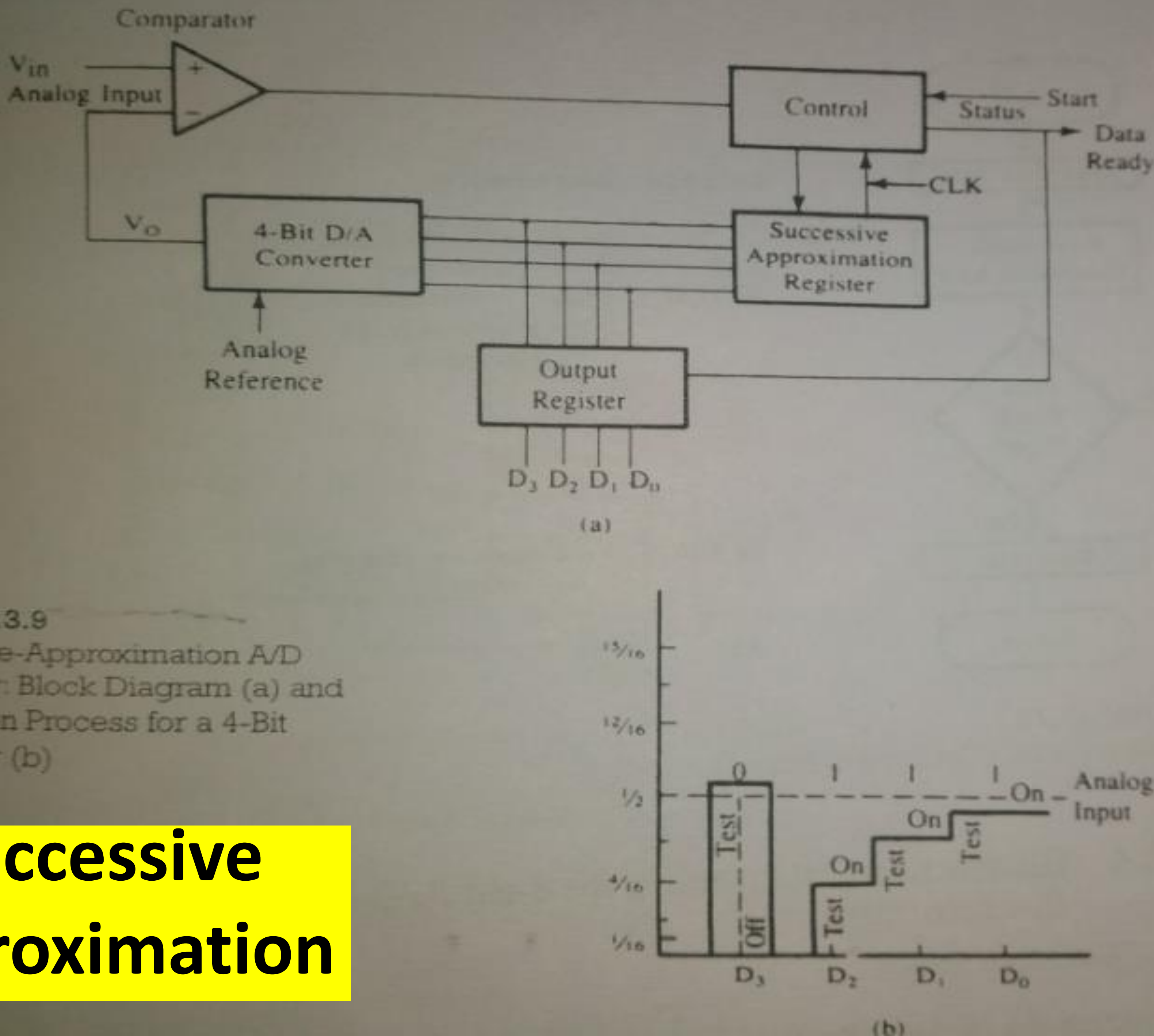
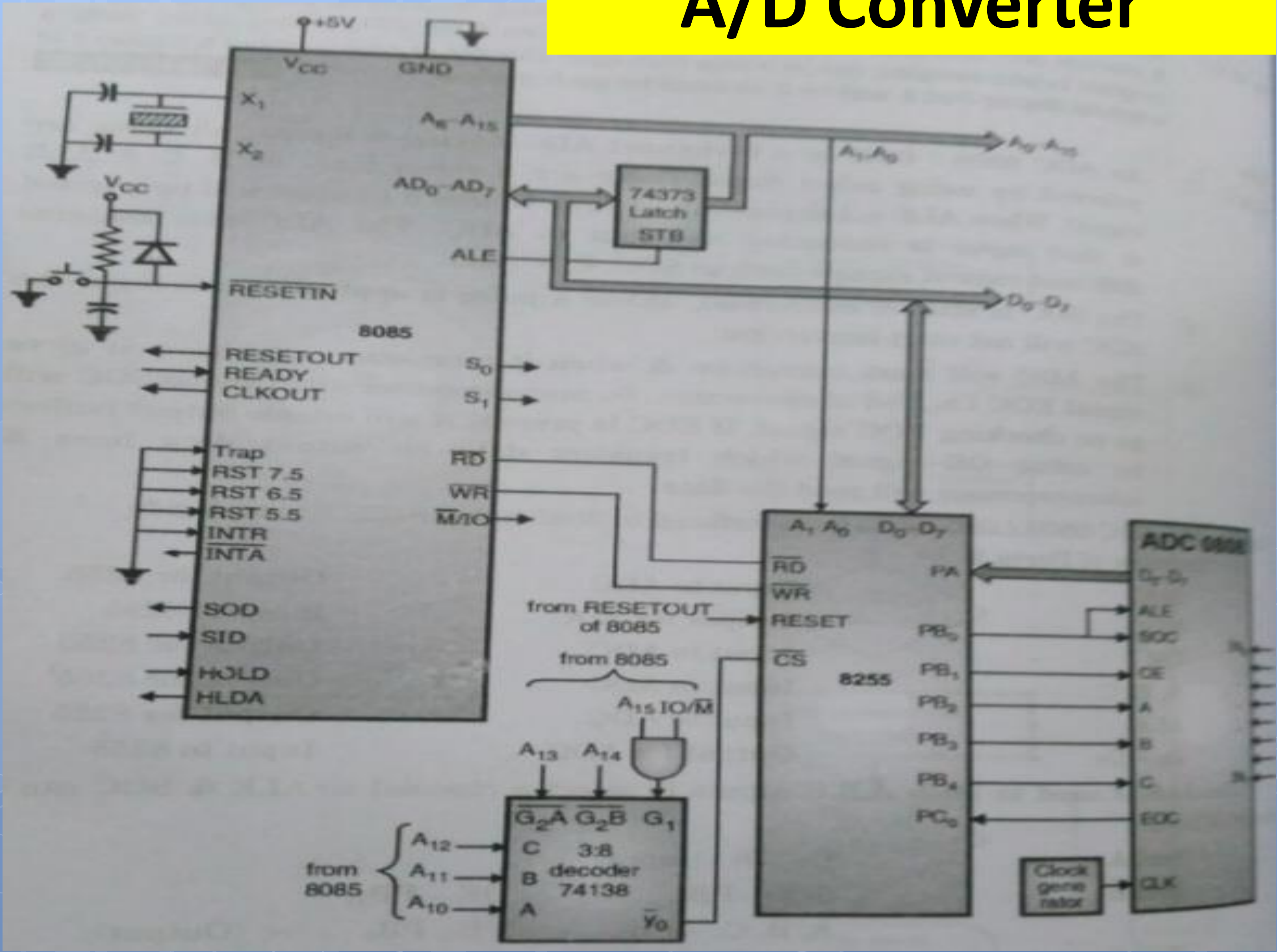


FIGURE 13.9
Successive-Approximation A/D Converter: Block Diagram (a) and Conversion Process for a 4-Bit Converter (b)

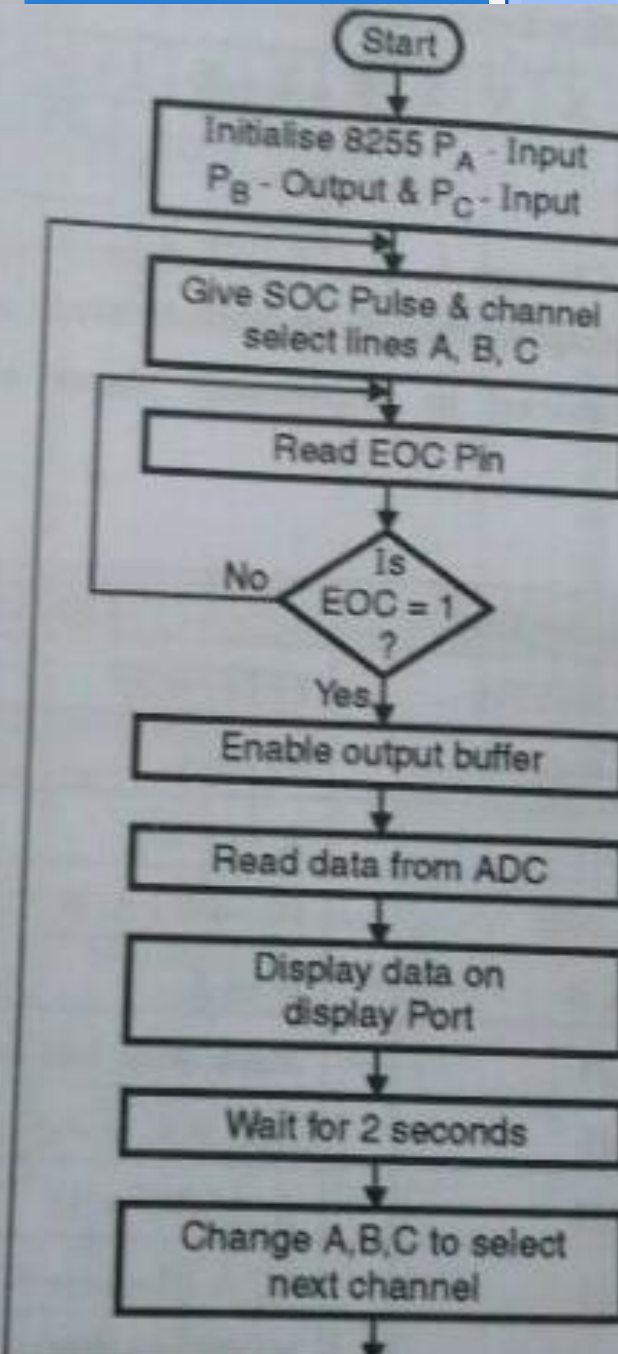
Successive Approximation

A/D Converter



Subroutines

Label	Instruction	Operations
	LXI SP, FFFFH	FFFF → SP
	MVI A, 99H	99 → A
	OUT 63H	A → CWR
	MVI B, 00H	00 → B (A, B, C)
Loop :	MOV A, B	B → A
	ANI 1CH	Mask other bits except A, B, C
	ORI 01H	Add SOC to data
	OUT 61H	A (SOC = 1)
	ANI 1CH	1C A (Remove SOC from data)
	OUT 61	A Port → B(SOC = 0)
Up :	IN 62H	P_C → A
	ANI 01H	Mask other bits except
	CPI 01H	Check bit (EOC)
	JNZ Up	If P C_0 = reset, go to up
	MVI A, 02H	02 → A
	OUT 61H	(OE = 1)
	IN 60H	P_A → A
	OUT display	A → Display
	CALL delay	Wait for 2 sec.
	MOV A, B	B → A
	ADI 04H	A + 04 → A (Change A, B, C to select next ch.)
	MOV B, A	A → B
	JMP loop	Go to loop



THANK YOU !