University of San Carlos | Department of
COMPUTER ENGINEERING

CpE 3104
Microprocessors

I/O Interfacing

Interfacing Parallel I/O (Keypad & LCD)

Contents

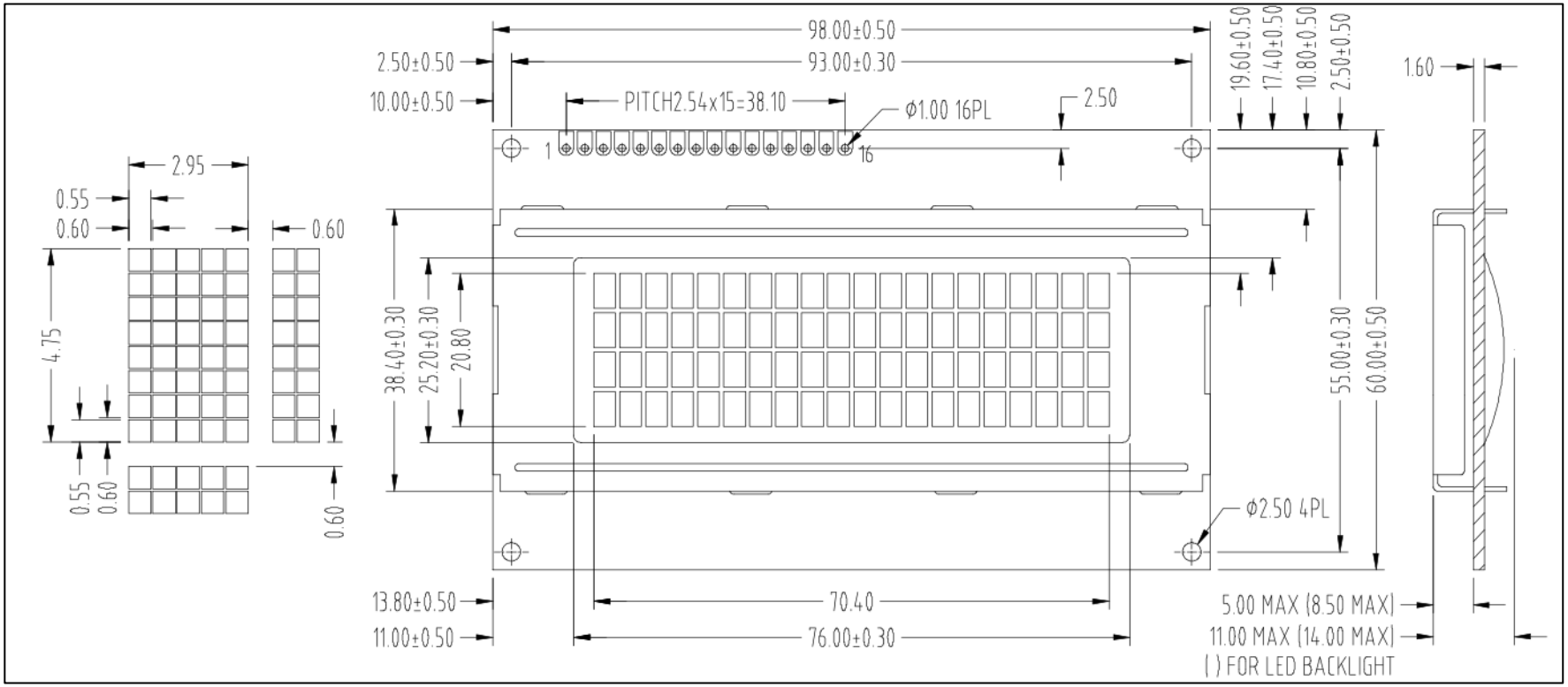
- Interfacing Displaytech 204a LCD
- Interfacing Numeric Keypad
  - Direct interface
  - Using 74C922 keypad encoder

Interfacing Displaytech 204a LCD

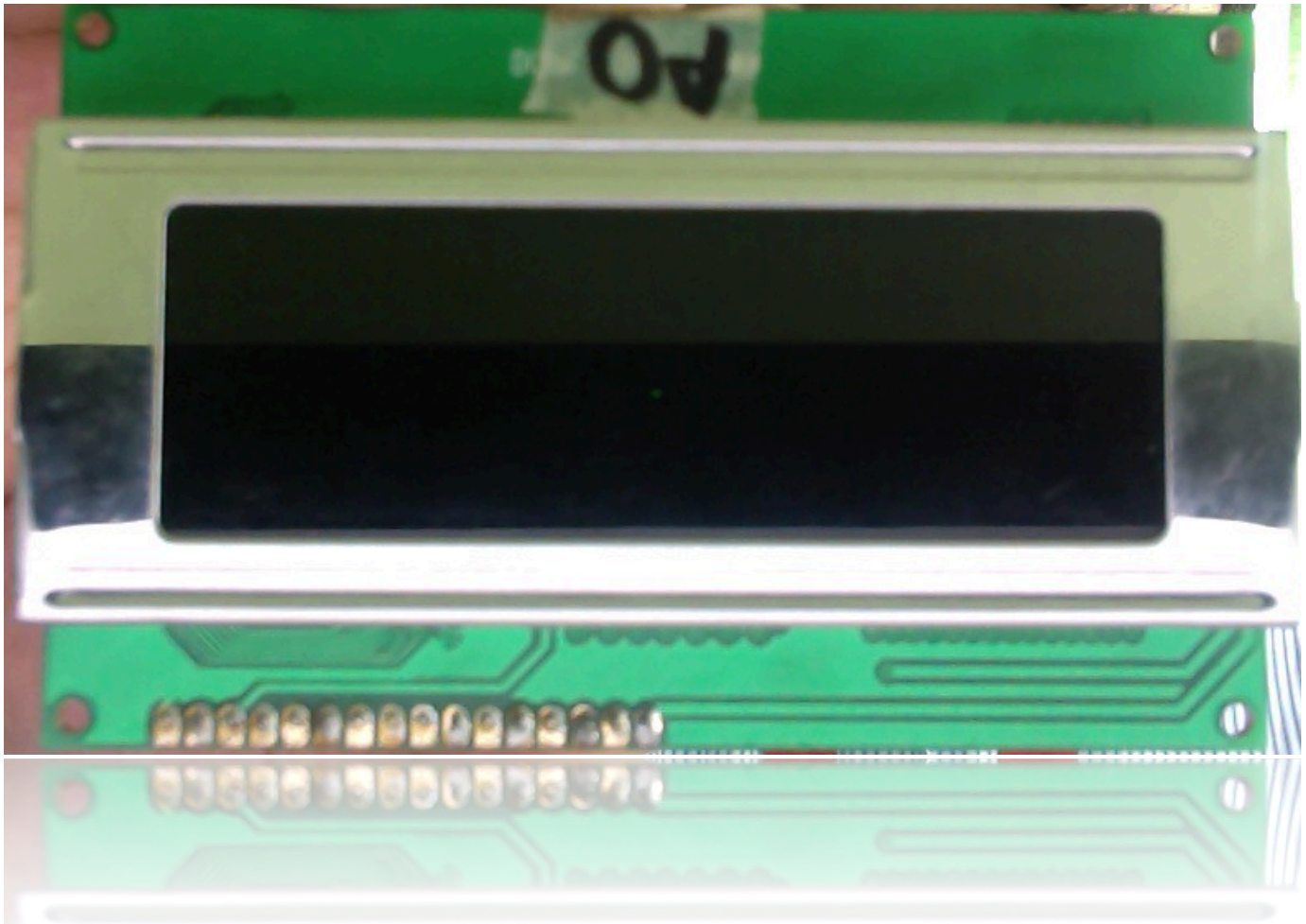
About the Displaytech 204a

Item	Contents	Unit
LCD type	TN / STN / FSTN	---
LCD duty	1/16	---
LCD bias	1/5	---
Viewing direction	6 / 12	o'clock
Module size (W×H×T)	98 × 60 × 11.0 MAX (14.0 MAX W/LED BACKLIGHT)	mm
Viewing area (W×H)	76 × 25.2	mm
Number of characters (characters×lines)	20 × 4	---
Character matrix (W×H)	5 × 8	dots
Character size (W×H)	2.95 × 4.75	mm
Dot size (W×H)	0.55 × 0.55	mm
Dot pitch (W×H)	0.60 × 0.60	mm

External Dimensions

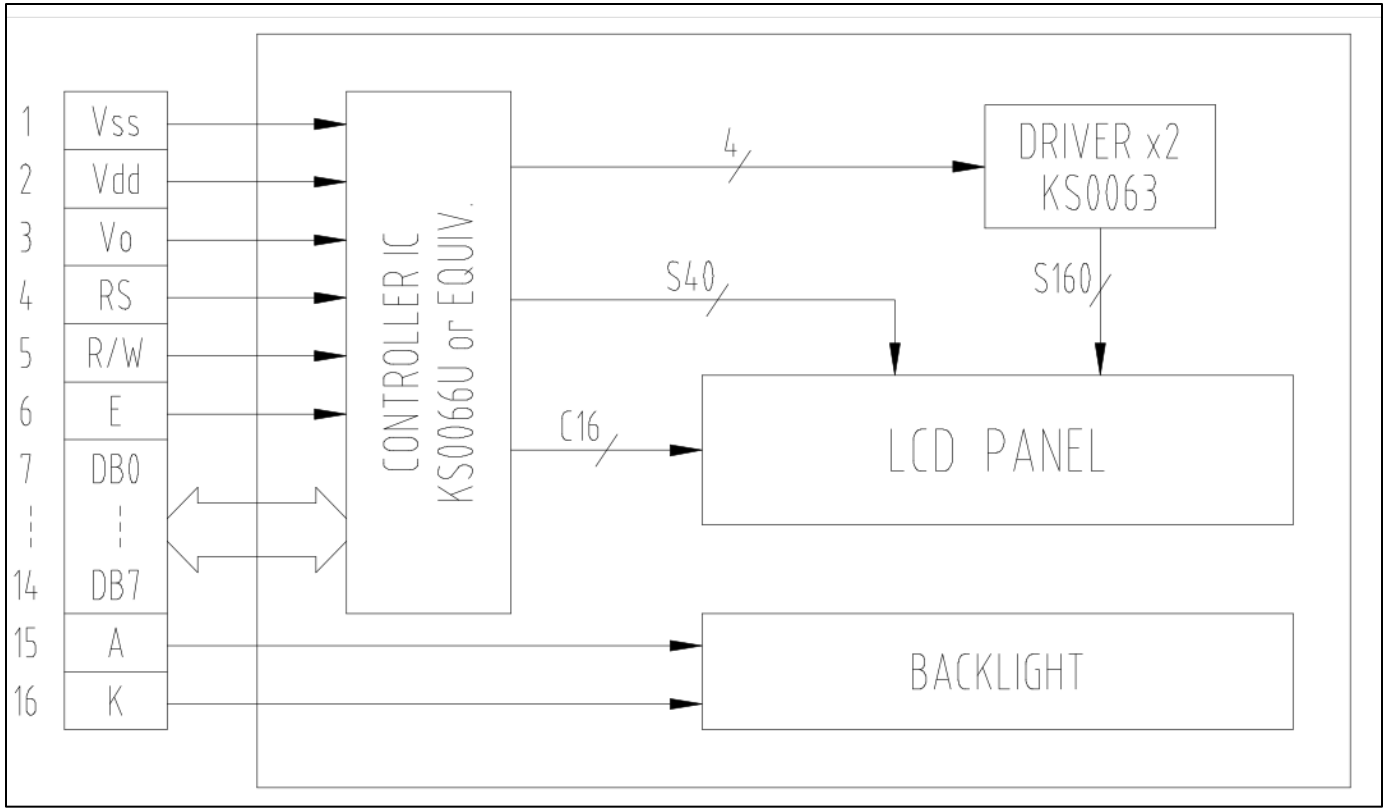


Actual Picture



* The above LCD is a negative display type.

Block Diagram



Interface Signals

Pin NO.	Symbol	Level	Description
1	VSS	0V	Ground
2	VDD	5.0V	Supply voltage for logic
3	VO	---	Input voltage for LCD
4	RS	H/L	H : Data signal, L : Instruction signal
5	R/W	H/L	H : Read mode, L : Write mode
6	E	H, H → L	Enable signal for KS0076
7	DB0	H/L	Data bit 0
8	DB1	H/L	Data bit 1
9	DB2	H/L	Data bit 2
10	DB3	H/L	Data bit 3
11	DB4	H/L	Data bit 4
12	DB5	H/L	Data bit 5
13	DB6	H/L	Data bit 6
14	DB7	H/L	Data bit 7
15	A	---	Back light anode
16	K	---	Back light cathode

Registers

- *Instruction Register (IR)**
  - stores instruction codes such as “clear display” or “shift cursor” and also stores address information for the display data RAM
- *Data Register**
  - is used for temporarily storing data during data transactions with the MCU

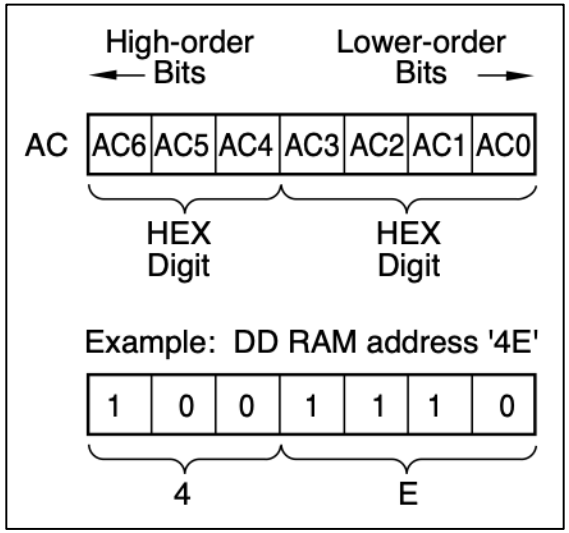
* RS should be logic-0 to access Instruction Register and logic-1 for the Data Register.

Registers

- *Busy Flag (BF)*
  - When the busy flag is set at a logical "1", the LCD unit is executing an internal operation, and no instruction will be accepted
- *Address Counter (AC)*
  - *The address counter generates the address for the display data RAM and character generator RAM*

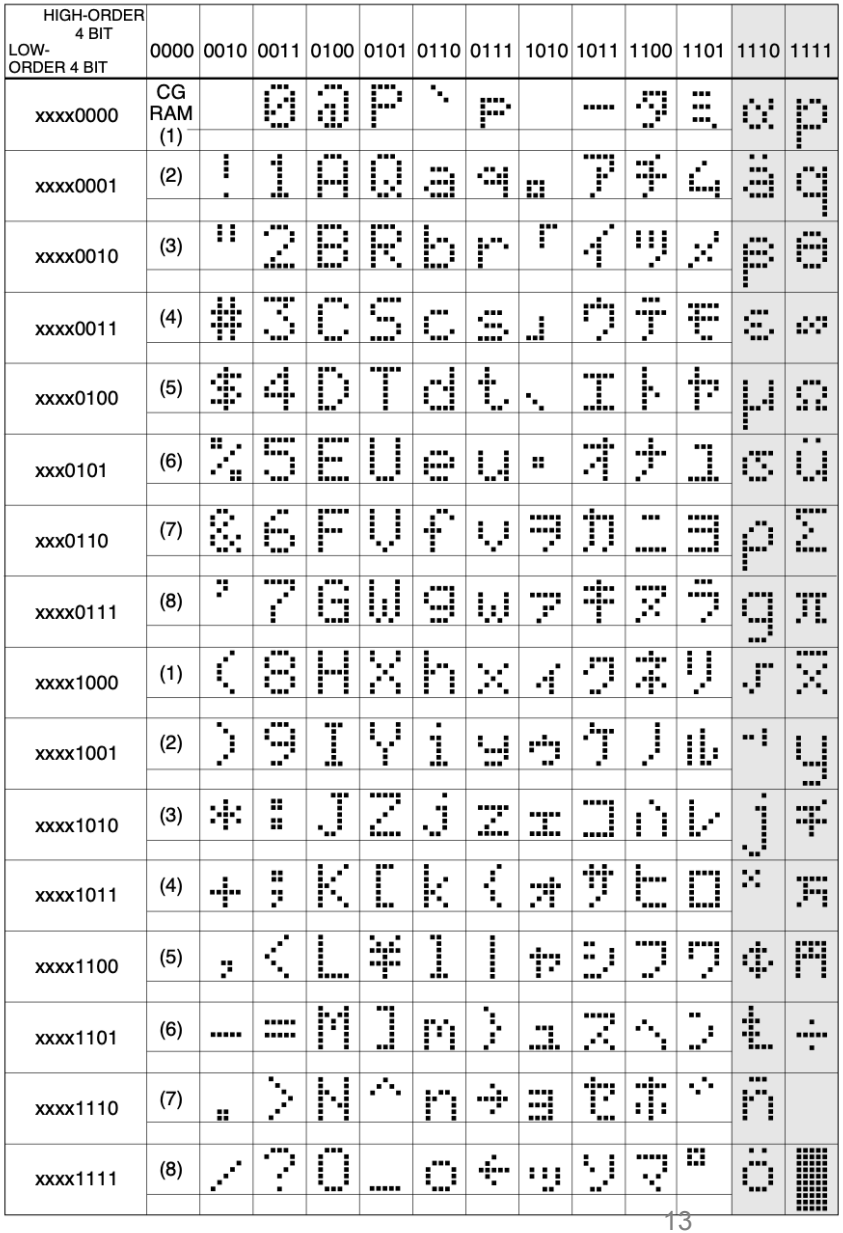
Registers

- *Display Data RAM (DD RAM)*
  - This 80 x 8 bit RAM stores up to 80 8-bit character codes as display data



Registers

- *Character Generator ROM (CG ROM)*
  - This ROM generates a 5 × 7 dot-matrix character pattern for each of 160 different 8-bit character codes
- *Character Generator RAM (CG RAM)*
  - This RAM stores eight arbitrary 5 x 7 dot-matrix character patterns, as programmed by the user



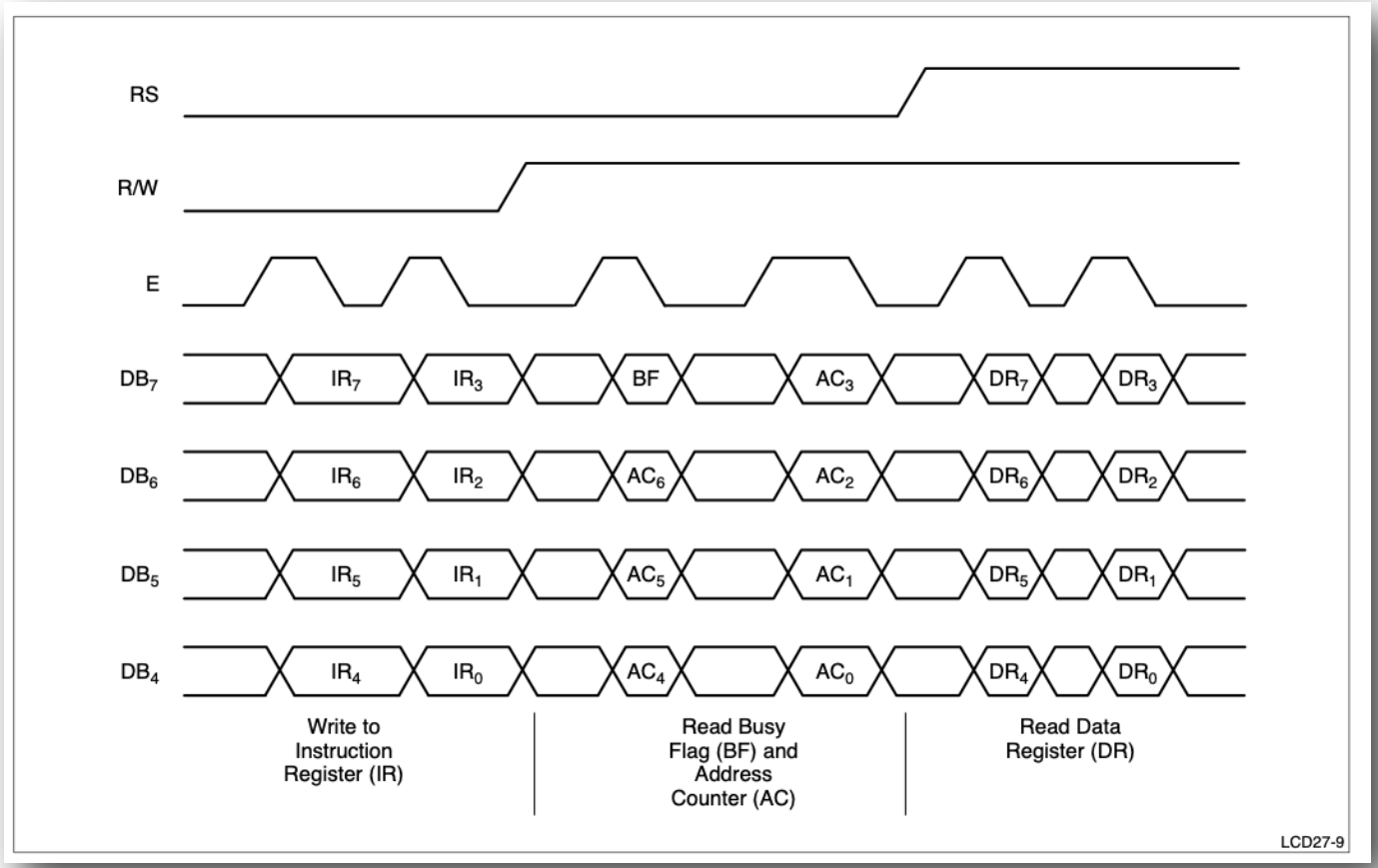
Character Codes

Microprocessor Interface

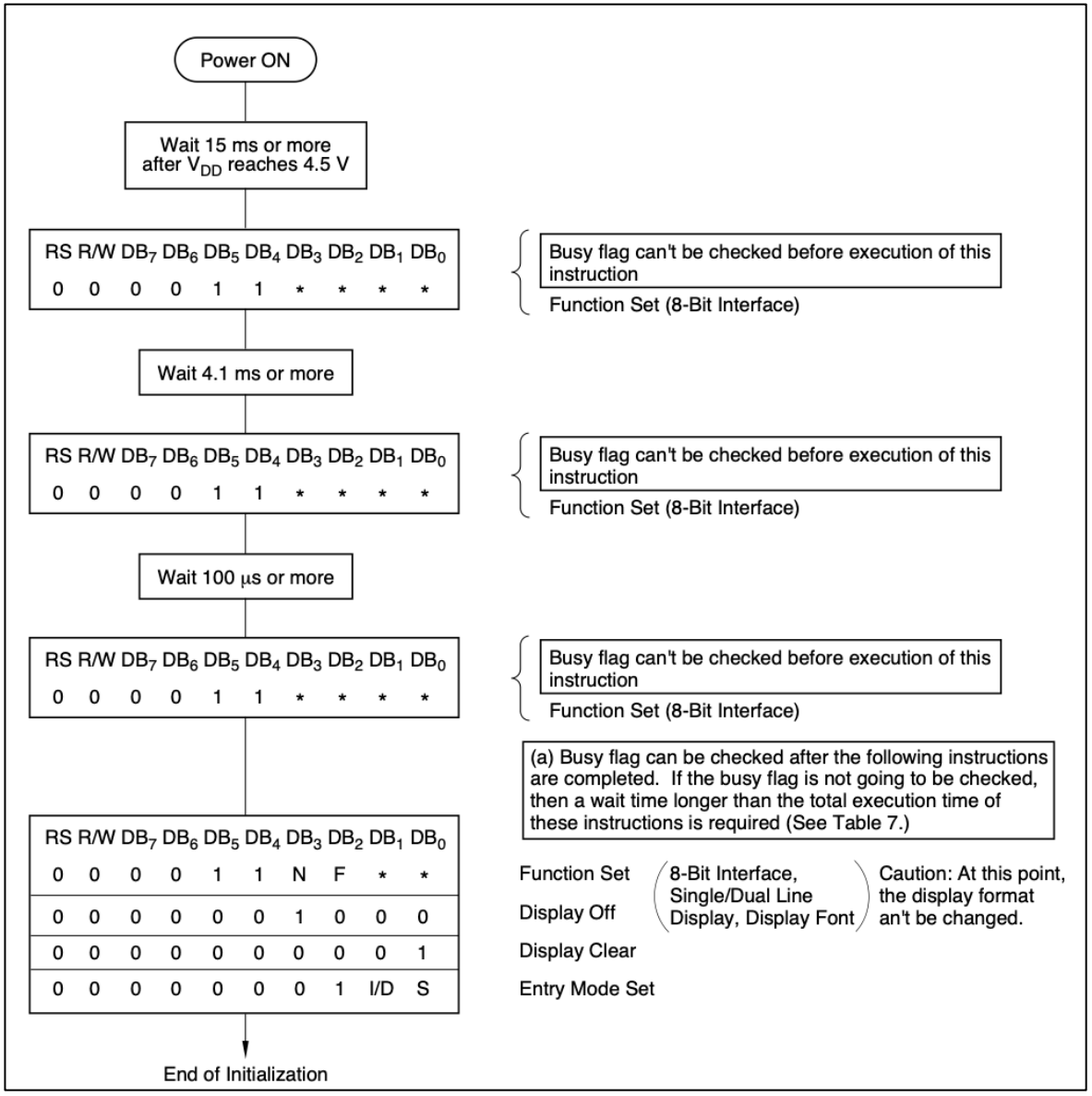
- The LCD unit performs either dual 4-bit or single 8-bit data transfers, allowing the user to interface with either a 4-bit or 8-bit microprocessor
- *4-Bit Microprocessor Interface.*
  - Data lines DB4-DB7 are used for data transfers. Data transactions with the external microprocessor take place in two 4-bit data transfer operations.

Microprocessor Interface

- *8-bit Microprocessor Interface*
  - Each 8-bit piece of data is transferred in a single operation using the entire data bus DB0-DB7.
  - In this chapter, we will be dealing with 8-bit interface only.



Timing Diagram for the 4-bit Data Transfer

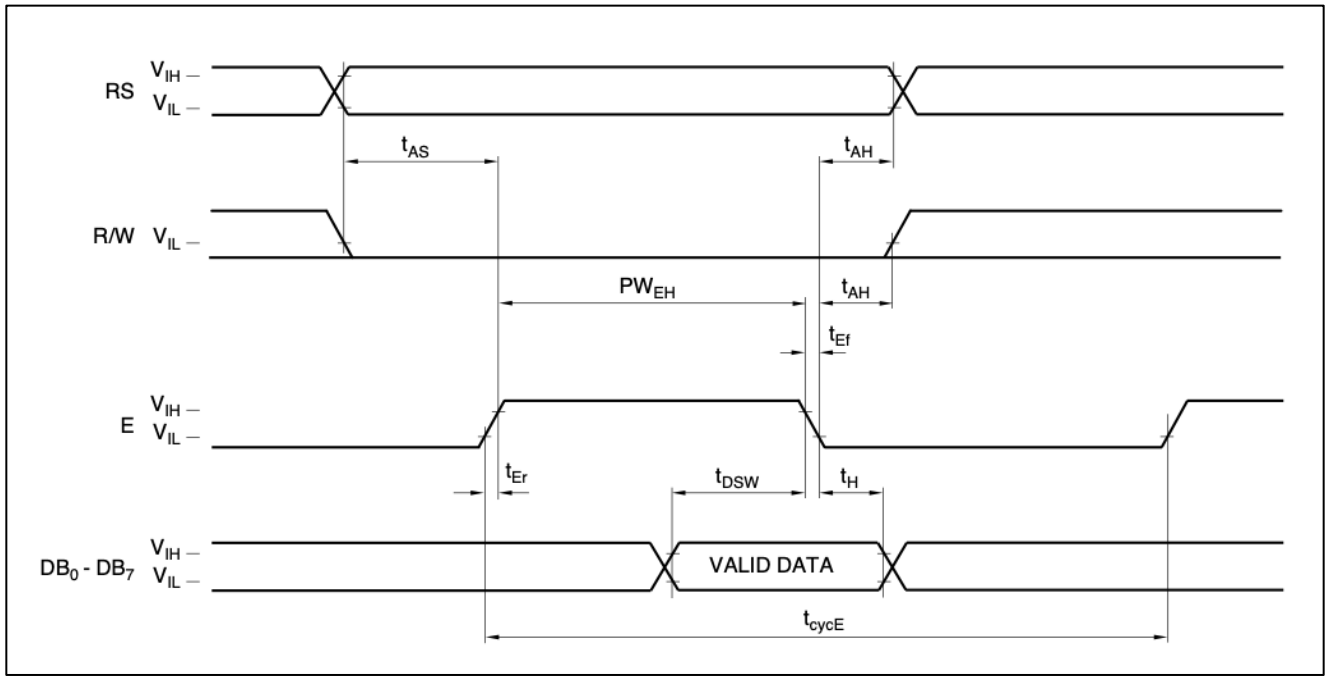


Initialization by Instruction using the 8-bit Interface

Instruction Set

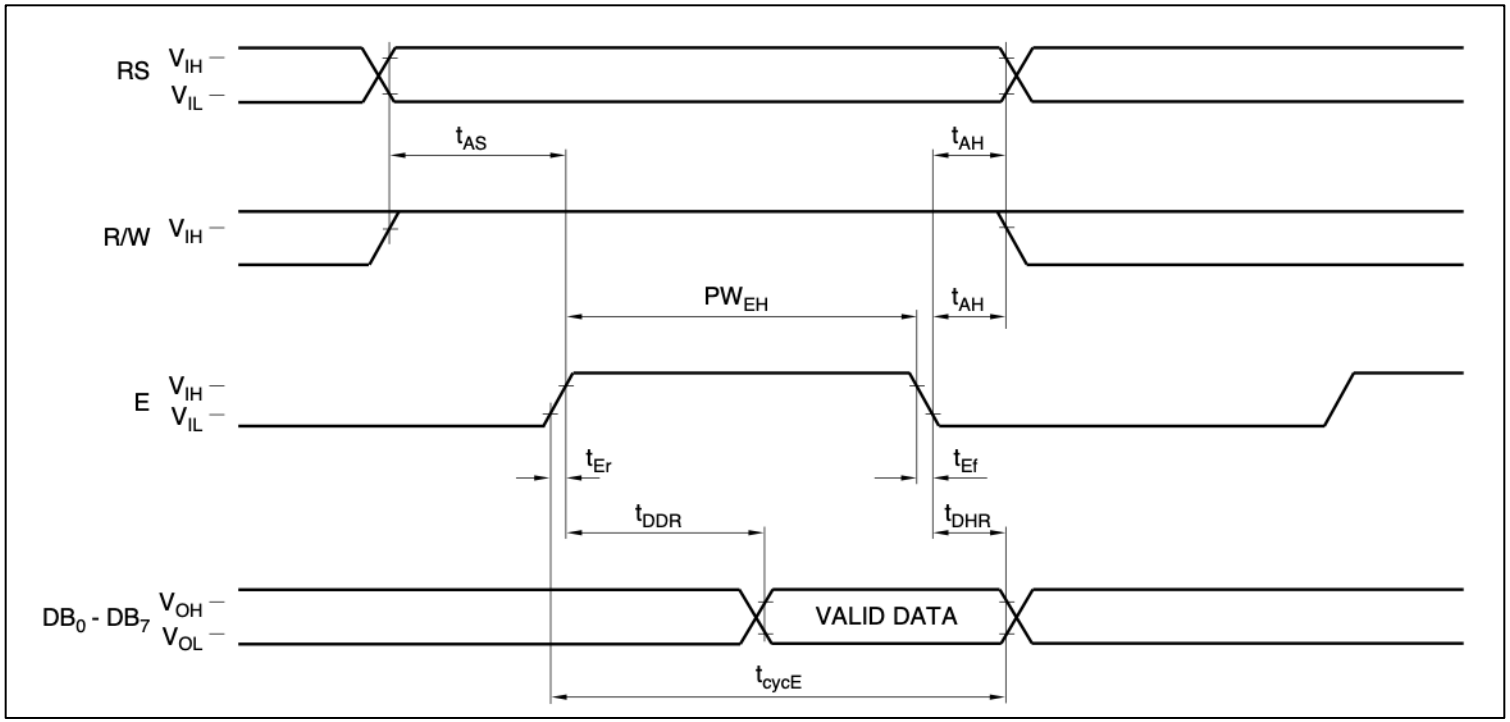
INSTRUCTION	CODE										FUNCTION	EXECUTION TIME (max) (f_{CP} or f_{OSC} = 250 kHz)
	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0		
Display Clear	0	0	0	0	0	0	0	0	0	1	Clear enter display area, restore display from shift, and load address counter with DD RAM address 00_H.	1.64 ms
Display/Cursor Home	0	0	0	0	0	0	0	0	1	*	Restore display from shift and load address counter with DD RAM address 00_H.	1.64 ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Specify cursor advance direction and display shift mode. This operation takes place after each data transfer.	40 μs
Display ON/OFF	0	0	0	0	0	0	1	D	C	B	Specify activation of display (D), cursor (C), and blinking of character at cursor position (B).	40 μs
Display/Cursor Shift	0	0	0	0	0	1	S/C	R/L	*	*	Shift display or move cursor.	40 μs
Function Set	0	0	0	0	1	DL	N	0	*	*	Set interface data length (DL) and number of display lines (N).	40 μs
CG RAM Address Set	0	0	0	1	A_{CG}						Load the address counter with a CG RAM address. Subsequent data is CG RAM data.	40 μs
DD RAM Address Set	0	0	1	A_{DD}							Load the address counter with a DD RAM address. Subsequent data is DD RAM data.	40 μs
Busy Flag/Address Counter Read	0	1	BF	AC							Read busy flag (BF) and contents of address counter (AC).	0 μs
CG RAM/DD RAM Data Write	1	0	Write data								Write data to CG RAM or DD RAM.	40 μs
CG RAM/DD RAM Data Read	1	1	Read data								Read data from CG RAM or DD RAM.	40 μs
	I/D = 1: Increment, I/D = 0: Decrement S = 1: Display Shift On S/C = 1: Shift Display, S/C = 0: Move Cursor R/L = 1: Shift Right, R/L = 0: Shift Left DL = 1: 8-Bit, DL = 0: 4-Bit N = 1: Dual Line, N = 0: Single Line BF = 1: Internal Operation, BF = 0: Ready for Instruction										DD RAM: Display Data RAM CG RAM: Character Generator RAM A_{CG}: Character Generator RAM Address A_{DD}: Display Data RAM Address AC: Address Counter	

Write Operation Timing Characteristics



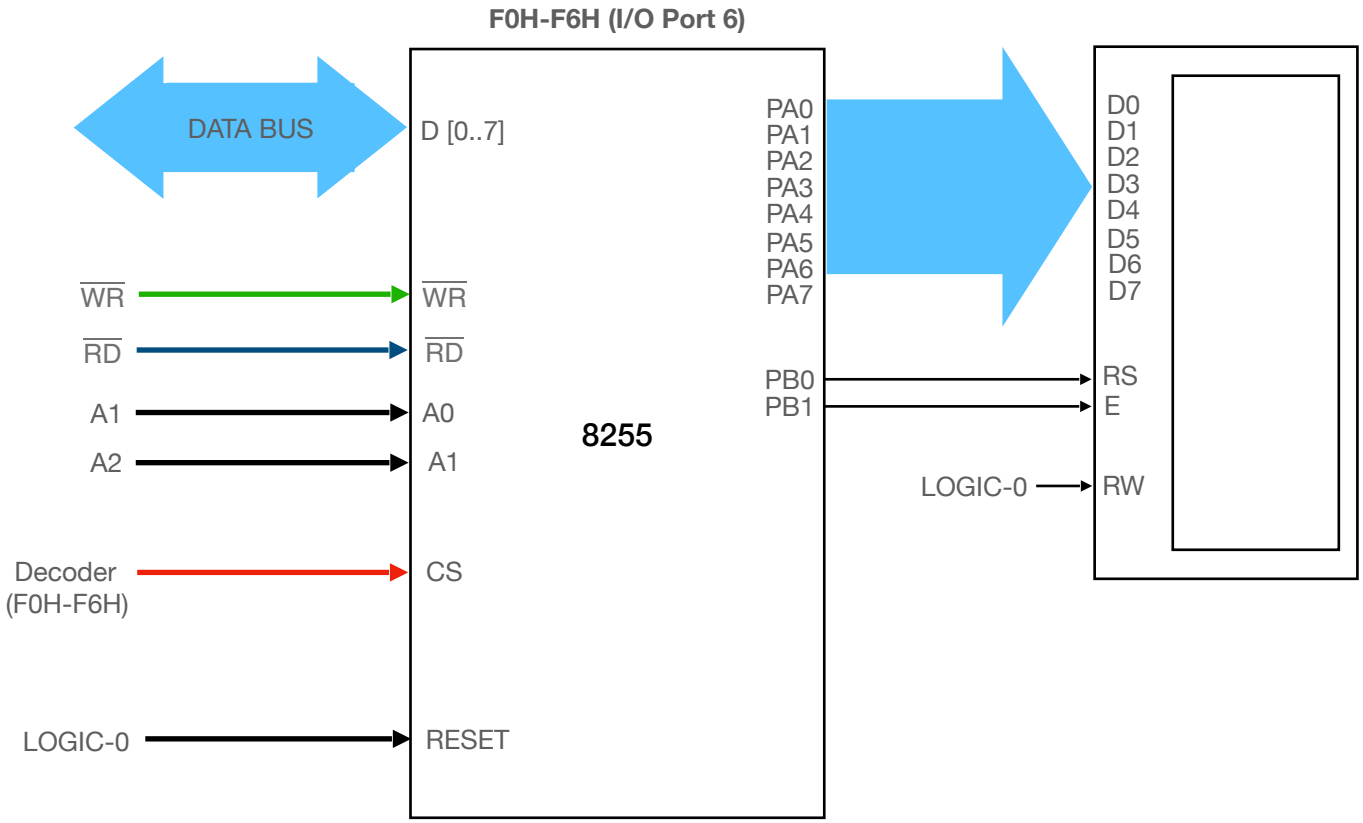
PARAMETER		SYMBOL	VALUE		UNIT
			MIN.	MAX.	
Enable Cycle Time		$t_{CYC}E$	1000	—	ns
Enable Pulse Width	"High" Level	PW_{EH}	450	—	ns
Enable Rise/Fall Time		t_{Er}, t_{Ef}	—	25	ns
Setup Time	RS, R/W-E	t_{AS}	140	—	ns
Address Hold Time		t_{AH}	10	—	ns
Data Setup Time		t_{DSW}	195	—	ns
Data Hold Time		t_H	10	—	ns

Read Operation Timing Characteristics



PARAMETER		SYMBOL	VALUE		UNIT
			MIN.	MAX.	
Enable Cycle Time		$t_{cyc}E$	1000	—	ns
Enable Pulse Width	"High" Level	PW_{EH}	450	—	ns
Enable Rise/Fall Time		t_{Er}, t_{Ef}	—	25	ns
Setup Time	RS, R/W-E	t_{AS}	140	—	ns
Address Hold Time		t_{AH}	10	—	ns
Data Delay Time		t_{DDR}	—	320	ns
Data Hold Time		t_{OHR}	20	—	ns

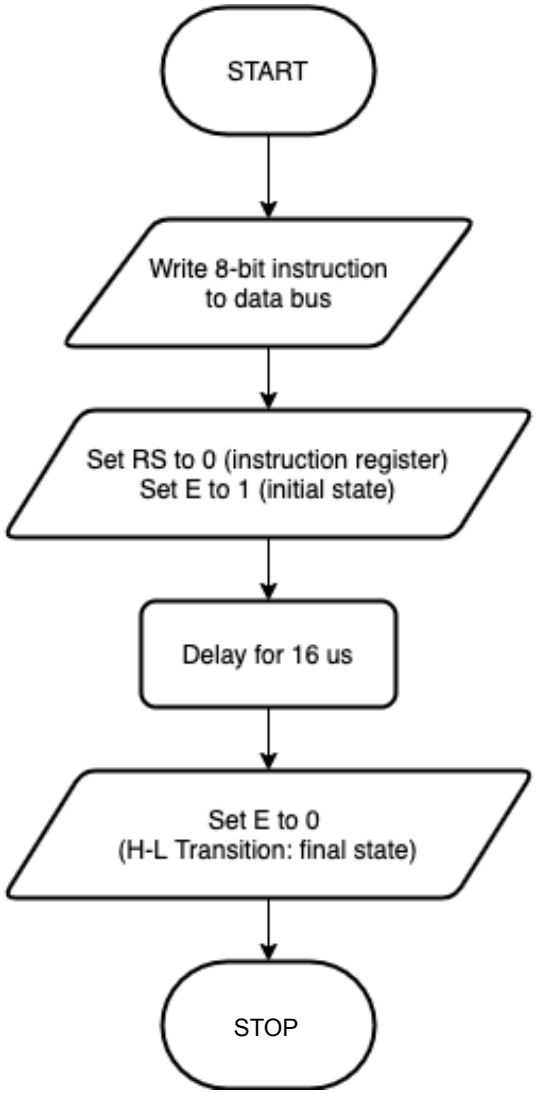
Interfacing via 8255 PPI



Programming

- Step 1: Write a function to send **Instructions** to the LCD
- Step 2: Write a function to *Initialize* the LCD
- Step 3: Write a function to send **Data** to the LCD
- Step 4*: Write a function to display a string to the LCD (optional)

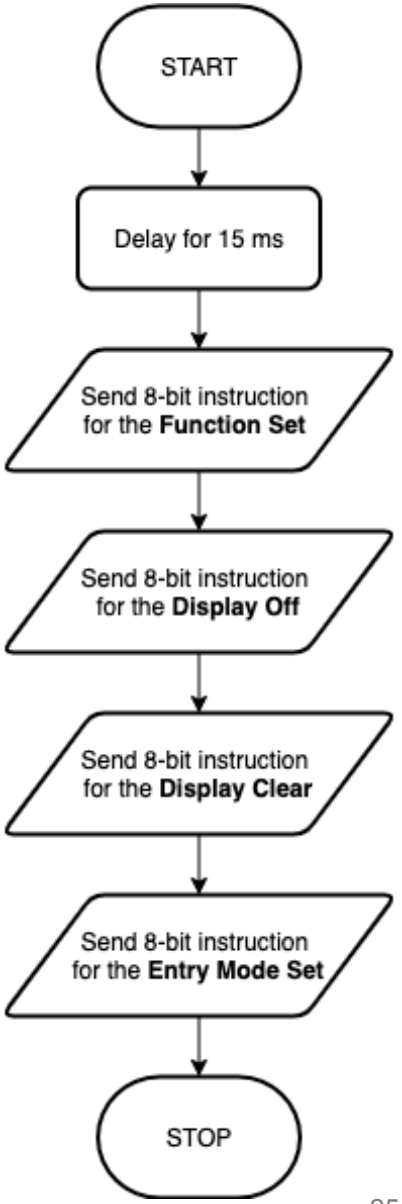
* this procedure will simplify the operation of displaying a string



Flowchart for writing 8-bit instruction to the LCD

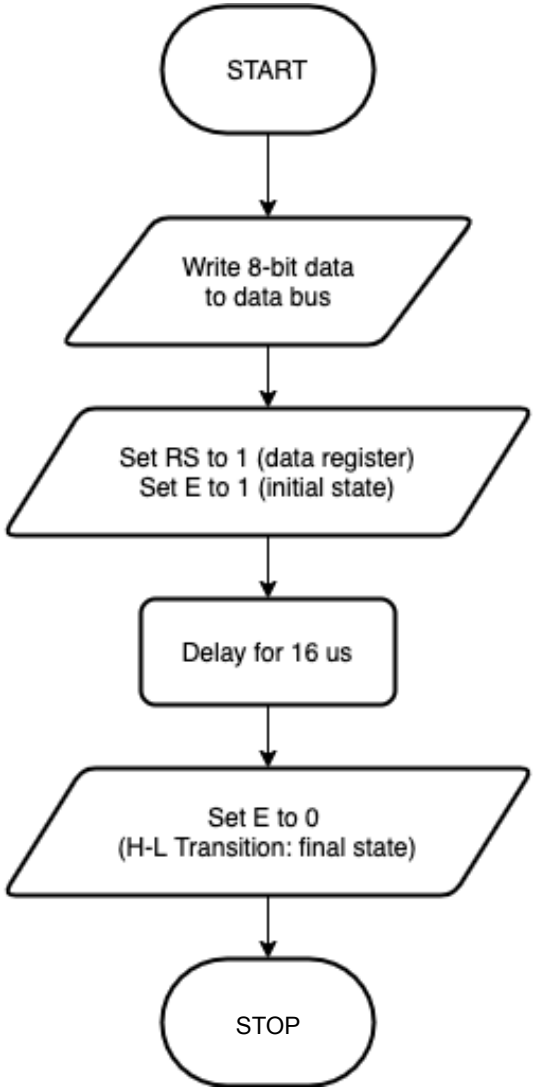
* 16 us is the optimal delay for fast instruction response

```
INST_CTRL:
    PUSH AX          ; preserve value of AL
    MOV DX, PORTA    ; set port of LCD data bus (PORTA)
    OUT DX, AL       ; write data in AL to PORTA
    MOV DX, PORTB    ; set port of LCD control lines (PORTB)
    MOV AL, 02H      ; E=1, RS=0 (access instruction reg)
    OUT DX, AL       ; write data in AL to PORTB
    CALL DELAY_1MS   ; delay for 1 ms
    MOV DX, PORTB    ; set port of LCD control lines (PORTB)
    MOV AL, 00H      ; E=0, RS=0
    OUT DX, AL       ; write data in AL to PORTB
    POP AX           ; restore value of AL
    RET
```



Flowchart for initializing the LCD

```
INIT_LCD:
    MOV AL, 38H     ; 8-bit interface, dual-line display
    CALL INST_CTRL  ; write instruction to LCD
    MOV AL, 08H     ; display off, cursor off, blink off
    CALL INST_CTRL  ; write instruction to LCD
    MOV AL, 01H     ; clear display
    CALL INST_CTRL  ; write instruction to LCD
    MOV AL, 06H     ; increment cursor, display shift off
    CALL INST_CTRL  ; write instruction to LCD
    MOV AL, 0CH     ; display on, cursor off, blink off
    CALL INST_CTRL  ; write instruction to LCD
    RET
```



Flowchart in writing 8-bit data to the LCD

* 16 us is the optimal delay for fast display refresh

```
DATA_CTRL:
    PUSH AX          ; preserve value of AL
    MOV DX, PORTA    ; set port of LCD data bus (PORTA)
    OUT DX, AL       ; write data in AL to PORTA
    MOV DX, PORTB    ; set port of LCD control lines (PORTB)
    MOV AL, 03H      ; E=1, RS=1 (access data register)
    OUT DX, AL       ; write data in AL to PORTB
    CALL DELAY_1MS   ; delay for 1 ms
    MOV DX, PORTB    ; set port of LCD control lines (PORTB)
    MOV AL, 01H      ; E=0, RS=1
    OUT DX, AL       ; write data in AL to PORTB
    POP AX           ; restore value of AL
    RET
```

* Similar code to INST_CTRL except that RS should be logic-1.

Example

- Display character ‘0’ at current cursor position:

```
MOV AL, 30H     ; set ASCII code of character
CALL DATA_CTRL  ; display the character
```

- Move the cursor at position at third column, first line:

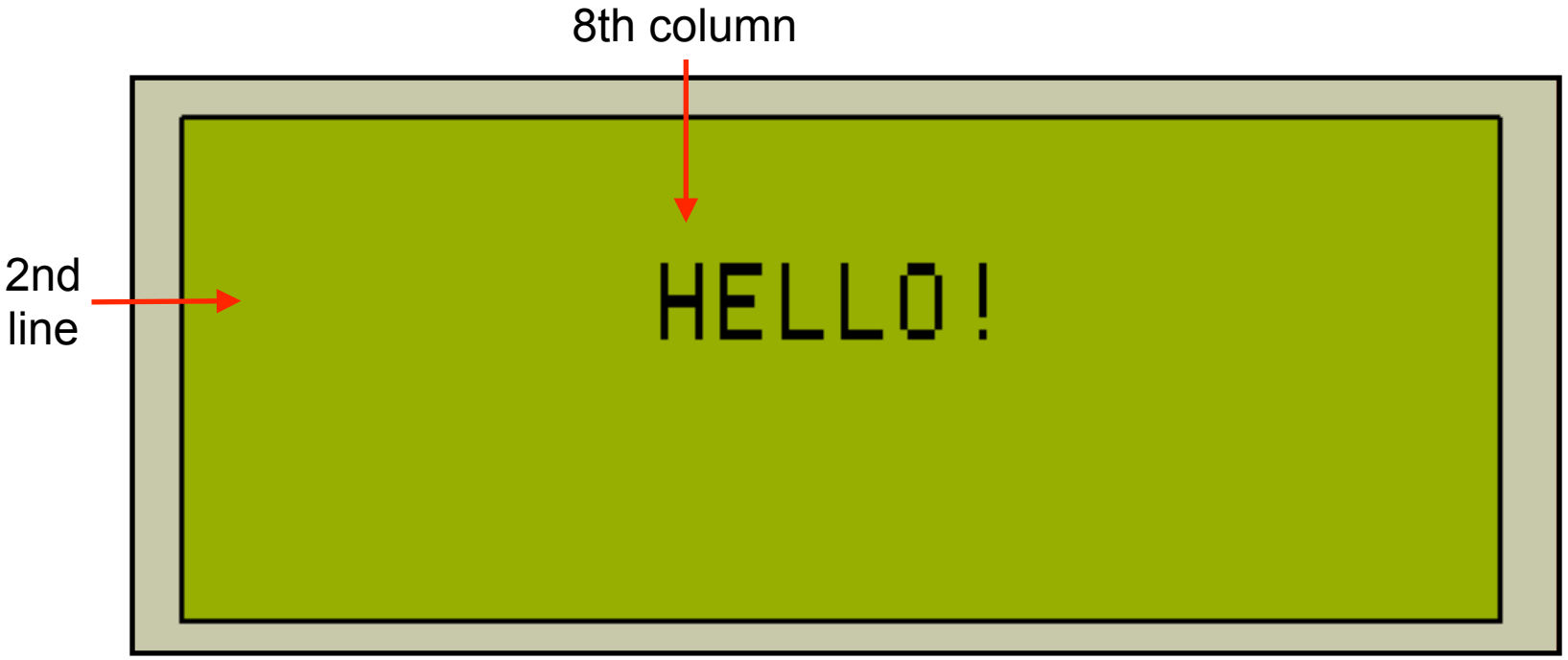
```
MOV AL, 0C2H    ; set address of third column, first line*
CALL INST_CTRL  ; send instruction to move cursor to position
```

* see character display address

Example

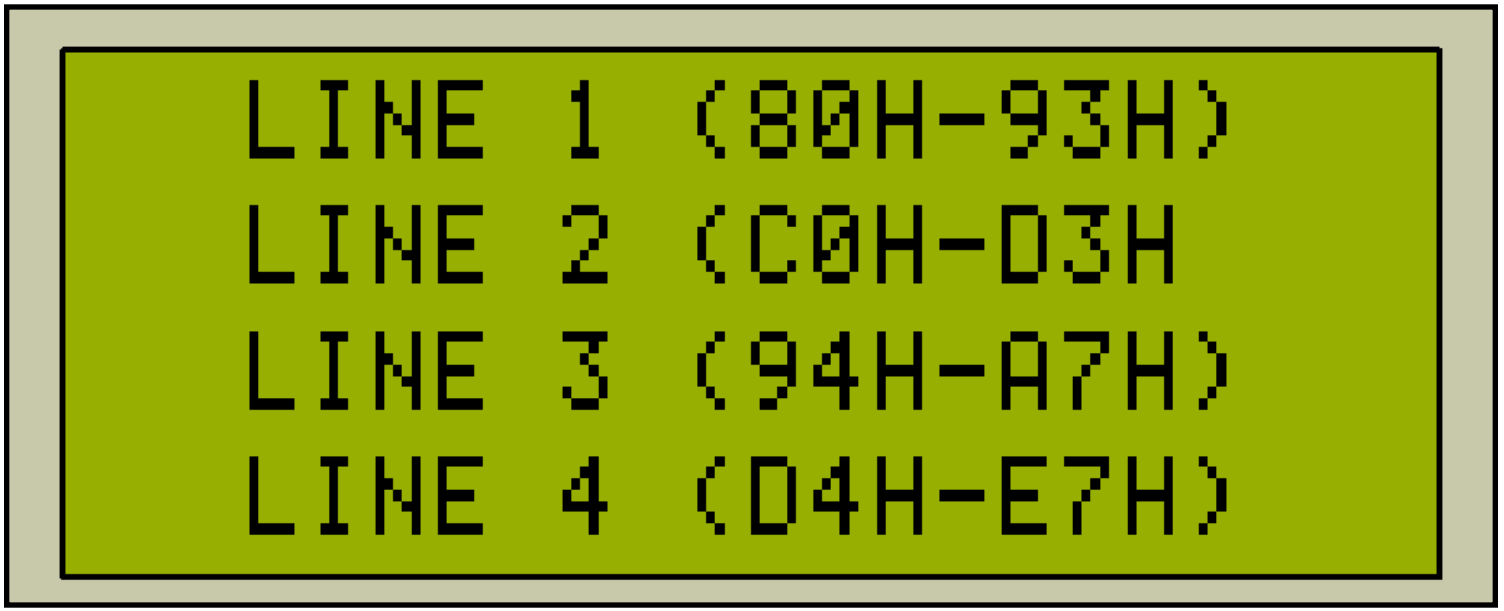
- Display string “HELLO!” at the middle of second line:

```
MOV AL, 0C7H   ; move cursor to 8th column of 2nd line
CALL INST_CTRL ; send instruction to LCD
MOV AL, 'H'    ; set ASCII code of character 'H'
CALL DATA_CTRL ; display the character
MOV AL, 'E'    ; set ASCII code of character 'E'
CALL DATA_CTRL ; display the character
MOV AL, 'L'    ; set ASCII code of character 'L'
CALL DATA_CTRL ; display the character
MOV AL, 'L'    ; set ASCII code of character 'L'
CALL DATA_CTRL ; display the character
MOV AL, 'O'    ; set ASCII code of character 'O'
CALL DATA_CTRL ; display the character
MOV AL, '!'    ; set ASCII code of character '!'
CALL DATA_CTRL ; display the character
```



String displayed beginning at 7th column of the second line.

Character Display Addresses



Line 1 address continue to Line 3 while Line 2 continues at Line 4.

Demonstration #1

- The LCD will be interfaced to the 8086 via the 8255 at address F0H-F6H (see page 21).
- LCD configuration: 8-bit, cursor off, blink off, increment cursor, display shift off
- LCD is initialized first, then “HELLO WORLD!” will be displayed at the middle of the 3rd line.

```
DATA SEGMENT
      PORTA EQU 0F0H          ; address to access PORTA
      PORTB EQU 0F2H          ; address to access PORTB
      PORTC EQU 0F4H          ; address to access PORTC
      PORT_CON EQU 0F6H       ; address to access Command Register
DATA ENDS

CODE SEGMENT PUBLIC 'CODE'
           ASSUME CS:CODE
START:
      ; Program 8255 (PORTA & PORTB as output, PORTC as input)
      MOV DX, PORT_CON        ; set port to Command Register
      MOV AL, 89H             ; set command byte
      OUT DX, AL              ; send data in AL to Command Register

      CALL INIT_LCD           ; initialize LCD

      MOV AL, 98H             ; move cursor to 3rd line, 5th column
      CALL INST_CTRL          ; send instruction to LCD

      ; display string "HELLO WORLD!" at current cursor position
      MOV AL, 'H'     ; set ASCII code of character 'H'
      CALL DATA_CTRL ; display the character
      MOV AL, 'E'     ; set ASCII code of character 'E'
      CALL DATA_CTRL ; display the character
      MOV AL, 'L'     ; set ASCII code of character 'L'
      CALL DATA_CTRL ; display the character
      MOV AL, 'L'     ; set ASCII code of character 'L'
      CALL DATA_CTRL ; display the character
      MOV AL, 'O'     ; set ASCII code of character 'O'
      CALL DATA_CTRL ; display the character
      MOV AL, ' '     ; set ASCII code of character ' '
      CALL DATA_CTRL ; display the character
      MOV AL, 'W'     ; set ASCII code of character 'W'
      CALL DATA_CTRL ; display the character
      MOV AL, 'O'     ; set ASCII code of character 'O'
      CALL DATA_CTRL ; display the character
      MOV AL, 'R'     ; set ASCII code of character 'R'
      CALL DATA_CTRL ; display the character
```

```
      MOV AL, 'L'     ; set ASCII code of character 'L'
      CALL DATA_CTRL ; display the character
      MOV AL, 'D'     ; set ASCII code of character 'D'
      CALL DATA_CTRL ; display the character
      MOV AL, '!'     ; set ASCII code of character '!'
      CALL DATA_CTRL ; display the character

ENDLESS:
      JMP ENDLESS

      ; this function will send an 8-bit instruction to the LCD
INST_CTRL:
      PUSH AX         ; preserve value of AL
      MOV DX, PORTA   ; set port of LCD data bus (PORTA)
      OUT DX, AL      ; write data in AL to PORTA
      MOV DX, PORTB   ; set port of LCD control lines (PORTB)
      MOV AL, 02H     ; E=1, RS=0 (access instruction reg)
      OUT DX, AL      ; write data in AL to PORTB
      CALL DELAY_1MS ; delay for 1 ms
      MOV DX, PORTB   ; set port of LCD control lines (PORTB)
      MOV AL, 00H     ; E=0, RS=0
      OUT DX, AL      ; write data in AL to PORTB
      POP AX          ; restore value of AL
      RET

      ; this function will send an 8-bit data (character) to the LCD
DATA_CTRL:
      PUSH AX         ; preserve value of AL
      MOV DX, PORTA   ; set port of LCD data bus (PORTA)
      OUT DX, AL      ; write data in AL to PORTA
      MOV DX, PORTB   ; set port of LCD control lines (PORTB)
      MOV AL, 03H     ; E=1, RS=1 (access data reg)
      OUT DX, AL      ; write data in AL to PORTB
      CALL DELAY_1MS ; delay for 1 ms
      MOV DX, PORTB   ; set port of LCD control lines (PORTB)
      MOV AL, 01H     ; E=0, RS=1
      OUT DX, AL      ; write data in AL to PORTB
      POP AX          ; restore value of AL
      RET
```

```
      ; this function will initialize the LCD
INIT_LCD:
      MOV AL, 38H    ; 8-bit interface, dual-line display
      CALL INST_CTRL ; write instruction to LCD
      MOV AL, 08H    ; display off, cursor off, blink off
      CALL INST_CTRL ; write instruction to LCD
      MOV AL, 01H    ; clear display
      CALL INST_CTRL ; write instruction to LCD
      MOV AL, 06H    ; increment cursor, display shift off
      CALL INST_CTRL ; write instruction to LCD
      MOV AL, 0CH    ; display on, cursor off, blink off
      CALL INST_CTRL ; write instruction to LCD
      RET

DELAY_1MS:
      MOV BX, 02CAH
L1:
      NOP
      JNZ
      L1
      RET

CODE ENDS
END START
```

Questions?

Interfacing Numeric Keypad

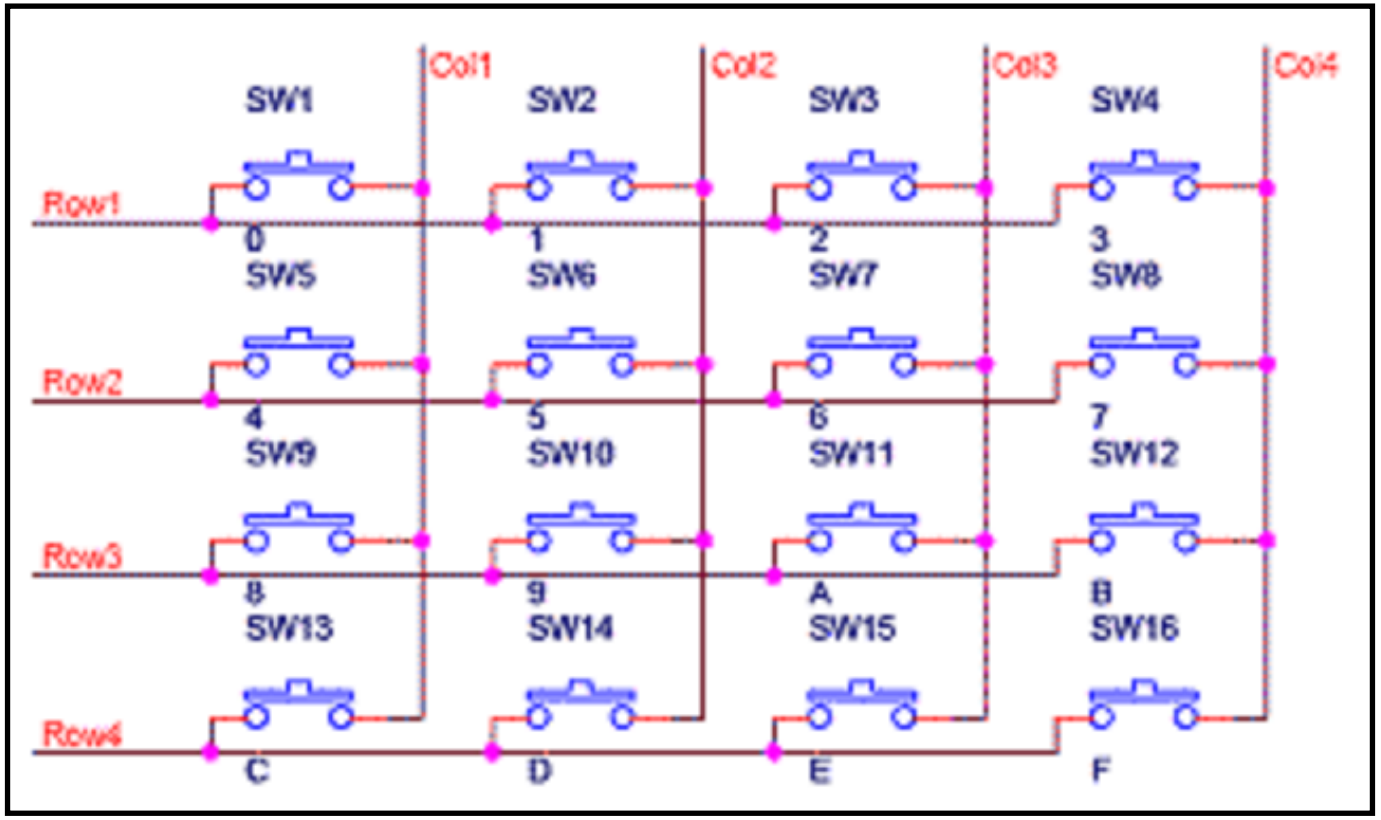
Numeric Keypad

- Most common input device in embedded systems appliances
- Follows the telephony standard layout (3x4 or 4x4)
- Uses array of pushbutton switches with common columns (X) and rows (Y); eliminates individual control of all switches
- Can be interfaced with encoder integrated circuit

4x4

3x4

Schematic Diagram

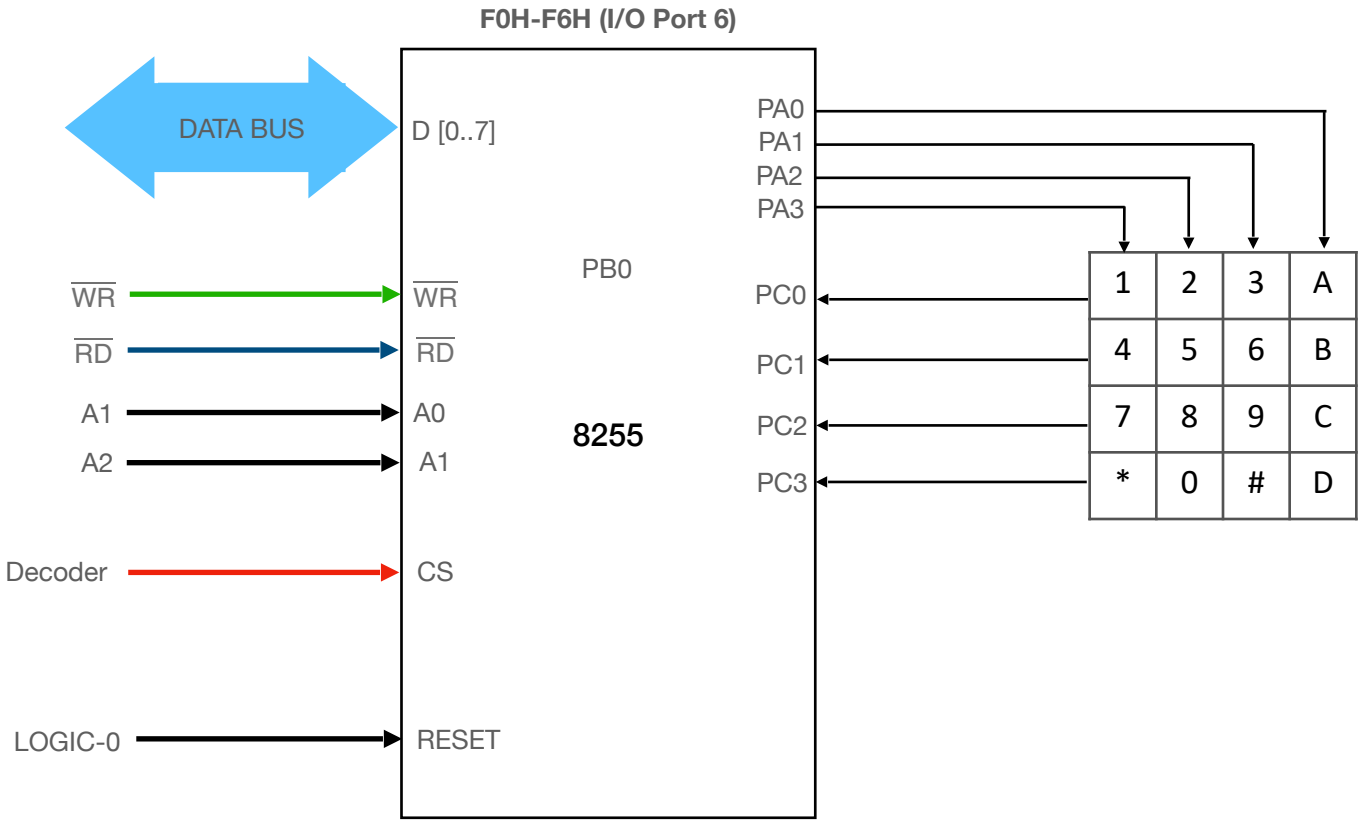


4x4 layout

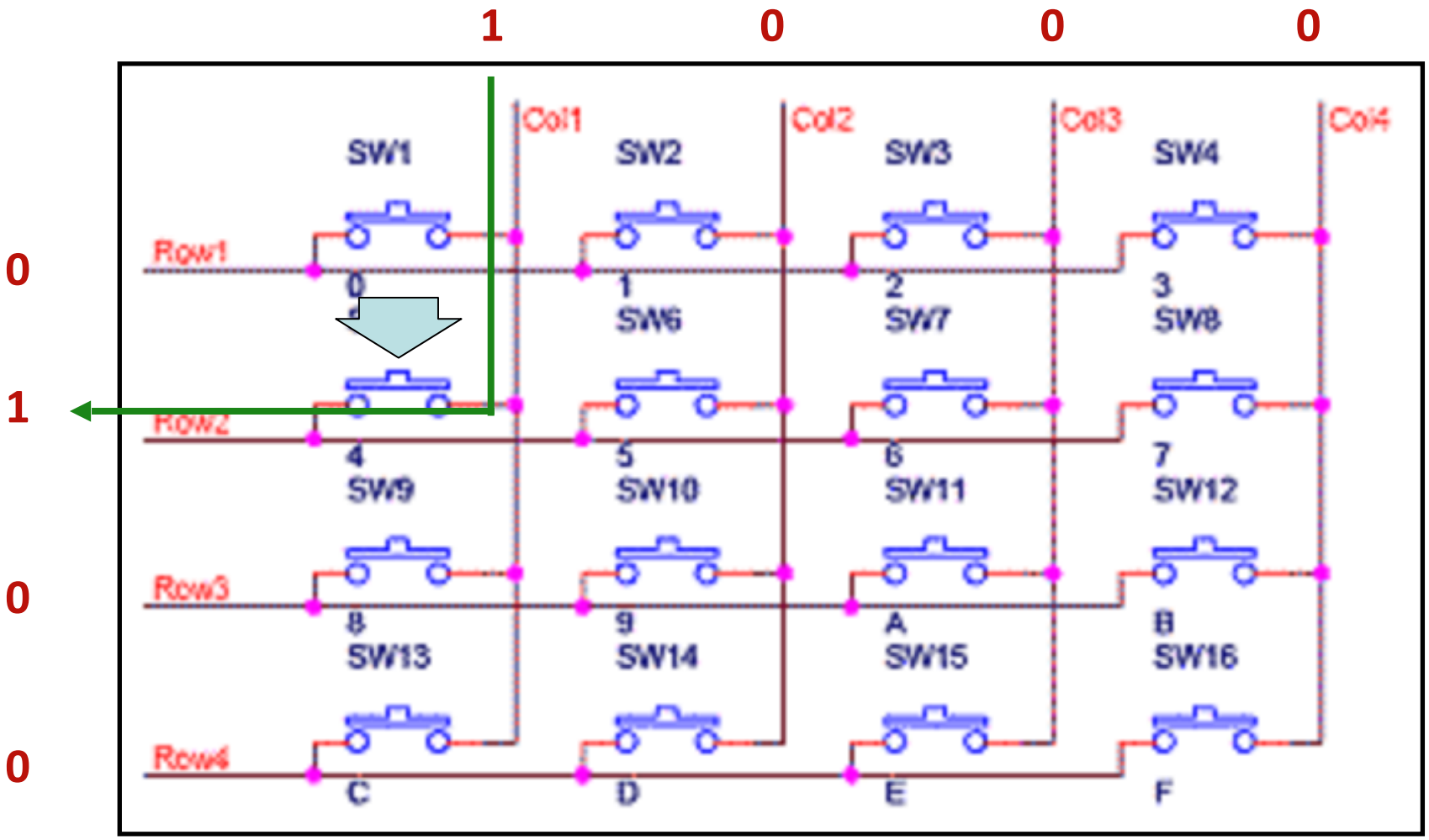
How to Control?

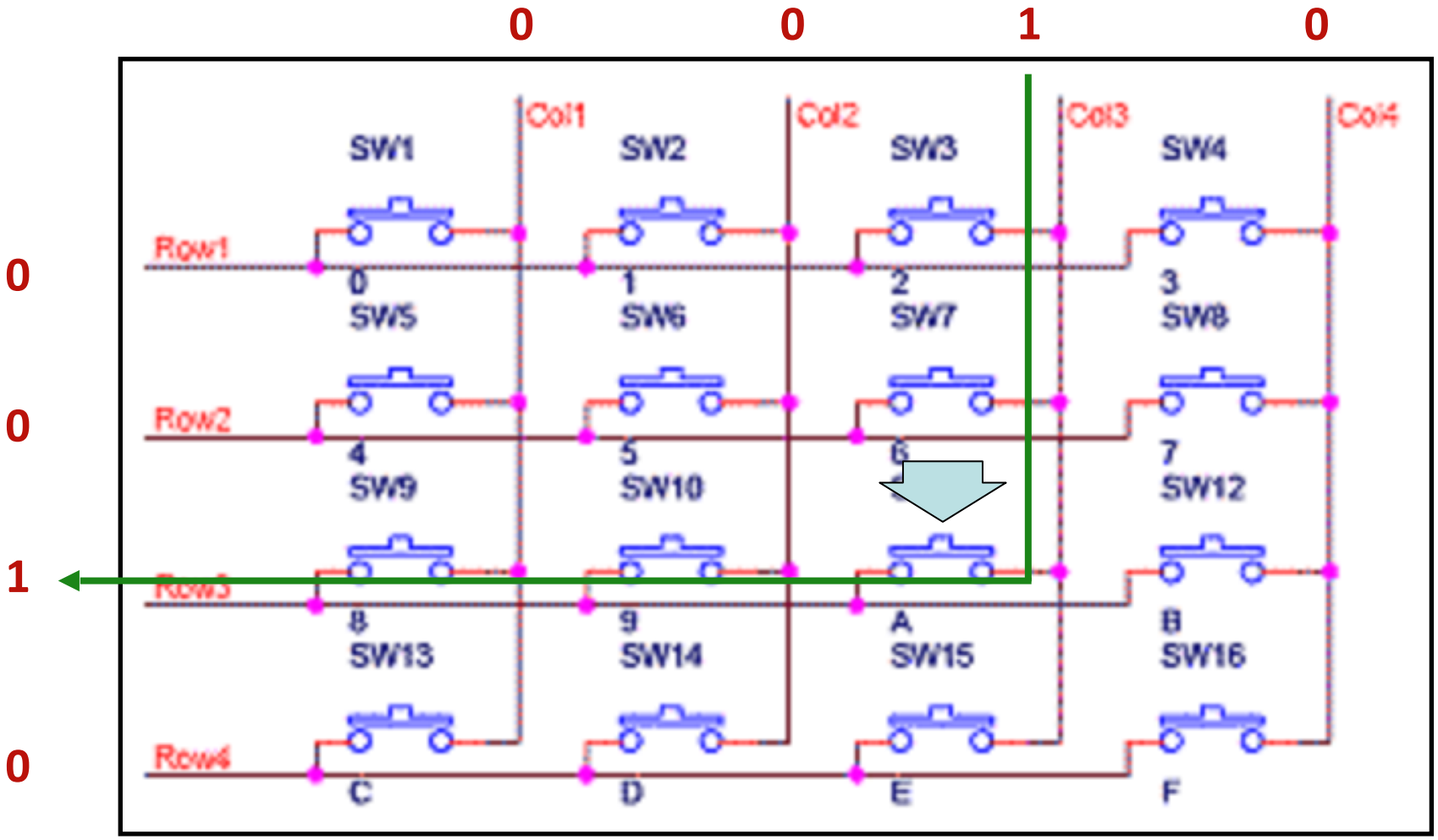
- Supply one of the columns for example Column 1 (X1) with logic-1 while the rest is logic-0.
- If the one of the buttons in that column is pressed, current will flow to the corresponding row.
- Therefore, the column and row data will now be used to identify the specific key pressed.
- Move to the next column and read the row data.

Interfacing via 8255 PPI



* PORTA is an output port while PORTB is input


1
0
0
0
0
1
0
0
Col1
Col2
Col3
Col4
SW1
SW2
SW3
SW4
Row1
0
1
2
3
SW6
SW7
SW8
Row2
4
5
6
7
SW9
SW10
SW11
SW12
Row3
8
9
A
B
SW13
SW14
SW15
SW16
Row4
C
D
E
F

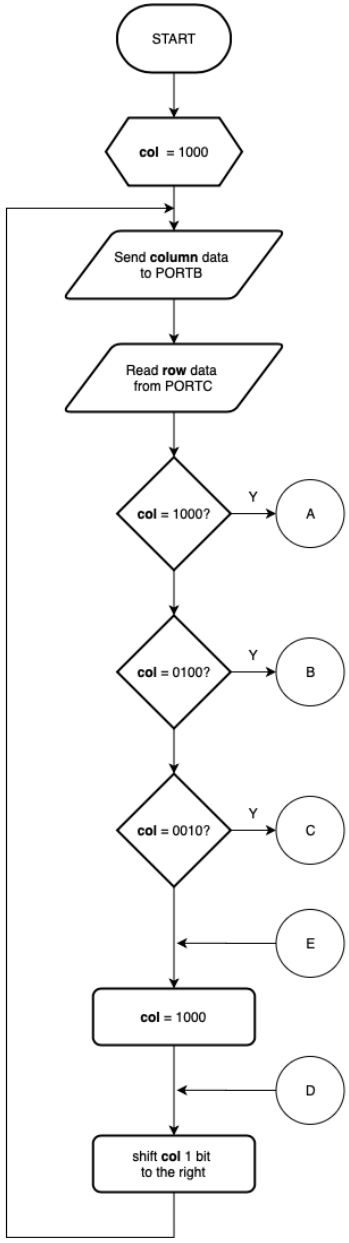

0
0
1
0
Col1
Col2
Col3
Col4
SW1
SW2
SW3
SW4
Row1
0
1
2
3
SW5
SW6
SW7
SW8
0
Row2
4
5
6
7
SW9
SW10
SW12
1
8
9
A
B
SW13
SW14
SW15
SW16
0
Row4
C
D
E
F

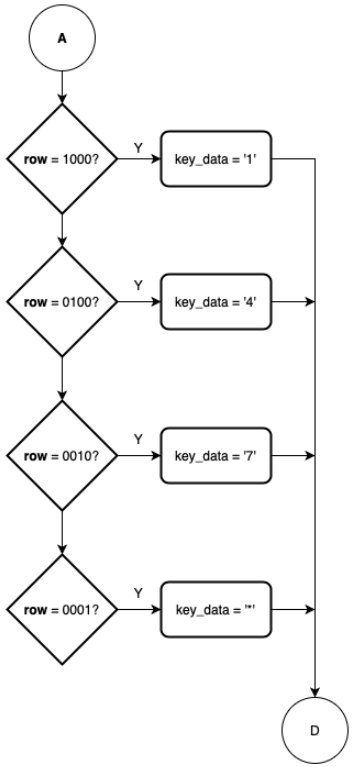
Row and Column Data

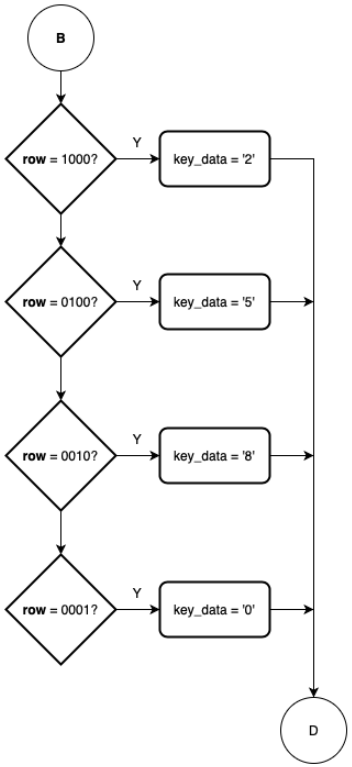
Key	SW No.	Column	Row
1	SW1	1000b	1000b
4	SW5		0100b
7	SW9		0010b
*	SW13		0001b
2	SW2	0100b	1000b
5	SW6		0100b
8	SW10		0010b
0	SW14		0001b
3	SW3	0010b	1000b
6	SW7		0100b
9	SW11		0010b
#	SW15		0001b
A	SW4	0001b	1000b
B	SW8		0100b
C	SW12		0010b
D	SW16		0001b

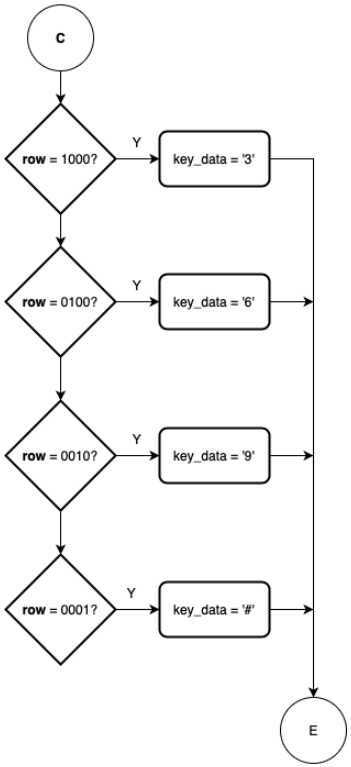
Flowchart in Controlling a 3x4 Numeric Keypad

(4th column is not read)







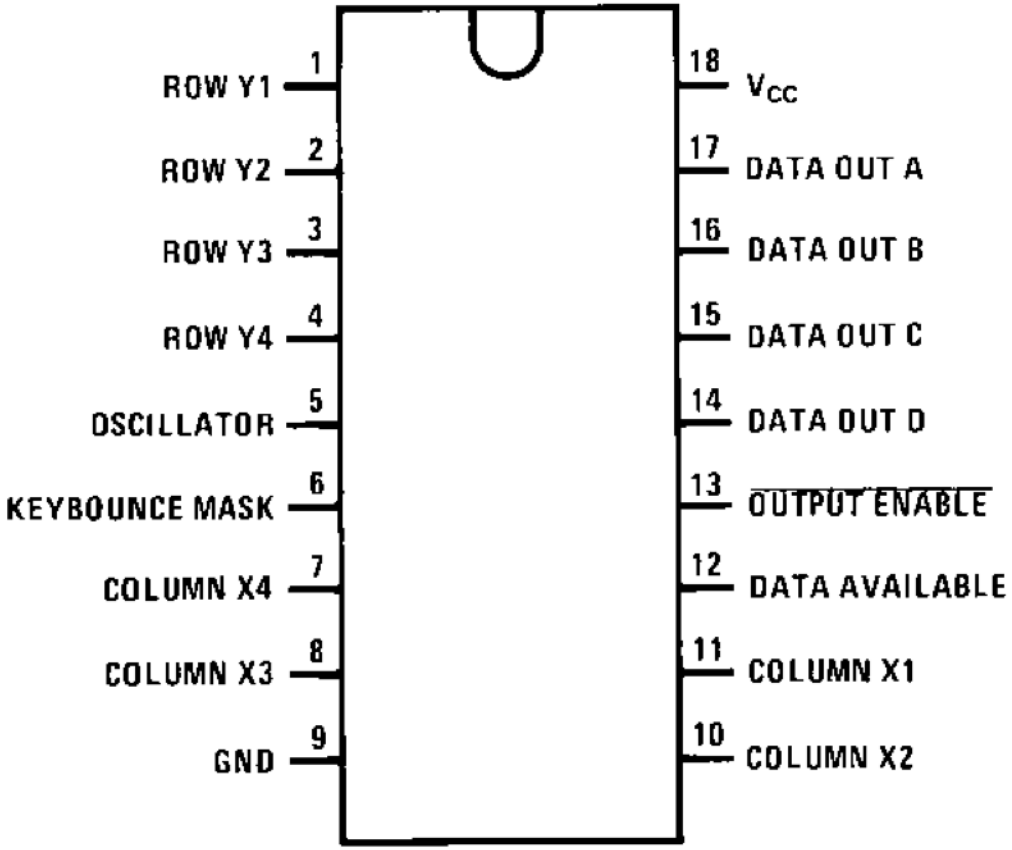


Control Using 74C922

- 74C922* is a CMOS key encoder that encode an array of SPST (**s**ingle-**p**ole, **s**ingle-**t**hrow) switches.
- Keyboard (keypad) scanning is implemented automatically by the IC.
- It has an internal debounce circuitry
- Data Available (DAVBL) output returns a logic-1 when any of the key is pressed.

* Datasheet can be downloaded here.

Pin Assignment for DIP



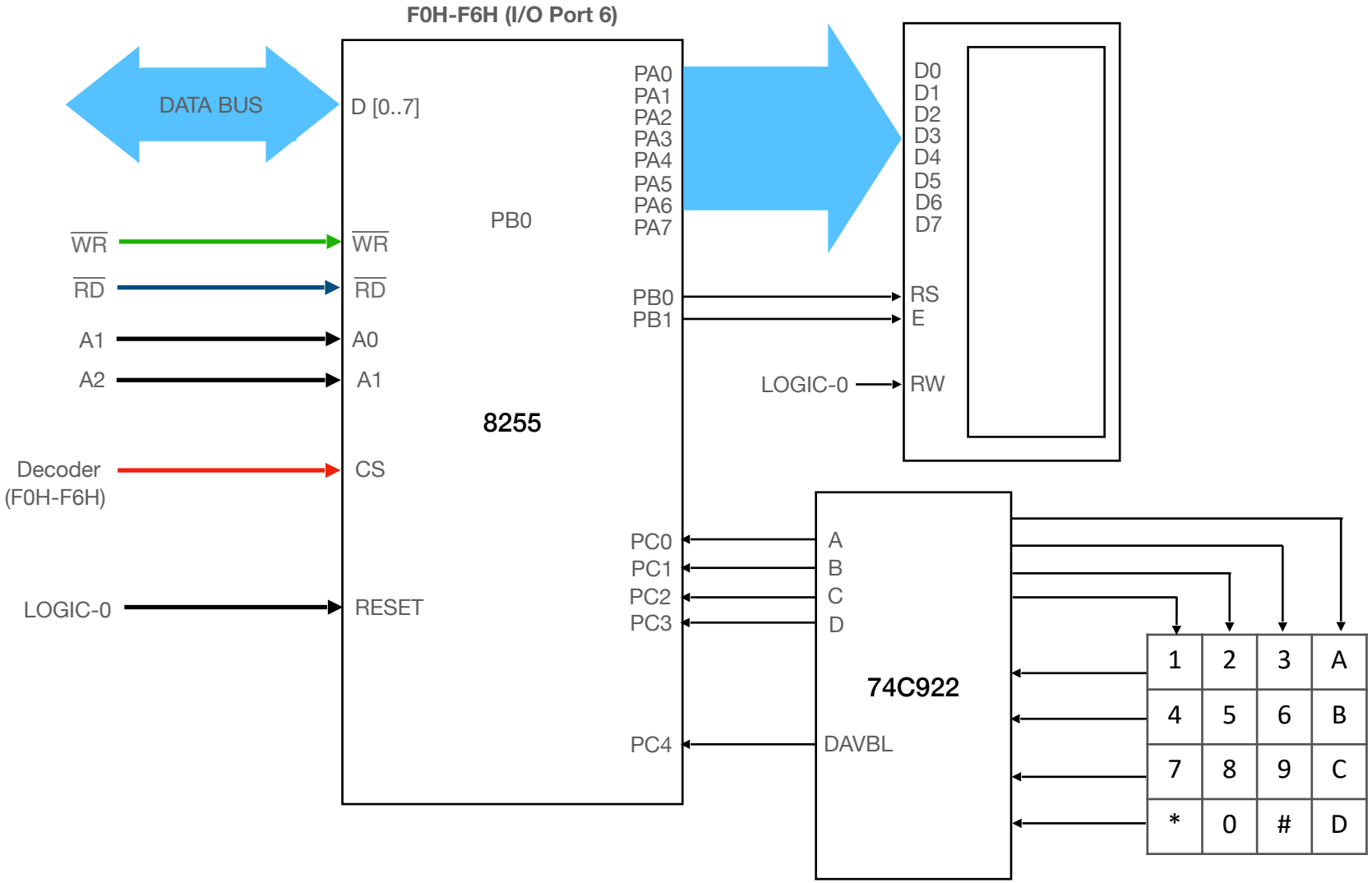
Pinout of the 74C922 keypad encoder

Data from 74C922

Key	74C922 Data Output (bin)	Hex
1	0000b	0x00
2	0001b	0x01
3	0010b	0x02
A	0011b	0x03
4	0100b	0x04
5	0101b	0x05
6	0110b	0x06
B	0111b	0x07
7	1000b	0x08
8	1001b	0x09
9	1010b	0x0A
C	1011b	0x0B
*	1100b	0x0C
0	1101b	0x0D
#	1110b	0x0E
D	1111b	0x0F

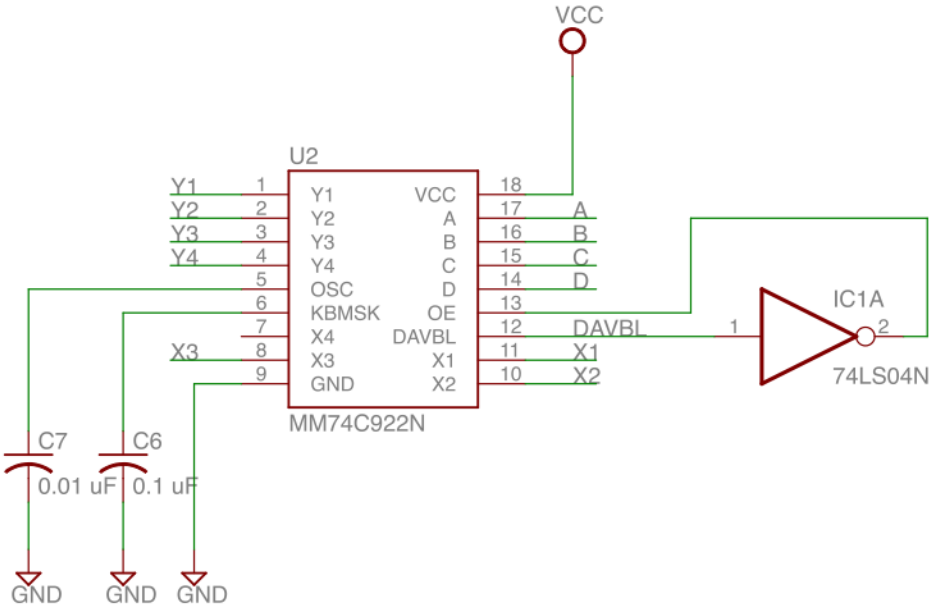
* keys in yellow cannot be accessed in 3x4 layout

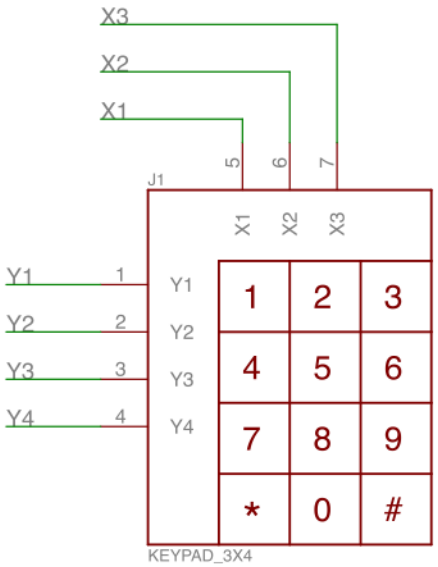
Interfacing via 8255 PPI



Electrical Connection of 74C922

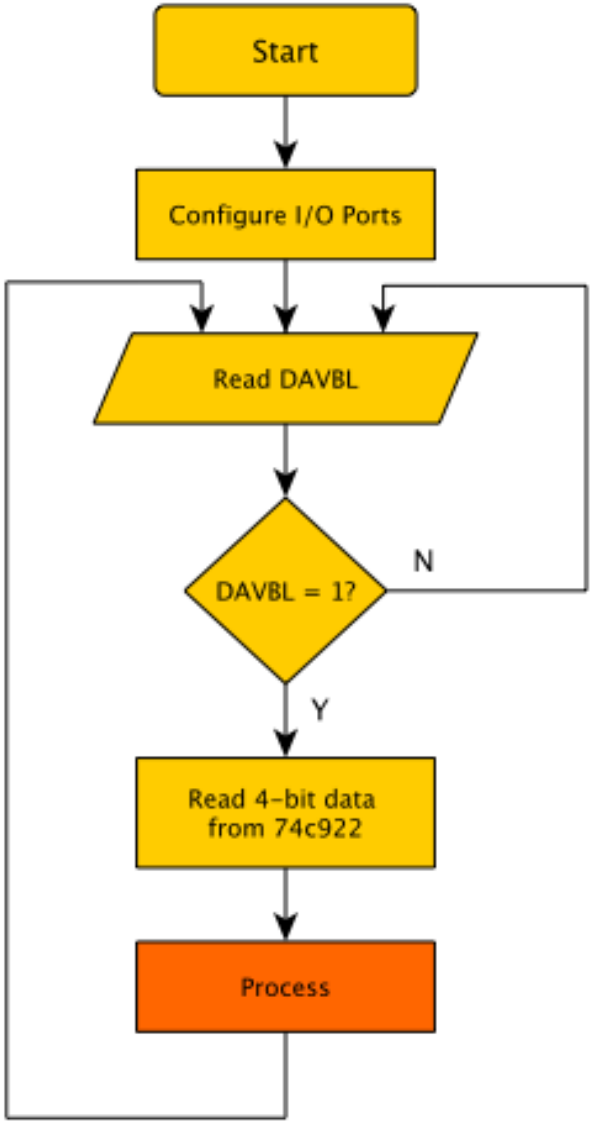
(with 3x4 keypad)





Software Control

- The DAVBL pin will go high (logic-1) when any key is pressed. This can be used as a signal to read the keypad data at A, B, C & D.



Read **DAVBL** before reading 4-bit data from 74C922

Example

- Read keypad data when DAVBL is high:

```
CHECK_DAVBL:
    MOV DX, PORTC ; set port of DAVBL(PORTC)
    IN AL, DX     ; read PORTC
    TEST AL, 10H  ; check if DAVBL is high
    JZ CHECK_DAVBL ; if low then check again
    IN AL, DX     ; read 4-bit keypad data
    AND AL, 0FH   ; mask upper nibble
```

Software Control Issues

- The example before shows that the keypad data will only be available if DAVBL is high. The software will continue to read DAVBL and exit the read loop only when DAVBL be high.
- Not very efficient but it works.
- Most efficient method:
  - external interrupt (will be discussed in Unit 7)
  - scheduling (will be covered in Operating Systems)

Demonstration #2

- The keypad+74C922 and a single 7-segment display (no decoder) will be interfaced to the 8086 via the 8255 at address E8H-EEH. 7-segment display is connected to PORTA and 74C922 is connected to PORTC: data (PC0-PC3), DAVBL (PC4).
- The program will display the key pressed on the keypad to the 7-segment display. For keys '*' and "#', the display will be '-'.
- Display is '0' initially then DAVBL will be read until to it goes high (logic-1) then update the display.

```
DATA SEGMENT
      PORTA EQU 0E8H          ; address to access PORTA
      PORTB EQU 0EAH          ; address to access PORTB
      PORTC EQU 0ECH          ; address to access PORTC
      PORT_CON EQU 0EEH       ; address to access Command Register
DATA ENDS

CODE SEGMENT PUBLIC 'CODE'
           ASSUME CS:CODE
START:
      ; Program 8255 (PORTA & PORTB as output, PORTC as input)
      MOV DX, PORT_CON        ; set port to Command Register
      MOV AL, 89H             ; set command byte
      OUT DX, AL              ; send data in AL to Command Register

CHECK_DAVBL:
      MOV DX, PORTC       ; set port of DAVBL(PORTC)
      IN AL, DX           ; read PORTC
      TEST AL, 10H        ; check if DAVBL is high
      JZ CHECK_DAVBL      ; if low then check again
      IN AL, DX           ; read 4-bit keypad data
      AND AL, 0FH         ; mask upper nibble

      CMP AL, 00H         ; check if key pressed is 1 (00H)
      JE D1               ; display 1
      CMP AL, 01H         ; check if key pressed is 2 (01H)
      JE D2               ; display 2
      CMP AL, 02H         ; check if key pressed is 3 (02H)
      JE D3               ; display 3
      CMP AL, 04H         ; check if key pressed is 4 (04H)
      JE D4               ; display 4
      CMP AL, 05H         ; check if key pressed is 5 (05H)
      JE D5               ; display 5
      CMP AL, 06H         ; check if key pressed is 6 (06H)
      JE D6               ; display 6
      CMP AL, 08H         ; check if key pressed is 7 (08H)
      JE D7               ; display 7
```

```
        CMP AL, 09H         ; check if key pressed is 8 (09H)
        JE D8               ; display 8
        CMP AL, 0AH         ; check if key pressed is 9 (0AH)
        JE D9               ; display 9
        CMP AL, 0CH         ; check if key pressed is * (0CH)
        JE D10              ; display -
        CMP AL, 0DH         ; check if key pressed is 0 (0DH)
        JE D0               ; display 0
        CMP AL, 0EH         ; check if key pressed is * (0EH)
        JE D10              ; display -

        CALL DELAY_1MS
        JMP CHECK_DAVBL
D0:   MOV AL, 00111111B  ; display '0'
      JMP CONT
D1:   MOV AL, 00000110B  ; display '1'
      JMP CONT
D2:   MOV AL, 01011011B  ; display '2'
      JMP CONT
D3:   MOV AL, 01001111B  ; display '3'
      JMP CONT
D4:   MOV AL, 01100110B. ; display '4'
      JMP CONT
D5:   MOV AL, 01101101B  ; display '5'
      JMP CONT
D6:   MOV AL, 01111101B  ; display '6'
      JMP CONT
D7:   MOV AL, 00000111B  ; display '7'
      JMP CONT
D8:   MOV AL, 01111111B  ; display '8'
      JMP CONT
D9:   MOV AL, 01100111B. ; display '9'
      JMP CONT
D10:  MOV AL, 01000000B  ; display '-'
```

```
CONT:
      MOV DX, PORTA
      OUT DX, AL
      CALL DELAY_1MS
      JMP CHECK_DAVBL

DELAY_1MS:
      MOV BX, 02CAH
L1:
      NOP
      JNZ
      L1
      RET

CODE ENDS
END START
```

Demonstration #3

- The keypad+74C922 and LCD will be interfaced to the 8086 via the 8255 at address F0H-F6H (see page 50).
- LCD configuration: 8-bit, cursor off, blink off, increment cursor, display shift off
- LCD is initialized first, then check keypad input.
  - if ***1*** is pressed, print "`University of`" middle of 1st line
  - if ***2*** is pressed, print "`San Carlos`" middle of 2nd line
  - if ***3*** is pressed, print "`Department of`" middle of 3rd line
  - if ***4*** is pressed, print "`Computer Engineering`" middle of 4th line
  - if * is pressed, clear the display
  - if any other key is pressed, do nothing

```
DATA SEGMENT
      PORTA EQU 0F0H          ; address to access PORTA
      PORTB EQU 0F2H          ; address to access PORTB
      PORTC EQU 0F4H          ; address to access PORTC
      PORT_CON EQU 0F6H       ; address to access Command Register
DATA ENDS

CODE SEGMENT PUBLIC 'CODE'
           ASSUME CS:CODE
START:
      ; Program 8255 (PORTA & PORTB as output, PORTC as input)
      MOV DX, PORT_CON        ; set port to Command Register
      MOV AL, 89H             ; set command byte
      OUT DX, AL              ; send data in AL to Command Register

      CALL INIT_LCD           ; initialize LCD

CHECK_DAVBL:
      MOV DX, PORTC       ; set port of DAVBL(PORTC)
      IN AL, DX           ; read PORTC
      TEST AL, 10H        ; check if DAVBL is high
      JZ CHECK_DAVBL      ; if low then check again
      IN AL, DX           ; read 4-bit keypad data
      AND AL, 0FH         ; mask upper nibble

      CMP AL, 00H         ; check if key pressed is 1 (00H)
      JE PRNT_LINE1       ; print line 1
      CMP AL, 01H         ; check if key pressed is 2 (01H)
      JE PRNT_LINE2       ; print line 2
      CMP AL, 02H         ; check if key pressed is 3 (02H)
      JE PRNT_LINE3       ; print line 3
      CMP AL, 04H         ; check if key pressed is 4 (04H)
      JE PRNT_LINE4       ; print line 4
      CMP AL, 0CH         ; check if key pressed is * (0CH)
      JE CLR_DISP

      JMP CHECK_DAVBL
```

```
      ; display string "University of" at middle of 1st line
PRNT_LINE1:
      MOV AL, 83H    ; position cursor at 1st line, 4th column
      CALL INST_CTRL ; send command to LCD
      MOV AL, 'U'    ; set ASCII code of character 'U'
      CALL DATA_CTRL ; display the character
      MOV AL, 'n'    ; set ASCII code of character 'n'
      CALL DATA_CTRL ; display the character
      MOV AL, 'i'    ; set ASCII code of character 'i'
      CALL DATA_CTRL ; display the character
      MOV AL, 'v'    ; set ASCII code of character 'v'
      CALL DATA_CTRL ; display the character
      MOV AL, 'e'    ; set ASCII code of character 'e'
      CALL DATA_CTRL ; display the character
      MOV AL, 'r'    ; set ASCII code of character 'r'
      CALL DATA_CTRL ; display the character
      MOV AL, 's'    ; set ASCII code of character 's'
      CALL DATA_CTRL ; display the character
      MOV AL, 'i'    ; set ASCII code of character 'i'
      CALL DATA_CTRL ; display the character
      MOV AL, 't'    ; set ASCII code of character 't'
      CALL DATA_CTRL ; display the character
      MOV AL, 'y'    ; set ASCII code of character 'y'
      CALL DATA_CTRL ; display the character
      MOV AL, ' '    ; set ASCII code of character ' '
      CALL DATA_CTRL ; display the character
      MOV AL, 'o'    ; set ASCII code of character 'o'
      CALL DATA_CTRL ; display the character
      MOV AL, 'f'    ; set ASCII code of character 'f'
      CALL DATA_CTRL ; display the character
      JMP CHECK_DAVBL
```

```
      ; display string "San Carlos" at middle of 2nd line
PRNT_LINE2:
      MOV AL, 0C5H    ; position cursor at 2nd line, 6th column
      CALL INST_CTRL ; send command to LCD
      MOV AL, 'S'     ; set ASCII code of character 'S'
      CALL DATA_CTRL ; display the character
      MOV AL, 'a'     ; set ASCII code of character 'a'
      CALL DATA_CTRL ; display the character
      MOV AL, 'n'     ; set ASCII code of character 'n'
      CALL DATA_CTRL ; display the character
      MOV AL, ' '     ; set ASCII code of character ' '
      CALL DATA_CTRL ; display the character
      MOV AL, 'C'     ; set ASCII code of character 'C'
      CALL DATA_CTRL ; display the character
      MOV AL, 'a'     ; set ASCII code of character 'a'
      CALL DATA_CTRL ; display the character
      MOV AL, 'r'     ; set ASCII code of character 'r'
      CALL DATA_CTRL ; display the character
      MOV AL, 'l'     ; set ASCII code of character 'l'
      CALL DATA_CTRL ; display the character
      MOV AL, 'o'     ; set ASCII code of character 'o'
      CALL DATA_CTRL ; display the character
      MOV AL, 's'     ; set ASCII code of character 's'
      CALL DATA_CTRL ; display the character
      JMP CHECK_DAVBL
```

```
      ; display string "Department of" at middle of 3rd line
PRNT_LINE3:
      MOV AL, 97H     ; position cursor at 3rd line, 4th column
      CALL INST_CTRL  ; send command to LCD
      MOV AL, 'D'     ; set ASCII code of character 'D'
      CALL DATA_CTRL  ; display the character
      MOV AL, 'e'     ; set ASCII code of character 'e'
      CALL DATA_CTRL  ; display the character
      MOV AL, 'p'     ; set ASCII code of character 'p'
      CALL DATA_CTRL  ; display the character
      MOV AL, 'a'     ; set ASCII code of character 'a'
      CALL DATA_CTRL  ; display the character
      MOV AL, 'r'     ; set ASCII code of character 'r'
      CALL DATA_CTRL  ; display the character
      MOV AL, 't'     ; set ASCII code of character 't'
      CALL DATA_CTRL  ; display the character
      MOV AL, 'm'     ; set ASCII code of character 'm'
      CALL DATA_CTRL  ; display the character
      MOV AL, 'e'     ; set ASCII code of character 'e'
      CALL DATA_CTRL  ; display the character
      MOV AL, 'n'     ; set ASCII code of character 'n'
      CALL DATA_CTRL  ; display the character
      MOV AL, 't'     ; set ASCII code of character 't'
      CALL DATA_CTRL  ; display the character
      MOV AL, ' '     ; set ASCII code of character ' '
      CALL DATA_CTRL  ; display the character
      MOV AL, 'o'     ; set ASCII code of character 'o'
      CALL DATA_CTRL  ; display the character
      MOV AL, 'f'     ; set ASCII code of character 'f'
      CALL DATA_CTRL  ; display the character
      JMP CHECK_DAVBL
```

```
      ; display string "Computer Engineering" at middle of 4th line
PRNT_LINE4:
      MOV AL, 0D4H    ; position cursor at 4th line, 1st column
      CALL INST_CTRL ; send command to LCD
      MOV AL, 'C'     ; set ASCII code of character 'C'
      CALL DATA_CTRL ; display the character
      MOV AL, 'o'     ; set ASCII code of character 'o'
      CALL DATA_CTRL ; display the character
      MOV AL, 'm'     ; set ASCII code of character 'm'
      CALL DATA_CTRL ; display the character
      MOV AL, 'p'     ; set ASCII code of character 'p'
      CALL DATA_CTRL ; display the character
      MOV AL, 'u'     ; set ASCII code of character 'u'
      CALL DATA_CTRL ; display the character
      MOV AL, 't'     ; set ASCII code of character 't'
      CALL DATA_CTRL ; display the character
      MOV AL, 'e'     ; set ASCII code of character 'e'
      CALL DATA_CTRL ; display the character
      MOV AL, 'r'     ; set ASCII code of character 'r'
      CALL DATA_CTRL ; display the character
      MOV AL, ' '     ; set ASCII code of character ' '
      CALL DATA_CTRL ; display the character
      MOV AL, 'E'     ; set ASCII code of character 'E'
      CALL DATA_CTRL ; display the character
      MOV AL, 'n'     ; set ASCII code of character 'n'
      CALL DATA_CTRL ; display the character
      MOV AL, 'g'     ; set ASCII code of character 'g'
      CALL DATA_CTRL ; display the character
      MOV AL, 'i'     ; set ASCII code of character 'i'
      CALL DATA_CTRL ; display the character
      MOV AL, 'n'     ; set ASCII code of character 'n'
```

```
      CALL DATA_CTRL ; display the character
      MOV AL, 'e'    ; set ASCII code of character 'e'
      CALL DATA_CTRL ; display the character
      MOV AL, 'e'    ; set ASCII code of character 'e'
      CALL DATA_CTRL ; display the character
      MOV AL, 'r'    ; set ASCII code of character 'r'
      CALL DATA_CTRL ; display the character
      MOV AL, 'i'    ; set ASCII code of character 'i'
      CALL DATA_CTRL ; display the character
      MOV AL, 'n'    ; set ASCII code of character 'n'
      CALL DATA_CTRL ; display the character
      MOV AL, 'g'    ; set ASCII code of character 'g'
      CALL DATA_CTRL ; display the character
      JMP CHECK_DAVBL

      ; clear the display
CLR_DISP:
      MOV AL, 01H    ; set clear display instruction (01H)
      CALL INST_CTRL ; send instruction to LCD
      JMP CHECK_DAVBL

      ; this function will send an 8-bit instruction to the LCD
INST_CTRL:
      PUSH AX        ; preserve value of AL
      MOV DX, PORTA  ; set port of LCD data bus (PORTA)
      OUT DX, AL     ; write data in AL to PORTA
      MOV DX, PORTB  ; set port of LCD control lines (PORTB)
      MOV AL, 02H    ; E=1, RS=0 (access instruction reg)
      OUT DX, AL     ; write data in AL to PORTB
      CALL DELAY_1MS ; delay for 1 ms
      MOV DX, PORTB  ; set port of LCD control lines (PORTB)
      MOV AL, 00H    ; E=0, RS=0
      OUT DX, AL     ; write data in AL to PORTB
      POP AX         ; restore value of AL
      RET
```

```
      ; this function will send an 8-bit data (character) to the LCD
DATA_CTRL:
      PUSH AX         ; preserve value of AL
      MOV DX, PORTA   ; set port of LCD data bus (PORTA)
      OUT DX, AL      ; write data in AL to PORTA
      MOV DX, PORTB   ; set port of LCD control lines (PORTB)
      MOV AL, 03H     ; E=1, RS=1 (access data reg)
      OUT DX, AL      ; write data in AL to PORTB
      CALL DELAY_1MS  ; delay for 1 ms
      MOV DX, PORTB   ; set port of LCD control lines (PORTB)
      MOV AL, 01H     ; E=0, RS=1
      OUT DX, AL      ; write data in AL to PORTB
      POP AX          ; restore value of AL
      RET

      ; this function will initialize the LCD
INIT_LCD:
      MOV AL, 38H     ; 8-bit interface, dual-line display
      CALL INST_CTRL  ; write instruction to LCD
      MOV AL, 08H     ; display off, cursor off, blink off
      CALL INST_CTRL  ; write instruction to LCD
      MOV AL, 01H     ; clear display
      CALL INST_CTRL  ; write instruction to LCD
      MOV AL, 06H     ; increment cursor, display shift off
      CALL INST_CTRL  ; write instruction to LCD
      MOV AL, 0CH     ; display on, cursor off, blink off
      CALL INST_CTRL  ; write instruction to LCD
      RET

DELAY_1MS:
      MOV BX, 02CAH
L1:
      NOP
      JNZ
      L1
      RET

CODE ENDS
END START
```

* LCD and keypad is connected to the Fujitsu MB90F387S MCU via the Jouet Bleu kit

Questions?

University of San Carlos | Department of
COMPUTER ENGINEERING

CpE 3104
Microprocessors

End of Lecture

References:

- *Brey, Barry B., 2009. The Intel microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro processor, Pentium II, Pentium III, Pentium 4, and Core2 with 64-bit extensions: architecture, programming, and interfacing. 8th ed. New Jersey: Pearson Education.*
- *Sharp HD44870 LCD Datasheet*
- *Displaytech 204A Datasheet*
- *74C922 Datasheet*