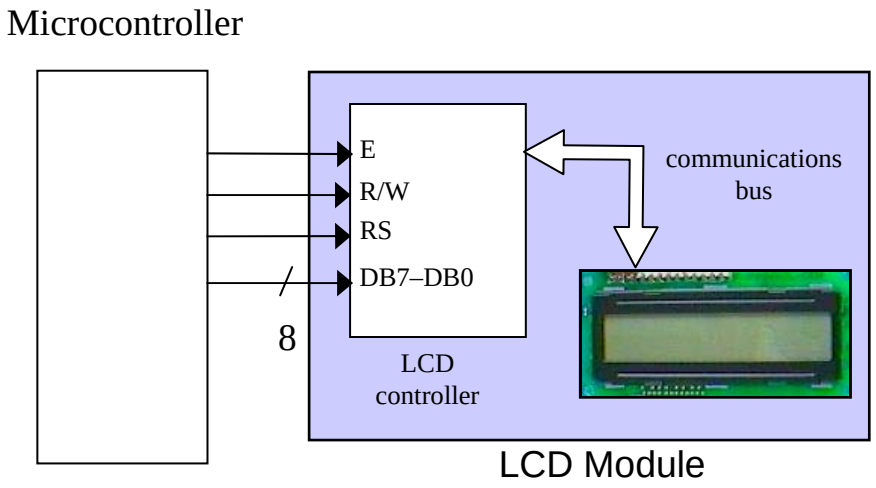
LCD Interfacing

- LCDs have become a cheap and easy way to get text display for an embedded system
  - Various configurations (1 line by 20 characters upto 8 lines by 80 characters), starting from around $5.
  - Graphics LCDs are also available
- Intelligent LCDs have internal ASCII decoders, Character Generators and LCD control circuitry
- Some also have custom character generation capacity
  - User defined character RAM
  - Program this RAM with the character pattern
  - Then use it like ordinary ASCII characters
    - Usually MSB decides between std ASCII and custom characters

Alphanumeric LCD Interfacing

- Pinout (statistical average)
  - 8 data pins D7:D0
  - RS: Data or Command Register Select
  - R/W: Read or Write
  - E: Enable (Latch data)
  - V_{EE} : contrast control



- RS – Register Select
  - RS = 0 → Command Register
  - RS = 1 → Data Register
- R/W = 0 → Write, R/W = 1 → Read
- E – Enable
  - Used to latch the data present on the data pins.
  - A high-to-low edge is needed to latch the data.
- D0 – D7
  - Bi-directional data/command pins.
  - Alphanumeric characters are sent in ASCII format.

LCD Commands

- The LCD's internal controller can accept several commands and modify the display accordingly. These commands would be things like:
  - Clear screen
  - Return home
  - Decrement/Increment cursor
  - Shift display right/left

  - Check the data sheet from the manufacturer.

- After writing to the LCD, it takes some time for it to complete its internal operations. During this time, it will not accept any new commands or data.
  - We can check if the LCD is busy or not by reading the busy flag.
    - Set R/W = 1 and RS = 0, then read from the LCD
      - If D7 = 1, then the LCD is still busy, wait.

Example LCD Interface Program

```
        ORG 100H
        MOV     A, #38H     ; Initialize, 2-lines, 5X7 matrix.
        ACALL   Command
        MOV     A, #0EH     ; LCD on, cursor on
        ACALL   Command
        MOV     A, #01H     ; Clear LCD Screen
        ACALL   Command
        MOV     A, #06H     ; Shift cursor right
        ACALL   Command
        MOV     A, #86H     ; Cursor, line 1 position 6
        ACALL   Command
        MOV     A, #'N'     ; Character N
        ACALL   Data
        MOV     A, #'O'     ; Character O
        ACALL   Data
Wait:   SJMP    Wait
```

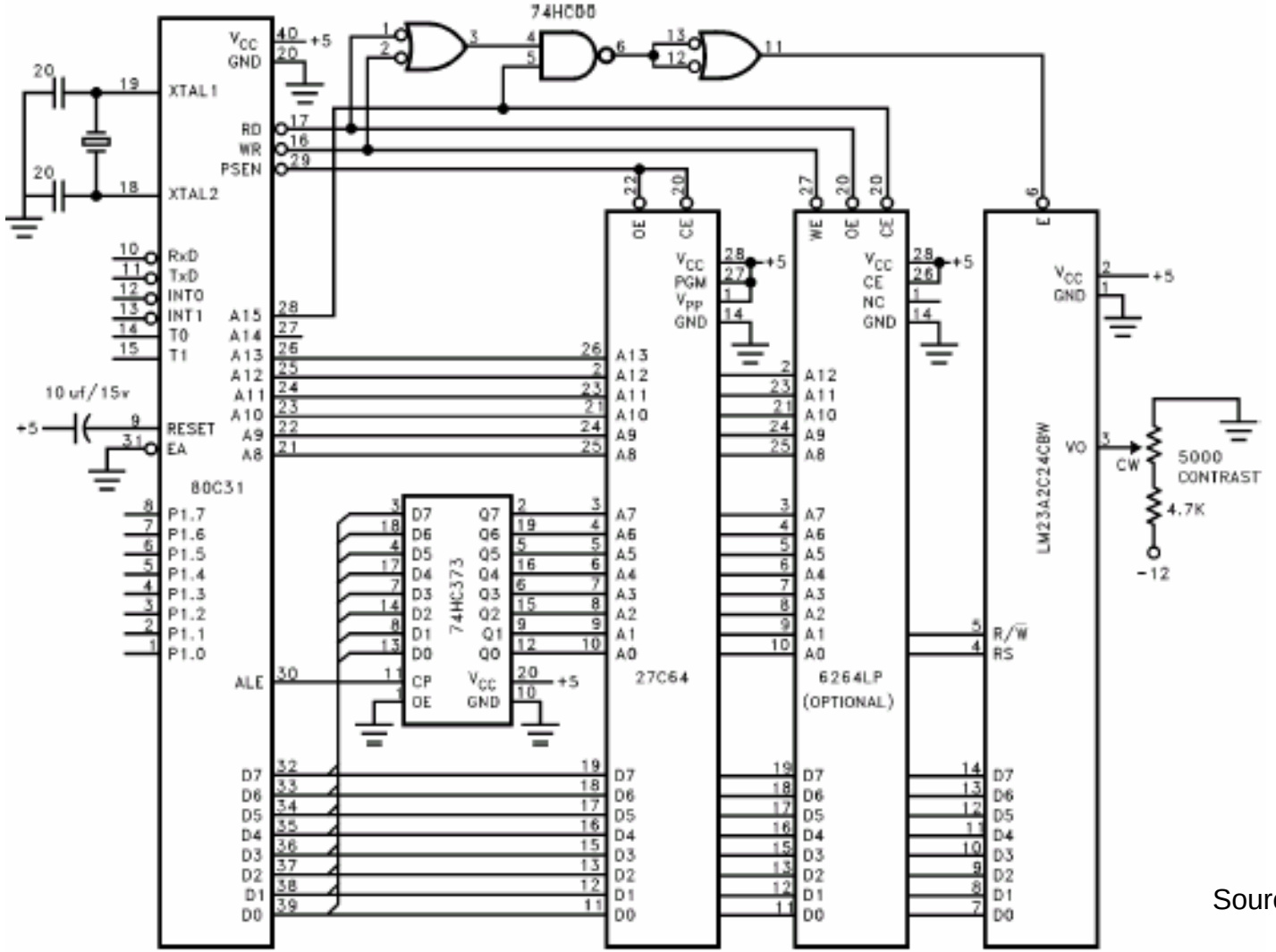
```
Command: ACALL   Ready
         MOV     P1, A
         CLR     P2.0       ; RS=0
         CLR     P2.1       ; R/W=0 (Write)
         SETB    P2.2       ; E = 1
         CLR     P2.2       ; E = 0 – complete the 1-to-0 pulse
         RET

Data:    ACALL   Ready
         MOV     P1, A
         SETB    P2.0       ; RS = 1
         CLR     P2.1       ; R/W = 0
         SETB    P2.2       ; Enable pulse
         CLR     P2.2
         RET

Ready:   SETB    P1.7       ; bit 7 of port 1 set to input
         CLR     P2.0       ; RS = 0 – Command Reg.
         SETB    P2.1       ; RW = 1 – Read
Back:    SETB    P2.2       ; Enable pulse
         CLR     P2.2
         JB      P1.7, Back ; Wait till not busy
         RET
```

The Cursor Location

- Whenever a new data character is sent to the LCD, the cursor position will be automatically updated one position to the right.

- It is also possible to place the cursor at a particular location using the “set address” command.
  - Set RS = 0 and R/W = 0
  - Set D7 = 1 – (set address command)
  - Set D6 = 0 for line 1 and D6 = 1 for line 2
  - Set D5 – D0 for the position on the line (in binary).
    - 1st position (extreme left) is 0.

Interfacing an 8031 to an LCD



Source: Intel Application Brief AB-39