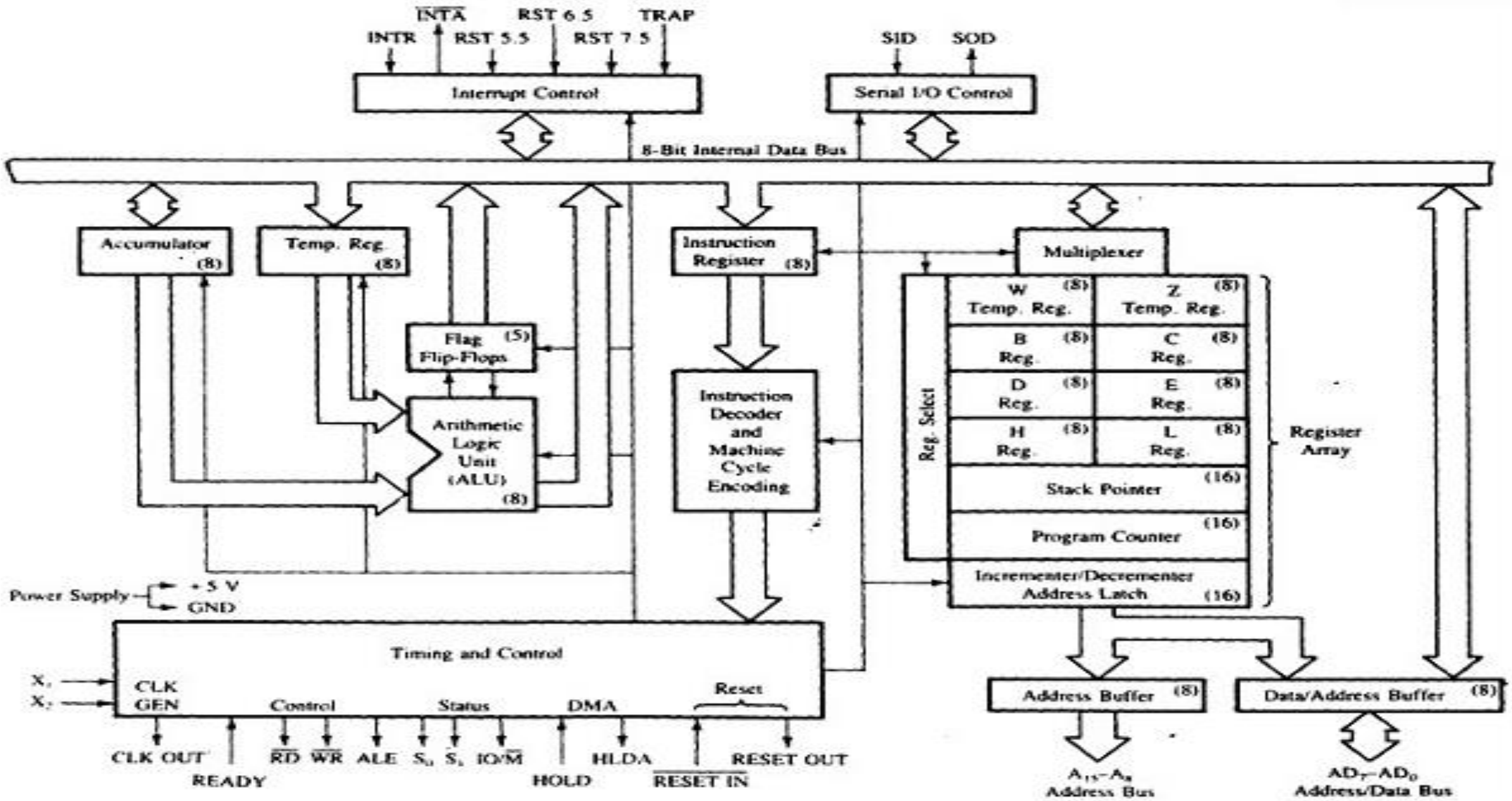
EE8551 MICROPROCESSOR AND MICROCONTROLLERS

Unit 1-8085 Microprocessor
Topic : Architecture of 8085 Microprocessor

Internal Architecture of 8085



The internal architecture of the 8085 microprocessor determines how and what operations can be performed with the data operations are,

- Store 8-bit data.
- Perform arithmetic and logical operations.
- Test for conditions.
- Sequence the execution of instructions.
- Store data temporarily in read write memory called stack.

Registers

- The 8085 has 6 general purpose registers to store 8-bit data during program execution.
- These 6 registers are identified as B, C, D, E, H and L.
- They can be combined as register pairs BC, DE and HL to perform some 16-bit operations.
- These registers are programmable. It can use to load or transfer data from the registers by using instructions.

Accumulator

- The accumulator (A) is an 8-bit register that is part of ALU.
- It is used to store 8-bit data and to perform ALU operations.
- The result of an operation is stored in Accumulator.
- The Accumulator is identified as register A.
- The data on which operations is to be performed is operand. One of the operands must be Accumulator.

Program Counter (PC)

- This 16-bit register deals with sequencing the execution of instructions.
- This register is a memory pointer. Memory locations have 16-bit address.
- The microprocessor uses this register to sequence the execution of the instructions.
- The function of the program counter is to point to the memory address from which the next byte is to be fetched.
- When a byte is being fetched, the program counter is automatically incremented by one to point to the next memory location.

Stack Pointer (SP)

- The stack pointer is also a 16-bit register, used as a memory pointer.
- It points to a memory location in R/W memory, called stack.
- The beginning of the stack is defined by loading 16- bit address in the stack pointer.

Timing and control unit

- This unit synchronizes all the microprocessor operations with the clock and generates the control signal necessary for communication between the microprocessor and peripherals.
- The control signals are similar to a sync pulse in an oscilloscope. The $\overline{RD}$ and $\overline{WR}$ signals are sync pulses indicating the availability of data on the data bus.

Instruction register and Decoder

- The instruction register and decoder are part of the ALU.
- When an instruction is fetched from the memory it is loaded in to the instruction register.
- The decoder decodes the instruction and establishes the sequence of events to follow.
- The instruction register is not programmable and cannot be accessed by any instruction.

Register Array

- There are two additional registers called temporary registers W and Z, which are included in the register array along with programmable registers namely B, C, D, E, H, L, SP and PC.
- These registers are used to hold 8-bit data during the execution of instructions. However, they are used internally by microprocessor, they are available to the program.

MUX / DEMUX unit

- This unit is used to select a register out of all the available registers.
- This unit behaves as Multiplexer (MUX) when data going from the register to the internal data bus.
- It behaves as Demultiplexer(DEMUX) when data is coming to a register from the internal data bus of the processor.
- The register select will behave as the function of selection lines at the Mux / Demux.

Address Buffer

- This is an 8-bit unidirectional buffer.
- It is used to drive external high order address bus (A15 – A8).
- It is also used to tri-state the high order address bus under certain conditions such as reset, hold, halt and when address lines are not in use.

Address / Data Buffer

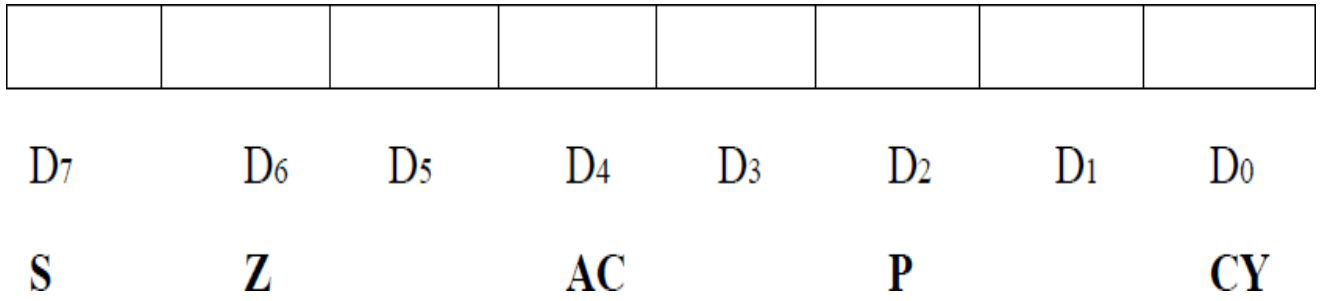
- This is an 8-bit bi-directional buffer.
- It is used to drive multiplexed address/data bus. i.e., low order address bus (A_7-A_0) and data bus (D_7-D_0).
- It is also to tri-state the multiplexed address/data bus under certain conditions such as reset, hold, halt and when the bus is not in use.

Incrementer/Decrementer address latch

- This 16-bit register is used to increment or decrement the contents of program counter or stack pointer as part of execution of instructions related to them.

Flag register

- The ALU includes five flip-flops, which are set (or) reset after an operation according to data condition of the result in the accumulator and other registers.
- They are called Zero (Z), Carry (CY), Sign (S), Parity (P) and Auxiliary Carry (AC) flags.
- Their bit positions in the flag register are shown in Fig. The microprocessor uses these flags to test data conditions.



Unit -2
ADDRESSING MODES OF 8085

Addressing Modes of 8085

- To perform any operation, we have to give the corresponding instructions to the microprocessor.
- In each instruction, programmer has to specify 3 things:
  - Operation to be performed.
  - Address of source of data.
  - Address of destination of result.

Addressing Modes of 8085

- The method by which the address of source of data or the address of destination of result is given in the instruction is called **Addressing Modes.**
- The term addressing mode refers to the way in which the operand of the instruction is specified.

Types of Addressing Modes

- Intel 8085 uses the following addressing modes:
  1. Direct Addressing Mode
  2. Register Addressing Mode
  3. Register Indirect Addressing Mode
  4. Immediate Addressing Mode
  5. Implicit Addressing Mode

Direct Addressing Mode

- In this mode, the address of the operand is given in the instruction itself.

LDA 2500 H	**Load the contents of memory location 2500 H in accumulator.**

- LDA is the operation.
- 2500 H is the address of source.
- Accumulator is the destination.

Register Addressing Mode

- In this mode, the operand is in general purpose register.

MOV A, B	Move the contents of register B to A.

- MOV is the operation.
- B is the source of data.
- A is the destination.

Register Indirect Addressing Mode

- In this mode, the address of operand is specified by a register pair.

MOV A, M	Move data from memory location specified by H-L pair to accumulator.

- MOV is the operation.
- M is the memory location specified by H-L register pair.
- A is the destination.

Immediate Addressing Mode

- In this mode, the operand is specified within the instruction itself.

MVI A, 05 H	**Move 05 H in accumulator.**

- MVI is the operation.
- 05 H is the immediate data (source).
- A is the destination.

Implicit Addressing Mode

- If address of source of data as well as address of destination of result is fixed, then there is no need to give any operand along with the instruction.

CMA	Complement accumulator.

- CMA is the operation.
- A is the source.
- A is the destination.

UNIT 3
8051 Microcontroller

PROGRAMMING 8051 TIMERS

- **Basic registers of the timer**
  - **Timer 0 and Timer 1 are 16 bits wide**
  - **each 16-bit timer is accessed as two separate registers of low byte and high byte.**

PROGRAMMING 8051 TIMERS

- **Timer 0 registers**
  - **low byte register is called TL0 (Timer 0 low byte) and the high byte register is referred to as TH0 (Timer 0 high byte)**
  - **can be accessed like any other register, such as A, B, R0, R1, R2, etc.**
  - **"MOV TL0, #4 FH" moves the value 4FH into TL0**
  - **"MOV R5, TH0" saves TH0 (high byte of Timer 0) in R5**

PROGRAMMING 8051 TIMERS

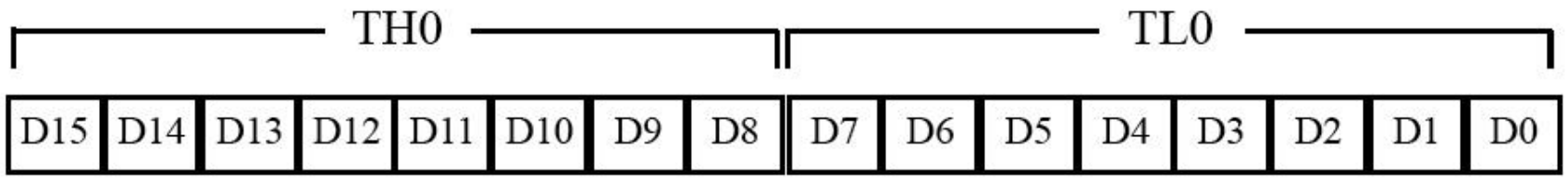


Figure 9–1 Timer 0 Registers

PROGRAMMING 8051 TIMERS

- **Timer 1 registers**
  - **also 16 bits**
  - **split into two bytes TL1 (Timer 1 low byte) and TH1 (Timer 1 high byte)**
  - **accessible in the same way as the registers of Timer 0.**

PROGRAMMING 8051 TIMERS

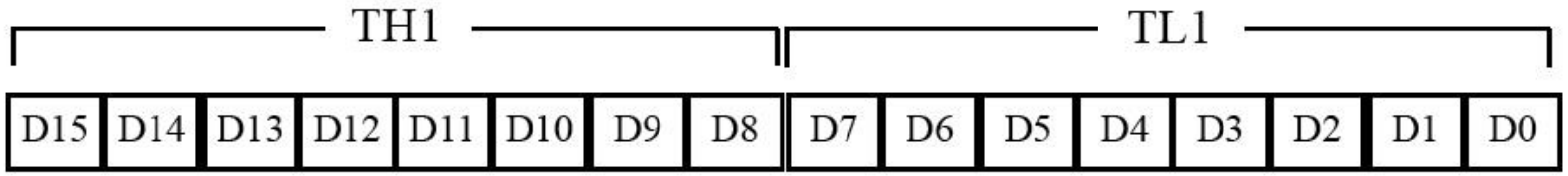


Figure 9–2 Timer 1 Registers

PROGRAMMING 8051 TIMERS

- **TMOD (timer mode) register**
  - **timers 0 and 1 use TMOD register to set operation modes (only learn Mode 1 and 2)**
  - **8-bit register**
  - **lower 4 bits are for Timer 0**
  - **upper 4 bits are for Timer 1**
  - **lower 2 bits are used to set the timer mode**
    - **(only learn Mode 1 and 2)**
  - **upper 2 bits to specify the operation**
    - **(only learn timer operation)**

PROGRAMMING 8051 TIMERS

(MSB) (LSB)

GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer 1				Timer 0			

GATE Gating control when set. The timer/counter is enabled only while the INTx pin is high and the TRx control pin is set. When cleared, the timer is enabled whenever the TRx control bit is set.

C/T Timer or counter selected cleared for timer operation (input from internal system clock). Set for counter operation (input from Tx input pin).

M1 Mode bit 1

M0 Mode bit 0

M1	M0	Mode	Operating Mode
0	0	0	13-bit timer mode 8-bit timer/counter THx with TLx as 5-bit prescaler
0	1	1	16-bit timer mode 16-bit timer/counters THx and TLx are cascaded; there is no prescaler
1	0	2	8-bit auto reload 8-bit auto reload timer/counter; THx holds a value that is to be reloaded into TLx each time it overflows.
1	1	3	Split timer mode

Figure 9–3 TMOD Register

PROGRAMMING 8051 TIMERS

- Clock source for timer
  - timer needs a clock pulse to tick
  - if C/T = 0, the crystal frequency attached to the 8051 is the source of the clock for the timer
  - frequency for the timer is always 1/12th the frequency of the crystal attached to the 8051
  - XTAL = 11.0592 MHz allows the 8051 system to communicate with the PC with no errors
  - In our case, the timer frequency is 1MHz since our crystal frequency is 12MHz

PROGRAMMING 8051 TIMERS

- Mode 1 programming
  - 16-bit timer, values of 0000 to FFFFH
  - TH and TL are loaded with a 16-bit initial value
  - timer started by "SETB TR0" for Timer 0 and "SETB TR1" for Timer l
  - timer count ups until it reaches its limit of FFFFH
  - rolls over from FFFFH to 0000H
  - sets TF (timer flag)
  - when this timer flag is raised, can stop the timer with "CLR TR0" or "CLR TR1“
  - after the timer reaches its limit and rolls over, the registers TH and TL must be reloaded with the original value and TF must be reset to 0

PROGRAMMING 8051 TIMERS

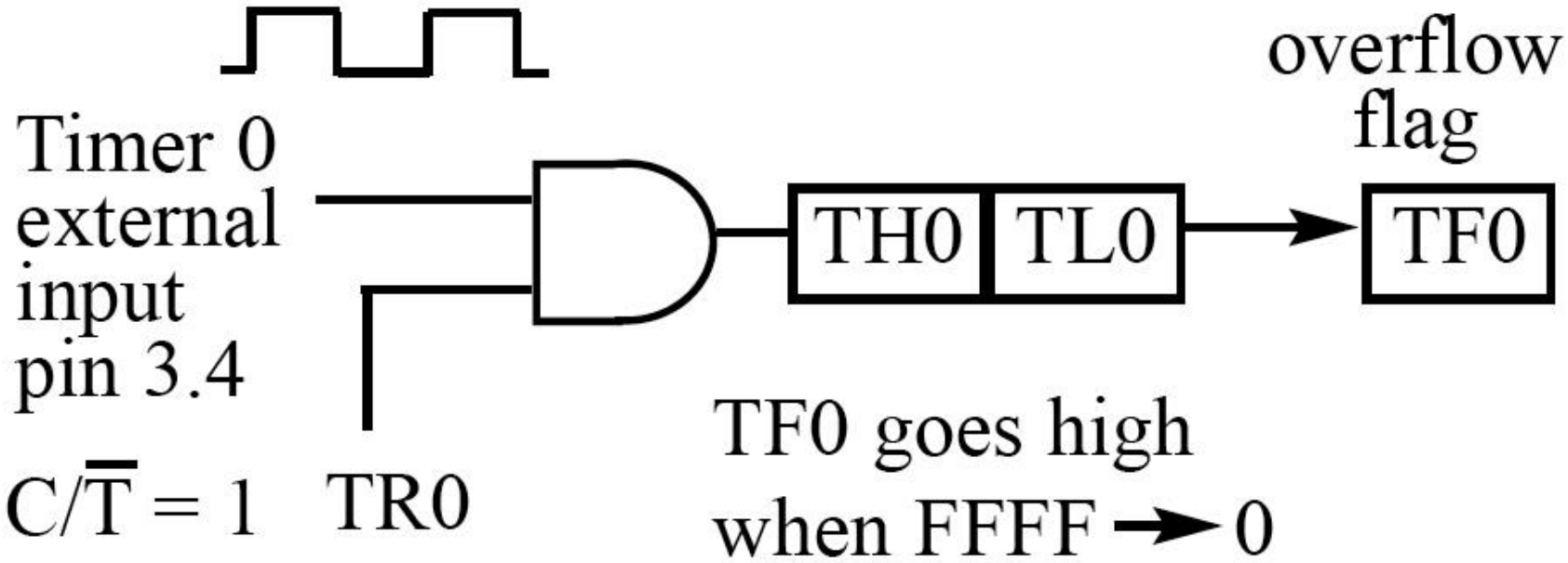


Figure 9–5a Timer 0 with External Input (Mode 1)

PROGRAMMING 8051 TIMERS (for information only)

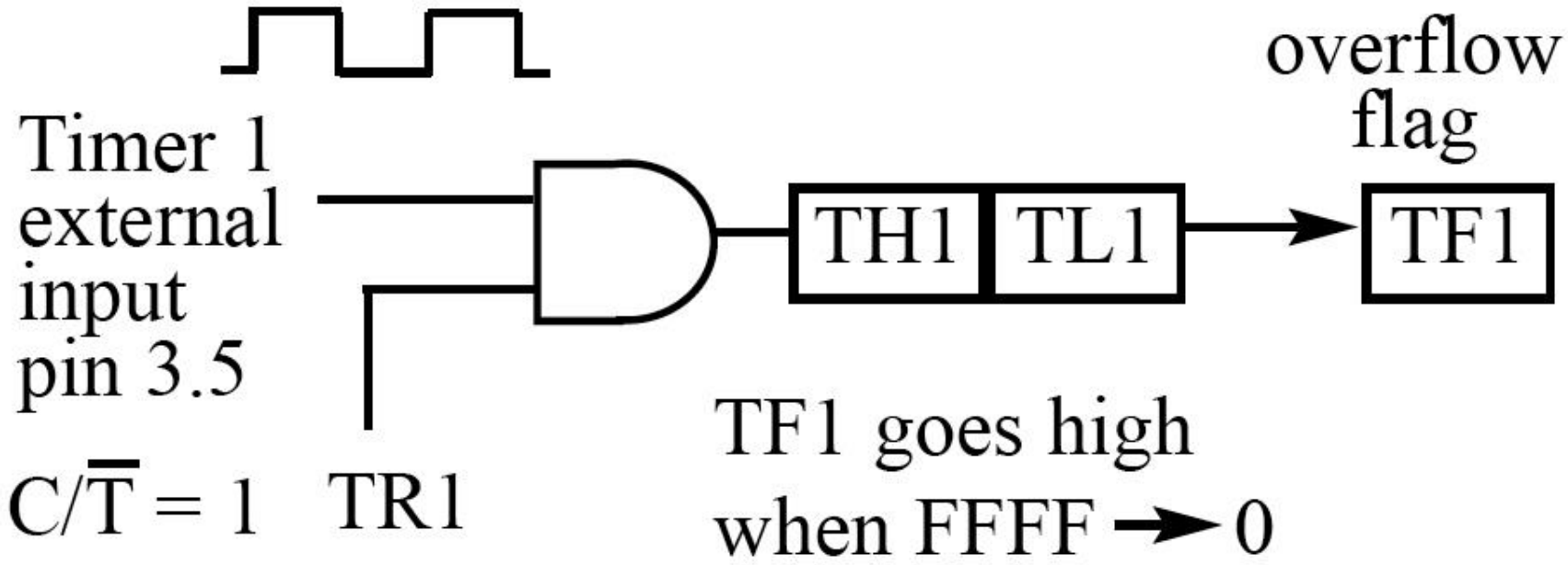


Figure 9–5b Timer 1 with External Input (Mode 1)

PROGRAMMING 8051 TIMERS

- **Steps to program in mode 1**
  - **Set timer mode 1 or 2**
  - **Set TL0 and TH0 (for mode 1 16 bit mode)**
  - **Set TH0 only (for mode 2 8 bit auto reload mode)**
  - **Run the timer**
  - **Monitor the timer flag bit**

Example
In the following program, we are creating a square wave of 50% duty cycle (with equal portions high and low) on the P1.5 bit.
Timer 0 is used to generate the time delay

```
01  MOV TMOD,#01            ;Timer 0, mode 1(16-bit mode)
02  HERE: MOV TL0,#0F2H     ;TL0 = F2H, the Low byte
03  MOV TH0,#0FFH           ;TH0 = FFH, the High byte
04  CPL P1.5                ;toggle P1.5
05  ACALL DELAY
06  SJMP HERE               ;load TH, TL again
07
08  DELAY:                  ;delay using Timer 0
09  SETB TR0                ;start Timer 0
10  AGAIN: JNB TF0,AGAIN    ;monitor Timer 0 flag until ;it rolls over
11  CLR TR0                 ;stop Timer 0
12  CLR TF0                 ;clear Timer 0 flag
13  RET
14
15  END
```

Example

The following program generates a square wave on pin P 1.5 continuously using Timer 1 for a time delay. Find the frequency of the square wave if XTAL = 11.0592 MHz. In your calculation do not include the overhead due to the timer setup instructions in the loop.

```
01  MOV TMOD,#10H                ;Timer 1, mode 1(16-bit)
02  AGAIN: MOV TL1,#34H          ;TL1 = 34H, Low byte
03  MOV TH1,#76H                 ;TH1 = 76H, High byte
04                               ;(7634H = timer value)
05  SETB TR1                     ;start Timer 1
06  BACK: JNB TF1,BACK           ;stay until timer rolls over
07  CLR TR1                      ;stop Timer 1
08  CPL P1.5                     ;comp. P1.5 to get hi, lo
09  CLR TF1                      ;clear Timer 1 flag
10  SJMP AGAIN                   ;reload timer since Mode 1
11                               ;is not auto-reload
12  END
13
14  ;Since FFFFH - 7634H = 89CBH + 1 = 89CCH
15  ;and 89CCH = 35276 clock count.
16  ;35276 x 1.085 us = 38.274 ms for half of the square wave.
17  ;The entire square wave length is 38.274 x 2 = 76.548 ms
18  ;and has a frequency = 13.064 Hz.
19  ;The high and low portions of the
20  ;square wave pulse are equal.
21  ;The overhead due to all the
22  ;instructions in the loop
23  ;is not included.
```

SECTION 9.1: PROGRAMMING 8051 TIMERS

- **Finding values to be loaded into the timer**
  - **XTAL = 11.0592 MHz (12MHz)**
  - **divide the desired time delay by 1.085μs (1μs) to get *n***
  - **65536 – *n* = *N***
  - **convert *N* to hex yyxx**
  - **set TL = xx and TH = yy**

Example
Assuming XTAL = 11.0592 MHz, write a program to generate a square wave of 50 Hz frequency on pin P2.3.

- T = 1/50 Hz = 20 ms
- 1/2 of it for the high and low portions of the pulse = 10 ms
- 10 ms / 1.085 us = 9216
- 65536 - 9216 = 56320 in decimal = DC00H
- TL = 00 and TH = DCH
- The calculation for 12MHz crystal uses the same steps

Example (cont)
Assuming XTAL = 11.0592 MHz, write a program to generate a square wave of 50 Hz frequency on pin P2.3.

```
01  MOV TMOD,#10H           ;Timer 1 mode 1 (16-bit)
02  AGAIN: MOV TL1,#00      ;TL1 = 00, Low byte
03  MOV TH1,#0DCH           ;TH1 = 0DCH, High byte
04
05  SETB TR1                ;start Timer 1
06  BACK: JNB TF1,BACK      ;stay until timer rolls over
07  CLR TR1                 ;stop Timer 1
08  CPL P2.3                ;compliment P2.3 to get hi, lo
09  CLR TF1                 ;clear Timer 1 flag
10  SJMP AGAIN              ;reload timer since
11                          ;mode 1 is not auto reload
12
13  END
```

PROGRAMMING 8051 TIMERS

- **Generating a large time delay**
  - **size of the time delay depends**
    - **crystal frequency**
    - **timer's 16-bit register in mode 1**
  - **largest time delay is achieved by making both TH and TL zero**
  - **what if that is not enough?**

PROGRAMMING 8051 TIMERS

- **Using Windows calculator to find TH, TL**
  - Windows scientific calculator can be use to find the TH, TL values
  - Lets say we would like to find the TH, TL values for a time delay that uses 35,000 clocks of 1.085μs
    1. open scientific calculator and select decimal
    2. enter 35,000
    3. select hex - converts 35,000 to hex 88B8H
    4. select +/- to give -35000 decimal (7748H)
    5. the lowest two digits (48) of this hex value are for TL and the next two (77) are for TH

Example

Examine the following program and find the time delay in seconds. Exclude the time delay due to the instructions in the loop.

```
01  MOV TMOD,#10H              ;Timer 1, mode 1(16-bit)
02  MOV R3,#200                ;counter for multiple delay
03
04  AGAIN: MOV TL1,#08H        ;TL1 = 08, Low byte
05  MOV TH1,#01H               ;TH1 = 01, High byte
06  SETB TR1                   ;start Timer 1
07  BACK: JNB TF1,BACK         ;stay until timer rolls over
08  CLR TR1                    ;stop Timer 1
09  CLR TF1                    ;clear Timer 1 flag
10  DJNZ R3,AGAIN              ;if R3 not zero then
11                             ;reload timer
12  END
13
14  ;TH-TL=0108H=264 in decimal
15  ;65536-264=65272
16  ;65272x1.085us=70.820ms
17  ;200x70.820ms=14.164024s
18
```

PROGRAMMING 8051 TIMERS (for information only)

- **Mode 0**
  - **works like mode 1**
  - **13-bit timer instead of 16bit**
  - **13-bit counter hold values 0000 to 1FFFH**
  - **when the timer reaches its maximum of 1FFFH, it rolls over to 0000, and TF is set**

PROGRAMMING 8051 TIMERS

- **Mode 2 programming**
  - **8-bit timer, allows values of 00 to FFH**
  - **TH is loaded with the 8-bit value**
  - **a copy is given to TL**
  - **timer is started by ,"SETB TR0" or "SETB TR1“**
  - **starts to count up by incrementing the TL register**
  - **counts up until it reaches its limit of FFH**
  - **when it rolls over from FFH to 00, it sets high TF**
  - **TL is reloaded automatically with the value in TH**
  - **To repeat, clear TF**
  - **mode 2 is an auto-reload mode**

PROGRAMMING 8051 TIMERS

- **Steps to program in mode 2**
  1. **load TMOD, select mode 2**
  2. **load the TH**
  3. **start timer**
  4. **monitor the timer flag (TF) with "JNB”**
  5. **get out of the loop when TF=1**
  6. **clear TF**
  7. **go back to Step 4 since mode 2 is auto-reload**

Example
Assuming that XTAL = 11.0592 MHz, find (a) the frequency of the square wave generated on pin P1.0 and (b) the smallest frequency achievable in this program, and the TH value to do that.

```
01  MOV TMOD,#20H          ;T1/mode 2/8-bit/auto-reload
02  MOV TH1,#5             ;TH1 = 5
03  SETB TR1               ;start Timer 1
04  BACK: JNB TF1,BACK     ;stay until timer rolls
05  CPL P1.0               ;comp. P1.0 to get hi, lo
06  CLR TF1                ;clear Timer 1 flag
07  SJMP BACK              ;mode 2 is auto-reload
08
09  END
10
11  ;(a)T = 2 x 272.33 gs = 544.67 us and the frequency = 1.83597 kHz
12  ;(b)smallest frequency, T = 00, T = 2 x 256 x 1.085 us = 555.52 us
13  ;   frequency = 1.8 kHz
```

PROGRAMMING 8051 TIMERS

- **Assemblers and negative values**
  - **can let the assembler calculate the value for TH and TL which makes the job easier**
  - **"MOV TH1, # -100", the assembler will calculate the -100 = 9CH**
  - **"MOV TH1,#high(-10000) "**
  - **"MOV TL1,#low(-10000) "**

COUNTER PROGRAMMING (for information only)

- **C/T bit in TMOD register**
  - **used as a timer, the 8051's crystal is used as the source of the frequency**
  - **used as a counter, pulse outside the 8051 increments the TH, TL registers**
  - **counter mode, TMOD and TH, TL registers are the same as for the timer**
  - **timer modes are the same as well**

COUNTER PROGRAMMING (for information only)

- **C/T bit in TMOD register**
  - C/T bit in the TMOD register decides the source of the clock for the timer
  - C/T = 0, timer gets pulses from crystal
  - C/T = 1, the timer used as counter and gets pulses from outside the 8051
  - C/T = 1, the counter counts up as pulses are fed from pins 14 and 15
  - pins are called T0 (Timer 0 input) and T1 (Timer 1 input)
  - these two pins belong to port 3
  - Timer 0, when C/T = 1, pin P3.4 provides the clock pulse and the counter counts up for each clock pulse coming from that pin
  - Timer 1, when C/T = 1 each clock pulse coming in from pin P3.5 makes the counter count up

COUNTER PROGRAMMING

Pin	Port Pin	Function	Description
14	P3.4	T0	Timer/Counter 0 external input
15	P3.5	T1	Timer/Counter 1 external input

(MSB) (LSB)

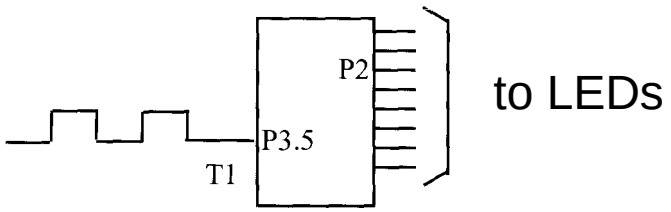
GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer 1				Timer 0			

Table 9–1 Port 3 Pins Used For Timers 0 and 1

Example 18

Assuming that clock pulses are fed into pin T1, write a program for counter 1 in mode 2 to count the pulses and display the state of the TL1 count on P2. (for information only)

```
01  MOV TMOD,#01100000B        ;counter 1,mode 2,C/T=1
02                              ;external pulses
03  MOV TH1,#0                 ;clear TH1
04  SETB P3.5                  ;make T1 input
05  AGAIN: SETB TR1            ;start the counter
06  BACK: MOV A,TL1            ;get copy of count TL1
07  MOV P2,A                   ;display it on port 2
08  JNB TF1,BACK               ;keep doing it if TF=0
09  CLR TR1                    ;stop the counter 1
10  CLR TF1                    ;make TF=0
11  SJMP AGAIN                 ;keep doing it
12
13  END
14
```



to LEDs

P2 is connected to 8 LEDs and input T1 to pulse.

COUNTER PROGRAMMING

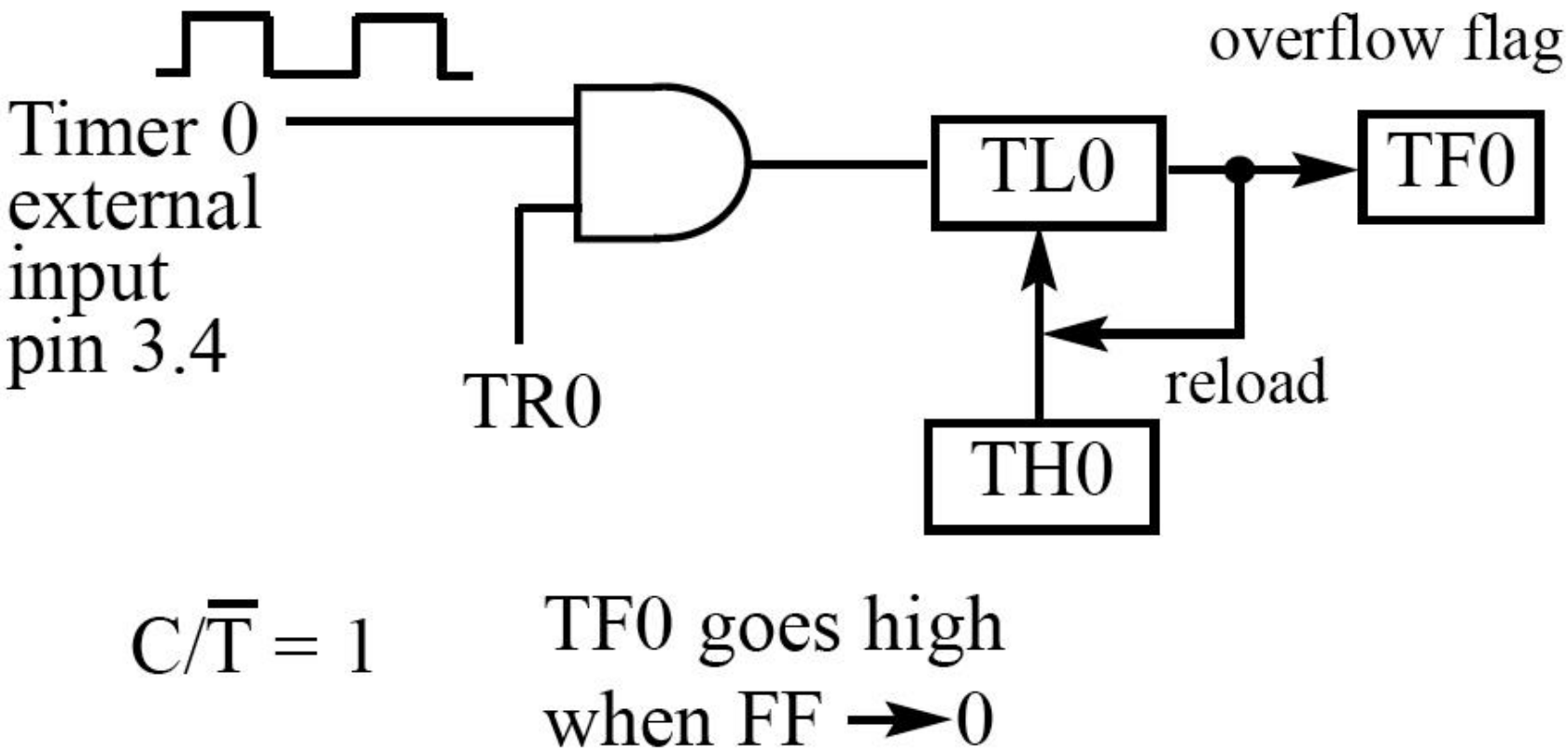


Figure 9–6 Timer 0 with External Input (Mode 2)

COUNTER PROGRAMMING

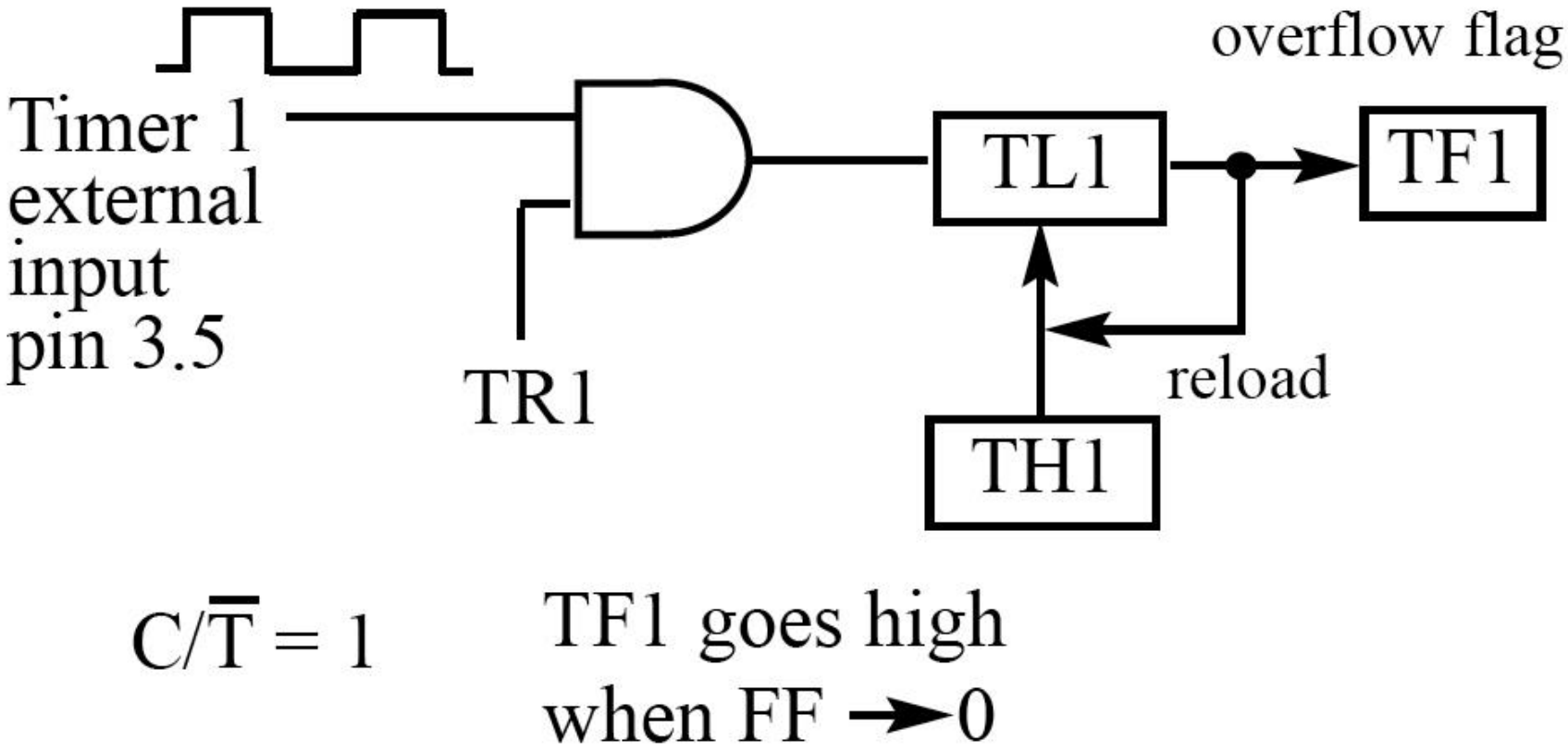
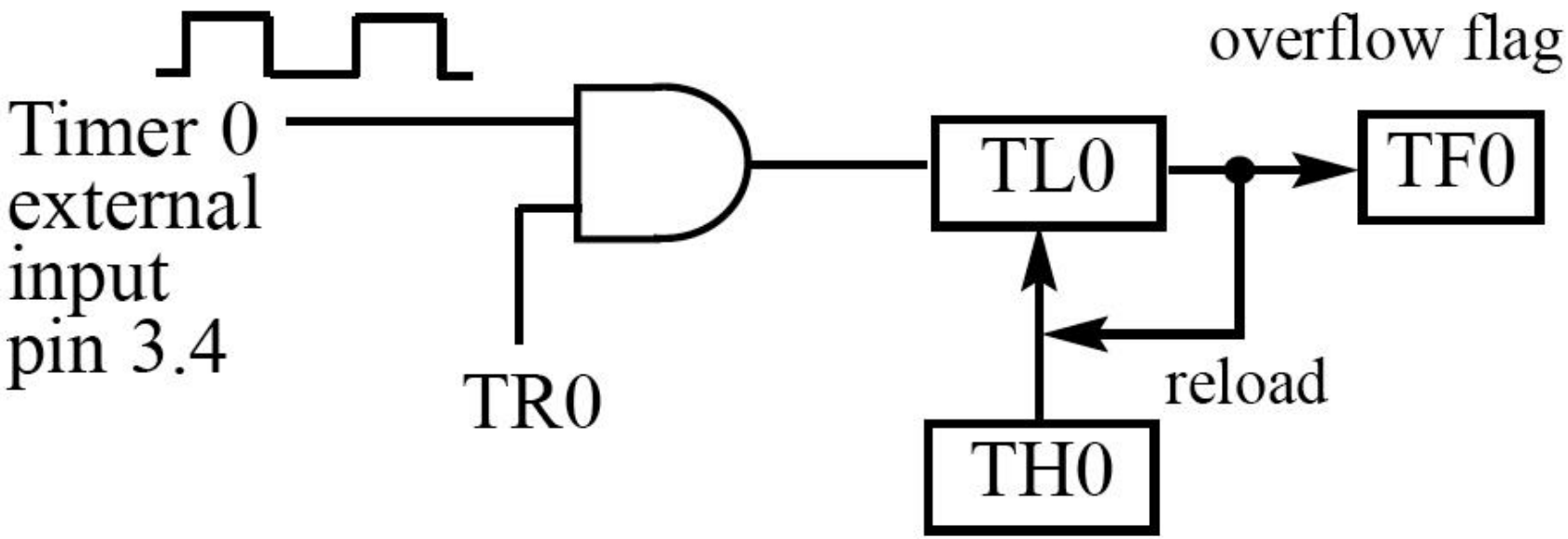


Figure 9–7 Timer 1 with External Input (Mode 2)

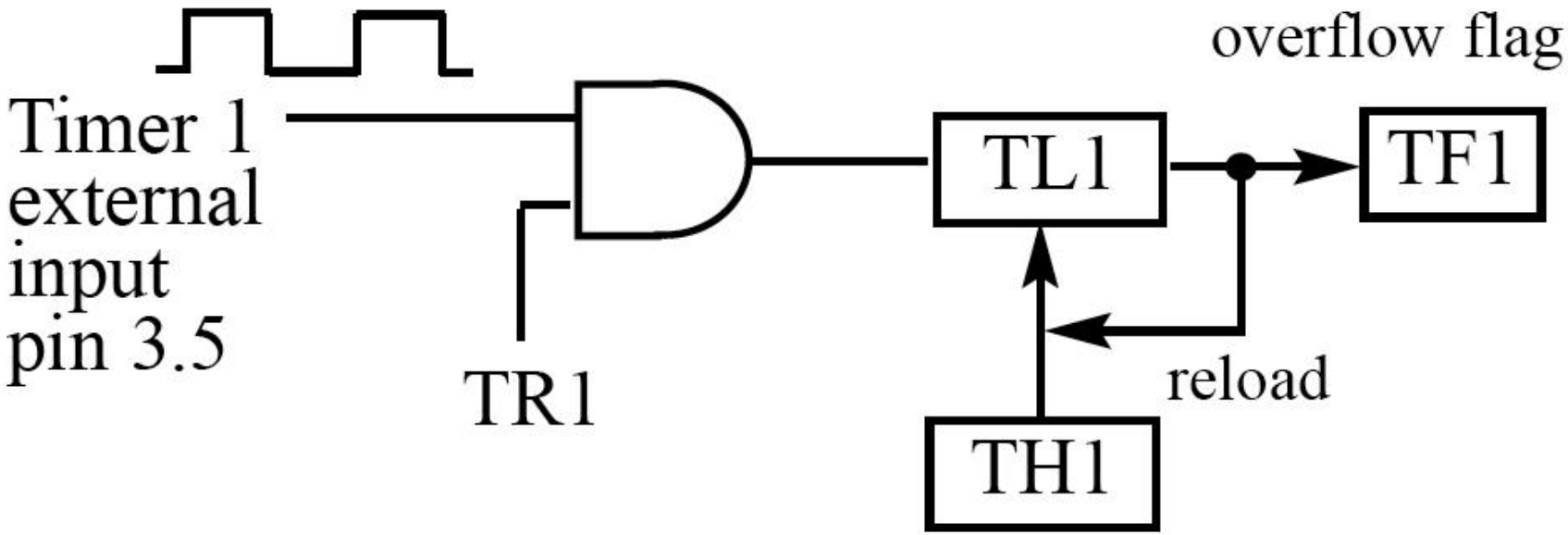
COUNTER PROGRAMMING



$C/\overline{T} = 1$

TF0 goes high
when FF → 0

COUNTER PROGRAMMING



$C/\overline{T} = 1$

TF1 goes high
when FF → 0

COUNTER PROGRAMMING

For Timer 0			
	SETB TR0	=	SETB TCON.4
	CLR TR0	=	CLR TCON.4
	SETB TF0	=	SETB TCON.5
	CLR TF0	=	CLR TCON.5
For Timer 1			
	SETB TR1	=	SETB TCON.6
	CLR TR1	=	CLR TCON.6
	SETB TF1	=	SETB TCON.7
	CLR TF1	=	CLR TCON.7

TCON: Timer/Counter Control Register

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Table 9–1 Port 3 Pins Used For Timers 0 and 1

COUNTER PROGRAMMING

- **TCON register**
  - **TR0 and TR1 flags turn on or off the timers**
  - **bits are part of a register called TCON (timer control)**
  - **upper four bits are used to store the TF and TR bits of both Timer 0 and Timer 1**
  - **lower four bits are set aside for controlling the interrupt bits**
  - **"SETB TRl" and "CLR TRl“**
  - **"SETB TCON. 6" and "CLR TCON. 6“**

COUNTER PROGRAMMING

For Timer 0			
	SETB TR0	=	SETB TCON.4
	CLR TR0	=	CLR TCON.4
	SETB TF0	=	SETB TCON.5
	CLR TF0	=	CLR TCON.5
For Timer 1			
	SETB TR1	=	SETB TCON.6
	CLR TR1	=	CLR TCON.6
	SETB TF1	=	SETB TCON.7
	CLR TF1	=	CLR TCON.7

TCON: Timer/Counter Control Register

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Table 9–2 Equivalent Instructions for the Timer Control Register (TCON)

COUNTER PROGRAMMING

- **The case of GATE = 1 in TMOD**
  - **GATE = 0, the timer is started with instructions** "SETB TR0" and "SETB TR1“
  - **GATE = 1, the start and stop of the timers are done externally through pins P3.2 and P3.3**
  - **allows us to start or stop the timer externally at any time via a simple switch**

COUNTER PROGRAMMING

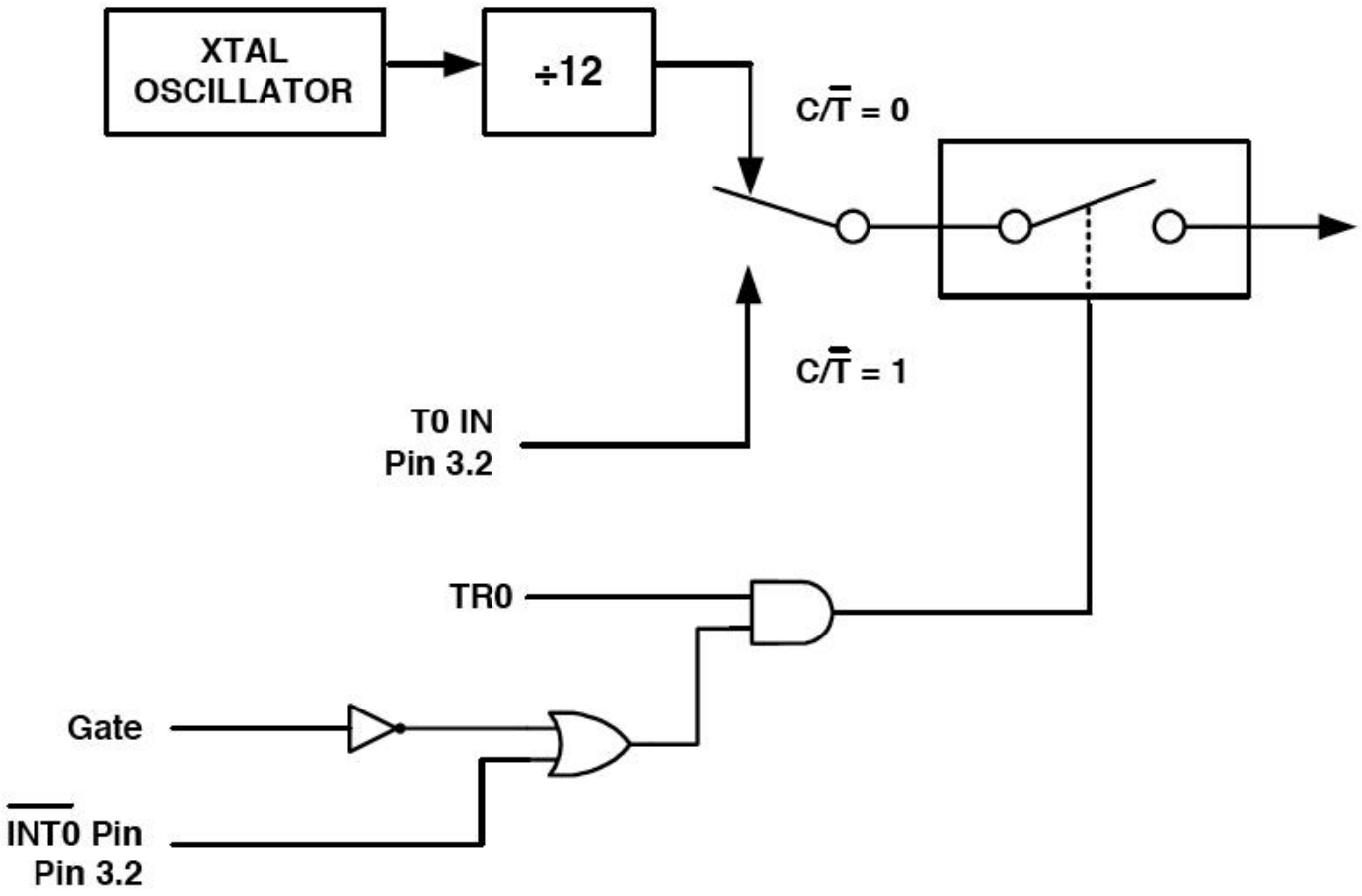


Figure 9–8 Timer/Counter 0

COUNTER PROGRAMMING

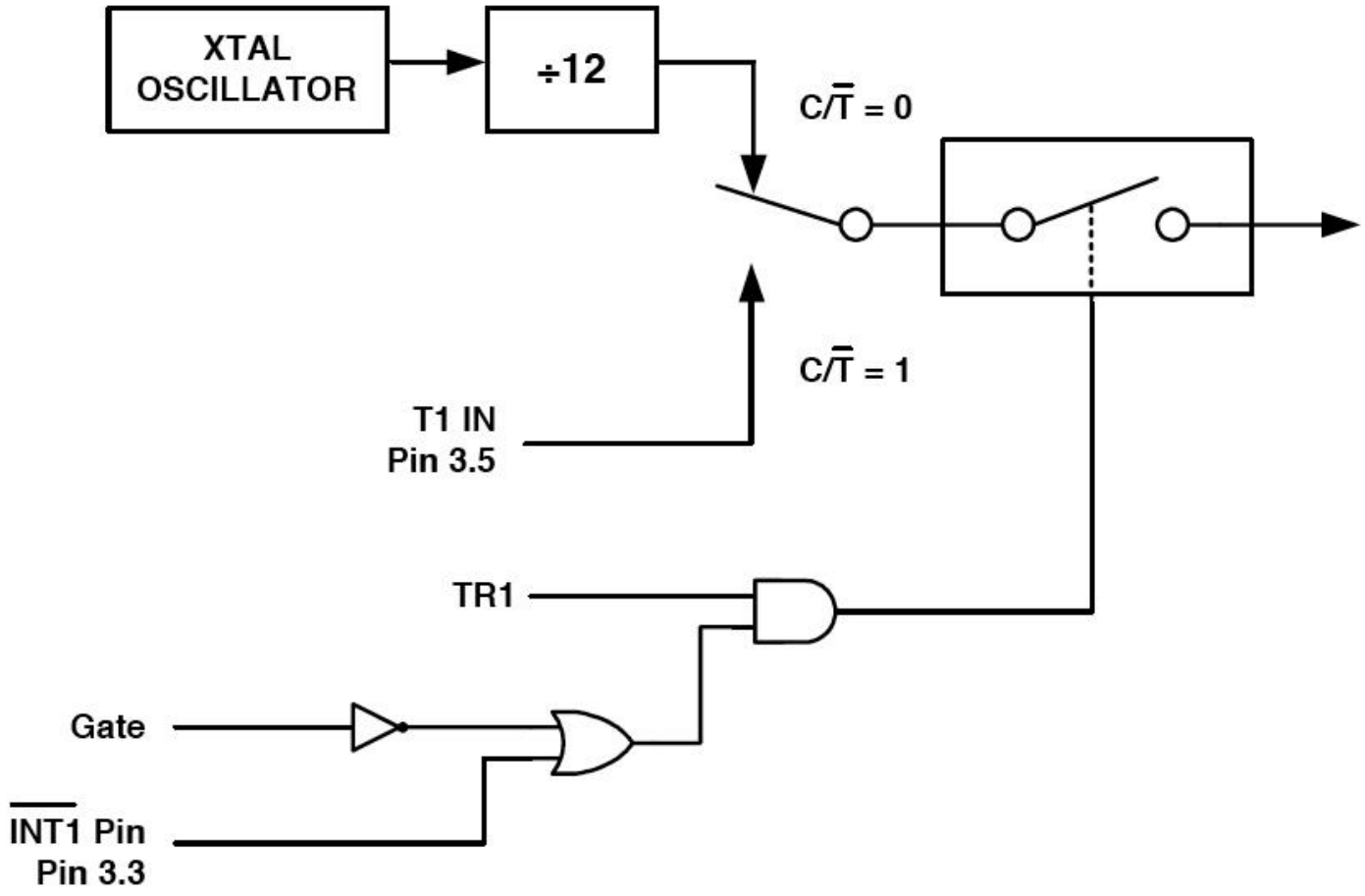


Figure 9–9 Timer/Counter 1

Unit -4
KEYBOARD/DISPLAY CONTROLLER - 8279

Features of 8279

The important features of 8279 are,

- Simultaneous keyboard and display operations.
- Scanned keyboard mode.
- Scanned sensor mode.
- 8-character keyboard FIFO.
- 1 6-character display.
- Right or left entry 1 6-byte display RAM.
- Programmable scan timing.

Pin details

- **A0: Selects data (0) or control/status (1) for reads and writes between micro and 8279.**

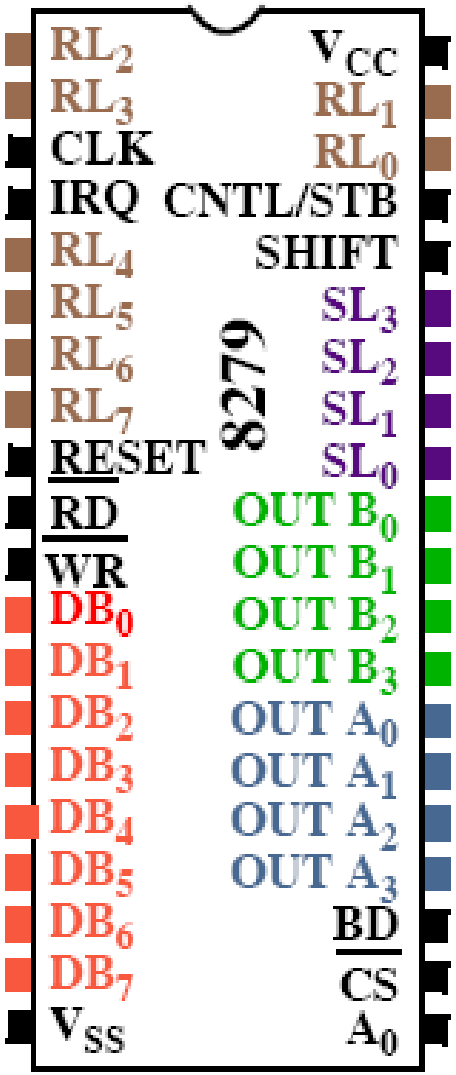
- **$\overline{BD}$: Output that blanks the displays.**

- **CLK: Used internally for timing. Max is 3 MHz.**

- **CN/ST: Control/strobe, connected to the control key on the keyboard.**

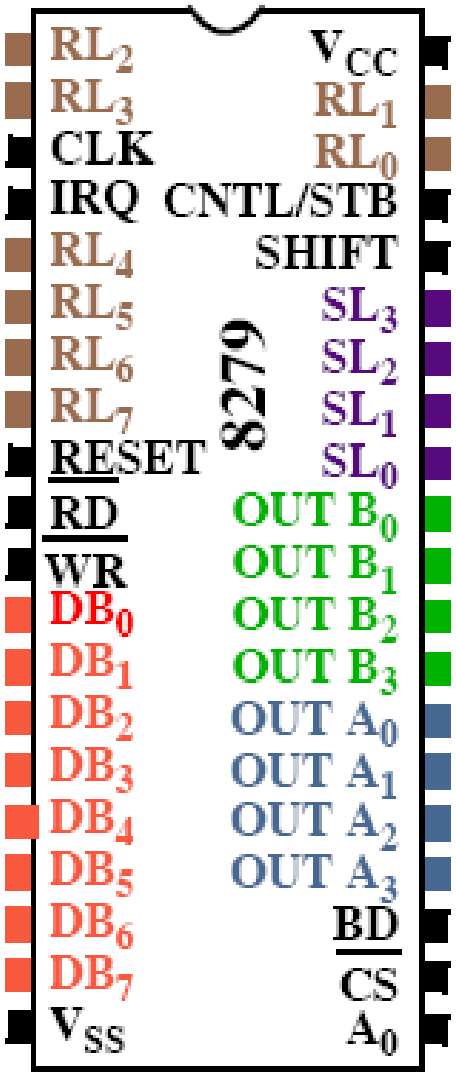
- **$\overline{CS}$: Chip select that enables programming, reading the keyboard, etc.**

- **DB7-DB0: Consists of bi-directional pins that connect to data bus on micro.**

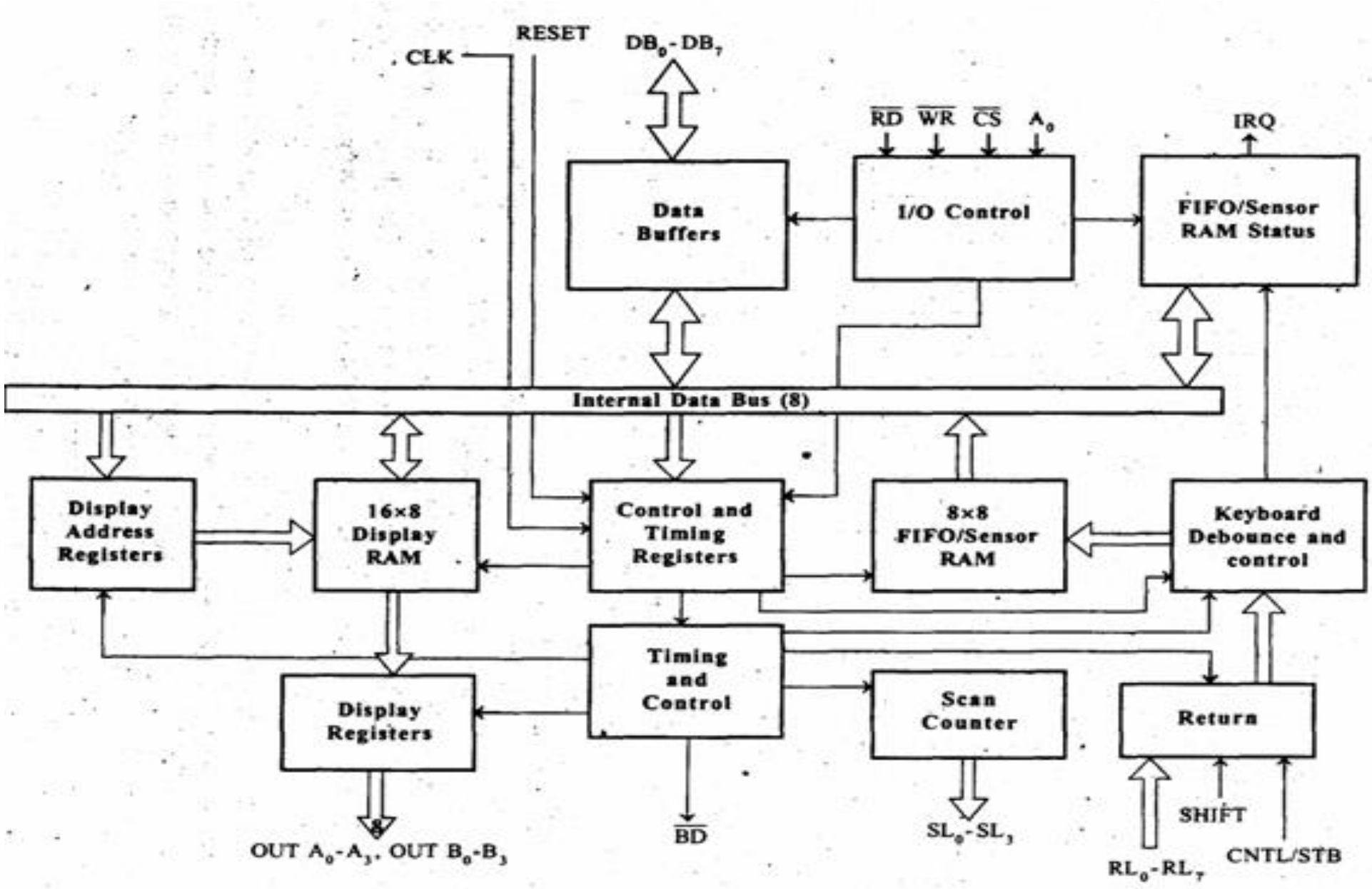


Pin details

- **IRQ: Interrupt request, becomes 1 when a key is pressed, data is available.**
- **OUT A3-A0/B3-B0: Outputs that sends data to the most significant/least significant nibble of display.**
- **$\overline{RD}$($\overline{WR}$) Connects to micro's IORC or RD signal, reads data/status registers.**
- **RESET: Connects to system RESET.**
- **RL7-RL0: Return lines are inputs used to sense key depression in the keyboard matrix.**
- **Shift: Shift connects to Shift key on keyboard.**
- **SL3-SL0: Scan line outputs scan both the keyboard and displays.**



Block diagram of 8279



Sections

- Keyboard
- Display
- Scan
- CPU interface

Keyboard section

- The keyboard section consists of eight return lines RL0 - RL7 that can be used to form the columns of a keyboard matrix.
- It has two additional input : shift and control/strobe. The keys are automatically debounced.
- The two operating modes of keyboard section are 2-key lockout and N-key rollover.
- In the 2-key lockout mode, if two keys are pressed simultaneously, only the first key is recognized.
- In the N-key rollover mode simultaneous keys are recognized and their codes are stored in FIFO.
- The keyboard section also have an 8 x 8 FIFO (First In First Out) RAM.
- The FIFO can store eight key codes in the scan keyboard mode. The status of the shift key and control key are also stored along with key code.
- The 8279 generate an interrupt signal when there is an entry in FIFO.

Display section

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

Scan section

- The scan section has a scan counter and four scan lines, SL0 to SL3.
- In decoded scan mode, the output of scan lines will be similar to a 2-to-4 decoder.
- In encoded scan mode, the output of scan lines will be binary count, and so an external decoder should be used to convert the binary count to decoded output.
- The scan lines are common for keyboard and display.
- The scan lines are used to form the rows of a matrix keyboard and also connected to digit drivers of a multiplexed display, to turn ON/OFF.

CPU interface section

- The CPU interface section takes care of data transfer between 8279 and the processor.
- This section has eight bidirectional data lines DB0 to DB7 for data transfer between 8279 and CPU.
- It requires two internal address A =0 for selecting data buffer and A = 1 for selecting control register of8279.
- The control signals WR (low), RD (low), CS (low) and A0 are used for read/write to 8279.
- It has an interrupt request line IRQ, for interrupt driven data transfer with processor.
- The 8279 require an internal clock frequency of 100 kHz. This can be obtained by dividing the input clock by an internal prescaler.
- The RESET signal sets the 8279 in 16-character display with two -key lockout keyboard modes.

Control Word Description:

First three bits given below select one of 8 control registers (opcode).

- ***000DDMMM***

***Mode set*: Opcode 000.**

DD sets displays mode.

MMM sets keyboard mode.

DD field selects either:

- **8- or 16-digit display**
- **Whether new data are entered to the rightmost or leftmost display position.**

DD	*Function*
00	8-digit display with left entry
01	16-digit display with left entry
10	8-digit display with right entry
11	16-digit display with right entry

Control Word Description:

MMM field:

MMM

DD	*Function*
000	Encoded keyboard with 2-key lockout
001	Decoded keyboard with 2-key lockout
010	Encoded keyboard with N-key rollover
011	Decoded keyboard with N-key rollover
100	Encoded sensor matrix
101	Decoded sensor matrix
110	Strobed keyboard, encoded display scan
111	Strobed keyboard, decoded display scan

- ***Encoded Mode:*** **SL outputs are active-high, follow binary bit pattern 0-7 or 0-15 depending on 8 or 16 digit display.**

- ***Decoded Mode:*** **SL outputs are active-low (only one of the four outputs will be low at any time).Pattern output: 1110, 1101, 1011, 0111.**

I/O Interface

Control Word Description:

- ***Strobe :*** **An active high pulse on the CN/ST input pin strobes data from the RL pins into an internal FIFO for reading by micro later.**

- ***2-key lockout/N-key rollover:*** **Prevents 2 keys from being recognized if pressed simultaneously/Accepts all keys pressed from 1st to last.**

Write display format

- ***100ZAAAA***

write display Selects address – to write address of one of the

Display. Z selects auto-increment so subsequent writes go to subsequent display positions.

Clear Display format

- ***1100CCFA***
- **The clear control word clears the display, FIFO or both**
- **Bit F clears FIFO and the display RAM status, and sets address pointer to 000.**
- **If CC are 00 or 01, all display RAM locations become 00000000.**
- **If CC is 10, --> 00100000,**
- **if CC is 11, --> 11111111.**

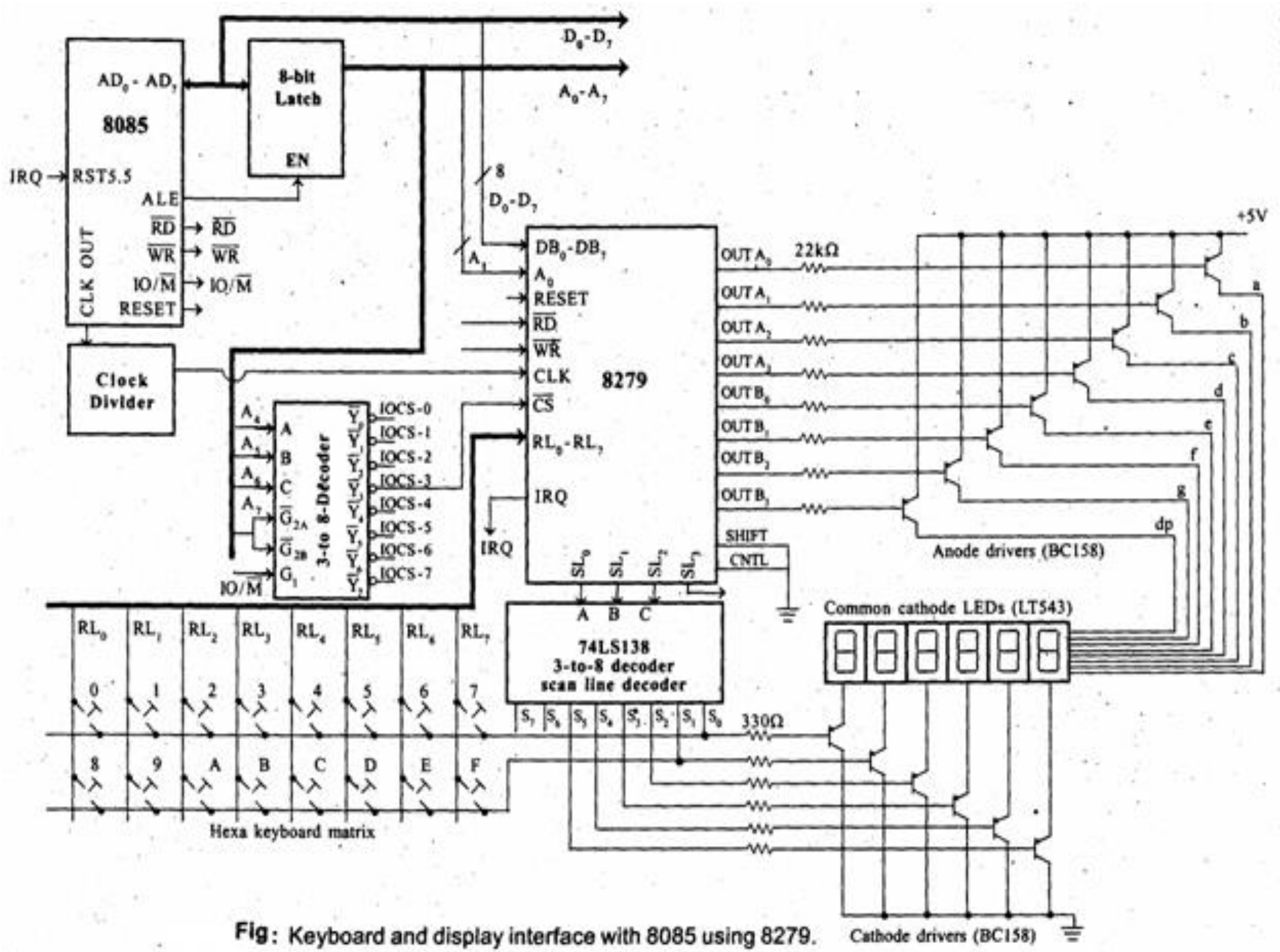


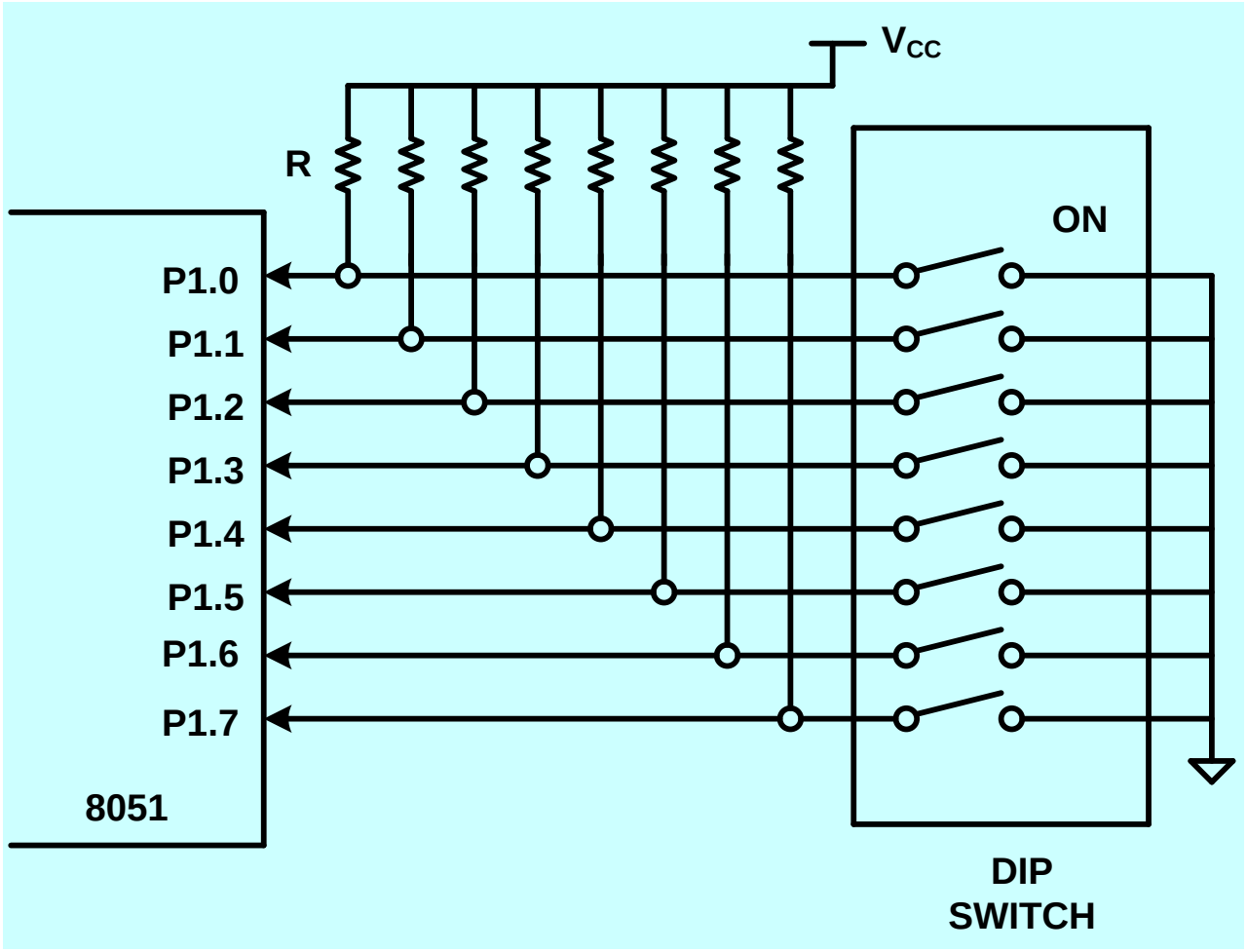
Fig: Keyboard and display interface with 8085 using 8279.

Unit 5
Applications of Microcontroller Systems

Interfacing Keyboard and Display Devices

- **Topics Covered:**
- **Interface switches and keyboard to the 8051**
- **Interface LED displays to the 8051**
- **Overcome Keybounce and multiple key press problems**
- **Design a microcontroller based system with keyboard and display devices**
- **Interface and program the LCD controller**

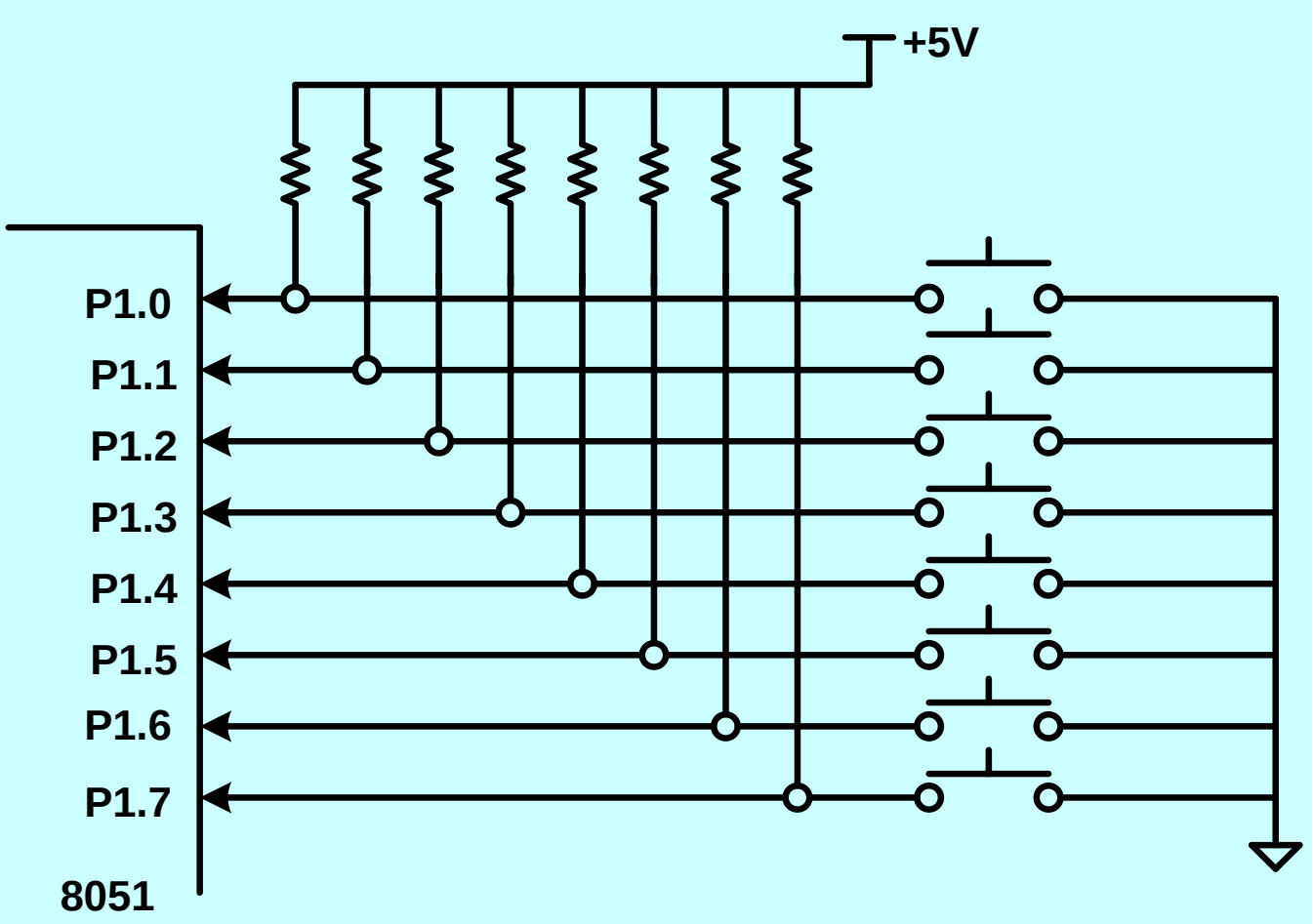
Interfacing Switches



What is a Keyboard ?

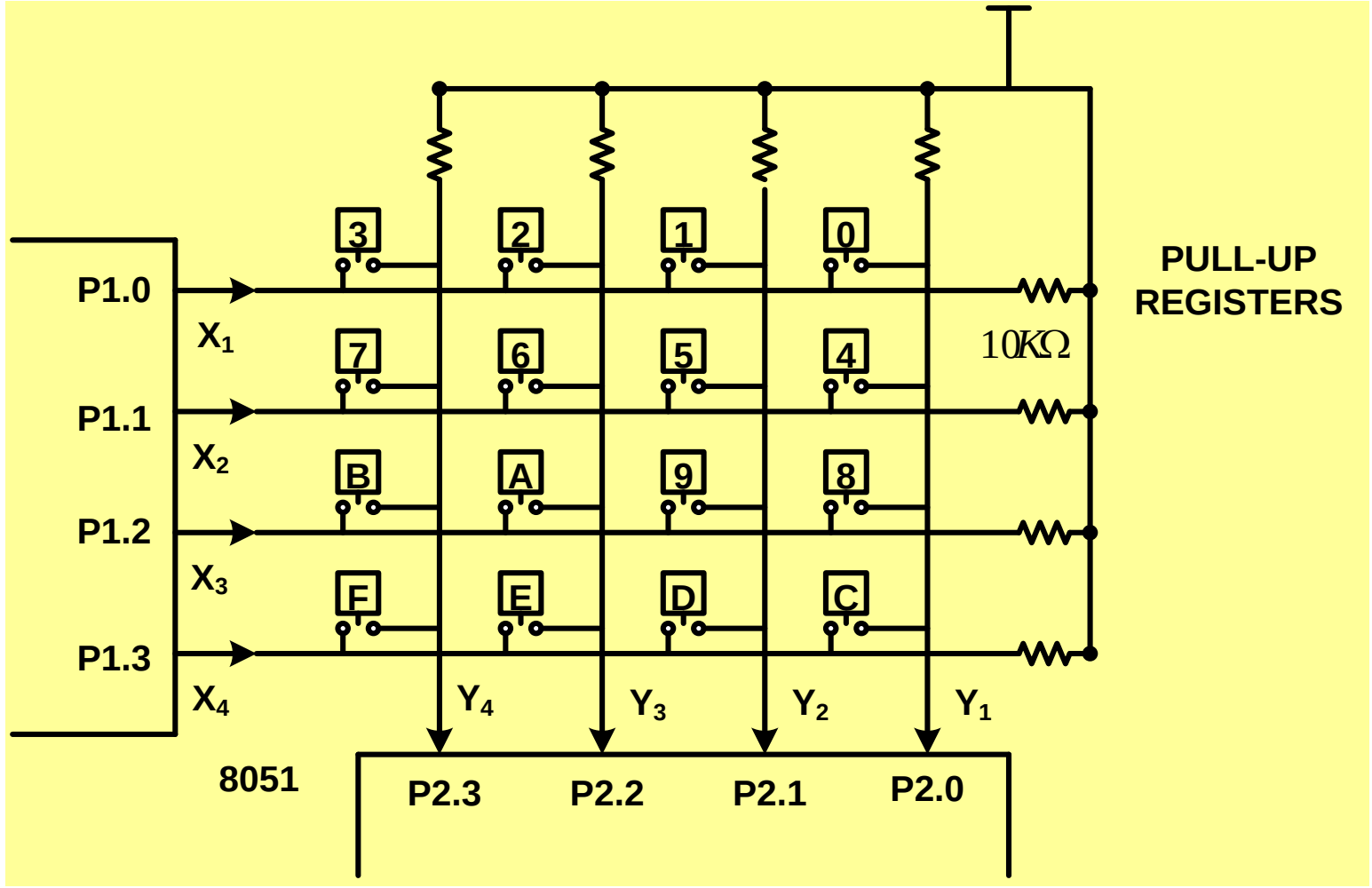
- **Collection of keys interfaced to the microcontroller**
- **Arranged in the form of two dimensional matrix**
- **Matrix arrangement used for minimizing the number of port lines**
- **Junction of each row and column forms the key**

Interfacing a Keyboard



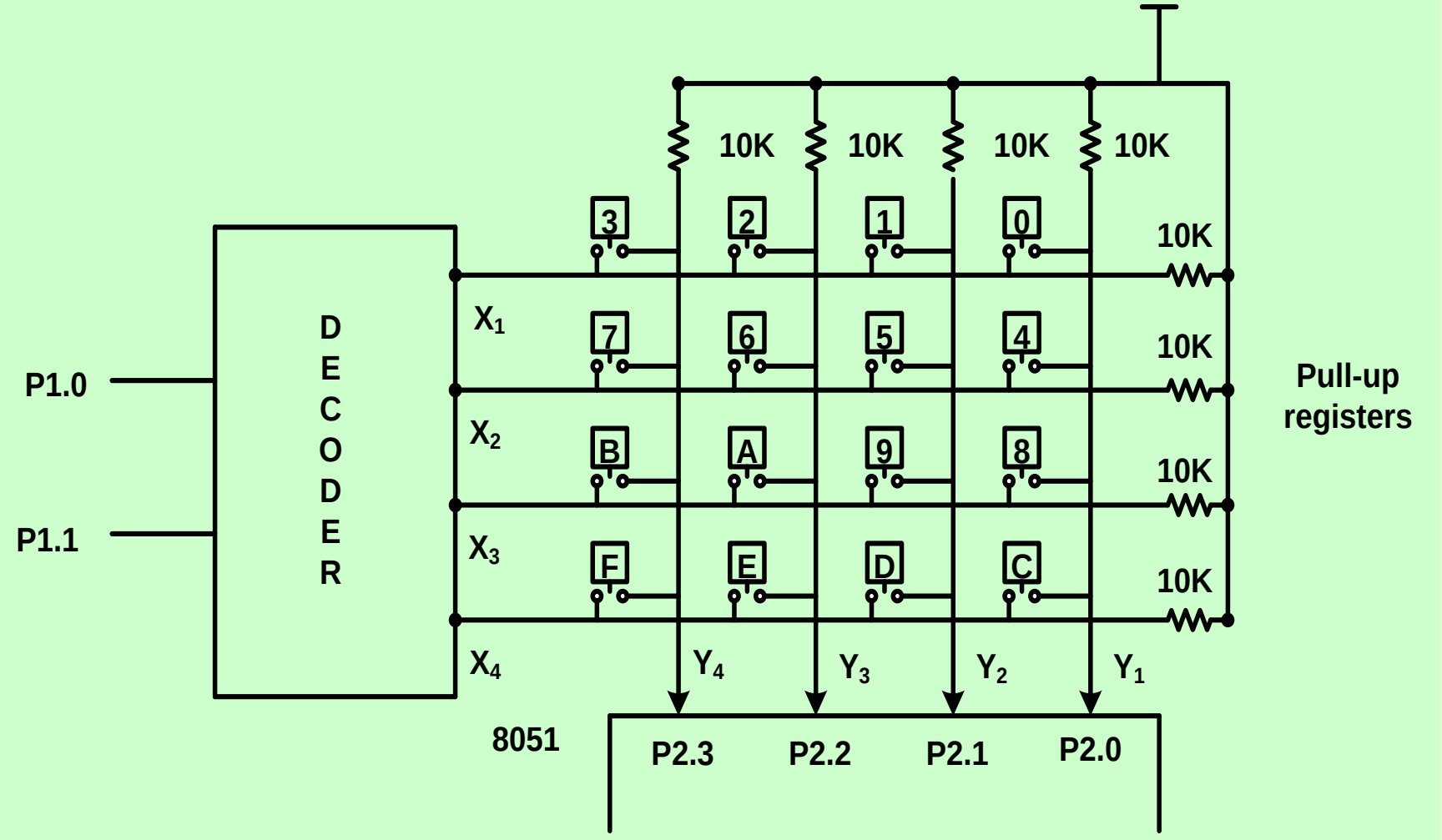
- One key per port line

Interfacing a Keyboard



- **Keys are organized in two-dimensional matrix to minimize the number of ports required for interfacing**

Interfacing a Keyboard

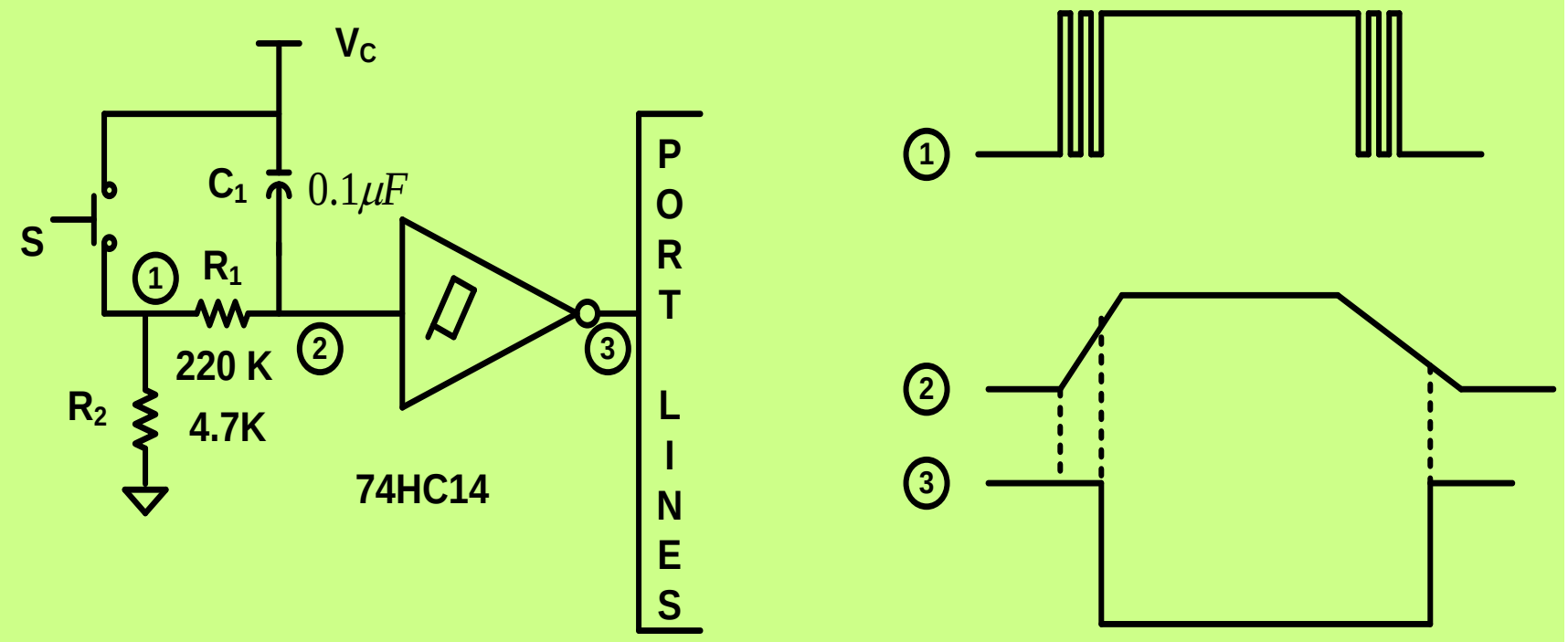


- Use of decoder further reduces the number of port lines required

Key Issues

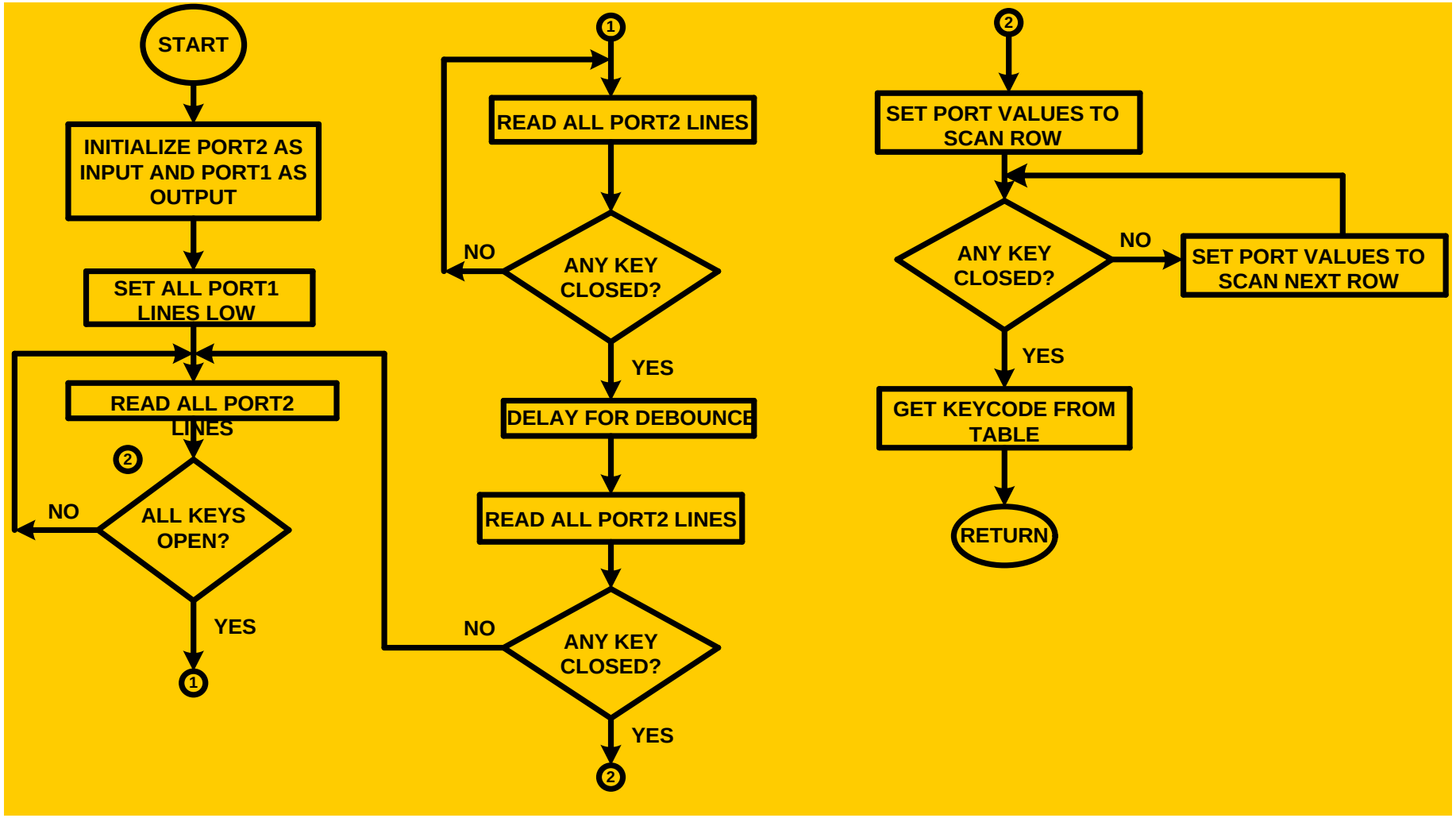
- **Key bounce can be overcome using Software/Hardware approach**
- **Keyboard Scanning**
- **Multiple Key Closure**
  - **2-key lockout**
  - **2-key rollover**
- **Minimize Hardware Requirement:**
  - **Use of Keyboard Encoder**
- **Minimize Software Overhead**

Key bounce



- **Hardware approach to overcome key-bounce**

Keyboard Scanning

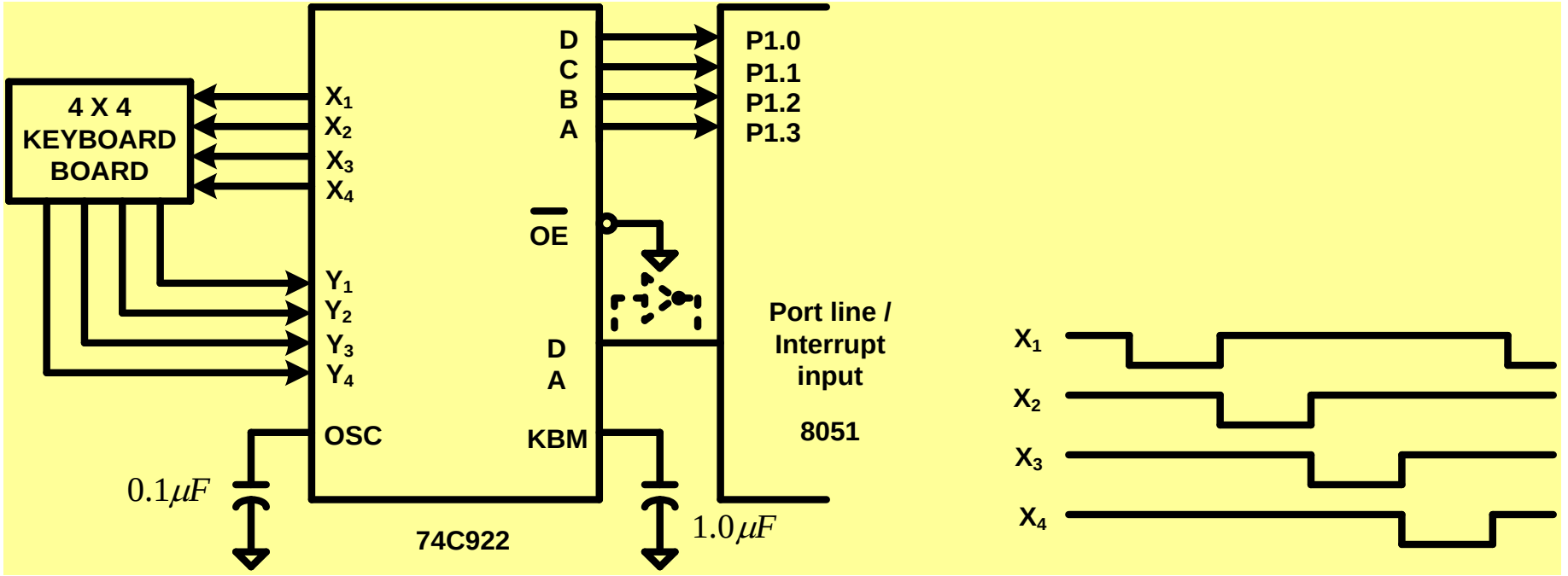


- **Software approach for keyboard scanning**

Hardware Approach

- **Use of an Encoder**
- **Automatically translates key press code into 4-bit number**
- **Built-in scanning circuit**
- **Overcomes key bounce using a single capacitor (1 µF for debounce time of 10 msec)**
- **Keyclosure indicated by an output (DA) line**
- **Last key pressed is stored in a latch**
- **Examples of Encoder**
  - **20 key encoder – 74C923**
  - **16 key encoder - 74C922**

Scanning by Hardware



- **Minimizes software overhead at the expense of extra hardware**

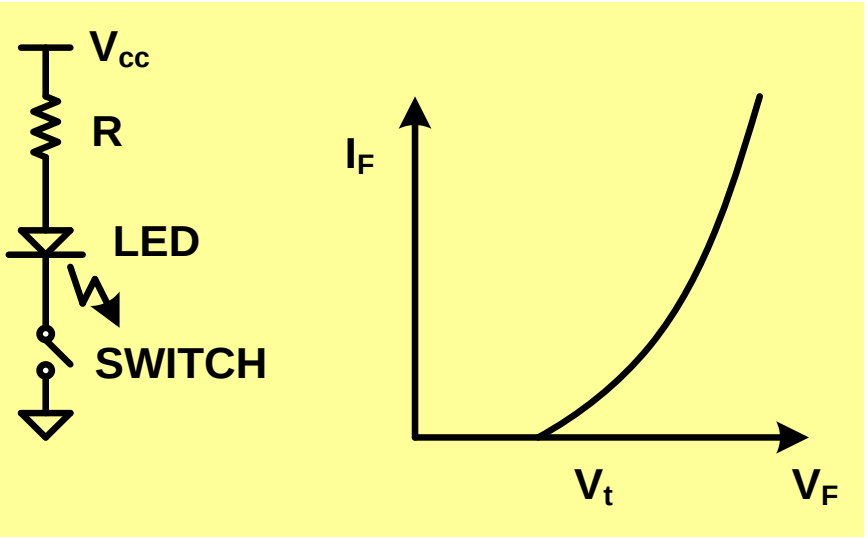
SWITCH CLOSED	DATA OUTPUT
	D C B A
Y_1 X_1	0 0 0 0
Y_1 X_2	0 0 0 1
Y_1 X_3	0 0 1 0
Y_1 X_4	0 0 1 1
Y_2 Y_1	0 1 0 0
Y_2 Y_2	0 1 0 1
Y_2 Y_3	0 1 1 0
Y_2 Y_4	0 1 1 1
⋮	⋮
Y_4 X_4	1 1 1 1

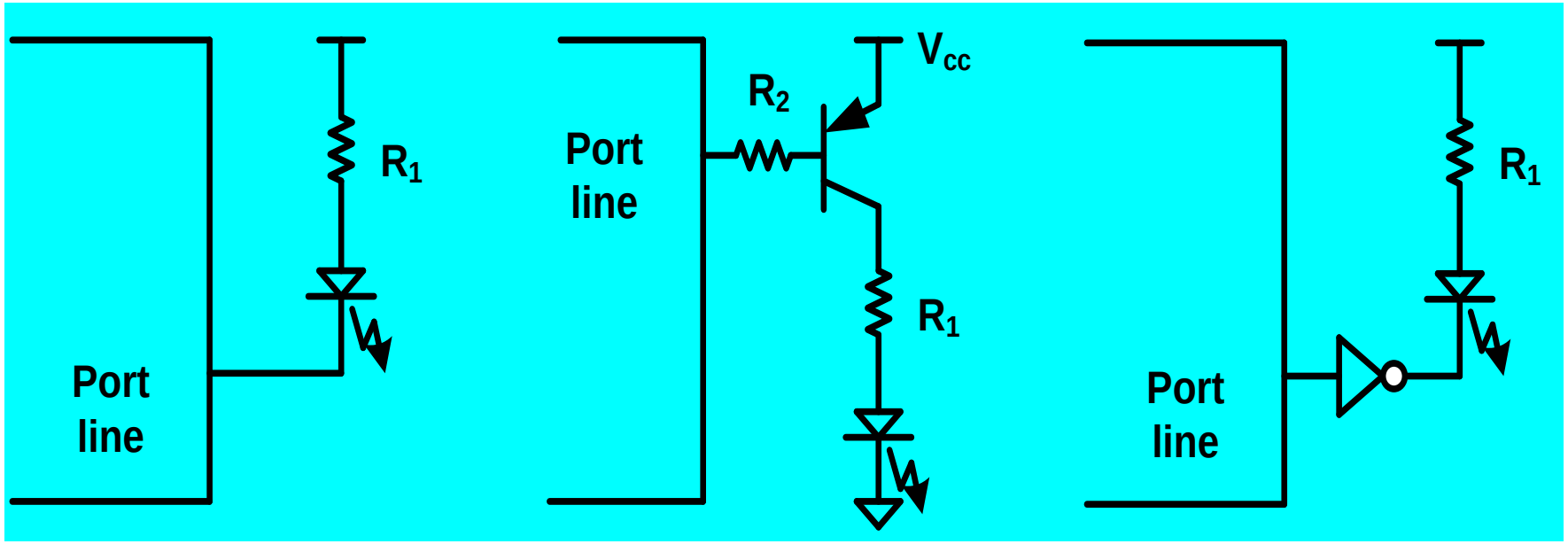
Display Devices

- **Most popular display device: LED**
  - **Very tiny in size**
  - **Available in many colors**
  - **Very reliable and rugged**
  - **Long life**
  - **Operates at low voltage**
  - **Small power consumption**
  - **Visible in darkness**
  - **Single LED**
  - **Bicolor LED**
  - **Seven Segment Displays**
    - **Common Cathode Form (ICM 7218D)**
    - **Common Anode Form (ICM 7218C)**
  - **Consumes large amount of current**

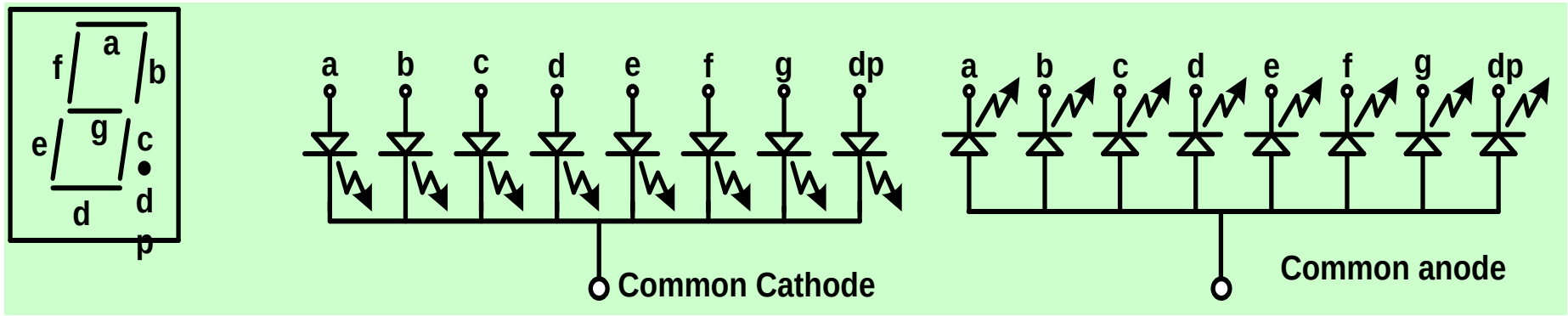
Interfacing a single LED

- **Driver circuit to interface a single LED**

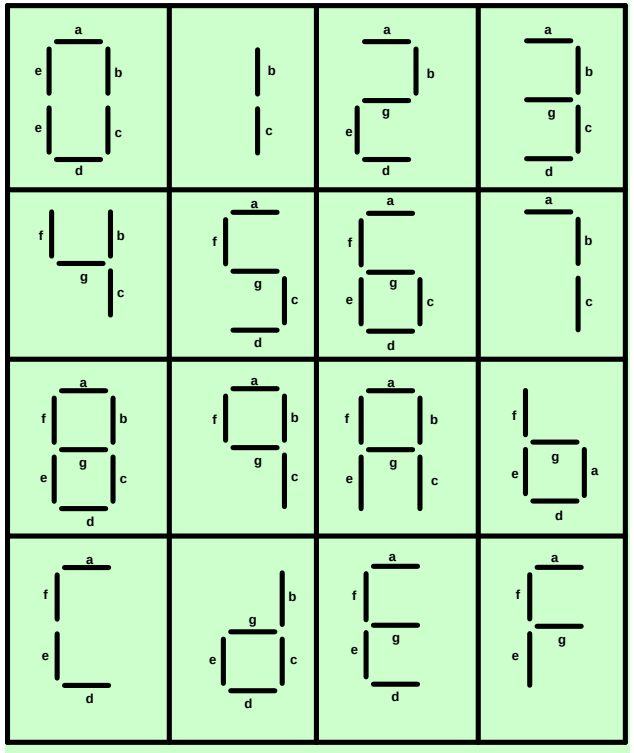




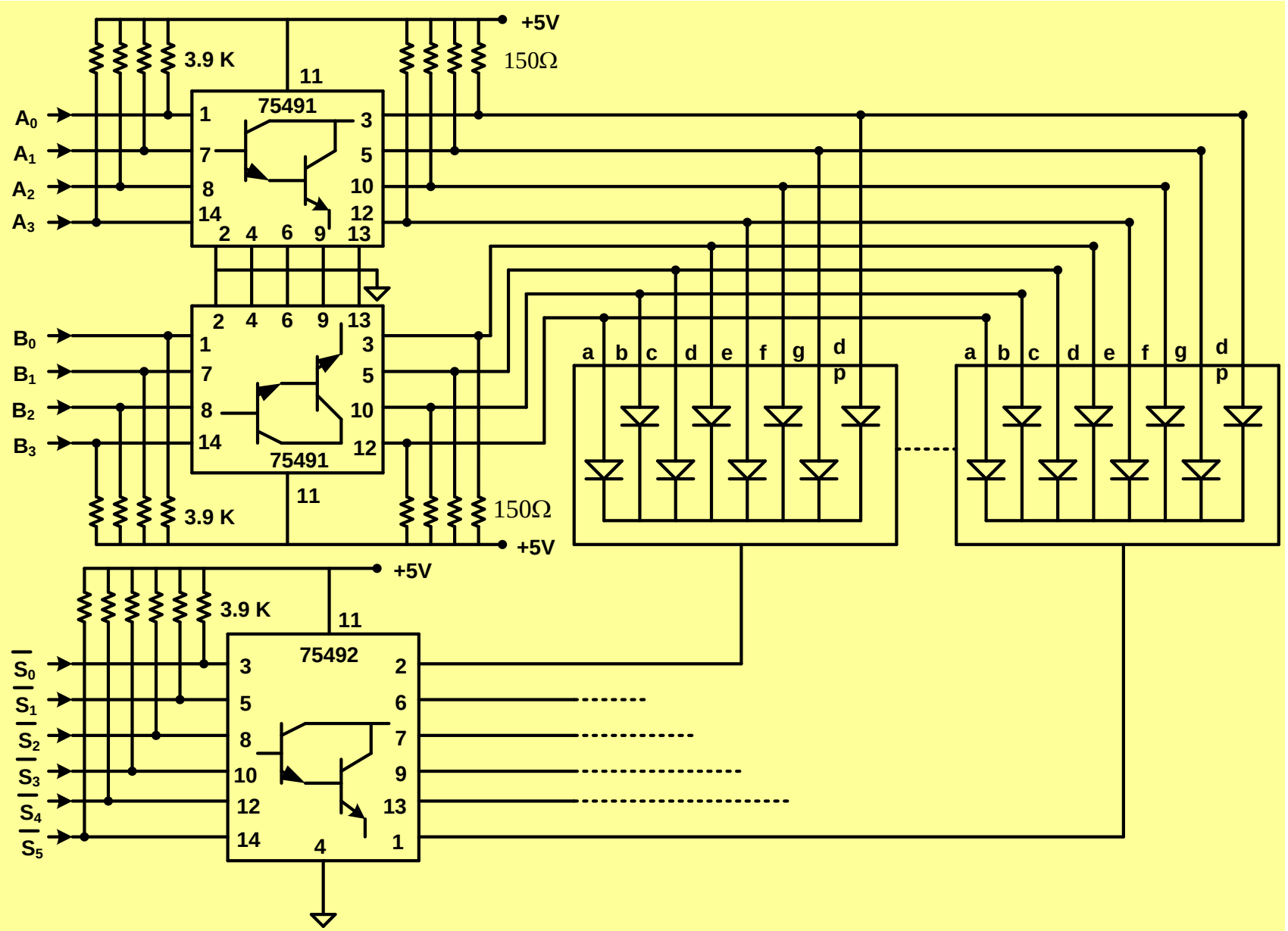
Seven Segment LEDs



- **Two types: Common cathode and common anode type**
- **Seven-segment LEDs can be conveniently used to display HEX characters**



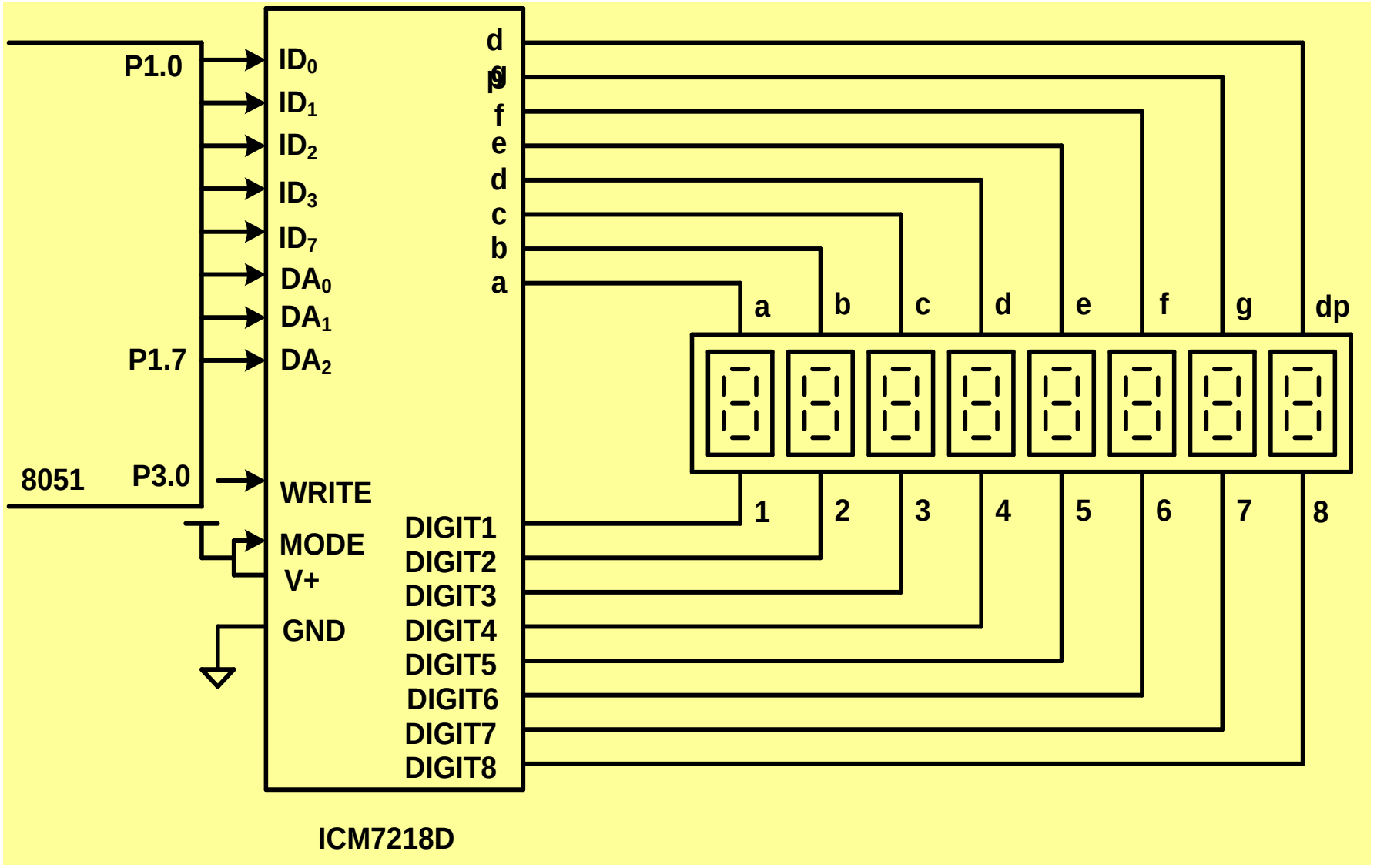
Interfacing multiple 7-Segment LEDs



Multidigit Driver

- **Features of Multidigit Driver**
  - **8-segment driver output lines**
  - **8-digit driver lines**
  - **20 mA peak current**
  - **LEDs can withstand high peak current**
- **Sequencing operation:**
  - **Select data using digit address lines DA_{0-2}**
  - **Write data using ID_{0-3} and ID_7 lines**
- **Three modes of operation:**
  - **HIGH: HEX, LOW: OFF, OPEN: CODED-HELP**

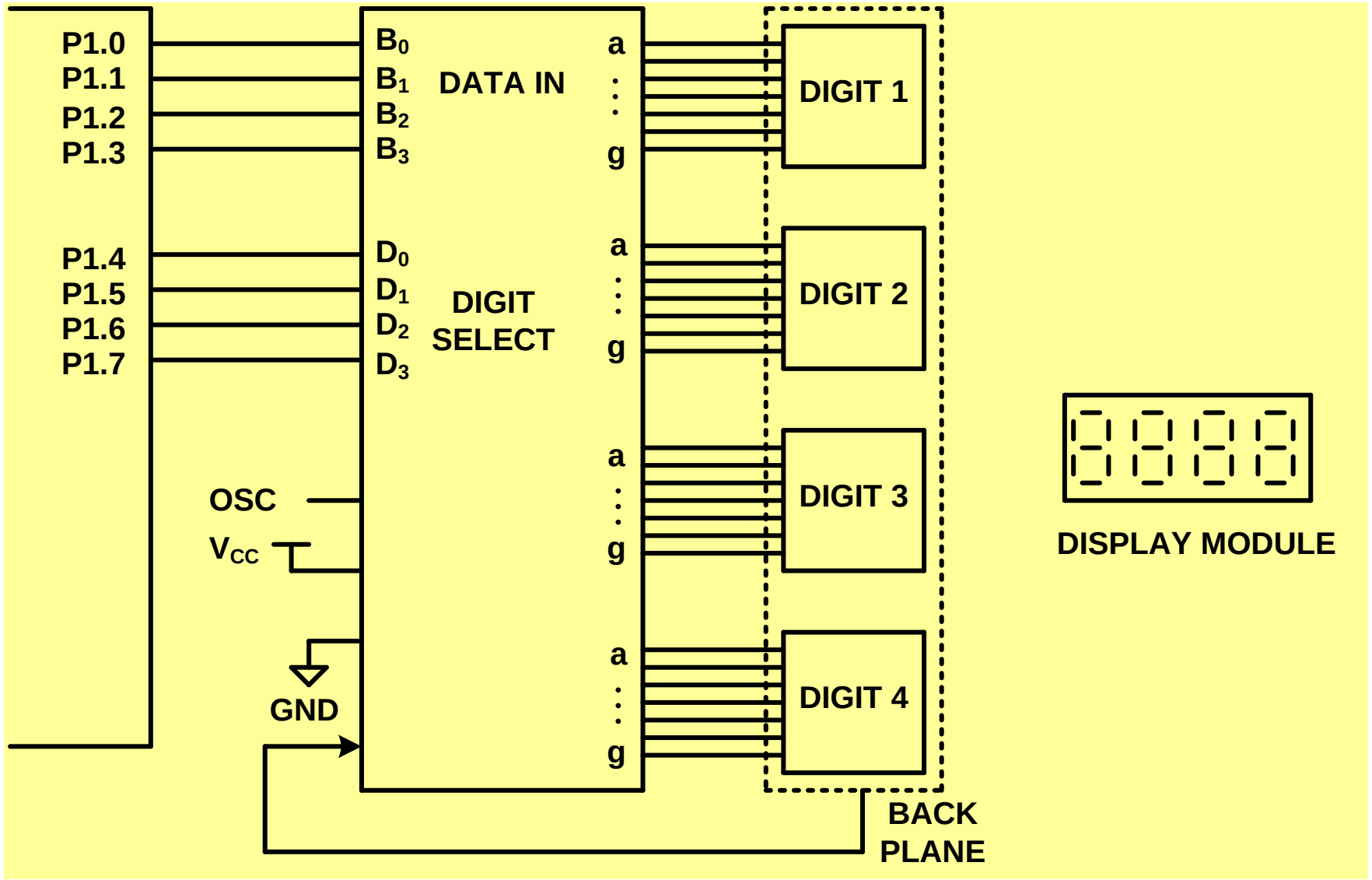
Interfacing using Multidigit Driver



Liquid Crystal Displays

- **Key features:**
  - **Low Power Consumption**
  - **Voltage Controlled**
  - **Easy to read in bright light**
  - **Declining Cost**
  - **Ability to display Characters/Graphics**
  - **Intelligent controller and LCD display panels readily available**

Liquid Crystal Displays

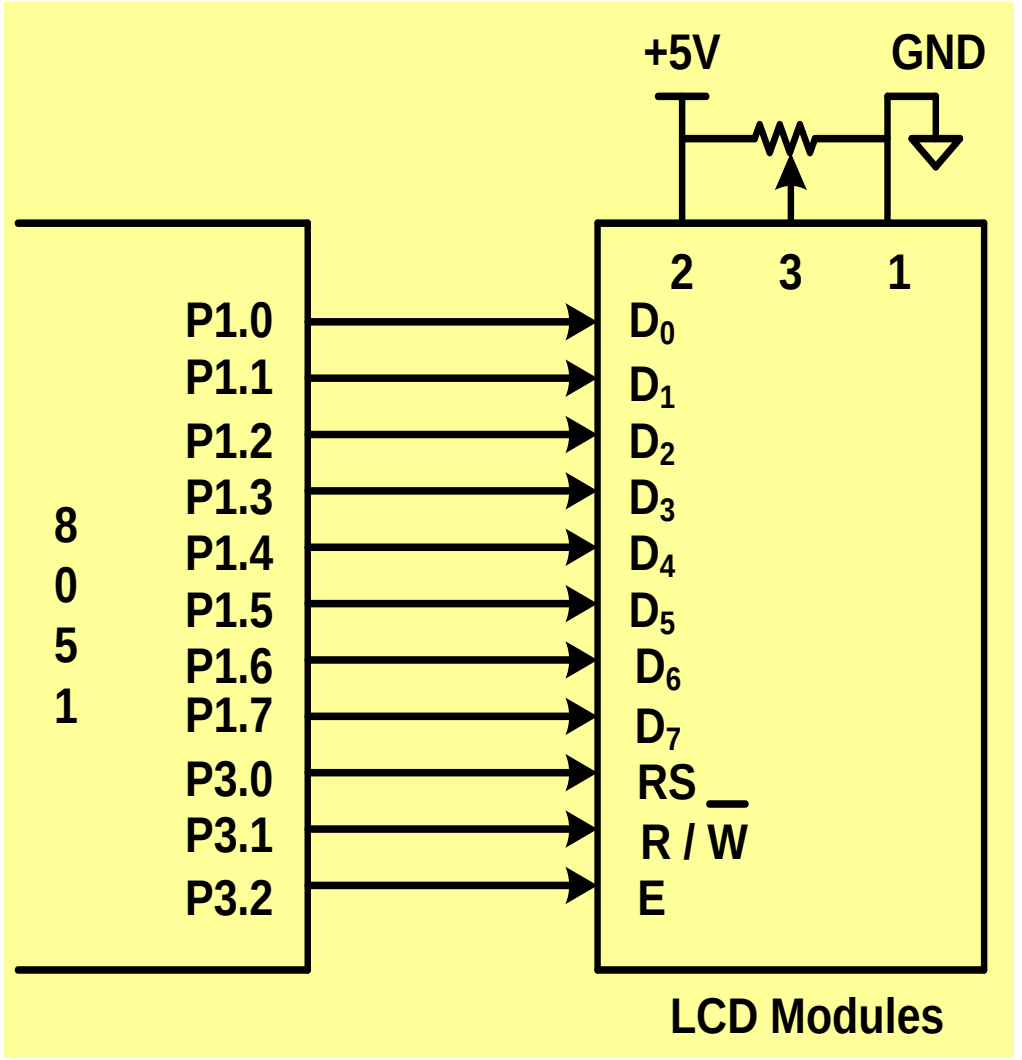


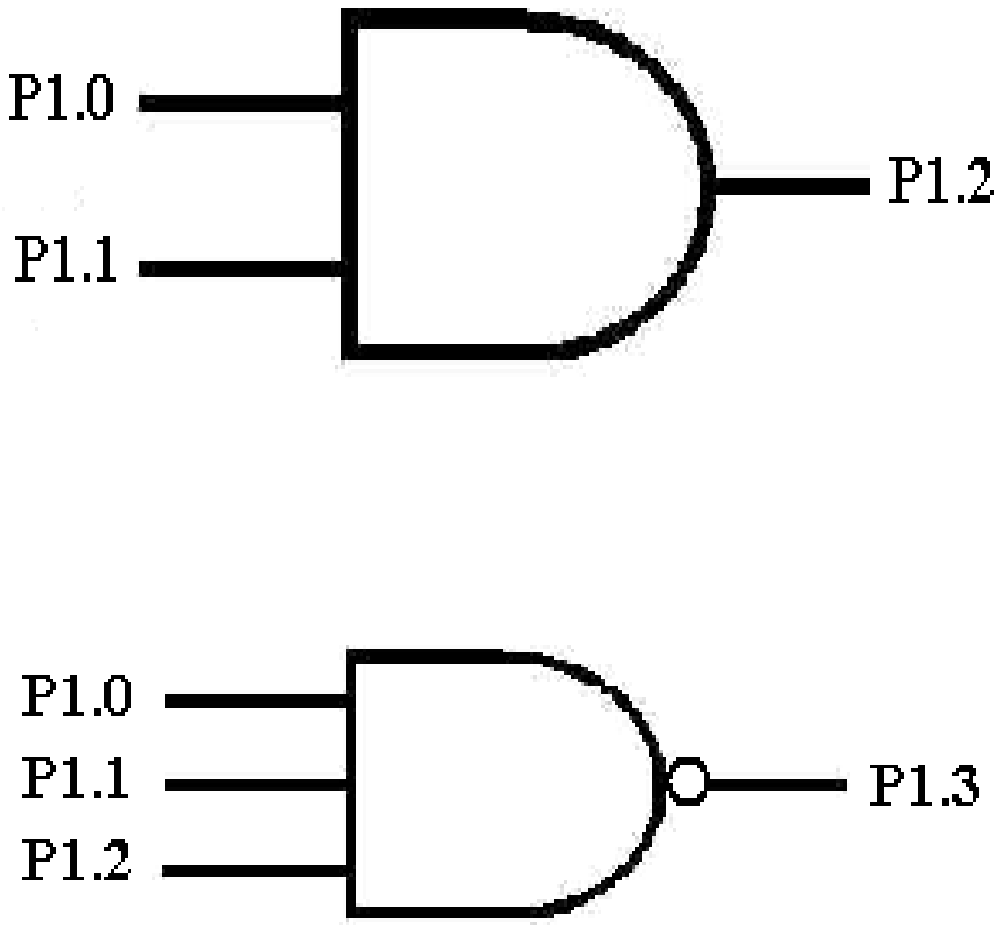
LCD Display Module

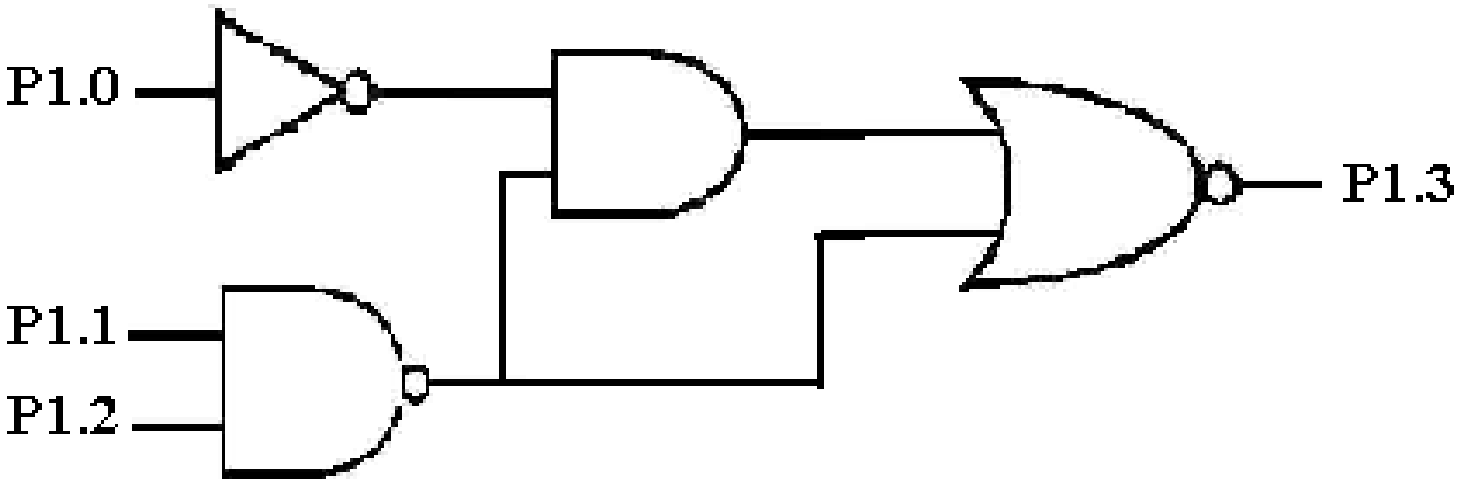
- **LCD modules:**
  - **An LCD panel and small circuit board containing the controller chip**
  - **14 – pin connections to microcontroller**
  - **HITACHI'S HD44780 controller can control up to 80 characters**
  - **Easy to program**
  - **2 rows, 20/40 character in each row**
  - **Each character can be 5X8 or 5X11 matrix**

LCD Display Module

- CG ROM stores segment pattern of 192 char.
- CG RAM stores segment patterns of 16 user-designed char.
- An 8-bit instruction reg.
- An 8-bit data reg.
- DD RAM stores up to 80 8-bit char. Codes
- 11 instructions clear display, return home




P1.0
P1.1
P1.2
P1.0
P1.1
P1.2
P1.3


P1.0
P1.1
P1.2
P1.3

Thank You