Principle and Interface Techniques of Microcontroller

--8051 Microcontroller and Embedded Systems Using Assembly and C

杭州 • 浙江大学 • 2022

Chapter 4

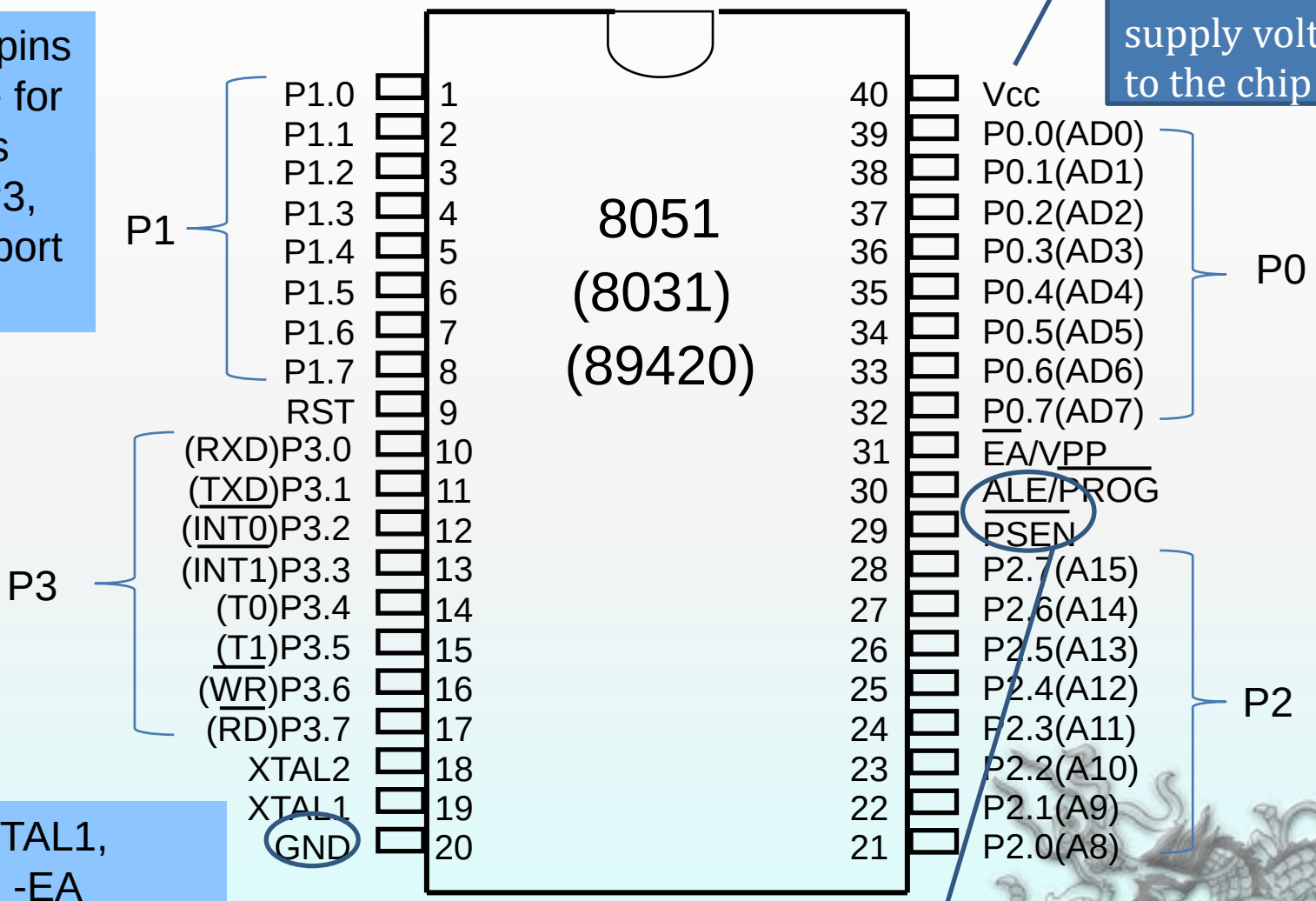
I/O Ports Configuration and Programming

Pin Description

- 8051 family members (e.g, 8751, 89C51, 89C52, DS89C4x0)
  - ✓Have 40 pins dedicated for various functions such as I/O, -RD, -WR, address, data, and interrupts
  - ✓Come in different packages, such as
    DIP(dual in-line package),
    QFP(quad flat package), and
    LLC(leadless chip carrier)
  - ✓Some companies provide a 20-pin version of the 8051 with a reduced number of I/O ports for less demanding applications

Pin Description

A total of 32 pins are set aside for the four ports P0,P1, P2, P3, where each port takes 8 pins



Provides +5V supply voltage to the chip

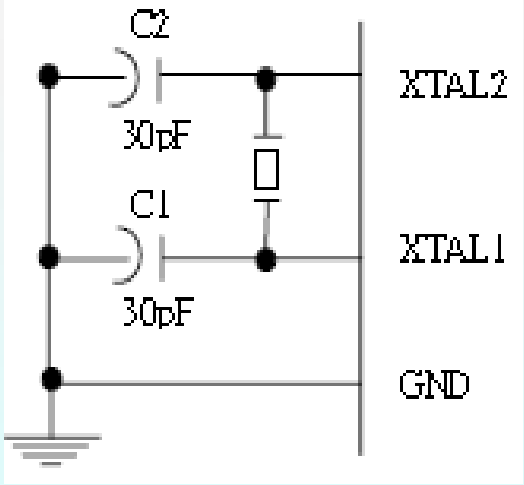
PSEN and ALE are used mainly in 8031-baded systems

Vcc, GND, XTAL1, XTAL2, RST, -EA are used by all members of 8051 and 8031 families

XTAL1 and XTAL2

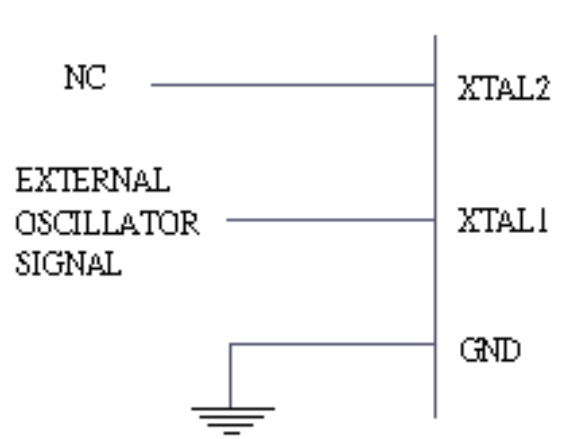
- The 8051 has an on-chip oscillator but requires an external clock to run it
  - ✓A quartz crystal oscillator is connected to inputs XTAL1 (pin19) and XTAL2 (pin18)

    The quartz crystal oscillator also needs two capacitors of 30 pF value



XTAL1 and XTAL2

- If you use a frequency source other than a crystal oscillator, such as a TTL oscillator
  - ✓ It will be connected to XTAL1
  - ✓ XTAL2 is left unconnected



XTAL1 and XTAL2

- The speed of 8051 refers to the maximum oscillator frequency connected to XTAL
  - ✓ex. A 12-MHz chip must be connected to a crystal with 12 MHz frequency or less
  - ✓ We can observe the frequency on the XTAL2 pin using the oscilloscope

RST

- RESET pin is an input and is active high (normally low)
  - ✓Upon applying a high pulse to this pin, the microcontroller will reset and terminate all activities
    - This is often referred to as a power-on reset
    - Activating a power-on reset will cause all values in the registers to be lost

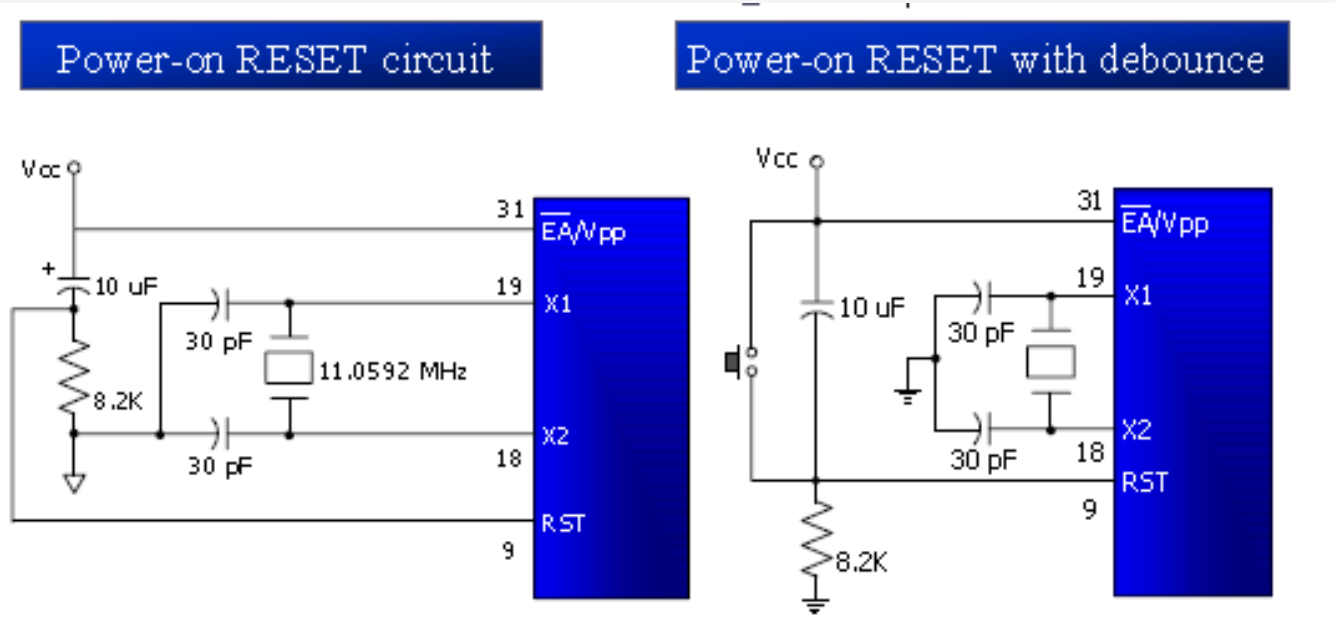
RESET value of some 8051 registers

we must place the first line of source code in ROM location 0

Register	Reset Value
PC	0000
DPTR	0000
ACC	00
PSW	00
SP	07
B	00
P0-P3	FF

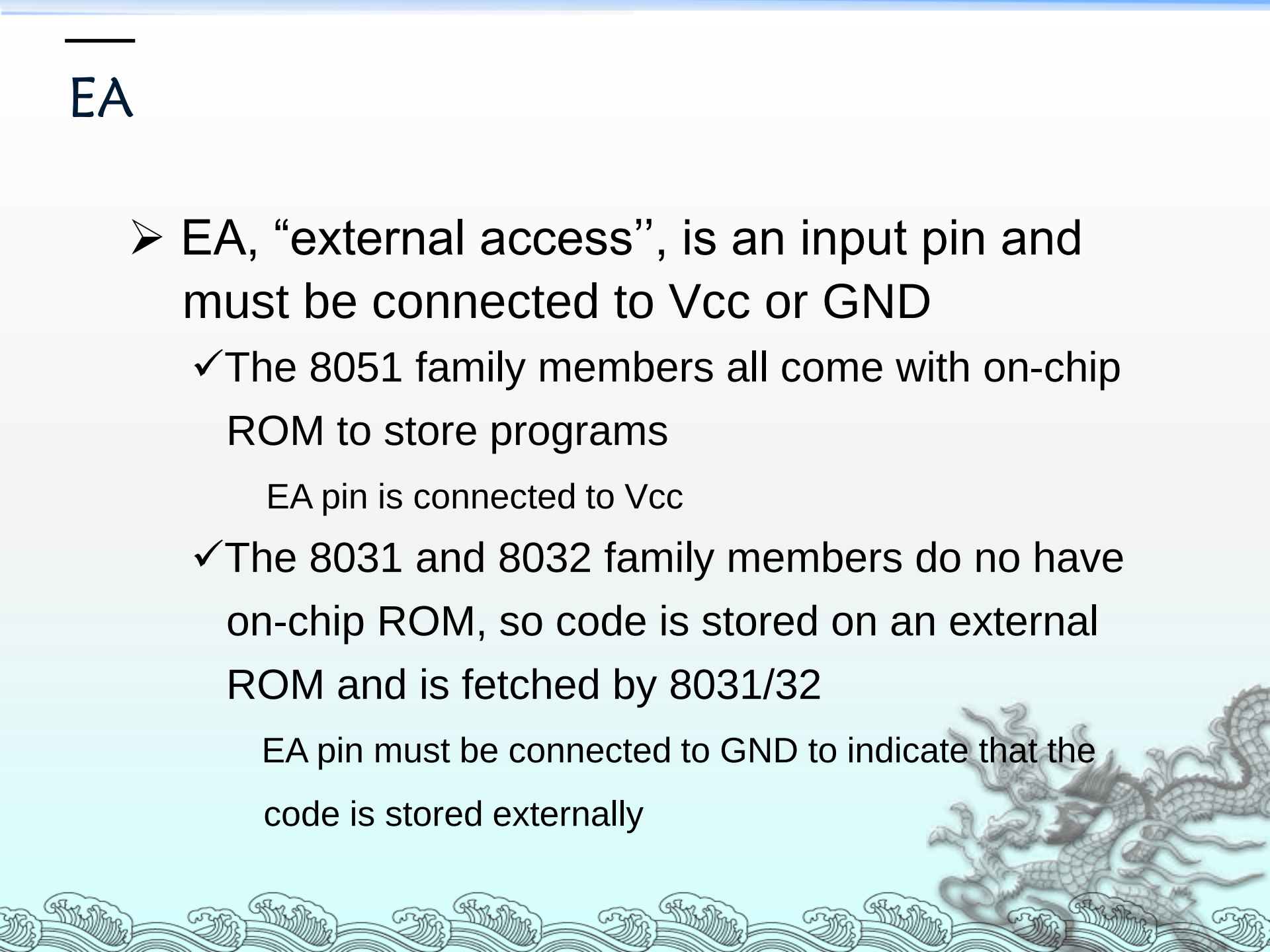
RST

- In order for the RESET input to be effective, it must have a minimum duration of 2 machine cycles
  - ✓ In other words, the high pulse must be high for a minimum of 2 machine cycles before it is allowed to go low



EA

- EA, “external access”, is an input pin and must be connected to Vcc or GND
  - ✓The 8051 family members all come with on-chip ROM to store programs
    - EA pin is connected to Vcc
  - ✓The 8031 and 8032 family members do no have on-chip ROM, so code is stored on an external ROM and is fetched by 8031/32
    - EA pin must be connected to GND to indicate that the code is stored externally



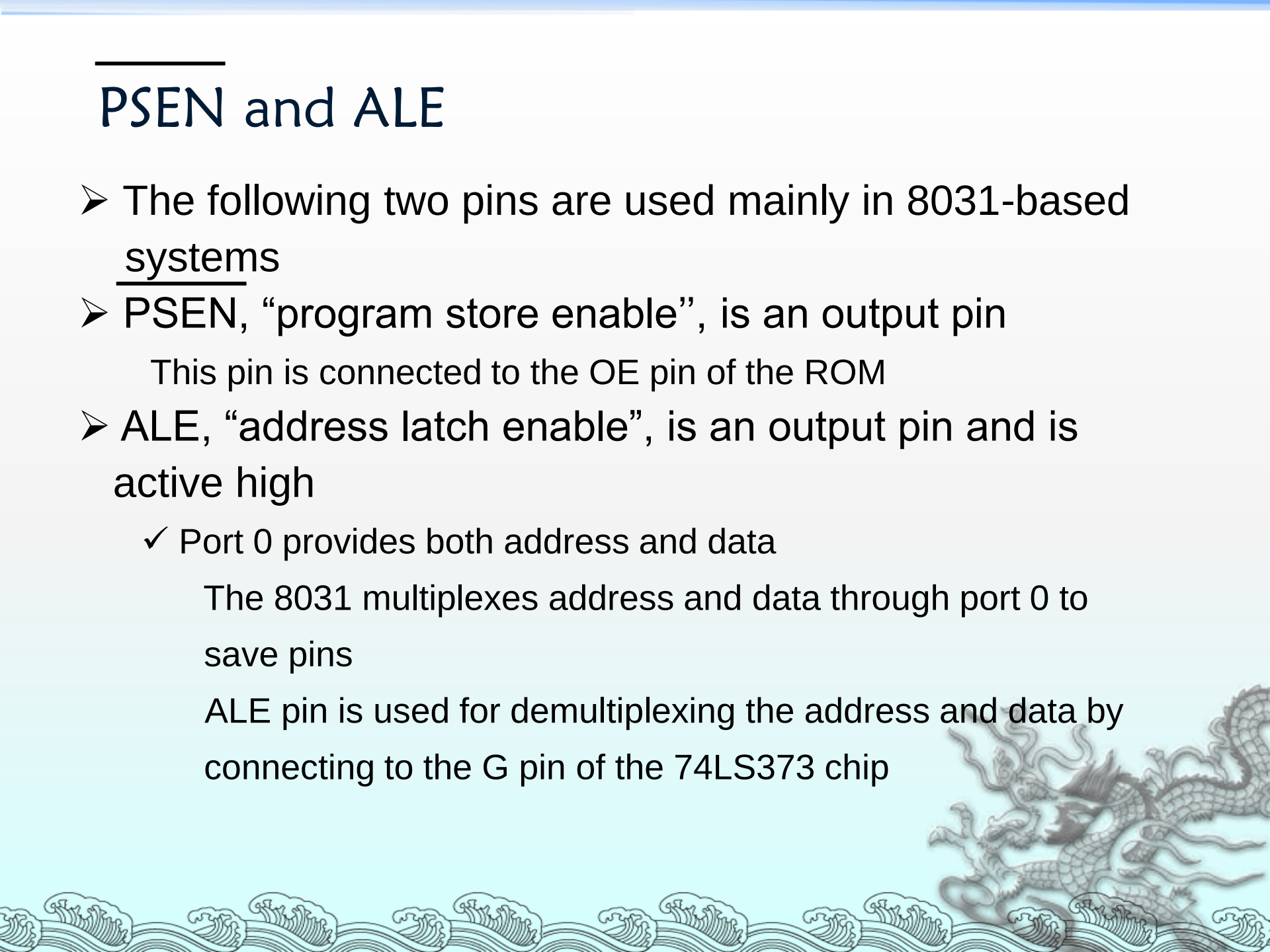
$\overline{\text{PSEN}}$ and ALE

- The following two pins are used mainly in 8031-based systems
- $\overline{\text{PSEN}}$, "program store enable", is an output pin

  This pin is connected to the OE pin of the ROM
- ALE, "address latch enable", is an output pin and is active high
  - ✓ Port 0 provides both address and data

    The 8031 multiplexes address and data through port 0 to save pins

    ALE pin is used for demultiplexing the address and data by connecting to the G pin of the 74LS373 chip



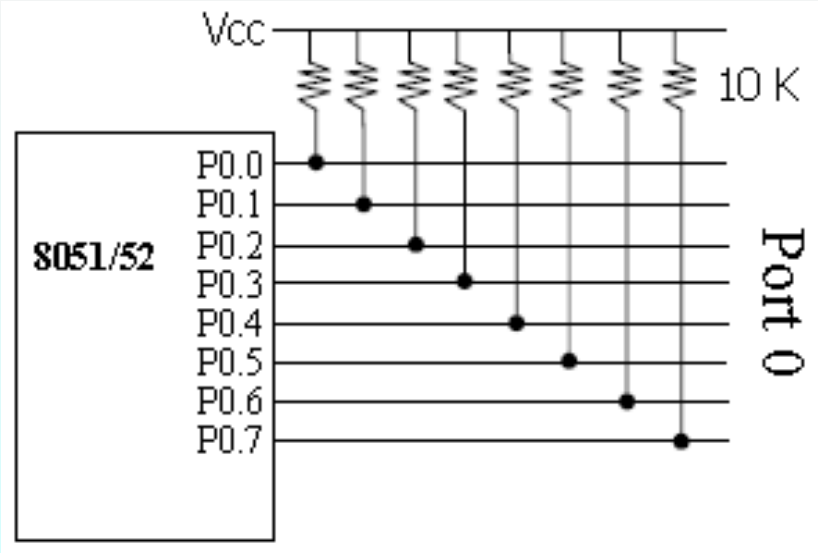
I/O Port Pins

- The four 8-bit I/O ports P0, P1, P2 and P3 each uses 8 pins
- All the ports upon RESET are configured as output, ready to be used as input ports

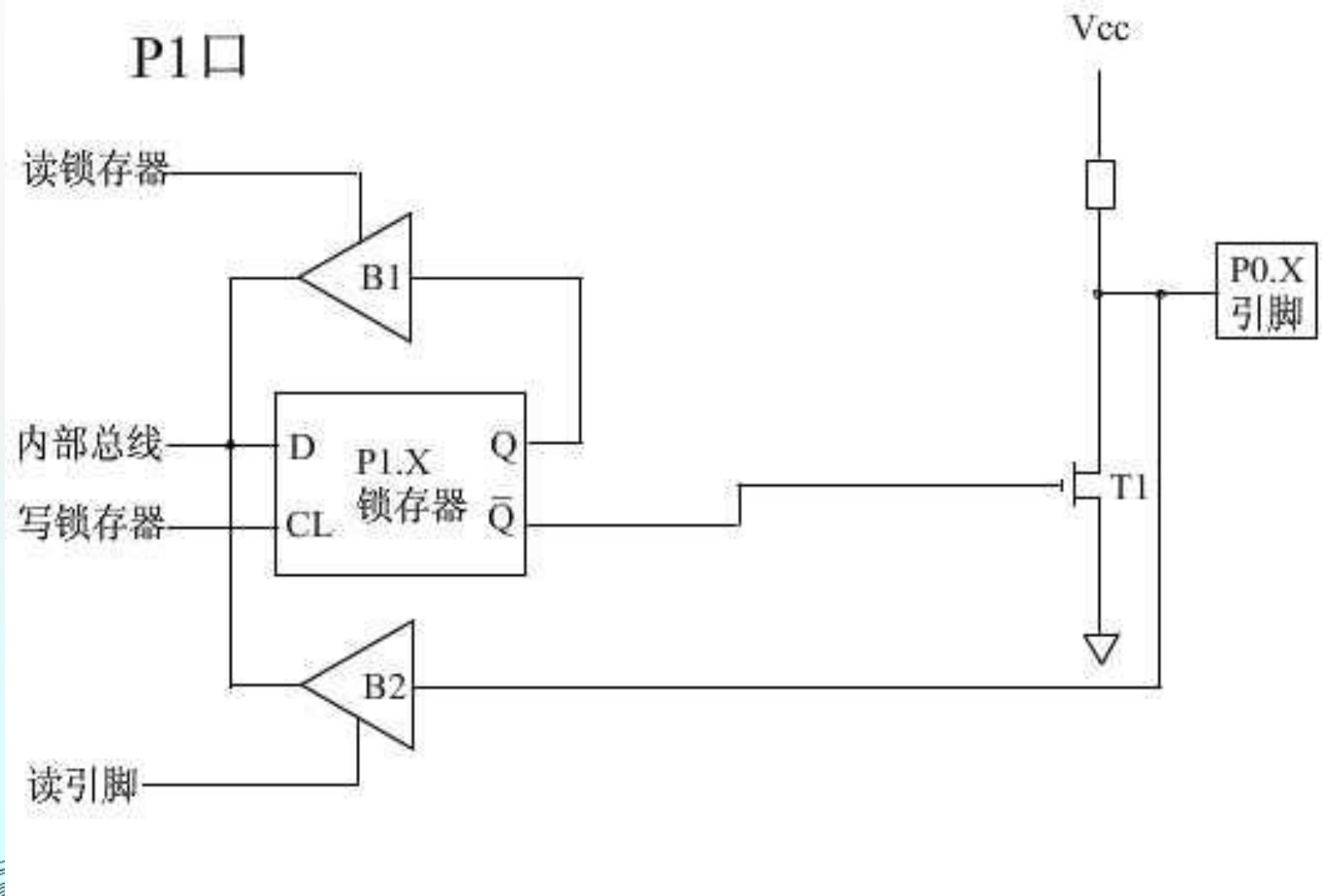
Port 0

- It can be used for input or output, each pin must be connected externally to a 10K ohm pull-up resistor
  - ✓ This is due to the fact that P0 is an open drain, unlike P1, P2, and P3

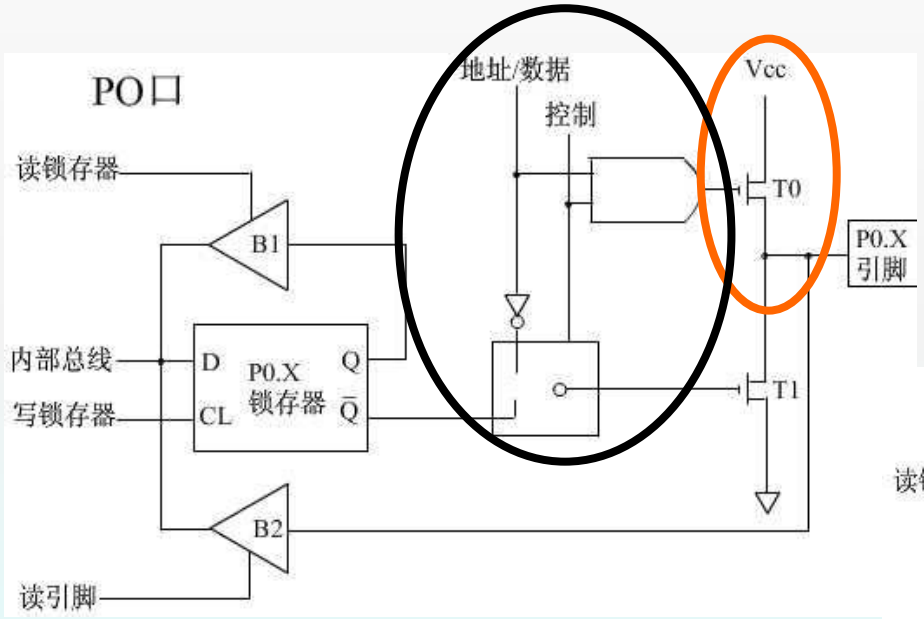
    Open drain is a term used for MOS chips in the same way that open collector is used for TTL chips

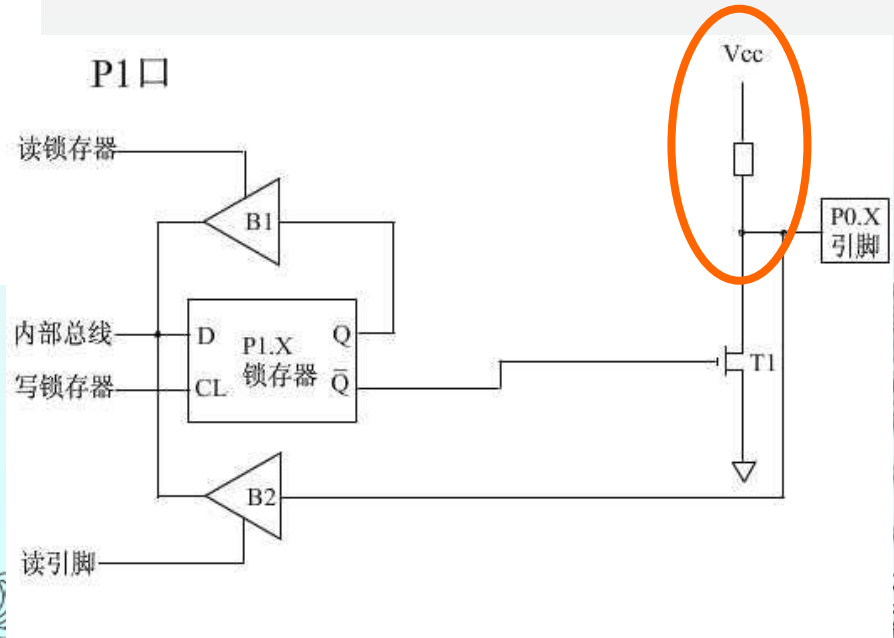


P1



P0 Vs P1


P0口
地址/数据
控制
Vcc
读锁存器
B1
T0
P0.X
引脚
内部总线
D
P0.X
锁存器
Q
写锁存器
CL
Q̄
T1
B2
读引脚


P1口
Vcc
读锁存器
B1
P0.X
引脚
内部总线
D
P1.X
锁存器
Q
写锁存器
CL
Q̄
T1
B2
读引脚

Port 1 and Port 2

- In 8051-based systems with no external memory connection
  - ✓ Both P1 and P2 are used as simple I/O
- In 8031/51-based systems with external memory connections
  - ✓ Port 2 must be used along with P0 to provide the 16-bit address for the external memory

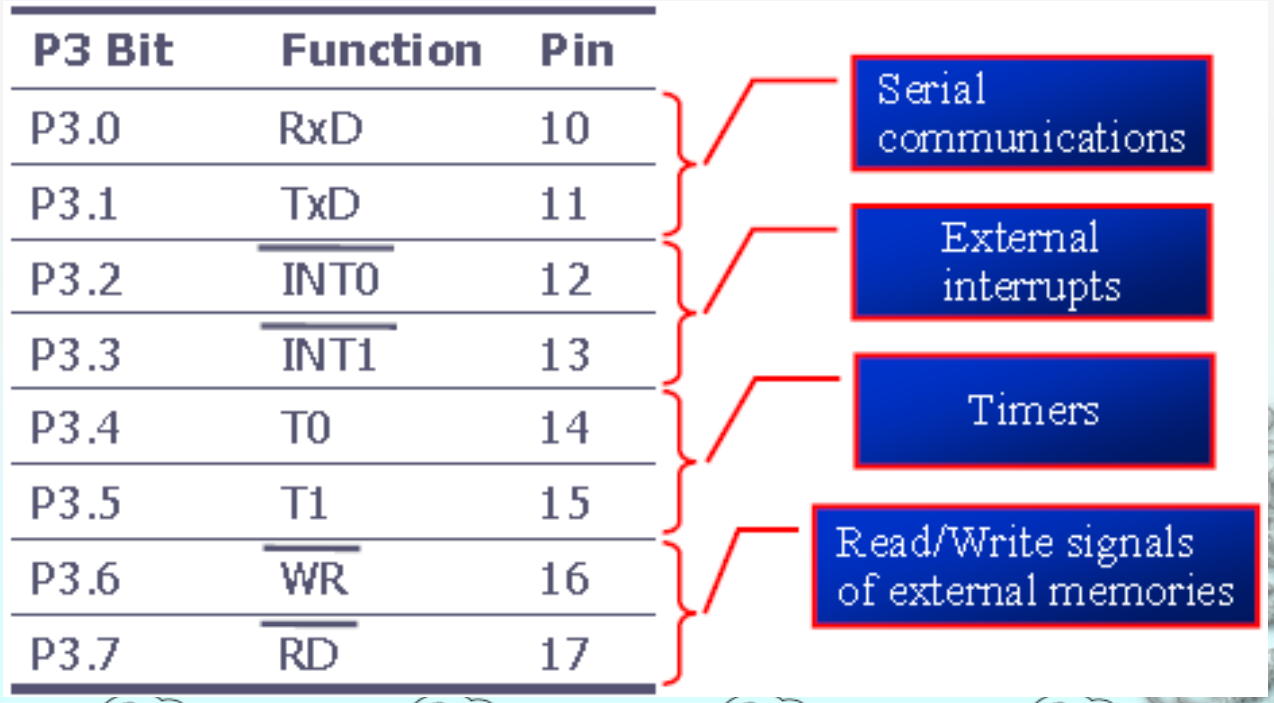
    P0 provides the lower 8 bits via A0 – A7

    P2 is used for the upper 8 bits of the 16-bit address, designated as A8 – A15, and it cannot be used for I/O

Port 3

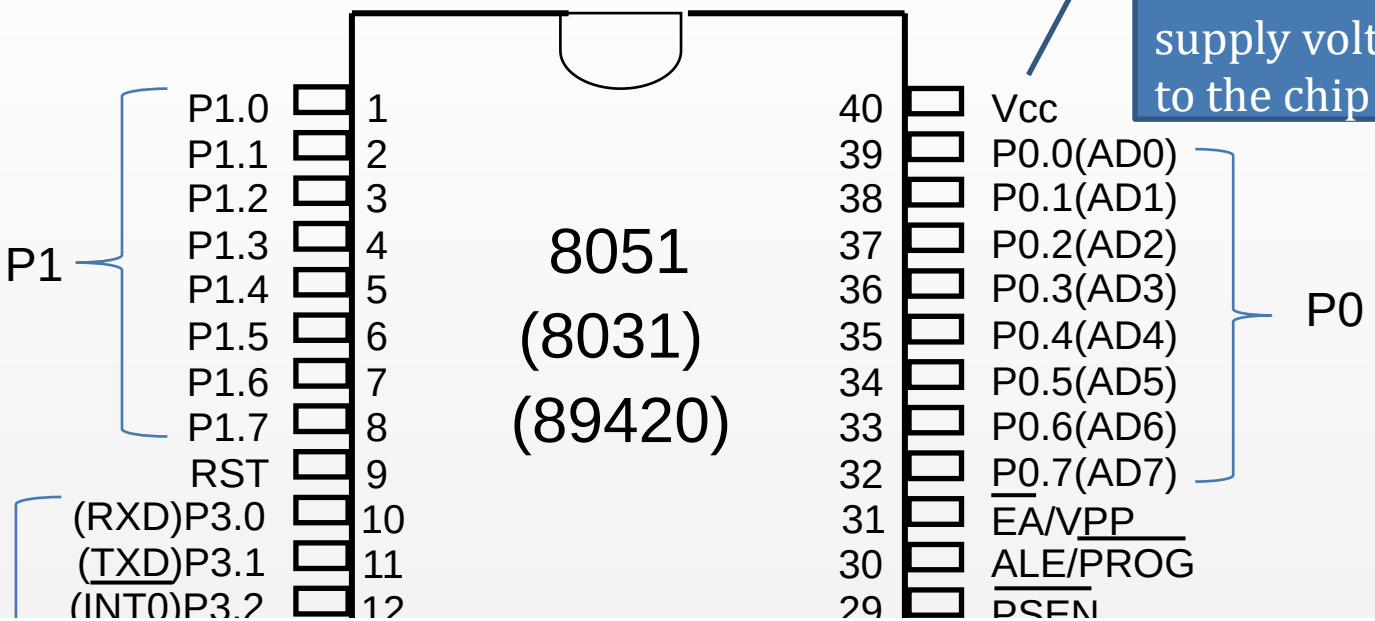
- Port 3 can be used as input or output
  - ✓Port 3 does not need any pull-up resistors
- Port 3 has the additional function of providing some extremely important signals

P3 Bit	Function	Pin
P3.0	RxD	10
P3.1	TxD	11
P3.2	$\overline{\text{INT0}}$	12
P3.3	$\overline{\text{INT1}}$	13
P3.4	T0	14
P3.5	T1	15
P3.6	$\overline{\text{WR}}$	16
P3.7	$\overline{\text{RD}}$	17



I/O Ports Configuration

A total of 32 pins are set aside for the four ports P0,P1, P2, P3, where each port takes 8 pins



- **All the ports upon RESET are configured as input, ready to be used as input ports**

  When the first 0 is written to a port, it becomes an output

  To reconfigure it as an input, a 1 must be sent to the port

  To use any of these ports as an input port, it must be programmed.

P0

The following code will continuously send out to port 0 the alternating value 55H and AAH

```
BACK:   MOV  A, #55H
        MOV  P0, A
        ACALL  DELAY
        MOV  A, #0AAH
        MOV  P0, A
        ACALL  DELAY
        SJMP  BACK
```

In order to make port 0 an input, the port must be programmed by writing 1 to all the bits

Port 0 is configured first as an input port by writing 1s to it, and then data is received from that port and sent to P1

```
        MOV  A, #0FFH      ;A=FF hex
        MOV  P0, A         ;make P0 an i/p port by writing it all 1s
BACK:   MOV  A, P0         ;get data from P0
        MOV  P1, A         ;send it to port 1
        SJMP  BACK         ;keep doing it
```

Dual Role of Port 0

Port 0 is also designated as AD0-AD7, allowing it to be used for both address and data

When connecting an 8051/31 to an external memory, port 0 provides both address and data.

Different ways of Accessing Entire 8 Bits

The entire 8 bits of Port 1 are accessed

```
BACK:   MOV  A, #55H
        MOV  P1, A
        ACALL  DELAY
        MOV  A, #0AAH
        MOV  P1, A
        ACALL  DELAY
        SJMP  BACK
```

Rewrite the code in a more efficient manner by accessing the port directly without going through the accumulator

```
BACK:   MOV P1,#55H
        ACALL DELAY
        MOV P1,#0AAH
        ACALL DELAY
        SJMP BACK
```

Another way of doing the same thing

```
        MOV  A, #55H
BACK:   MOV  P1, A
        ACALL  DELAY
        CPL  A
        SJMP  BACK
```

I/O Ports and Bit Addressability

- Sometimes we need to access only 1 or 2 bits of the port

```
BACK:    CPL  P1.2          ;complement P1.2
         ACALL DELAY
         SJMP BACK
;another variation of the above program
AGAIN:   SETB  P1.2         ;set only P1.2
         ACALL  DELAY
         CLR  P1.2          ;clear only P1.2
         ACALL  DELAY
         SJMP AGAIN
```

P0	P1	P2	P3	Port Bit
P0.0	P1.0	P2.0	P3.0	D0
P0.1	P1.1	P2.1	P3.1	D1
P0.2	P1.2	P2.2	P3.2	D2
P0.3	P1.3	P2.3	P3.3	D3
P0.4	P1.4	P2.4	P3.4	D4
P0.5	P1.5	P2.5	P3.5	D5
P0.6	P1.6	P2.6	P3.6	D6
P0.7	P1.7	P2.7	P3.7	D7

Example 4-1

Write the following programs.

Create a square wave of 50% duty cycle on bit 0 of port 1.

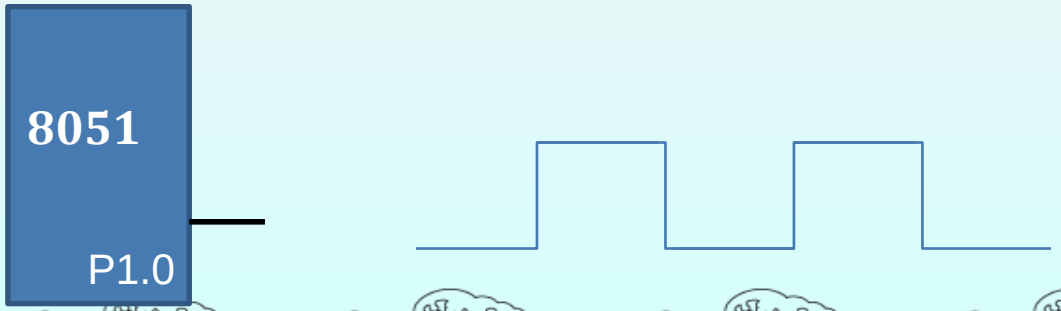
Solution:

The 50% duty cycle means that the “on” and “off” state (or the high and low portion of the pulse) have the same length. Therefore, we toggle P1.0 with a time delay in between each state.

```
HERE:   SETB P1.0          ;set to high bit 0 of port 1
        LCALL DELAY        ;call the delay subroutine
        CLR P1.0           ;P1.0=0
        LCALL DELAY
        SJMP HERE          ;keep doing it
```

Another way to write the above program is:

```
HERE:   CPL P1.0           ;set to high bit 0 of port 1
        LCALL DELAY        ;call the delay subroutine
        SJMP HERE          ;keep doing it
```



I/O Ports and Bit Addressability

- Instructions that are used for signal-bit operations are as following

Single-Bit Instructions

Instruction	Function
SETB bit	Set the bit (bit = 1)
CLR bit	Clear the bit (bit = 0)
CPL bit	Complement the bit (bit = NOT bit)
JB bit, target	Jump to target if bit = 1 (jump if bit)
JNB bit, target	Jump to target if bit = 0 (jump if no bit)
JBC bit, target	Jump to target if bit = 1, clear bit (jump if bit, then clear)

Checking an Input Bit

- The JNB and JB instructions are widely used single-bit operations

They allow you to monitor a bit and make a decision depending on whether its 0 or 1

These two instructions can be used for any bits of I/O ports 0, 1, 2, and 3

Instructions for Reading an Input Port

er single-bit or byte-wise

Mnemonic	Examples	Description
MOV A,PX	MOV A,P2	Bring into A the data at P2 pins
JNB PX.Y, ..	JNB P2.1,TARGET	Jump if pin P2.1 is low
JB PX.Y, ..	JB P1.3,TARGET	Jump if pin P1.3 is high
MOV C,PX.Y	MOV C,P2.4	Copy status of pin P2.4 to CY

Checking an Input Bit

Example 4-2

Write a program to perform the following:

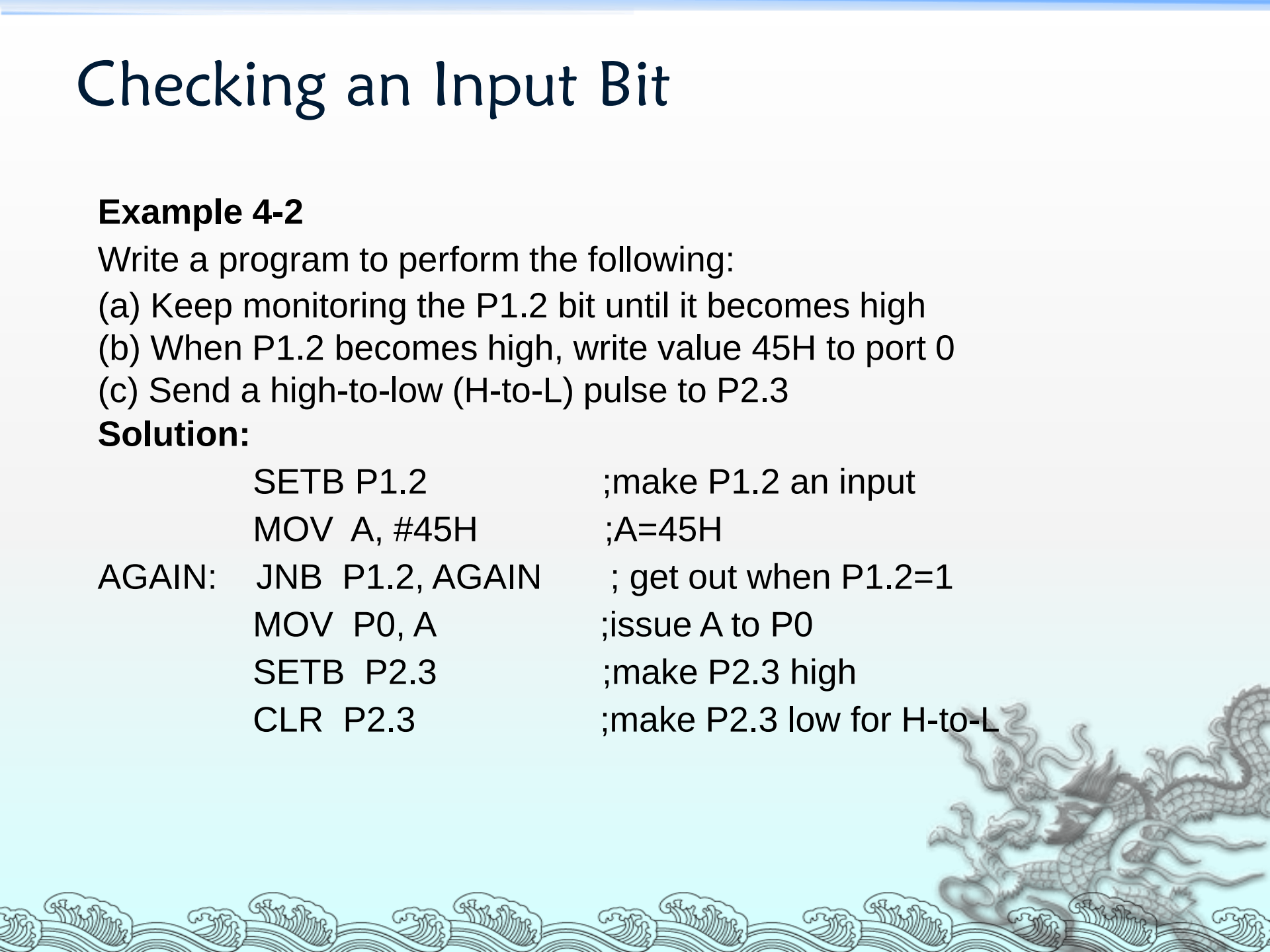
(a) Keep monitoring the P1.2 bit until it becomes high

(b) When P1.2 becomes high, write value 45H to port 0

(c) Send a high-to-low (H-to-L) pulse to P2.3

Solution:

```
        SETB P1.2          ;make P1.2 an input
        MOV  A, #45H       ;A=45H
AGAIN:  JNB  P1.2, AGAIN    ; get out when P1.2=1
        MOV  P0, A         ;issue A to P0
        SETB  P2.3         ;make P2.3 high
        CLR  P2.3          ;make P2.3 low for H-to-L
```



Checking an Input Bit

Example 4-3

Assume that bit P2.3 is an input and represents the condition of an oven. If it goes high, it means that the oven is hot. Monitor the bit continuously. Whenever it goes high, send a high-to-low pulse to port P1.5 to turn on a buzzer.

Solution:

```
HERE:   JNB  P2.3, HERE     ;keep monitoring for high
        SETB  P1.5          ;set bit P1.5=1
        CLR  P1.5           ;make high-to-low
        SJMP  HERE          ;keep repeating
```

Reading Single Bit into Carry Flag

Example 4-4

A switch is connected to pin P1.0 and an LED to pin P2.7. Write a program to get the status of the switch and send it to the LED

Solution:

```
        SETB P1.7          ;make P1.7 an input
AGAIN:  MOV C, P1.0        ;read SW status into CF
        MOV P2.7, C        ;send SW status to LED
        SJMP  AGAIN        ;keep repeating
```

The instruction 'MOV P2.7,P1.0' is wrong , since such an instruction does not exist

However 'MOV P2,P1' is a valid instruction

THANK YOU!!