# Principle and Interface Techniques of Microcontroller

## --8051 Microcontroller and Embedded Systems Using Assembly and C


**LI, Guang (李光)** **Prof. PhD, DIC, MIET**

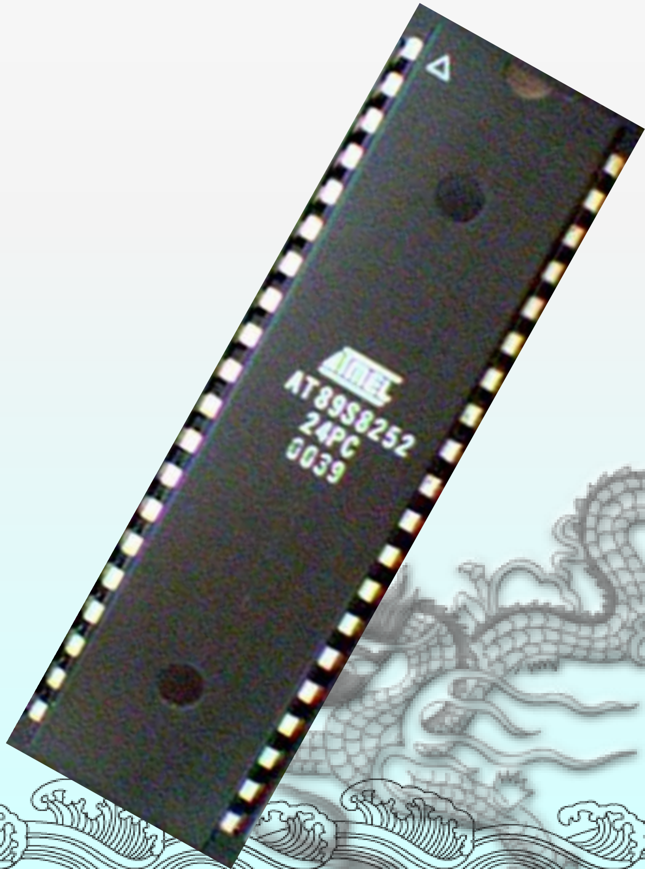
**WANG, You (王酉)** **PhD, MIET**


**杭州 • 浙江大学 • 2015**

# Chapter 13
# Real-world Interfacing
# LCD, ADC, and DAC

# Outline

- § 13-1  LCD and Keyboard Interfacing
- § 13-2  Interfacing to ADC
- § 13-3  Interfacing to DAC



# § 13-1 LCD and Keyboard Interfacing

## LCD Operation

- LCD is finding widespread use replacing LEDs
  - The declining prices of LCD
  - The ability to display numbers, characters, and graphics
  - Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD
  - Ease of programming for characters and graphics

# Pin Descriptions for LCD

| Pin | Symbol | I/O | Descriptions |
|---|---|---|---|
| 1 | VSS | -- | Ground |
| 2 | VCC | -- | +5V power supply |
| 3 | VEE | -- | Power supply to control contrast |
| 4 | RS | I | RS=0 to select command register, RS=1 to select data register |
| 5 | R/W | I | R/W=0 for write, R/W=1 for read |
| 6 | E | I/O | Enable |
| 7 | DB0 | I/O | The 8-bit data bus |
| 8 | DB1 | I/O | The 8-bit data bus |
| 9 | DB2 | I/O | The 8-bit data bus |
| 10 | DB3 | I/O | The 8-bit data bus |
| 11 | DB4 | I/O | The 8-bit data bus |
| 12 | DB5 | I/O | The 8-bit data bus |
| 13 | DB6 | I/O | The 8-bit data bus |
| 14 | DB7 | I/O | The 8-bit data bus |

used by the LCD to latch information presented to its data bus

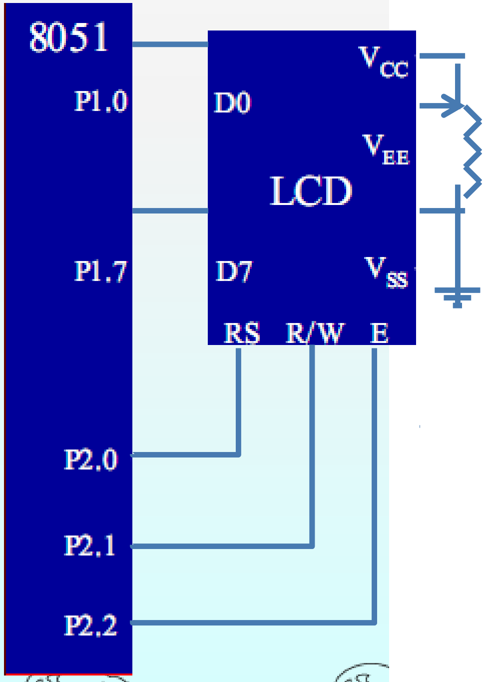
- Send displayed information or instruction command codes to the LCD
- Read the contents of the LCD's internal registers

# LCD Command Codes

| Code (Hex) | Command to LCD Instruction Register |
|---|---|
| 1 | Clear display screen |
| 2 | Return home |
| 4 | Decrement cursor (shift cursor to left) |
| 6 | Increment cursor (shift cursor to right) |
| 5 | Shift display right |
| 7 | Shift display left |
| 8 | Display off, cursor off |
| A | Display off, cursor on |
| C | Display on, cursor off |
| E | Display on, cursor blinking |
| F | Display on, cursor blinking |
| 10 | Shift cursor position to left |
| 14 | Shift cursor position to right |
| 18 | Shift the entire display to the left |
| 1C | Shift the entire display to the right |
| 80 | Force cursor to beginning to 1st line |
| C0 | Force cursor to beginning to 2nd line |
| 38 | 2 lines and 5x7 matrix |

# Sending Codes and Data to LCDs w/ Time Delay

To send any of the commands to the LCD, make pin RS=0. For data, make RS=1. Then send a high-to-low pulse to the E pin to enable the internal latch of the LCD. This is shown in the code below.




```
;calls a time delay before sending next data/command
;P1.0-P1.7 are connected to LCD data pins D0-D7
;P2.0 is connected to RS pin of LCD
;P2.1 is connected to R/W pin of LCD
;P2.2 is connected to E pin of LCD
        ORG
        MOV    A,#38H   ;INIT. LCD 2 LINES, 5X7 MATRIX
        ACALL  COMNWRT  ;call command subroutine
        ACALL  DELAY    ;give LCD some time
        MOV    A,#0EH   ;display on, cursor on
        ACALL  COMNWRT  ;call command subroutine
        ACALL  DELAY    ;give LCD some time
        MOV    A,#01    ;clear LCD
        ACALL  COMNWRT  ;call command subroutine
        ACALL  DELAY    ;give LCD some time
        MOV    A,#06H   ;shift cursor right
        ACALL  COMNWRT  ;call command subroutine
        ACALL  DELAY    ;give LCD some time
        MOV    A,#84H   ;cursor at line 1, pos. 4
        ACALL  COMNWRT  ;call command subroutine
        ACALL  DELAY    ;give LCD some time
```

```
        MOV    A,#'N'     ;display letter N
        ACALL  DATAWRT    ;call display subroutine
        ACALL  DELAY      ;give LCD some time
        MOV    A,#'O'     ;display letter O
        ACALL  DATAWRT    ;call display subroutine
AGAIN:  SJMP   AGAIN      ;stay here
COMNWRT:                  ;send command to LCD
        MOV    P1,A       ;copy reg A to port 1
        CLR    P2.0       ;RS=0 for command
        CLR    P2.1       ;R/W=0 for write
        SETB   P2.2       ;E=1 for high pulse
        ACALL  DELAY      ;give LCD some time
        CLR    P2.2       ;E=0 for H-to-L pulse
        RET
DATAWRT:                  ;write data to LCD
        MOV    P1,A       ;copy reg A to port 1
        SETB   P2.0       ;RS=1 for data
        CLR    P2.1       ;R/W=0 for write
        SETB   P2.2       ;E=1 for high pulse
        ACALL  DELAY      ;give LCD some time
        CLR    P2.2       ;E=0 for H-to-L pulse
        RET
DELAY:  MOV    R3,#50     ;50 or higher for fast CPUs
HERE2:  MOV    R4,#255    ;R4 = 255
HERE:   DJNZ   R4,HERE    ;stay until R4 becomes 0
        DJNZ   R3,HERE2
        RET
        END
```

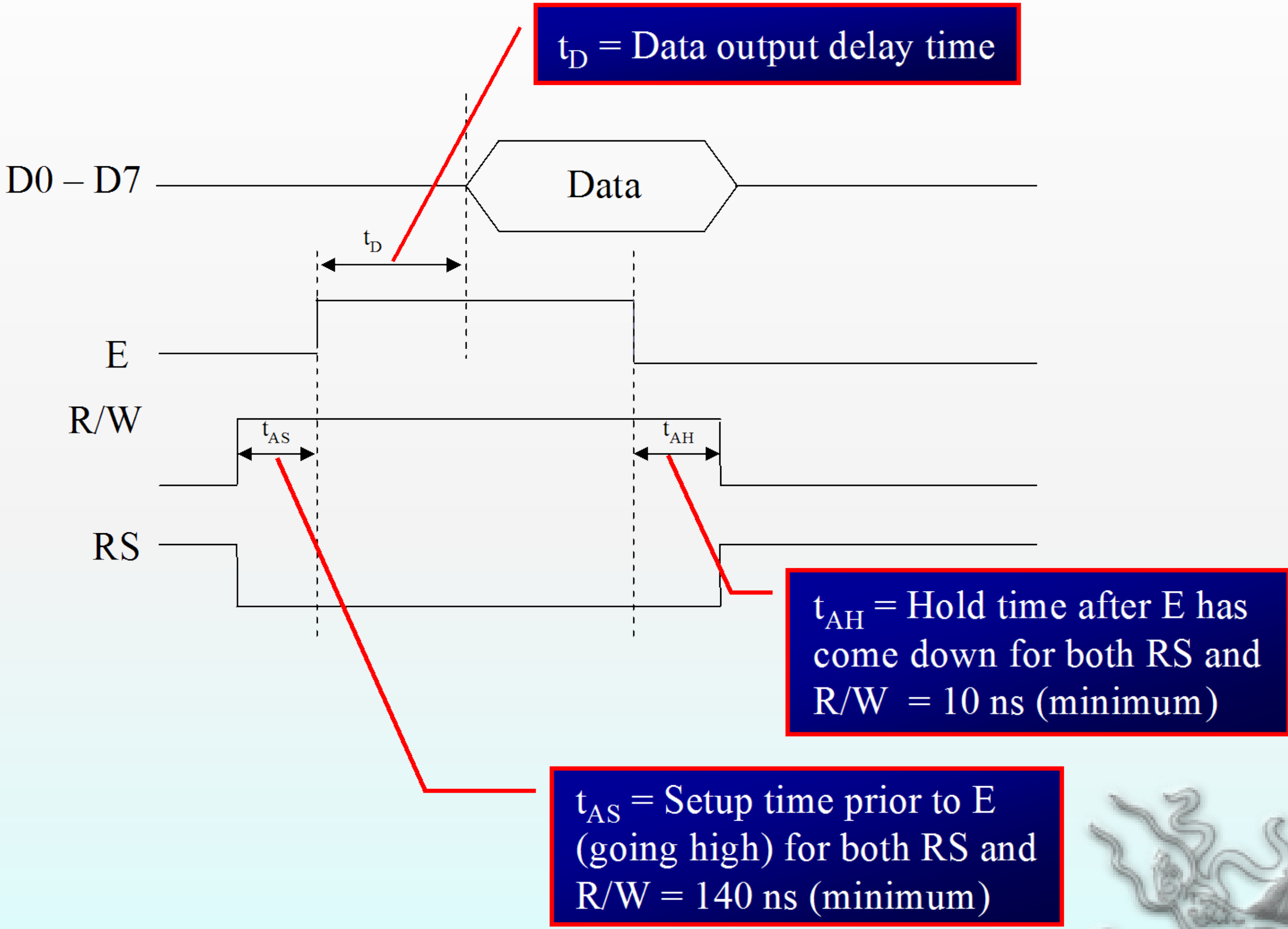
```
;Check busy flag before sending data, command to LCD
;p1=data pin
;P2.0 connected to RS pin
;P2.1 connected to R/W pin
;P2.2 connected to E pin
      ORG
      MOV   A,#38H        ;init. LCD 2 lines ,5x7 matrix
      ACALL COMMAND       ;issue command
      MOV   A,#0EH        ;LCD on, cursor on
      ACALL COMMAND       ;issue command
      MOV   A,#01H        ;clear LCD command
      ACALL COMMAND       ;issue command
      MOV   A,#06H        ;shift cursor right
      ACALL COMMAND       ;issue command
      MOV   A,#86H        ;cursor: line 1, pos. 6
      ACALL COMMAND       ;command subroutine
      MOV   A,#'N'        ;display letter N
      ACALL DATA_DISPLAY
      MOV   A,#'O'        ;display letter O
      ACALL DATA_DISPLAY
HERE:SJMP   HERE          ;STAY HERE
```

```
COMMAND:
        ACALL READY       ;is LCD ready?
        MOV   P1,A        ;issue command code
        CLR   P2.0        ;RS=0 for command
        CLR   P2.1        ;R/W=0 to write to LCD
        SETB  P2.2        ;E=1 for H-to-L pulse
        CLR   P2.2        ;E=0,latch in
        RET
DATA_DISPLAY:
        ACALL  READY      ;is LCD ready?
        MOV   P1,A        ;issue data
        SETB  P2.0        ;RS=1 for data
        CLR   P2.1        ;R/W =0 to write to LCD
        SETB  P2.2        ;E=1 for H-to-L pulse
        CLR   P2.2        ;E=0,latch in
        RET
READY:
        SETB  P1.7        ;make P1.7 input port
        CLR   P2.0        ;RS=0 access command reg
        SETB  P2.1        ;R/W=1 read command reg
;read command reg and check busy flag
BACK: SETB  P2.2          ;E=1 for H-to-L pulse
        CLR   P2.2        ;E=0 H-to-L pulse
        JB    P1.7, BACK  ;stay until busy flag=0
        RET
        END
```

To read the command register, we make R/W=1, RS=0, and a H-to-L pulse for the E pin.

If bit 7 (busy flag) is high, the LCD is busy and no information should be issued to it.

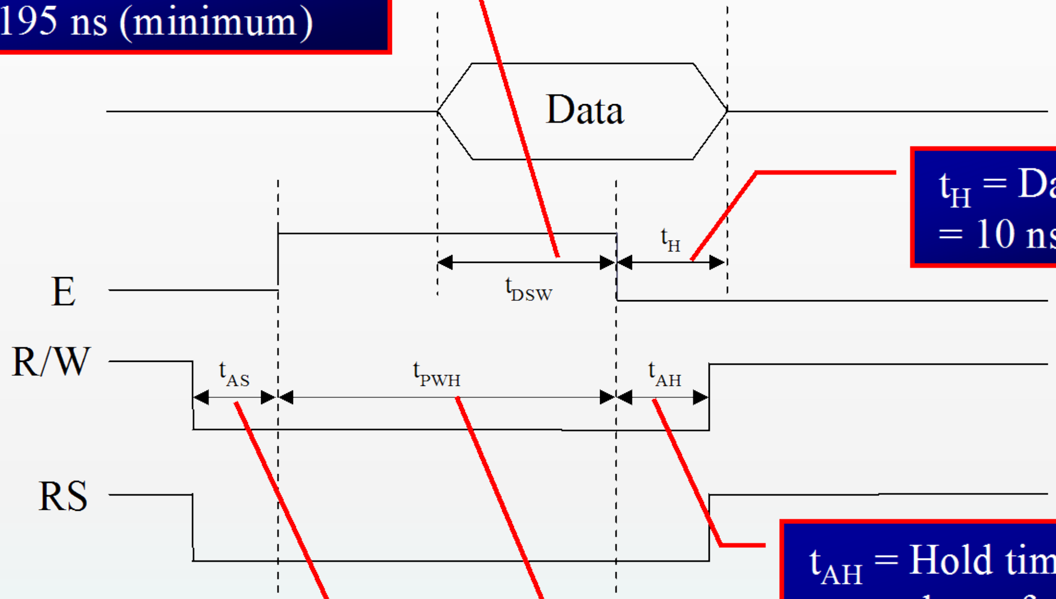
# LCD Timing for Read




Note : Read requires an L-to-H pulse for the E pin

# LCD Timing for Write

$t_{DSW}$ = Data set up time = 195 ns (minimum)




$t_H$ = Data hold time = 10 ns (minimum)

$t_{AH}$ = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

$t_{PWH}$ = Enable pulse width = 450 ns (minimum)

$t_{AS}$ = Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)

- One can put data at any location in the LCD and the following shows address locations and how they are accessed

| RS | R/W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | A | A | A | A | A | A | A |

  - AAAAAAA=000_0000 to 010_0111 fo
  - AAAAAAA=100_0000 to 110_0111 fo

The upper address range can go as high as 0100111 for the 40-character-wide LCD, which corresponds to locations 0 to 39

LCD Addressing for the LCDs of 40×2 size

| | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
|---|---|---|---|---|---|---|---|---|
| Line1 (min) | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Line1 (max) | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| Line2 (min) | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| Line2 (max) | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

Write an 8051 C program to send letters ‘M’, ‘D’, and ‘E’ to the LCD using the busy flag method.

Solution:

```
#include <reg51.h>
 sfr ldata = 0x90; //P1=LCD data pins
 sbit rs = P2^0;
 sbit rw = P2^1;
 sbit en = P2^2;
 sbit busy = P1^7;
void main(){
  lcdcmd(0x38);
  lcdcmd(0x0E);
  lcdcmd(0x01);
  lcdcmd(0x06);
  lcdcmd(0x86);   //line 1, position 6
  lcdcmd(‘M’);
  lcdcmd(‘D’);
  lcdcmd(‘E’);
}
.....
```

```
void lcdcmd(unsigned char value){
  lcdready();      //check the LCD busy flag
  ldata = value;   //put the value on the pins
  rs = 0;
  rw = 0;
  en = 1;          //strobe the enable pin
  MSDelay(1);
  en = 0;
  return;
}
void lcddata(unsigned char value){
  lcdready();      //check the LCD busy flag
  ldata = value;   //put the value on the pins
  rs = 1;
  rw = 0;
  en = 1;          //strobe the enable pin
  MSDelay(1);
  en = 0;
  return;
}
.....
```

```
.....
void lcdready(){
  busy = 1;  //make the busy pin at input
  rs = 0;
  rw = 1;
  while(busy==1){
                        //wait here for busy flag
    en = 0;         //strobe the enable pin
    MSDelay(1);
   }
   en = 1;
}
```

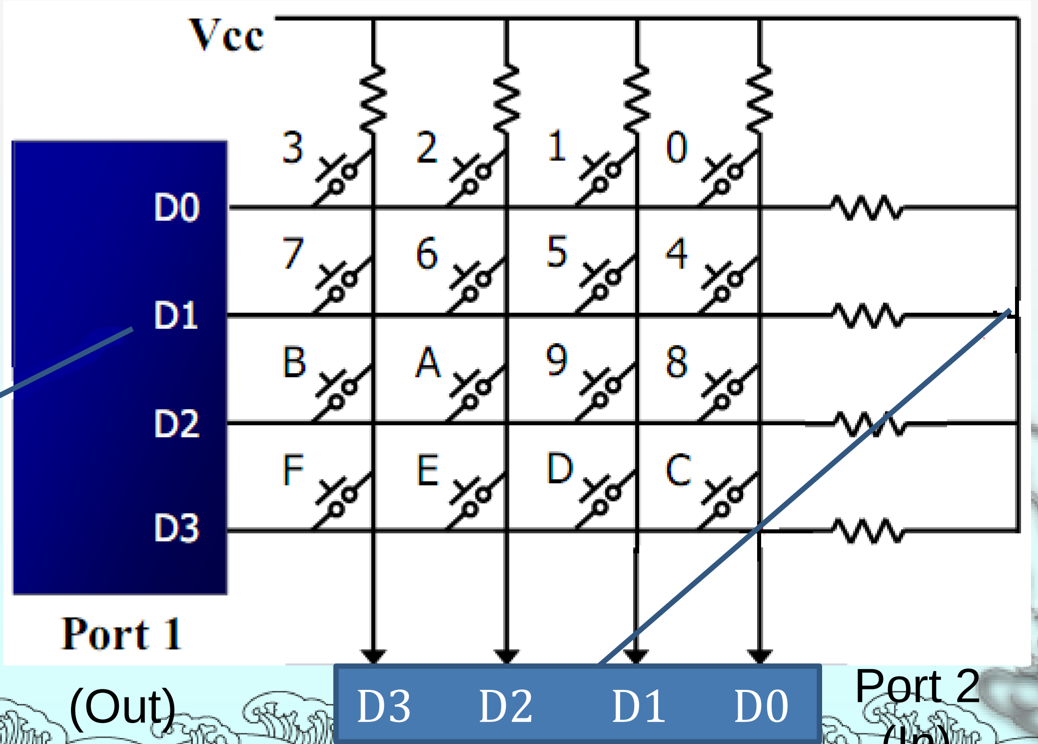
# Keyboard Interfacing

- Keyboards are organized in a matrix of rows and columns
  - The CPU accesses both rows and columns through ports
    - Therefore, with two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor
  - When a key is pressed, a row and a column make a contact
    - Otherwise, there is no connection between rows and columns
- In IBM PC keyboards, a single microcontroller takes care of hardware and software interfacing

# Scanning and Identifying the Key

- A 4x4 matrix connected to two ports
  - The rows are connected to an output port and the columns are connected to an input port

Matrix Keyboard Connection to ports




If all the rows are grounded and a key is pressed, one of the columns will have 0 since the key pressed provides the path to ground

If no key has been pressed, reading the input port will yield 1s for all columns since they are all connected to high (Vcc)

# Grounding Rows and Reading Columns

- It is the function of the microcontroller to scan the keyboard continuously to detect and identify the key pressed
- To detect a pressed key, the microcontroller grounds all rows by providing 0 to the output latch, then it reads the columns
  - If the data read from columns is D3 – D0 = 1111, no key has been pressed and the process continues till key press is detected
  - If one of the column bits has a zero, this means that a key press has occurred
    - For example, if D3 – D0 = 1101, this means that a key in the D1 column has been pressed  After detecting a key press, microcontroller will go through the process of identifying the key

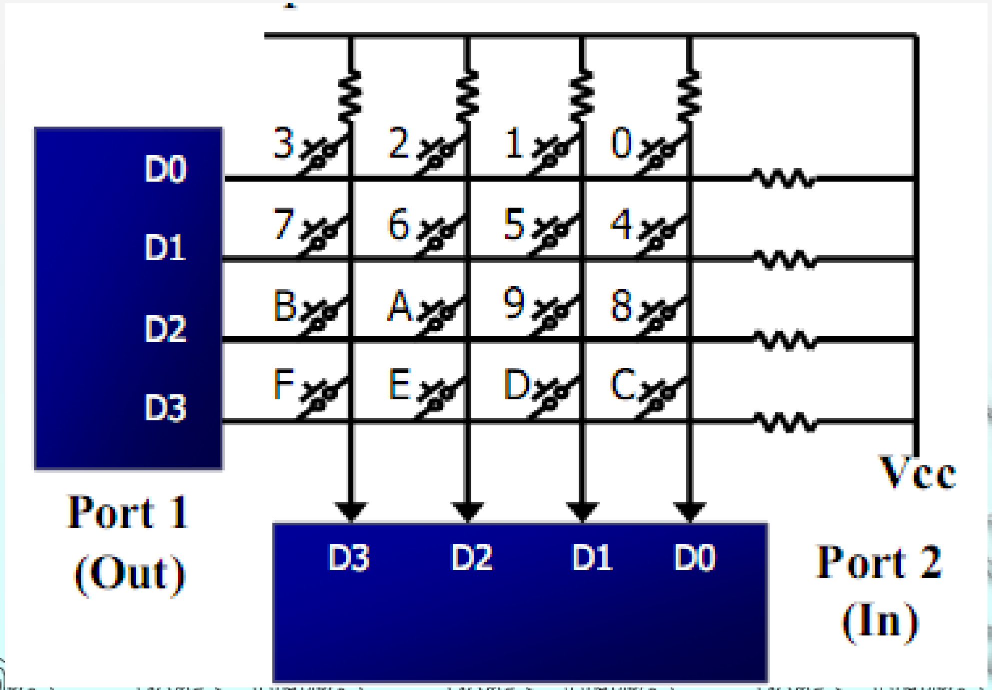
- Starting with the top row, the microcontroller grounds it by providing a low to row D0 only
  - It reads the columns, if the data read is all 1s, no key in that row is activated and the process is moved to the next row
- It grounds the next row, reads the columns, and checks for any zero
  - This process continues until the row is identified
- After identification of the row in which the key has been pressed
  - Find out which column the pressed key belongs to

From Figure 12-6, identify the row and column of the pressed key for each of the following.
(a) D3 – D0 = 1110 for the row, D3 – D0 = 1011 for the column
(b) D3 – D0 = 1101 for the row, D3 – D0 = 0111 for the column

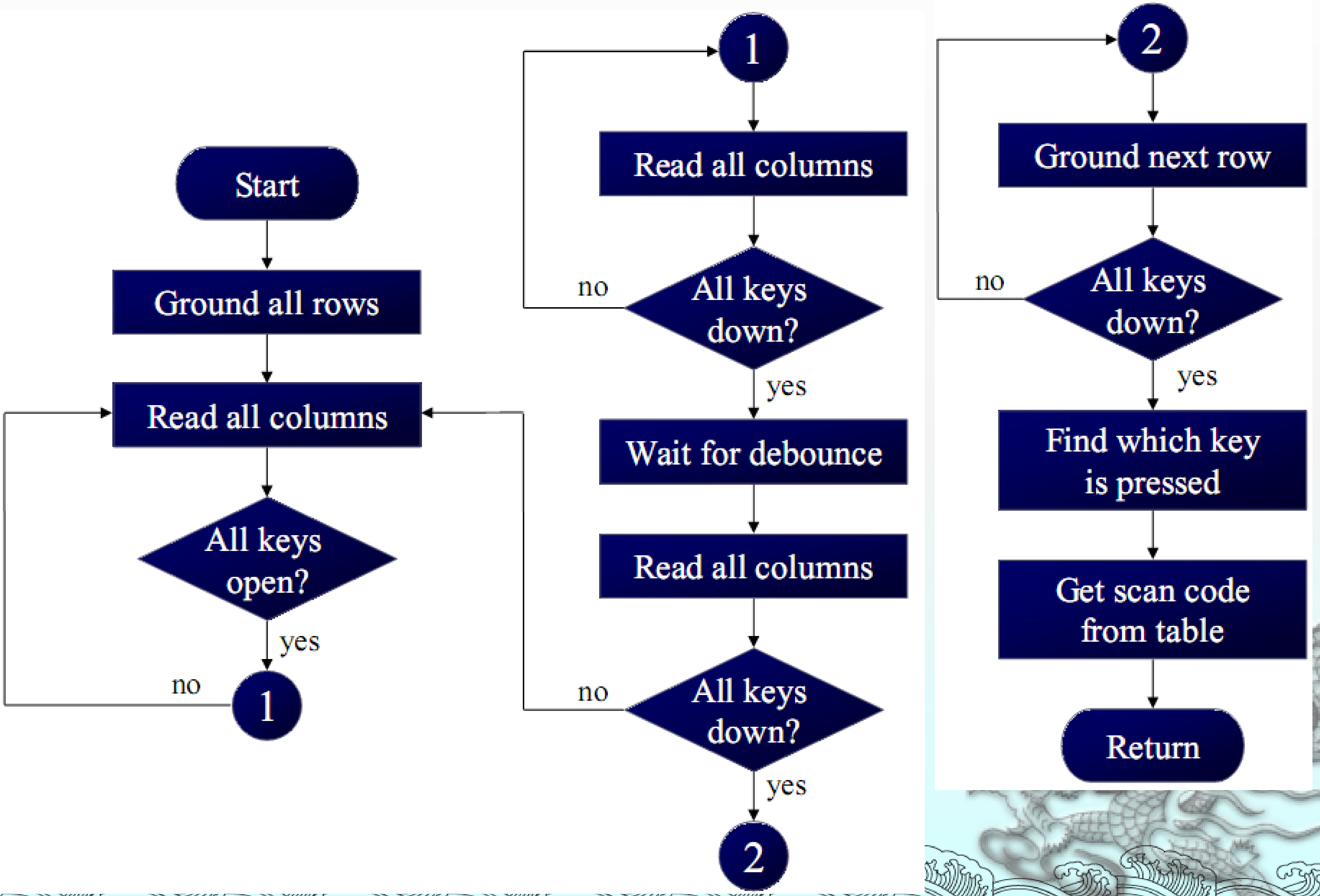
Solution :

From Figure 13-5 the row and column can be used to identify the key.

(a) The row belongs to D0 and the column belongs to D2; therefore, key number 2 was pressed.
(b) The row belongs to D1 and the column belongs to D3; therefore, key number 7 was pressed.



# Flowchart for Program



```
;keyboard subroutine. This program sends the ASCII
;code for pressed key to P0.1
;P1.0-P1.3 connected to rows, P2.0-P2.3 to column

        MOV  P2,#0FFH   ;make P2 an input port
K1:     MOV  P1,#0      ;ground all rows at once
        MOV  A,P2       ;read all col
                        ;(ensure keys open)
        ANL  A,00001111B      ;masked unused bits
        CJNE A,#00001111B,K1 ;till all keys release
K2:     ACALL DELAY     ;call 20 msec delay
        MOV  A,P2       ;see if any key is pressed
        ANL  A,00001111B      ;mask unused bits
        CJNE A,#00001111B,OVER;key pressed, find row
        SJMP K2         ;check till key pressed
OVER:   ACALL DELAY     ;wait 20 msec debounce time
        MOV  A,P2       ;check key closure
        ANL  A,00001111B      ;mask unused bits
        CJNE A,#00001111B,OVER1;key pressed, find row
        SJMP K2         ;if none, keep polling
```

```
....
OVER1:  MOV P1, #11111110B  ;ground row 0
        MOV A,P2                   ;read all columns
        ANL A,#00001111B           ;mask unused bits
       CJNE A,#00001111B,ROW_0  ;key row 0, find col.
        MOV P1,#11111101B          ;ground row 1
        MOV A,P2                   ;read all columns
        ANL A,#00001111B           ;mask unused bits
       CJNE A,#00001111B,ROW_1  ;key row 1, find col.
        MOV P1,#11111011B          ;ground row 2
        MOV A,P2                   ;read all columns
        ANL A,#00001111B           ;mask unused bits
       CJNE A,#00001111B,ROW_2  ;key row 2, find col.
        MOV P1,#11110111B          ;ground row 3
        MOV A,P2                   ;read all columns
        ANL A,#00001111B           ;mask unused bits
       CJNE A,#00001111B,ROW_3  ;key row 3, find col.
        LJMP K2             ;if none, false input,
                            ;repeat
....
```

```
....
ROW_0:  MOV   DPTR,#KCODE0    ;set DPTR=start of row 0
        SJMP FIND             ;find col. Key belongs to
ROW_1:  MOV   DPTR,#KCODE1    ;set DPTR=start of row
        SJMP FIND             ;find col. Key belongs to
ROW_2:  MOV   DPTR,#KCODE2    ;set DPTR=start of row 2
        SJMP FIND             ;find col. Key belongs to
ROW_3:  MOV   DPTR,#KCODE3    ;set DPTR=start of row 3
FIND:   RRC   A               ;see if any CY bit low
        JNC   MATCH           ;if zero, get ASCII code
        INC   DPTR            ;point to next col. addr
        SJMP FIND             ;keep searching
MATCH:  CLR   A               ;set A=0 (match is found)
        MOVC A,@A+DPTR        ;get ASCII from table
        MOV   P0,A            ;display pressed key
        LJMP K1
;ASCII LOOK-UP TABLE FOR EACH ROW
        ORG   300H
KCODE0: DB    '0','1','2','3' ;ROW 0
KCODE1: DB    '4','5','6','7' ;ROW 1
KCODE2: DB    '8','9','A','B' ;ROW 2
KCODE3: DB    'C','D','E','F' ;ROW 3
        END
```

# § 13-2  Interfacing to ADC and DAC

## ADC Devices

- ADCs (analog-to-digital converters) are among the most widely used devices for data acquisition
  - A physical quantity, like temperature, pressure, humidity, and velocity, etc., is converted to electrical (voltage, current) signals using a device called a transducer, or sensor
- We need an analog-to-digital converter to translate the analog signals to digital numbers, so microcontroller can read them

# ADC Principle

- 积分型
  - 输入电压通过积分电路转换成时间(脉冲宽度信号)或频率(脉冲频率)，转换速率极低;
- 逐次比较型
  - 由一个比较器和DA转换器通过逐次比较逻辑构成，速度较高、功耗低，低分辩率（<12位）时价格便宜;
- 并行比较型/串并行比较型
  - 电路规模极大，转换速率极高;

# ADC Principle（2）

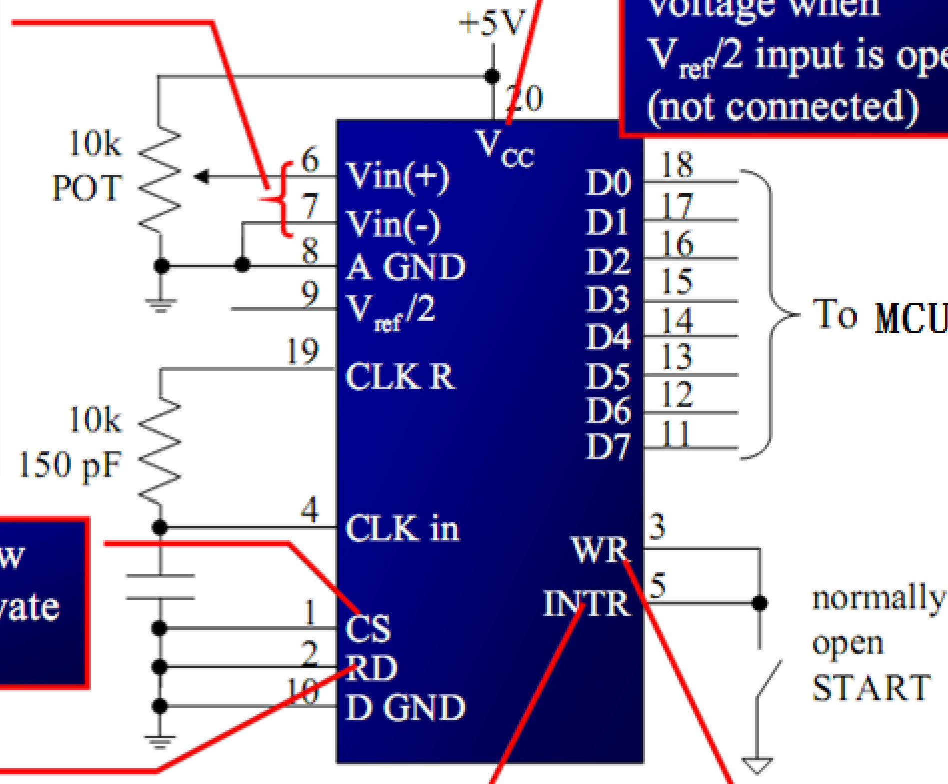
- Σ-Δ(Sigma/delta)调制型
  - 信号采样，负反馈网络对量化噪声进行低频衰减，高频放大，用数字滤波器滤除带外噪声；
- 压频变换型
  - 由计数器、控制门及一个具有恒定时间的时钟门控制信号组成，把输入的模拟电压转换成与模拟电压成正比的脉冲信号。

# ADC0804 Chip

- ADC0804 IC is an analog-to-digital converter
  - It works with +5 volts and has a resolution of 8 bits
  - Conversion time is another major factor in judging an ADC
    - Conversion time is defined as the time it takes the ADC to convert the analog input to a digital (binary) number
    - In ADC804 conversion time varies depending on the clocking signals applied to CLK R and CLK IN pins, but it cannot be faster than 110 μs

Differential analog inputs where $V_{in} = V_{in}(+) - V_{in}(-)$ Vin (-) is connected to ground and Vin (+) is used as the analog input to be converted

+5V power supply or a reference voltage when $V_{ref}/2$ input is open (not connected)




CS is an active low input used to activate ADC804

"output enable"
a high-to-low RD pulse is used to get the 8-bit converted data out of ADC804

"end of conversion"
When the conversion is finished, it goes low to signal the CPU that the converted data is ready to be picked up

"start conversion"
When WR makes a low-to-high transition, ADC804 starts converting the analog input value of $V_{in}$ to an 8-bit digital number

- CLK IN and CLK R
  - CLK IN is an input pin connected to an external clock source
  - To use the internal clock generator (also called self-clocking), CLK IN and CLK R pins are connected to a capacitor and a resistor, and the clock frequency is determined by

$$f = \frac{1}{1.1RC}$$

    - Typical values are R = 10K ohms and C = 150 pF
    - We get f = 606 kHz and the conversion time is 110 µs

- $V_{ref}/2$
  - It is used for the reference voltage
    - If this pin is open (not connected), the analog input voltage is in the range of 0 to 5 volts (the same as the Vcc pin)
    - If the analog input range needs to be 0 to 4 volts, $V_{ref}/2$ is connected to 2 volts

## Vref/2 Relation to Vin Range

| $V_{ref}/2$(v) | $V_{in}$(V) | Step Size ( mV) |
|---|---|---|
| Not connected* | 0 to 5 | 5/256=19.53 |
| 2.0 | 0 to 4 | 4/255=15.62 |
| 1.5 | 0 to 3 | 3/256=11.71 |
| 1.28 | 0 to 2.56 | 2.56/256=10 |
| 1.0 | 0 to 2 | 2/256=7.81 |
| 0.5 | 0 to 1 | 1/256=3.90 |

Step size is the smallest change can be discerned by an ADC

- D0-D7
  - The digital data output pins
  - These are tri-state buffered
    - The converted data is accessed only when CS = 0 and RD is forced low
  - To calculate the output voltage, use the following formula

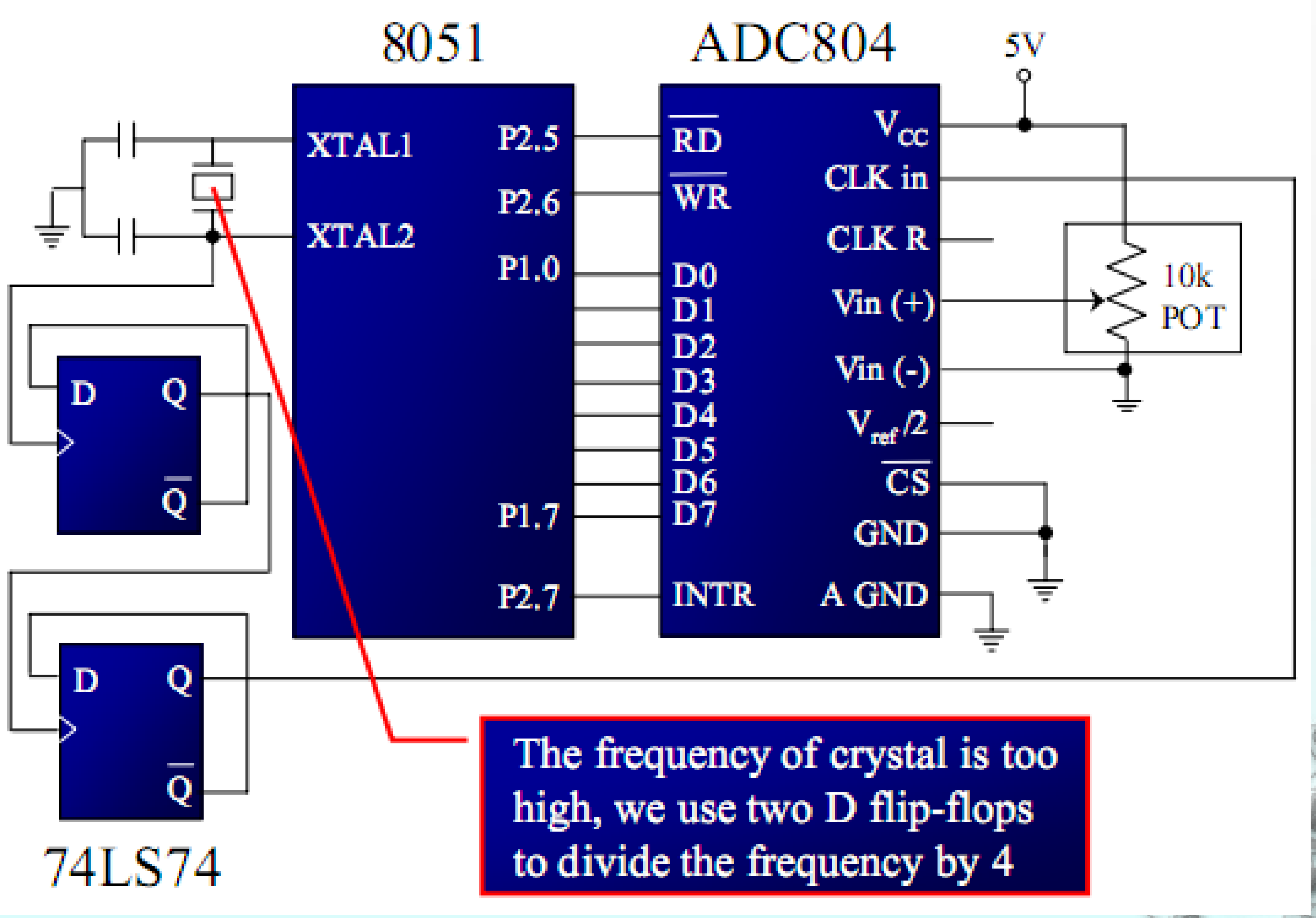
$$D_{\text{out}} = \frac{V_{in}}{stepsize}$$

    - Dout = digital data output (in decimal),
    - Vin = analog voltage, and
    - step size (resolution) is the smallest change size step

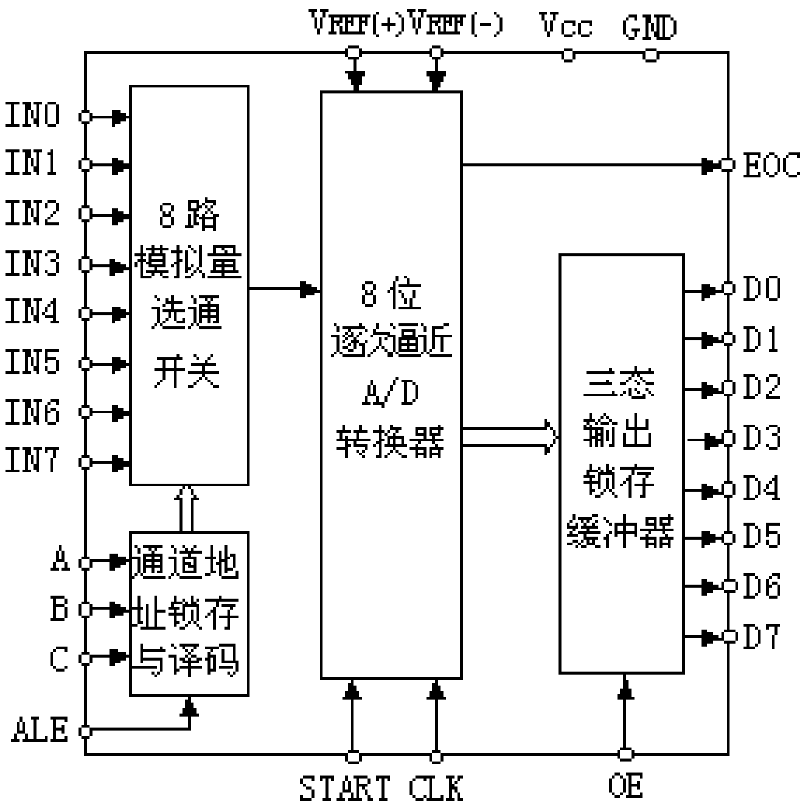
- Analog ground and digital ground
  - Analog ground is connected to the ground of the analog Vin
  - Digital ground is connected to the ground of the Vcc pin
- To isolate the analog Vin signal from transient voltages caused by digital switching of the output D0 – D7
  - This contributes to the accuracy of the digital data output

# ADC804 Clock from 8051 XTAL2

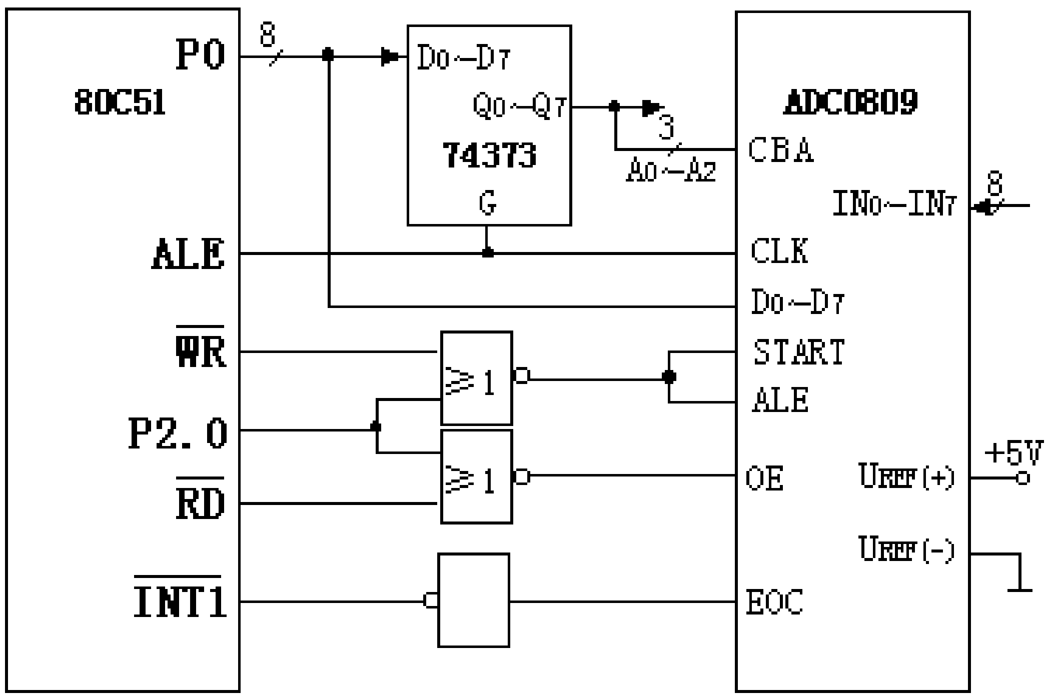
8051 Connection to ADC804 with Clock from XTAL2 of 8051



# 常用A/D转换器芯片ADC0809




ADC 0809 的结构框图




ADC 0809 与 80C51 的连接电路

# § 13.2.3 A/D转换与接口技术

## (1) ADC0809的特点

ADC0809是NS（National Semiconductor，美国国家半导体）公司生产的逐次逼近型A/D转换器。其特点如下：

① 分辨率为8位，误差1LSB；

② CMOS低功耗器件；

③ 转换时间为100 μs（当外部时钟输入频率$f_c$ = 640 kHz）；

④ 很容易与微处理器连接；

⑤ 单一电源+5V，采用单一电源+5V供电时量程为0～5V；

⑥无需零位或满量程调整，使用5V或采用经调整模拟间距的电压基准工作；

⑦ 带有锁存控制逻辑的8通道多路输入转换开关；

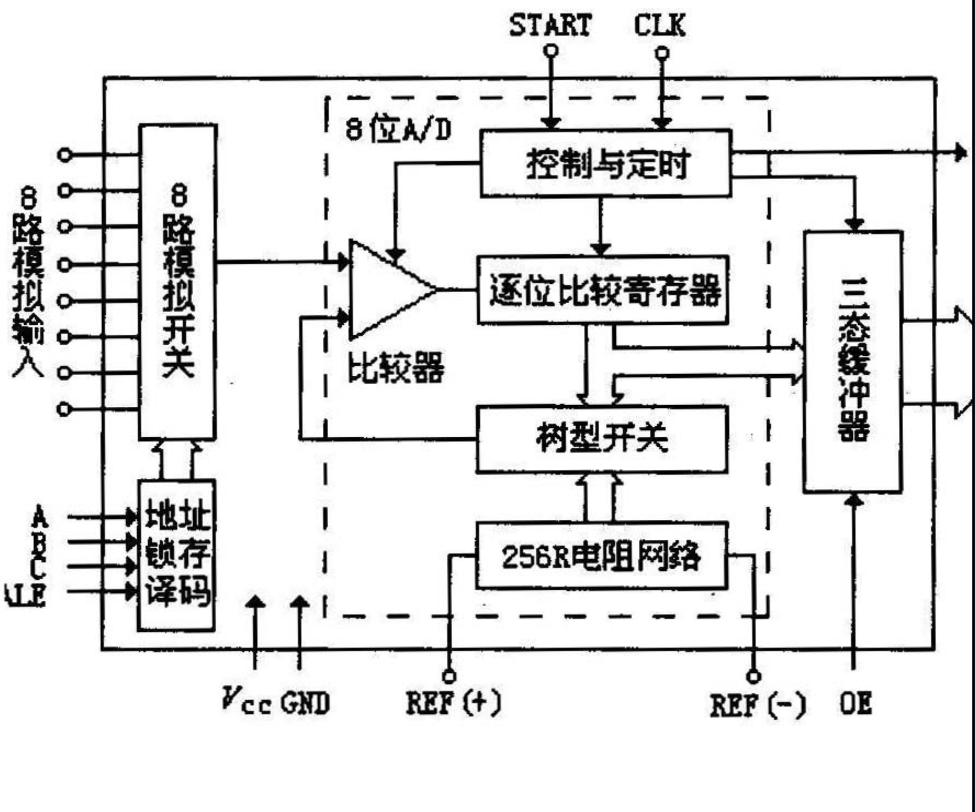
⑧ DIP28封装；

⑨带锁存器的三态数据输出。

⑩ 转换结果读取方式有延时读数、查询EOC=1、EOC申请中断。

# § 13.2.3 A/D转换与接口技术

◈ ADC0809的结构:




ADC0809 结构原理框图

(4)逐次逼近寄存器SAR（8位）：

在A/D转换过程中用以产生设定的数字量和获得正确的与输入模拟量相当的数字量。

(5)D/A部分:

包括电阻网络和树状开关，将SAR中设定的数字量按基准电压VRFE转换成模拟量。

(6)三态输出缓冲器:

A/D转换的结果被送到这里锁存、缓冲，等待结果输出。

(7)控制时序逻辑:

由START信号启动整个A/D转换过程，按CLK时钟节拍控制整个A/D转换过程，转换结束时可提供A/D转换结束信号EOC。

# § 13.2.3 A/D转换与接口技术

## （2）ADC0809引脚功能

**(6) ALE**：地址锁存允许信号输入端。ALE信号有效时将当前转换的通道地址锁存。

**(7) START**：启动A/D转换信号输入端。

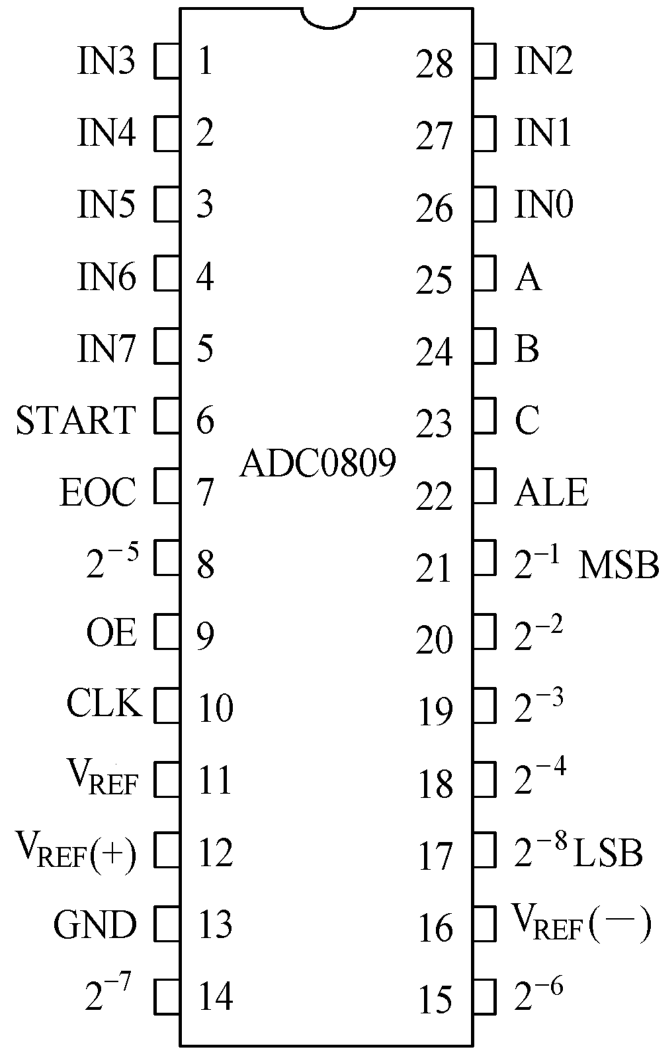
当START端输入一个正脉冲时，立即启动0809进行A/D转换。START端与ALE端连在一起，由80C51WR与0809片选端（例如P2.0）通过或非门相连。

**(8) EOC**：A/D转换结束信号输出端，高电平有效。

**(9) UREF（+）、UREF（-）**：正负基准电压输入端。

**(10) Vcc**：正电源电压（+5V）。

**GND**：接地端。



# § 13.2.3 A/D转换与接口技术

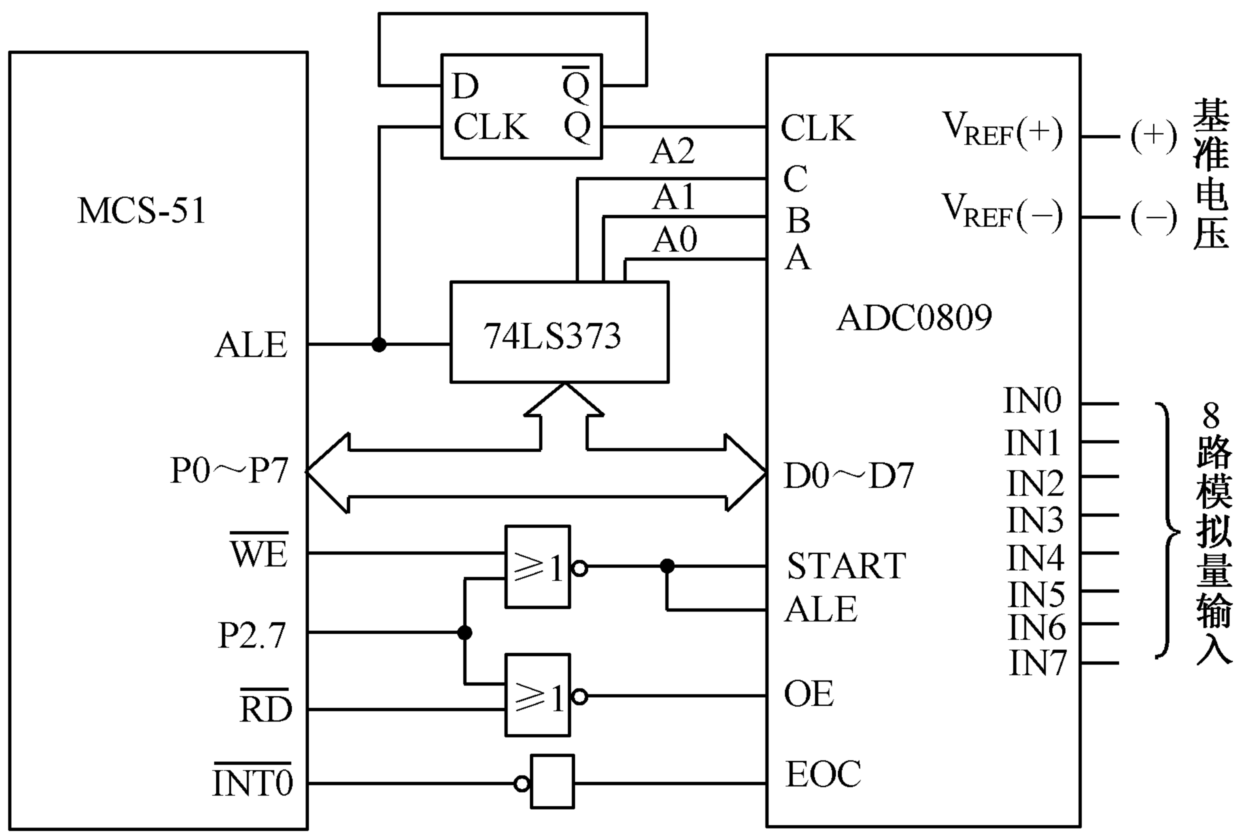
## ADC0809与单片机80C51接口

由于ADC0809输出含三态锁存，所以其数据输出可以直接连接MCS-51的数据总线P0口。数据传送方式:

1)中断方式

2) 查询方式

3)延时等待方式



# § 13.2.3 A/D转换与接口技术

## (1) 中断方式

用中断方式对8路模拟信号依次A/D转换一次，并把结果存入以30H为首址的内RAM中，试编制程序。

```
        ORG    0000H
        LJMP   STAT
        ORG    0013H                    ;中断服务子程序入口地址
        LJMP   PINT1
        ORG    0100H                    ;初始化程序首地址
STAT:   MOV    R1,#30H                  ;置数据区首址
        MOV    R7,#8                    ;置转换通道数
        SETB   IT1                      ;置边沿触发方式
        SETB   EX1                      ;开外中断
        SETB   EA                       ;CPU开中断
        MOV    DPTR,#07FF8H             ;置0809通道0地址
        MOVX   @DPTR,A                  ;启动0通道A/D
        SJMP   $                        ;等待A/D中断
```

# § 13.2.3 A/D转换与接口技术

```
         ORG    0200H
PINT1:   PUSH   ACC             ;保护现场
         PUSH   PSW
         MOVX   A,@DPTR         ;读A/D值
         MOV    @R1,A
         INC    DPTR            ;修正通道地址
         INC    R1              ;修正数据区地址
         MOVX   @DPTR,A         ;启动下一通道A/D
         DJNZ   R7,GORETI       ;判8路采集完否?
         CLR    EX1             ;8路采集已完,关中断
GORETI:  POP    PSW             ;恢复现场
         POP    ACC
         RETI                   ;中断返回
```

# § 13.2.3 A/D转换与接口技术

## (2) 查询方式

工作在查询方式时,0809 EOC端可直接与80C51 P1口或P3口中任一端线相连。设用P1.0直接与0809 EOC端相连，试用查询方式编制程序，对8路模拟信号依次A/D转换一次，并把结果存入以40H为首址的内RAM中。

```
MAIN:  MOV    R1,#40H          ;置数据区首址
       MOV    R7,#8            ;置通道数
       SETB   P1.0             ;置P1.0输入态
       MOV    DPTR,#07FF8H     ;置0809通道0地址
LOOP:  MOVX   @DPTR,A          ;启动A/D
       JNB    P1.0,$           ;查询A/D转换结束否? 未完继续查询等待
       MOVX   A,@DPTR          ;A/D已结束,读A/D值
       MOV    @R1,A            ;存A/D值
       INC    DPTR             ;修改通道地址
       INC    R1               ;修改数据区地址
       DJNZ   R7,LOOP          ;判8路采集完否?未完继续
       RET                     ;8路采集完毕,返回
```

# § 13.2.3　A/D转换与接口技术

## (3) 延时等待方式

工作在延时等待方式时，0809 EOC端可不必与80C51相连，是根据时钟频率计算出A/D转换时间，略微延长后直接读A/D转换值。0809 EOC端开路，fosc=6MHz，试用延时等待方式编制程序，对8路模拟信号依次A/D转换一次，并把结果存入以50H为首址的内RAM中。

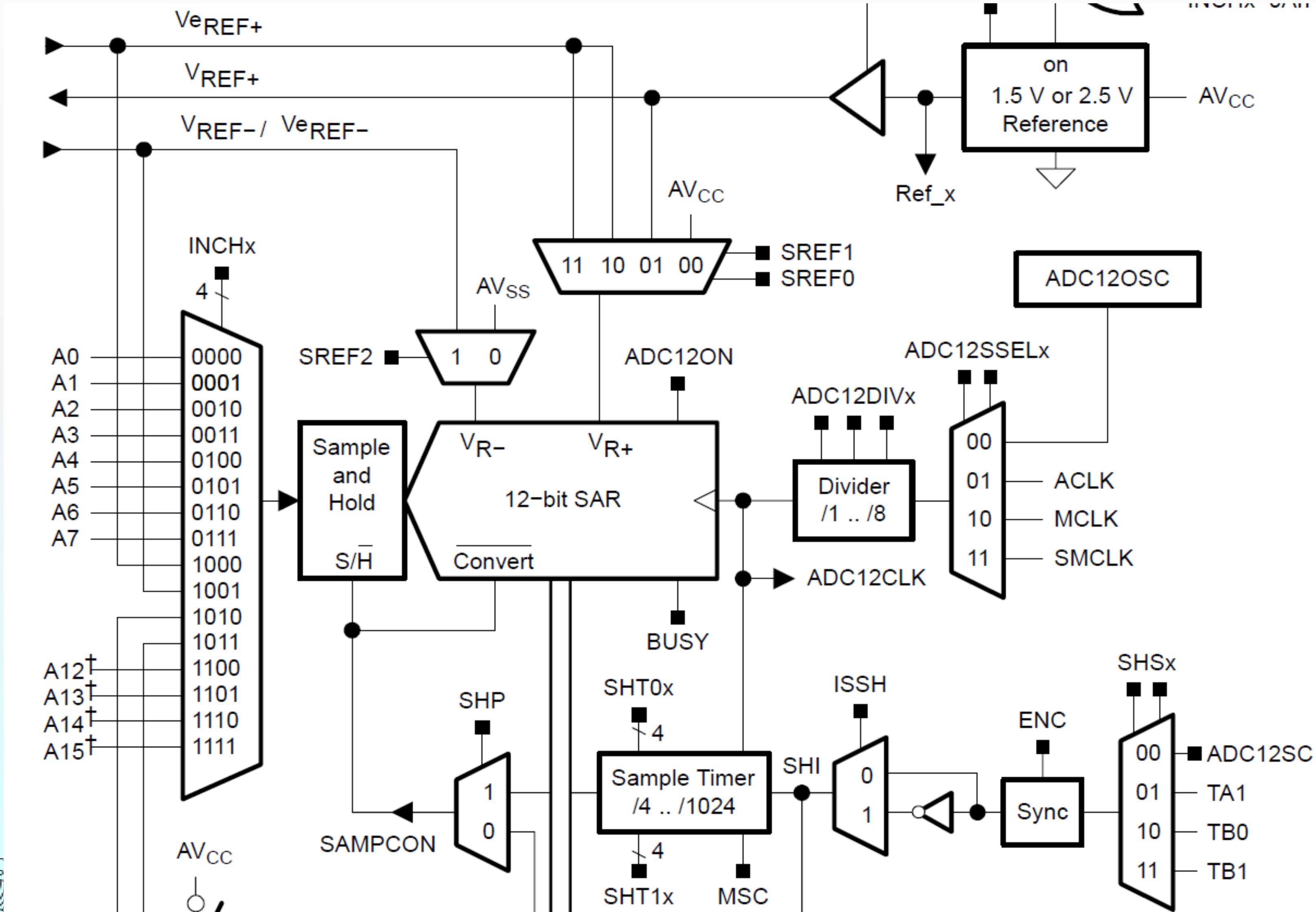
```
MAIN:   MOV     R1,#50H         ;置数据区首址
        MOV     R7,#8           ;置通道数
        MOV     DPTR,#07FF8H;置0809通道0地址
LOOP:   MOVX    @DPTR,A         ;启动A/D
        MOV     R6,#50
        DJNZ    R6,$            ;延时100μS:2μS×50=100μS
        MOVX    A,@DPTR         ;读A/D值
        MOV     @R1,A
        INC     DPTR            ;修正通道地址
        INC     R1              ;修正数据区地址
        DJNZ    R7,LOOP         ;判8路采集完否?未完继续
        RET                     ;8路采集完毕,返回
```

# Modern MCU ADC

Key features of the MSP430x4xx family include:

- Ultralow-power architecture extends battery life
  - 0.1-µA RAM retention
  - 0.8-µA real-time clock mode
  - 250-µA / MIPS active
- High-performance analog ideal for precision measurement
  - 12-bit or 10-bit ADC — 200 ksps, temperature sensor, $V_{Ref}$
  - 12-bit dual-DAC
  - Comparator-gated timers for measuring resistive elements
  - Supply voltage supervisor
- 16-bit RISC CPU enables new applications at a fraction of the code size.

# MSP430F4xx ADC structure



# ADC program

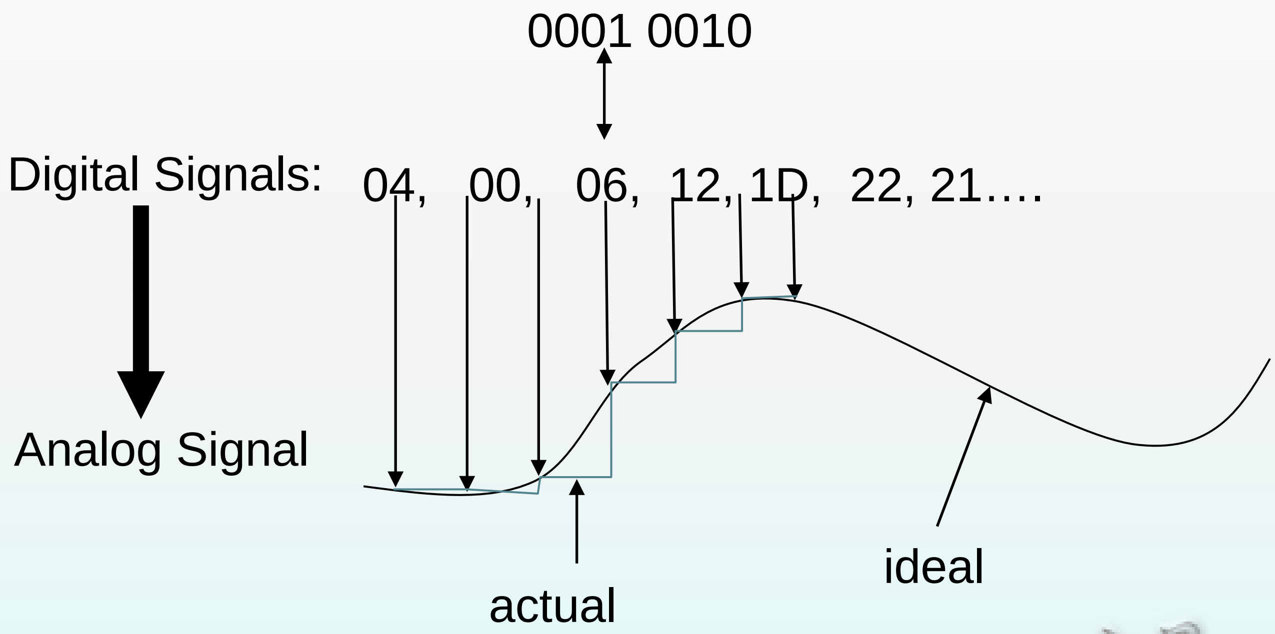
- //************************************************************
- // 功能：ADC12采样
- // 入口：uchar i：ADC通道（1----12）
- // 出口：uint ADC12MEM：AD采样值（12bit）
- // 说明：1、使用外部参考电源Vr+=VeRef+,Vr-=AVss;
- //  2、单通道单次转换模式;
- //  3、单次转换地址为ADC12MEM0
- //  4、采用ACLK时钟
- //************************************************************

# ADC program （2）

```
uint ADC_sample( uchar i )
{
 uint ADC_result;
 ADC12CTL0 &= ~ENC;
 ADC12CTL0 = ADC12ON + SHT0_2;          //打开ADC12内核,设置采样周期4*16*t(aclk)
 //定义ADC12MEM0为单次转换地址;采样信号来自采样定时器;单通道单次转换模式;内核
    时钟源为MCLK
 ADC12CTL1 = CSTARTADD_0 + SHP + CONSEQ_0 + ADC12SSEL_2;
 ADC12MCTL0 = (i) + SREF_2 + EOS;       //选择第i通道，参考电源Vr+=Veref+,Vr-=AVss;
 ADC12CTL0 |= ENC + ADC12SC;            //开始转换
 while ( ( ADC12CTL1 & ADC12BUSY ) == 1 );   //ADC12BUSY?
 ADC12CTL0 &= ~ENC;
 ADC12CTL0 &= ~ADC12ON;                 //关闭ADC内核电源
 ADC_result = ADCMEM[0];                //将ADC12MEMx给Result
 _NOP();
 return ADC_result;
}
```

# § 13-3 Interfacing to DAC

Digital-to-analog convert



# DAC Devices

- There are several series of DAC， which have different functions.
- Features

a. Format of digital numbers: binary number

8 bits, 10 bits, 12 bits, 14 bits, 16 bits

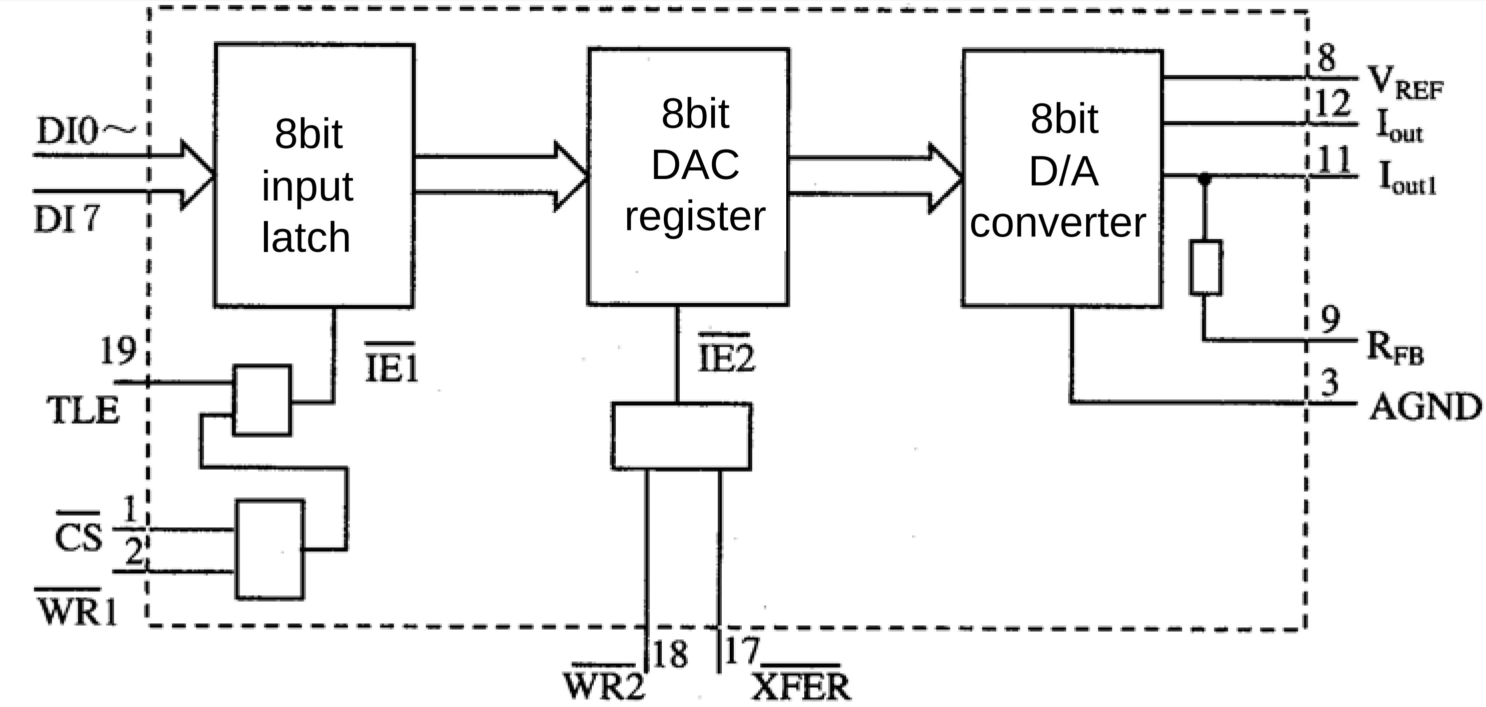
b. Output form : Current output and Voltage output

c. Self-contained reference voltage $V_{REF}$ and circumscribed reference voltage $V_{REF}$。

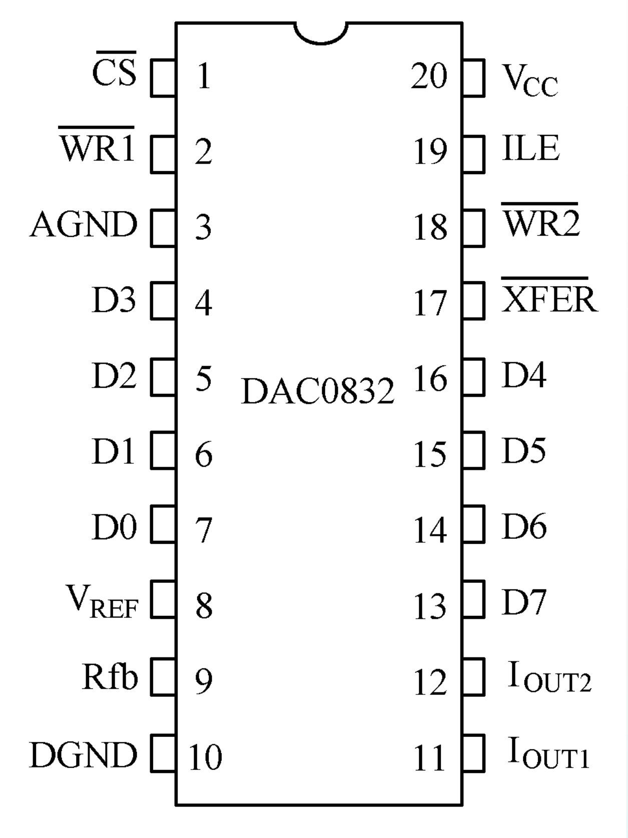
d. Output without latch 、 Output with latch 、 Buffer with two-stage

e. Input form: parallel and serial

- **DAC with latch**
  - DAC 0832 is a typical 8 bit D/A chip with two-data-buffer.
  - Produced by National Semiconductor



- **DAC0832 pin**

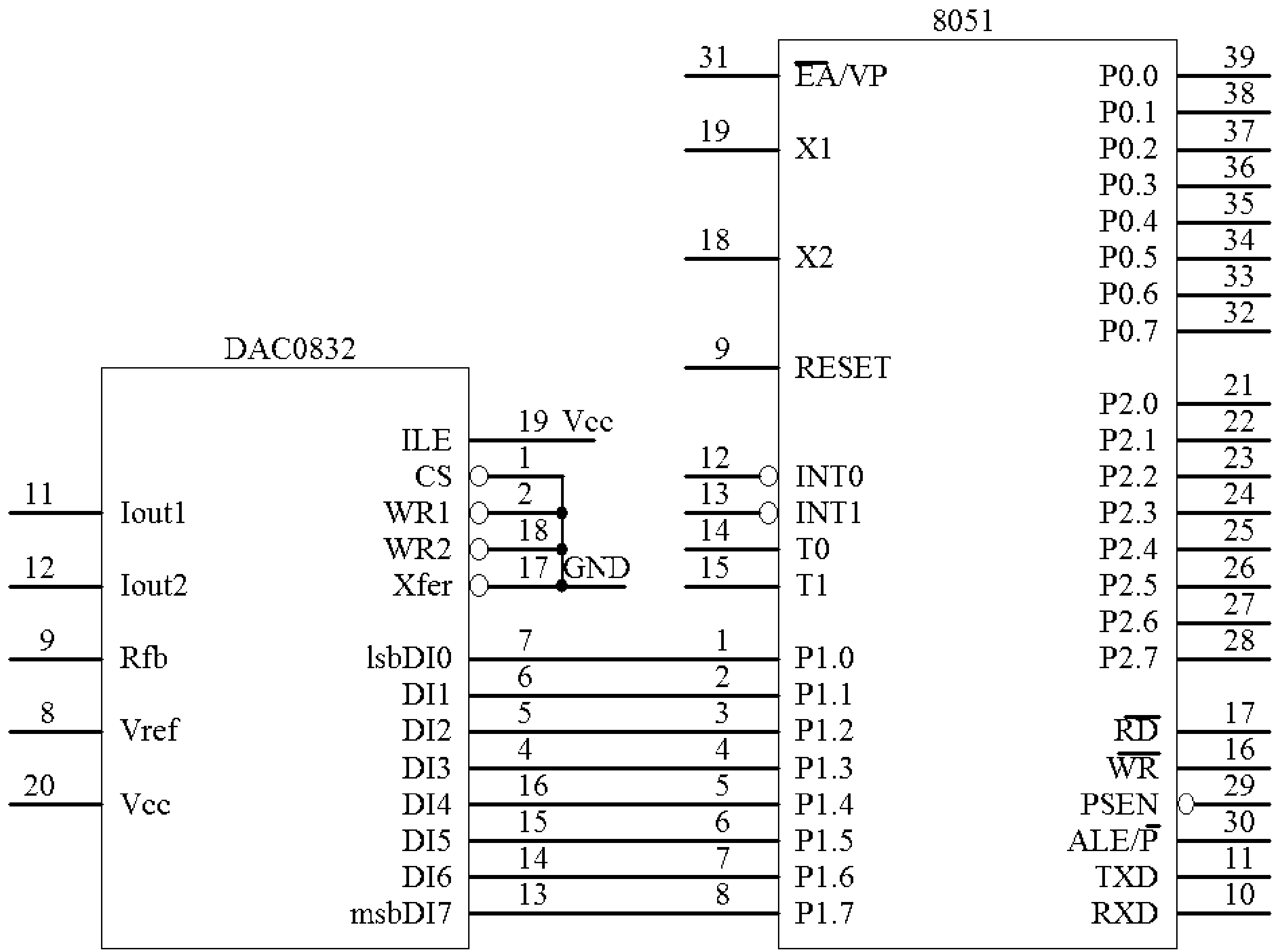

CS
WR1
AGND
D3
D2
D1
D0
VREF
Rfb
DGND
1
2
3
4
5
6
7
8
9
10
DAC0832
20
19
18
17
16
15
14
13
12
11
VCC
ILE
WR2
XFER
D4
D5
D6
D7
IOUT2
IOUT1

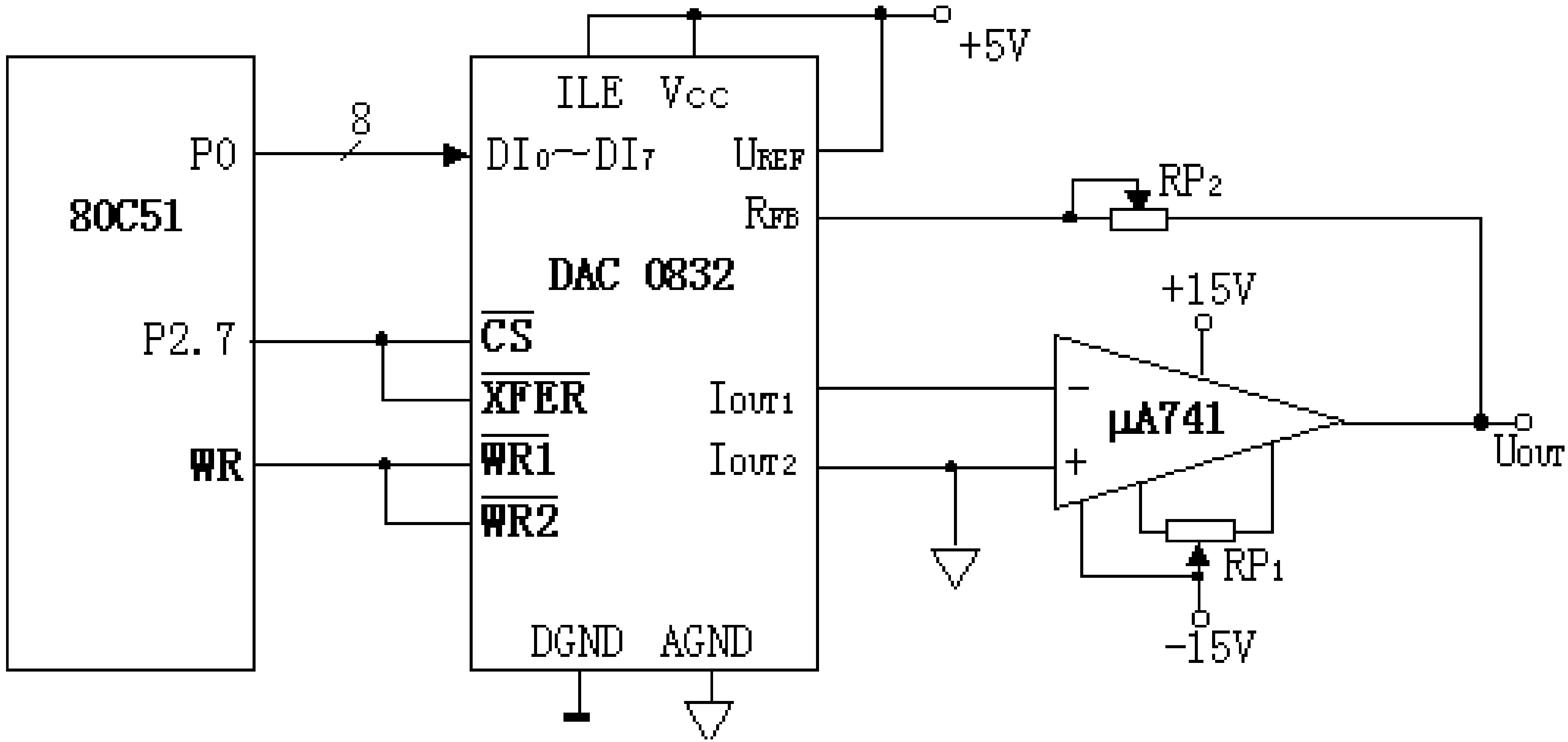
## DAC 0832 operating mode

Using command to control:ILE、CS、WR1、WR2、XFER

(1) Direct connection: 5 control ports are all effective, direct D/A
(2) Single buffering: 5 control ports being gated once
(3) Double buffering: 5 control ports being gated through two times

# (1) Direct connection:



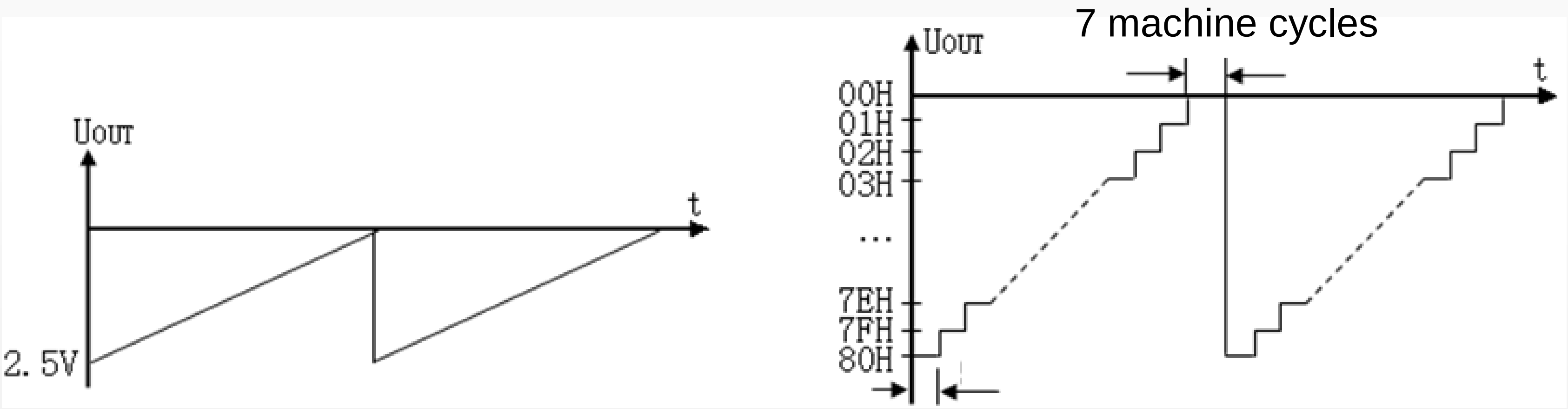



DAC 0832 Single buffering mode

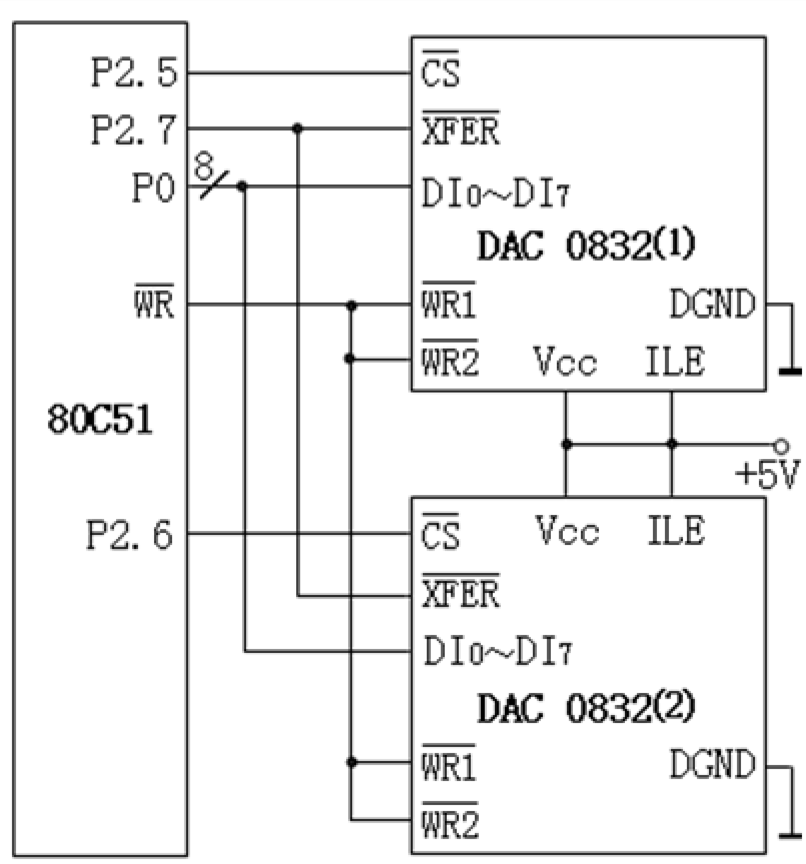
Figure 13-1

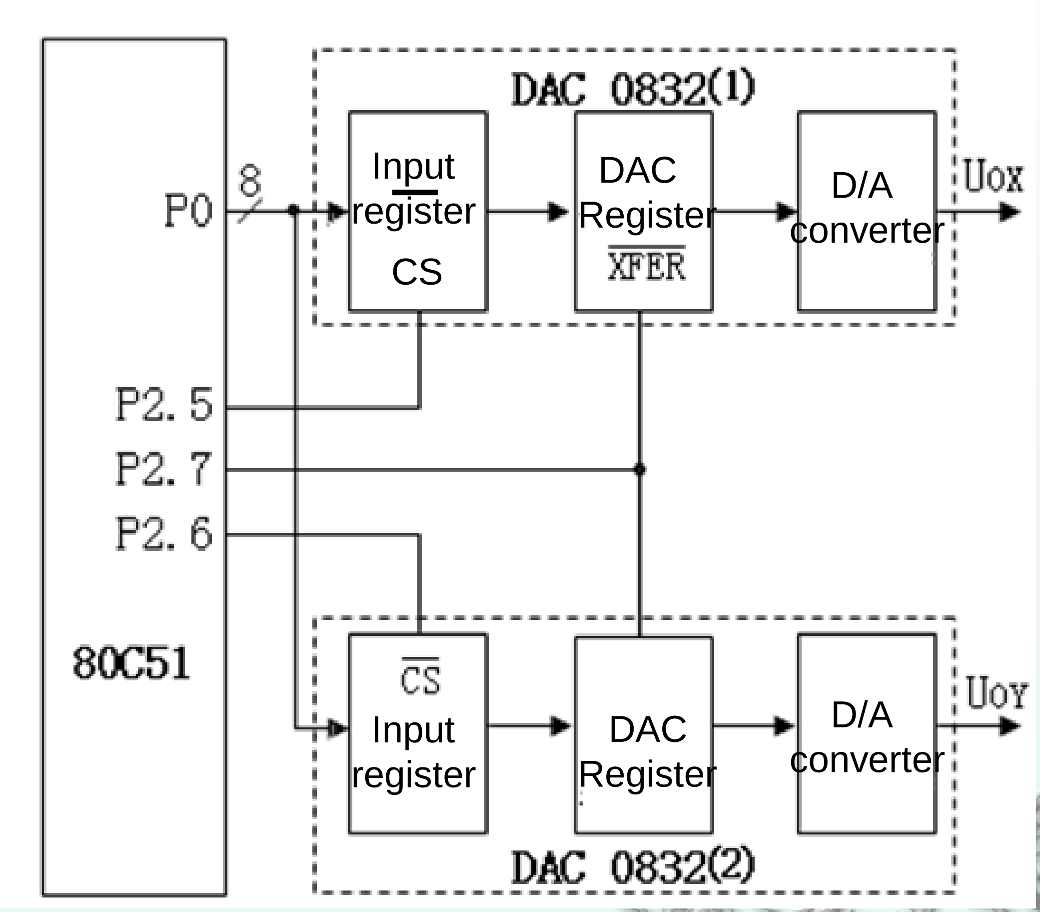
# For Figure 13-1

Output saw tooth wave as following, amplitude UREF/2=2.5V




```
START: MOV   DPTR,#7FFFH ;set DAC0832 address
LOOP1: MOV   R7,#80H     ;set saw tooth wave amplitude  1 machine cycle
LOOP2: MOV   A,R7        ;read output value                          1
       MOVX  @DPTR,A       ;output;                                  2
       DJNZ  R7,LOOP2      ;                                         2
       SJMP  LOOP1         ;                                         2
```




Double buffering mode

THANK YOU!!