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

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

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

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

- $t_D = $ Data output delay time
- $t_{AH} = $ Hold time after E has come down for both RS and R/W = 10 ns (minimum)
- $t_{AS} = $ Setup time prior to E (going high) for both RS and R/W = 140 ns (minimum)

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

- \( \text{AAAAAAA}=000\_0000 \text{ to } 010\_0111 \) for line 1
- \( \text{AAAAAAA}=100\_0000 \text{ to } 110\_0111 \) for line 2

**LCD Addressing for the LCDs of 40 \times 2 \text{ size}**

<table>
<thead>
<tr>
<th></th>
<th>DB7</th>
<th>DB6</th>
<th>DB5</th>
<th>DB4</th>
<th>DB3</th>
<th>DB2</th>
<th>DB1</th>
<th>DB0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line1 (min)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Line1 (max)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Line2 (min)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Line2 (max)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The upper address range can go as high as 0100111 for the 40-character-wide LCD, which corresponds to locations 0 to 39.
Write an 8051 C program to send letters ‘M’, ‘D’, and ‘E’ to the LCD using the busy flag method.

Solution:

```c
#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;
}
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 $D_3 - D_0 = 1111$, 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 $D_3 - D_0 = 1101$, 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) $D_3 - D_0 = 1110$ for the row, $D_3 - D_0 = 1011$ for the column
(b) $D_3 - D_0 = 1101$ for the row, $D_3 - D_0 = 0111$ for the column

Solution:
From Figure 13-5 the row and column can be used to identify the key.
(a) The row belongs to $D_0$ and the column belongs to $D_2$; therefore, key number 2 was pressed.
(b) The row belongs to $D_1$ and the column belongs to $D_3$; therefore, key number 7 was pressed.
Flowchart for Program

Start

Ground all rows

Read all columns

All keys open?

no

All keys down?

yes

Wait for debounce

Read all columns

no

All keys down?

yes

Get scan code from table

no

Find which key is pressed

yes

Ground next row

Return
; 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
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 a 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 $V_{in} = V_{in (+)} - V_{in (-)}$. $V_{in (-)}$ is connected to ground and $V_{in (+)}$ is used as the analog input to be converted.

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.

+5V power supply or a reference voltage when $V_{ref}/2$ input is open (not connected)
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 \( R = 10\, \text{K ohms} \) and \( C = 150\, \text{pF} \)
- We get \( f = 606\, \text{kHz} \) and the conversion time is 110 \( \mu \text{s} \)
**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

### V_{ref}/2 Relation to Vin Range

<table>
<thead>
<tr>
<th>( V_{ref}/2 (V) )</th>
<th>( V_{in} (V) )</th>
<th>Step Size (mV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not connected*</td>
<td>0 to 5</td>
<td>5/256 = 19.53</td>
</tr>
<tr>
<td>2.0</td>
<td>0 to 4</td>
<td>4/255 = 15.62</td>
</tr>
<tr>
<td>1.5</td>
<td>0 to 3</td>
<td>3/256 = 11.71</td>
</tr>
<tr>
<td>1.28</td>
<td>0 to 2.56</td>
<td>2.56/256 = 10</td>
</tr>
<tr>
<td>1.0</td>
<td>0 to 2</td>
<td>2/256 = 7.81</td>
</tr>
<tr>
<td>0.5</td>
<td>0 to 1</td>
<td>1/256 = 3.90</td>
</tr>
</tbody>
</table>

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
  \[ D_{\text{out}} = \frac{V_{\text{in}}}{\text{steps}ize} \]
  - 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

The frequency of crystal is too high, we use two D flip-flops to divide the frequency by 4.
常用A/D转换器芯片ADC0809
§ 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 \text{ 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。
ADC0809的结构:

1. **ADC0809的结构**

   - **⑴ 8路模拟开关**：
     ADC0809 A/D转换器芯片可以有8路模拟量输入，在地址锁存译码电路控制下由8路模拟开关选择一路模拟量进行A/D转换。
   
   - **⑵ 地址锁存译码**：
     由地址锁存允许信号ALE锁存A、B、C三个输入端的地址信息，借以决定从8路模拟量输入ACH0~ACH7中选择一路进行A/D转换。
   
   - **⑶ 模拟比较器**：
     用以将选中的模拟量与芯片内部设定的数字量所产生的模拟量进行比较；
   
   - **⑷ 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。
   
   - **⑸ 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。
   
   - **⑹ 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。
   
   - **⑺ 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
  PUSH PSW
  MOVX A,@DPTR
  MOV @R1,A
  INC DPTR
  INC R1
  MOVX @DPTR,A
  DJNZ R7,GORETI
  CLR EX1

GORETI:
  POP PSW
  POP ACC
  RETI
§ 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 ksp, temperature sensor, $V_{\text{Ref}}$
  - 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

/*****************************/
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 );   // ADC12_BUSY?
    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

Digital Signals: 04, 00, 06, 12, 1D, 22, 21, ...

Analog Signal

0001 0010

ideal

actual

04, 00, 06, 12, 1D, 22, 21, ....
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
- 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
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:
Single buffering mode

Figure 13-1
For Figure 13-1
Output saw tooth wave as following, amplitude $\frac{\text{UREF}}{2} = 2.5\text{V}$

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

7 machine cycles

5 machine cycles
Double buffering mode
THANK YOU!!