Principle and Interface Techniques of Microcontroller

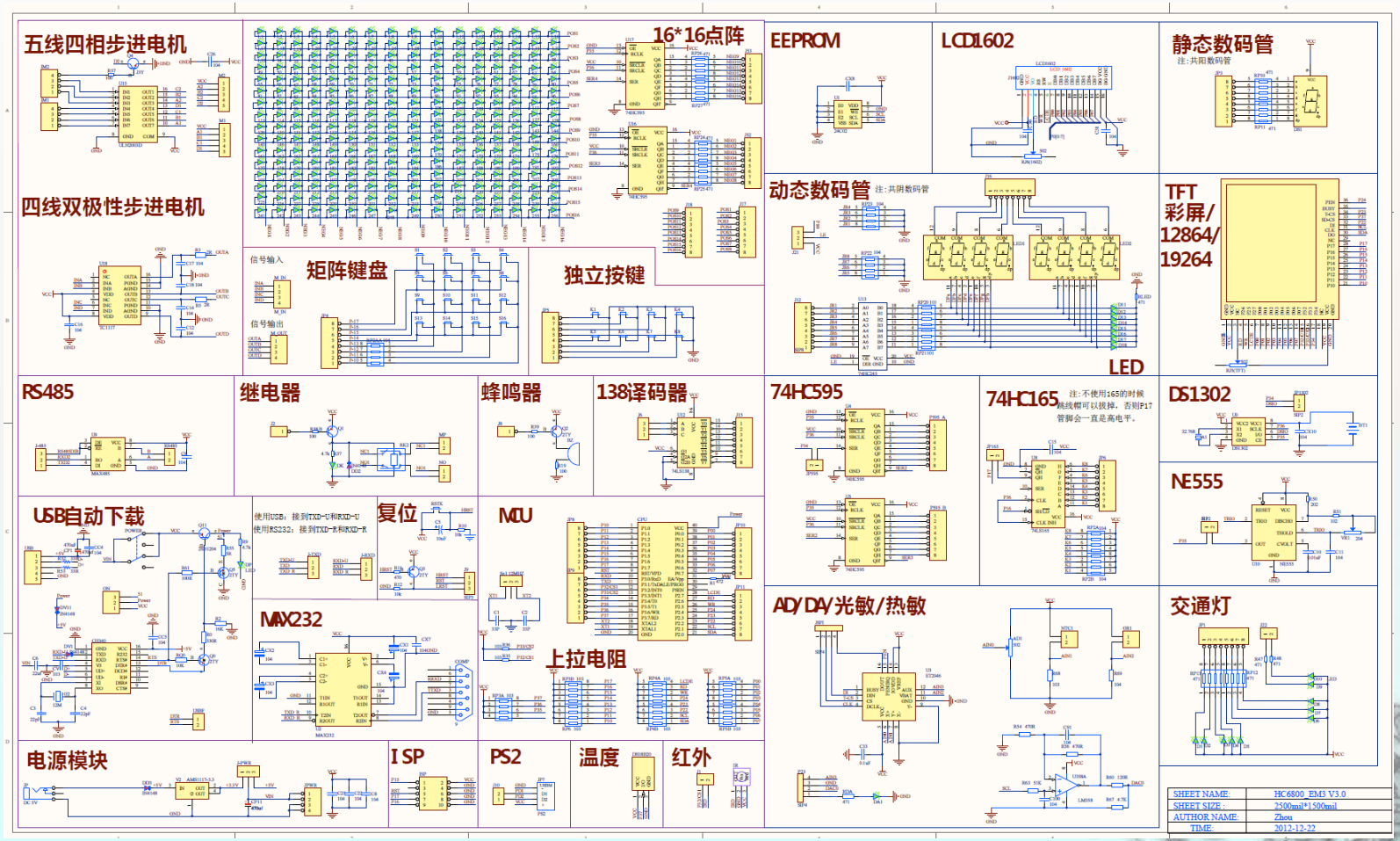
--8051 Microcontroller and Embedded Systems Using Assembly and C

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

WANG, You (王酉) **PhD, MIET**

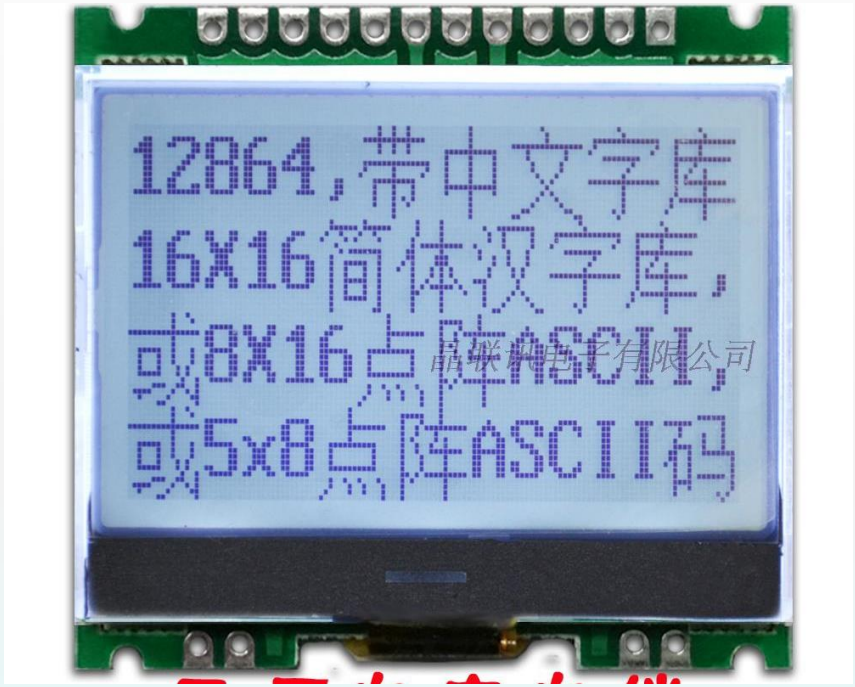
杭州 • 浙江大学 • 2021

开发板电路图


五线四相步进电机
四线双极性步进电机
16*16点阵
矩阵键盘
信号输入
信号输出
独立按键
EEPROM
LCD1602
静态数码管
注：共阳数码管
动态数码管
注：共阴数码管
LED
TFT
彩屏/
12864/
19264
RS485
继电器
蜂鸣器
138译码器
74HC595
74HC165
注：不使用165的时候
跳线帽可以拔掉，否则P17
管脚会一直是高电平。
DS1302
NE555
USB自动下载
使用USB：接到TXD-U和RXD-U
使用RS232：接到TXD-R和RXD-R
复位
MCU
MAX232
上拉电阻
AD/DA/光敏/热敏
交通灯
电源模块
ISP
PS2
温度
红外
SHEET NAME: HC6800_EM3 V3.0
SHEET SIZE: 2500mil*1500mil
AUTHOR NAME: Zhou
TIME: 2012-12-22

Chapter 14
LCD & Keyboard

LCD


12864,带中文字库
16X16简体汉字库,
或8X16点阵ASCII,
或5x8点阵ASCII码


Turbo
88
188
°C
188
min.
sec.
8.8:88

LCD

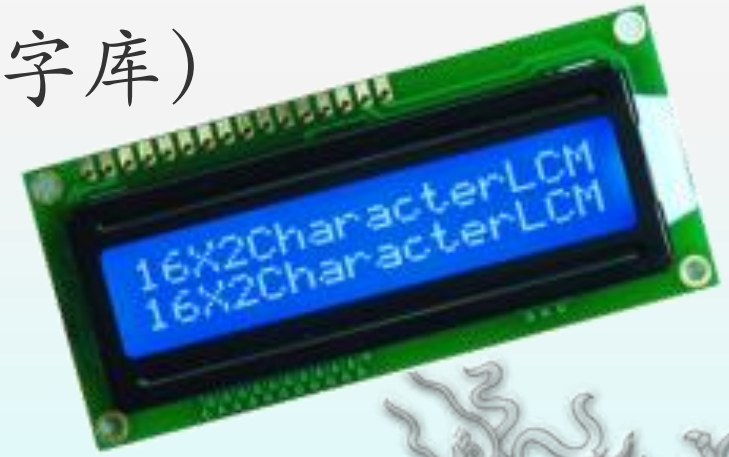
§13-1 LCD and Keyboard Interfacing

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

LCD1602

- 广泛使用的一种字符型液晶显示模块
- 控制驱动主电路为HD44780
- 显示16×2个字符（英文大小写+数字）
- 不支持中文（可以自编字库）



Pin Descriptions for LCD1602

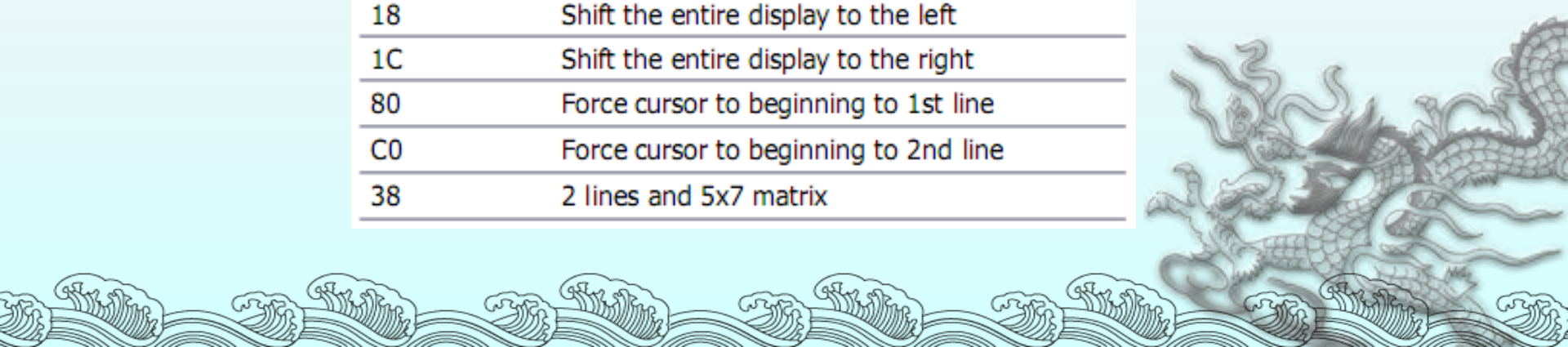
Pin	Symbol	I/O	Descriptions
1	VSS	--	Ground
2	VCC	--	+5V power supply
3	VEE	--	Power supply to control contrast
4	RS	I	RS=0 to select command register, RS=1 to select data register
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
7	DB0	I/O	The 8-bit data bus
8	DB1	I/O	The 8-bit data bus
9	DB2	I/O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

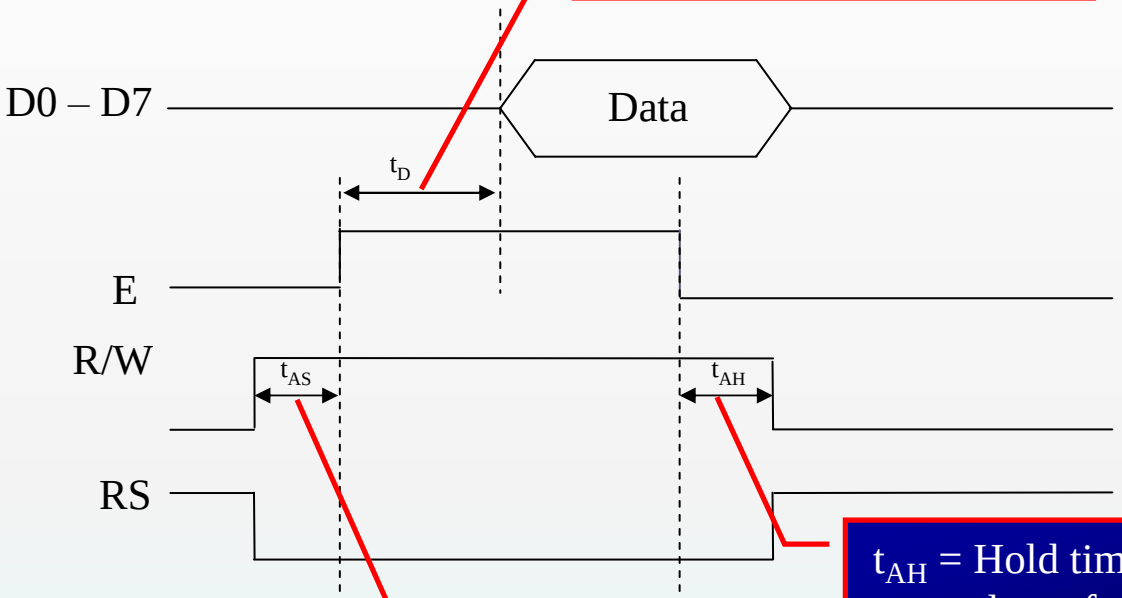
- Send displayed information or instruction command codes to the LCD
- Read the contents of the LCD's internal registers

LCD1602 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



LCD Timing for Read



t_D = Data output delay time

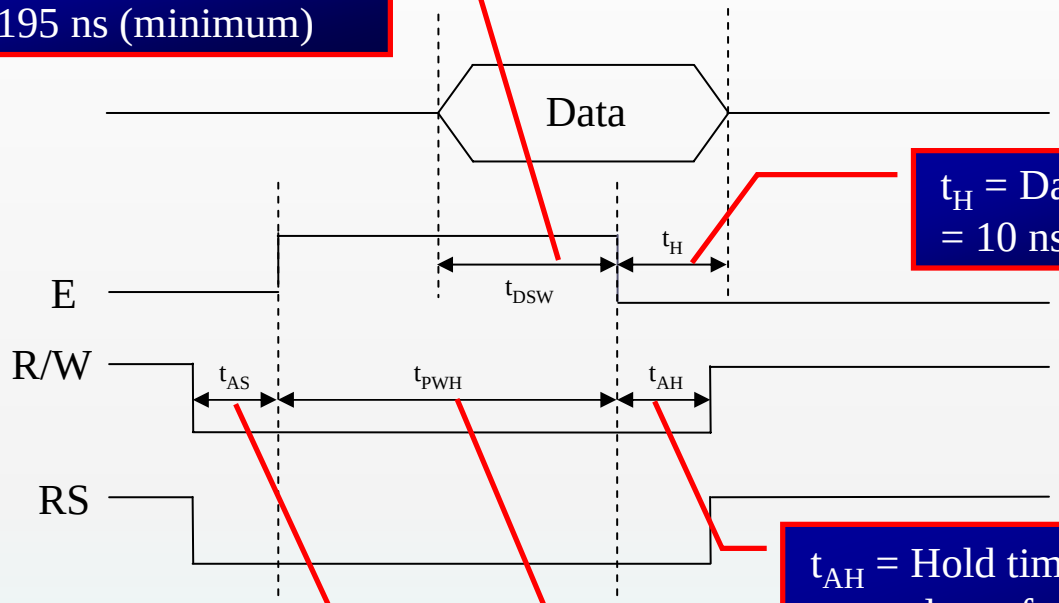
t_{AH} = Hold time after E has come down for both RS and R/W = 10 ns (minimum)

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

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

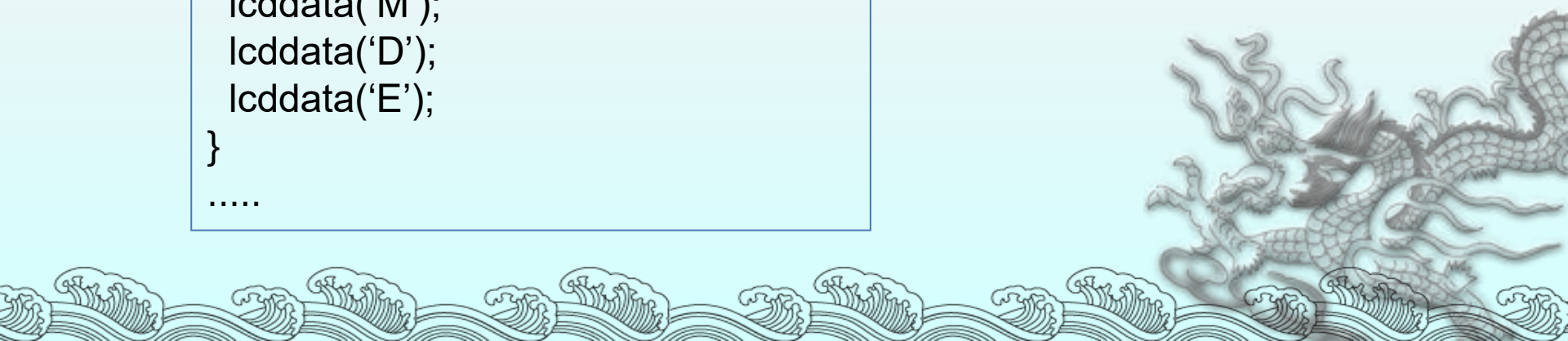
LCD Addressing for the LCDs of 40×2 size

	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.

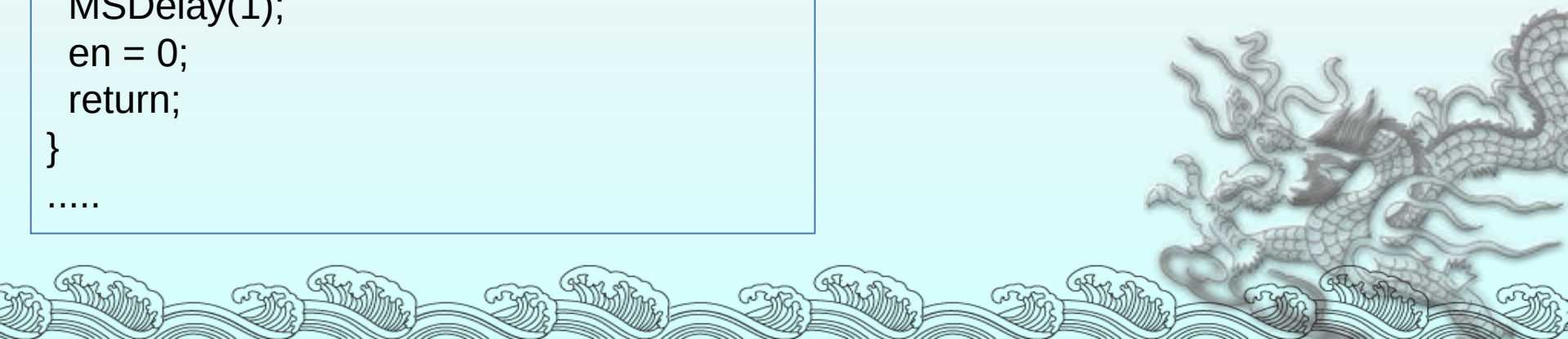
Solution:

```
#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
  lcddata(‘M’);
  lcddata(‘D’);
  lcddata(‘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;
}
```



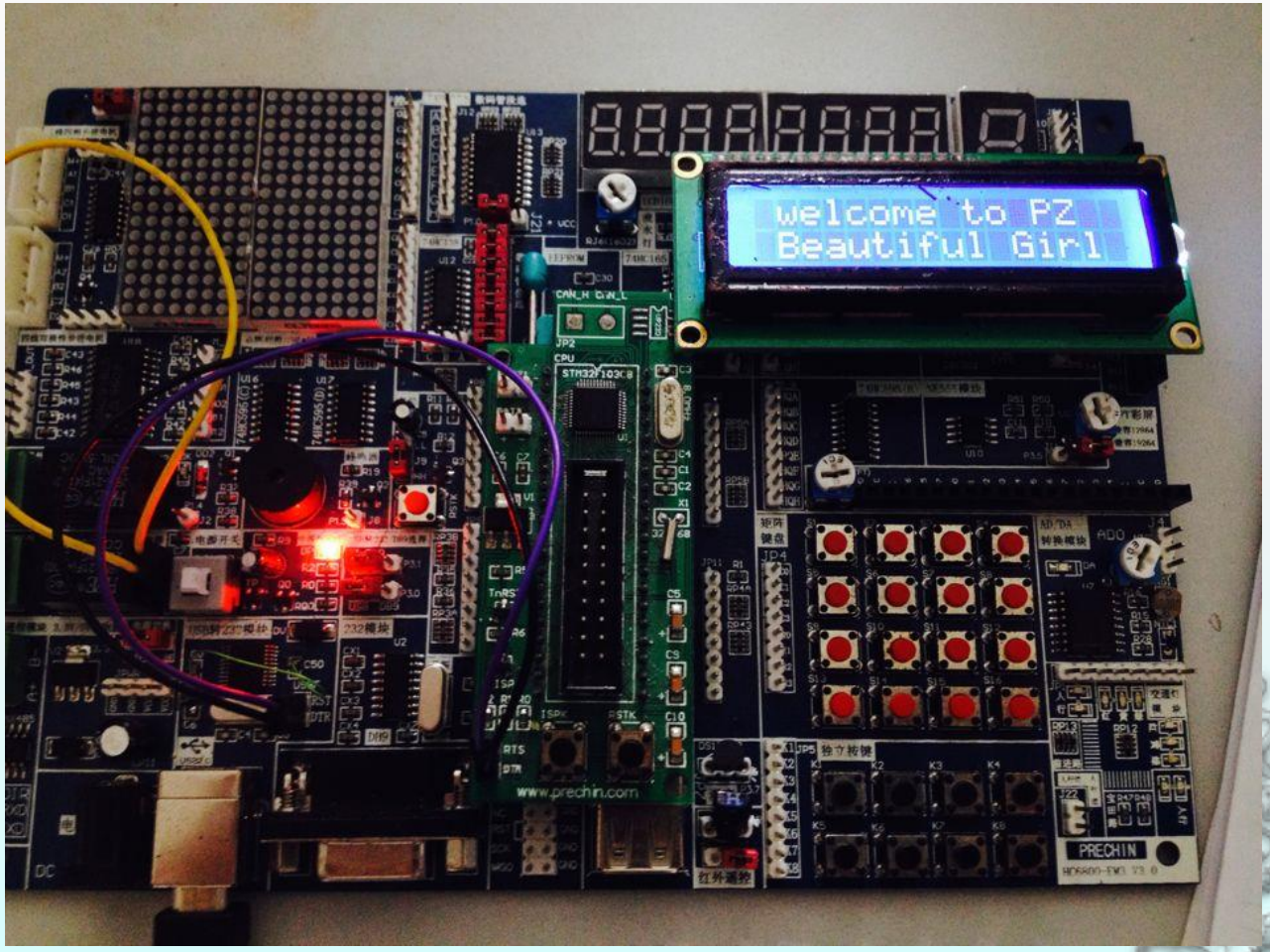
STM32驱动LCD1602显示程序

```
void lcdinit()          //LCD初始化
{
  delayms(15);
  lcdwrc4bit(0x32);
  delayms(5);
  lcdwrc4bit(0x28);
  delayms(5);
  lcdwrc4bit(0x08);
  delayms(5);
  lcdwrc4bit(0x01);
  delayms(5);
  lcdwrc4bit(0x06);
  delayms(5);
  lcdwrc4bit(0x0c);
  delayms(5);
}
```

```
void display()        //显示
{
  u8 i;
  lcdwrc4bit(0x00+0x80);
  for(i=0;i<16;i++)
  {
    lcdwrd(a[i]);
  }
  lcdwrc4bit(0x40+0x80);
  for(i=0;i<16;i++)
  {
    lcdwrd(b[i]);
  }
}
```

```
void lcdwrd(long dat)      //读八位数据通过4个引脚
{
  while(readbusy());
  GPIO_SetBits(GPIOB,rs);
  GPIO_ResetBits(GPIOB,rw);
  GPIO_ResetBits(GPIOB,e);
  delayms(1);
  GPIOB->BSRR = dat<<8 & 0xf000;  //将数据送到P0口
    GPIOB->BRR = ((~dat)<<8) & 0xf000;
  delayms(1);
  GPIO_SetBits(GPIOB,e);
  delayms(1);
  GPIO_ResetBits(GPIOB,e);
  delayms(1);
  GPIOB->BSRR = dat<<12 & 0xf000;  //将数据送到P0口
    GPIOB->BRR = ((~dat)<<12) & 0xf000;
  delayms(1);
  GPIO_SetBits(GPIOB,e);
  delayms(1);
  GPIO_ResetBits(GPIOB,e);
  delayms(1);
  GPIO_ResetBits(GPIOB,rs);
}
```

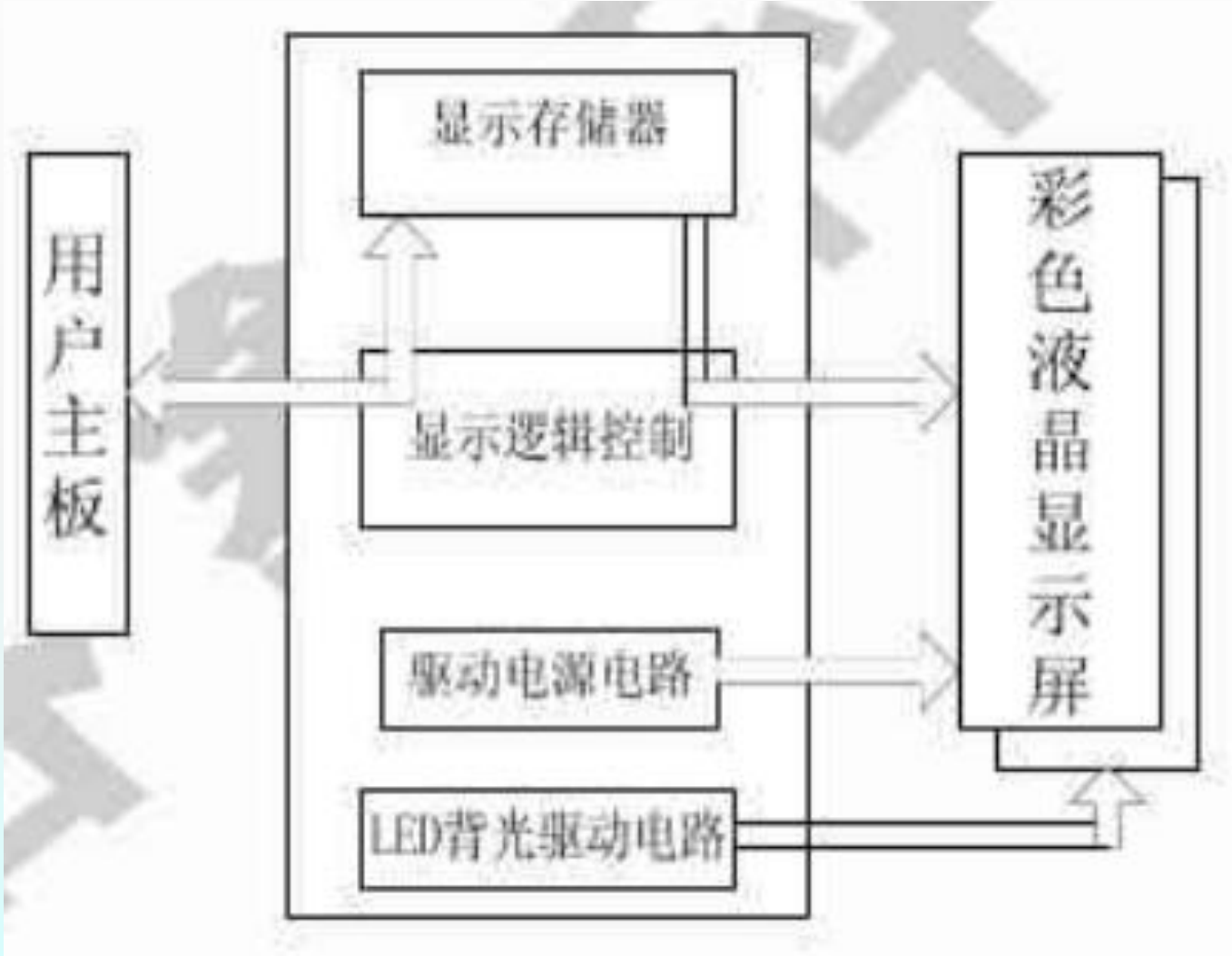
LCD1602显示效果



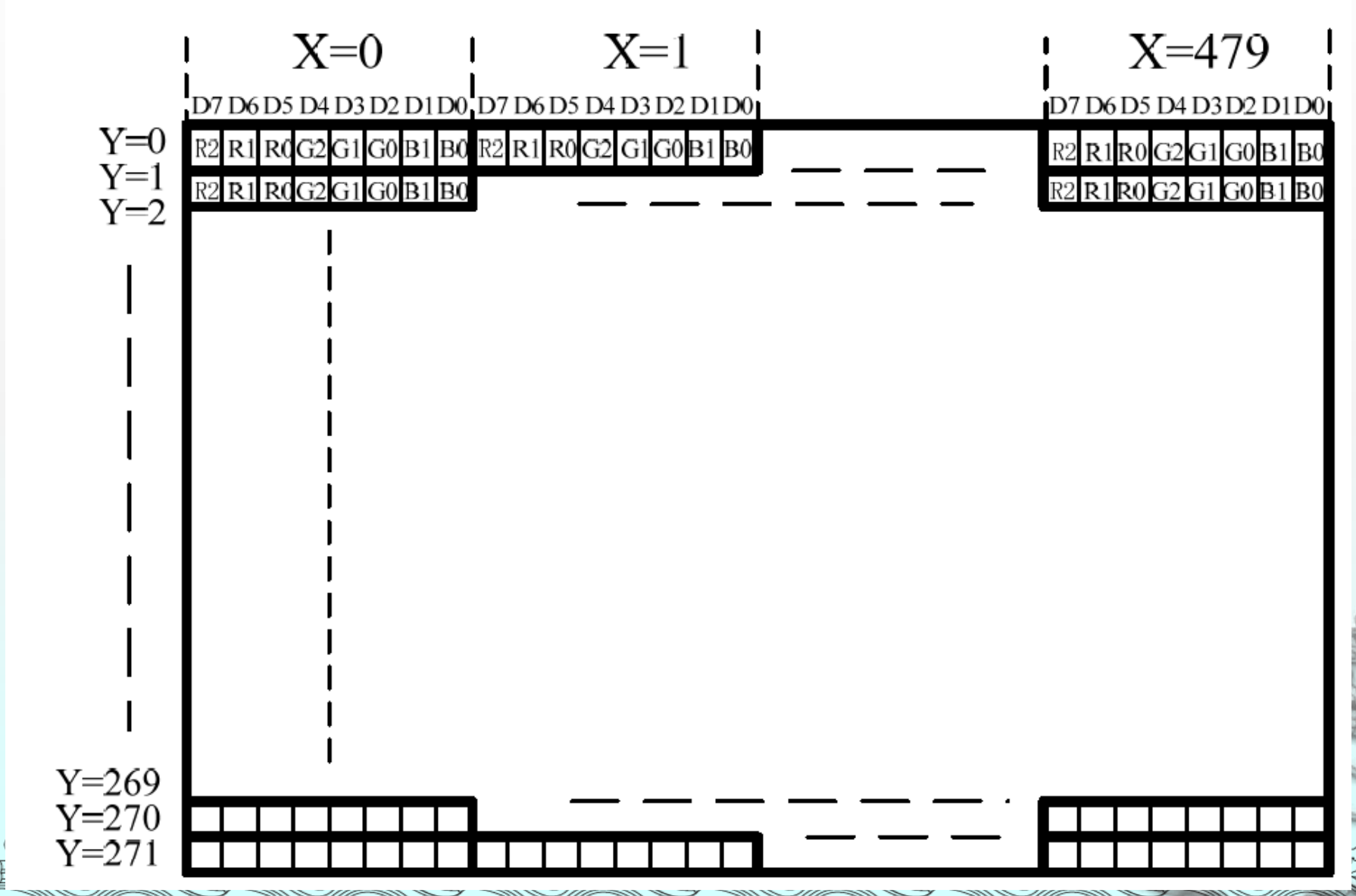
彩色LCD显示器

- TFT480272-4.3 是专门针对单片机用户而设计的液晶显示器(带触摸屏)
- 采用4.3 英寸、分辨率为480x272 的真彩TFT 屏
- 提供一个简单的高速8 位总线与单片机连接，支持256 色。
- 可以直接与MCS51、MCS96、MC68、ARM以及DSP相连。
- 直接输入X、Y 坐标，无须计算地址。

系统结构



显示存储器与像素对应关系



更多色彩

- 65536色

	Set_pixel_format	DFM	DB15	DB14	DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Command/Parameter Write	*	*									D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Command/Parameter Read	*	*									D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]

	Set_pixel_format	DFM	DB15	DB14	DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
16bpp Frame Memory Write	3'h5	*	R4]	R[3]	R[2]	R[1]	R[0]	G[5]	G[4]	G[3]	G[2]	G[1]	G[0]	B[4]	B[3]	B[2]	B[1]	B[0]
Frame Memory Read	*	*	r4]	r[3]	r[2]	r[1]	r[0]	g[5]	g[4]	g[3]	g[2]	g[1]	g[0]	b[4]	b[3]	b[2]	b[1]	b[0]

- 262K色

18-bit data bus DB[17:0] interface, IM[2:0] = 000

	Set_pixel_format	DFM	DB17	DB16	DB15	DB14	DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Command/Parameter Write	*	*											D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]
Command/Parameter Read	*	*											D[7]	D[6]	D[5]	D[4]	D[3]	D[2]	D[1]	D[0]

	Set_pixel_format	DFM	DB17	DB16	DB15	DB14	DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
18bpp Frame Memory Write	3'h6	*	R[5]	R4]	R[3]	R[2]	R[1]	R[0]	G[5]	G[4]	G[3]	G[2]	G[1]	G[0]	B[5]	B[4]	B[3]	B[2]	B[1]	B[0]
Frame Memory Read	*	*	r[5]	r4]	r[3]	r[2]	r[1]	r[0]	g[5]	g[4]	g[3]	g[2]	g[1]	g[0]	b[5]	b[4]	b[3]	b[2]	b[1]	b[0]

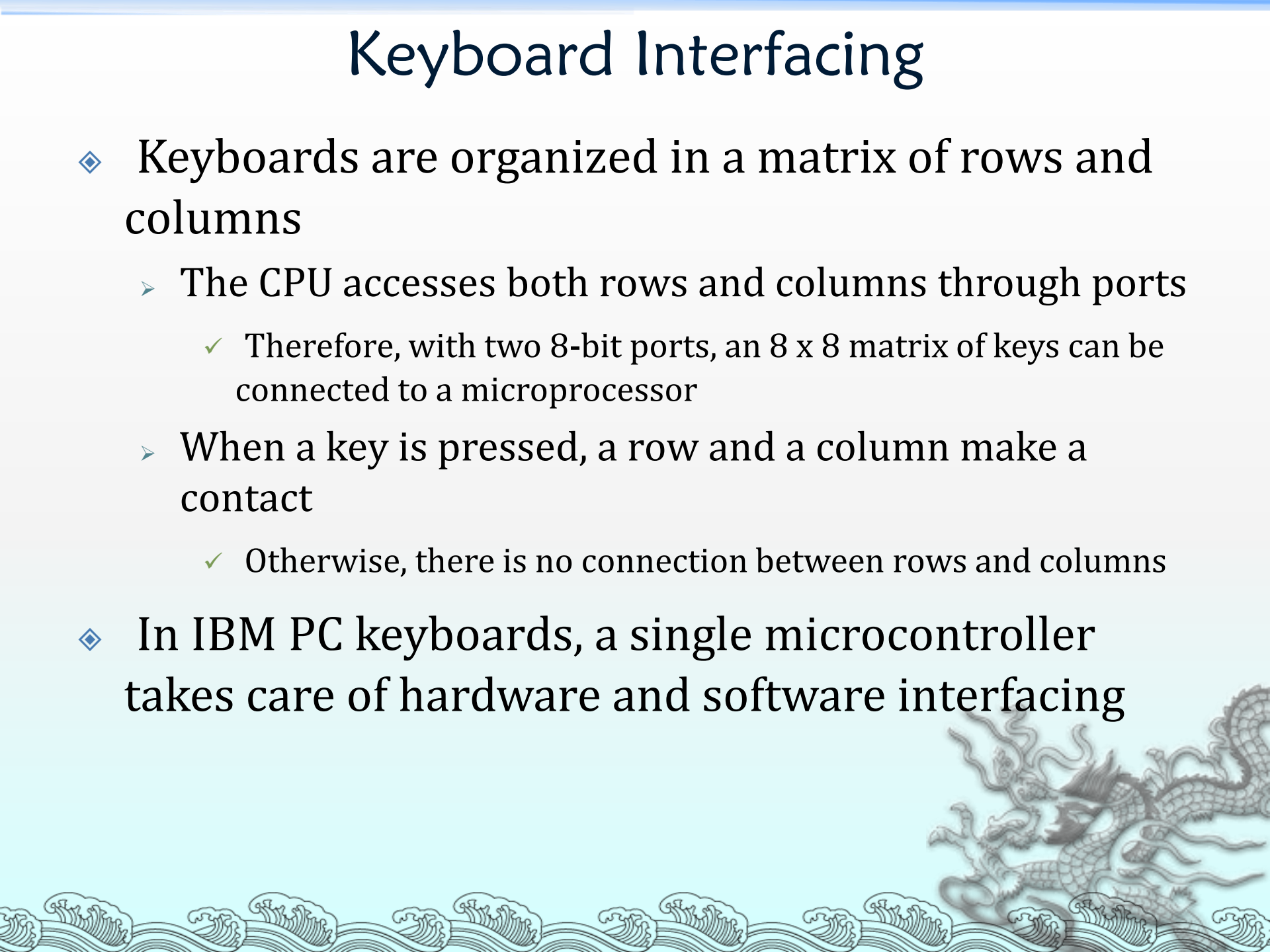
STM32点阵式LCD显示

```
  while(1)
  {
    for(j=0;j<100;j++)
    {
      for(i=0;i<32;i+=2)
      {
        change595(~word1[i+1],~word1[i],word0[i],word0[i+1]);
      }
    }
```

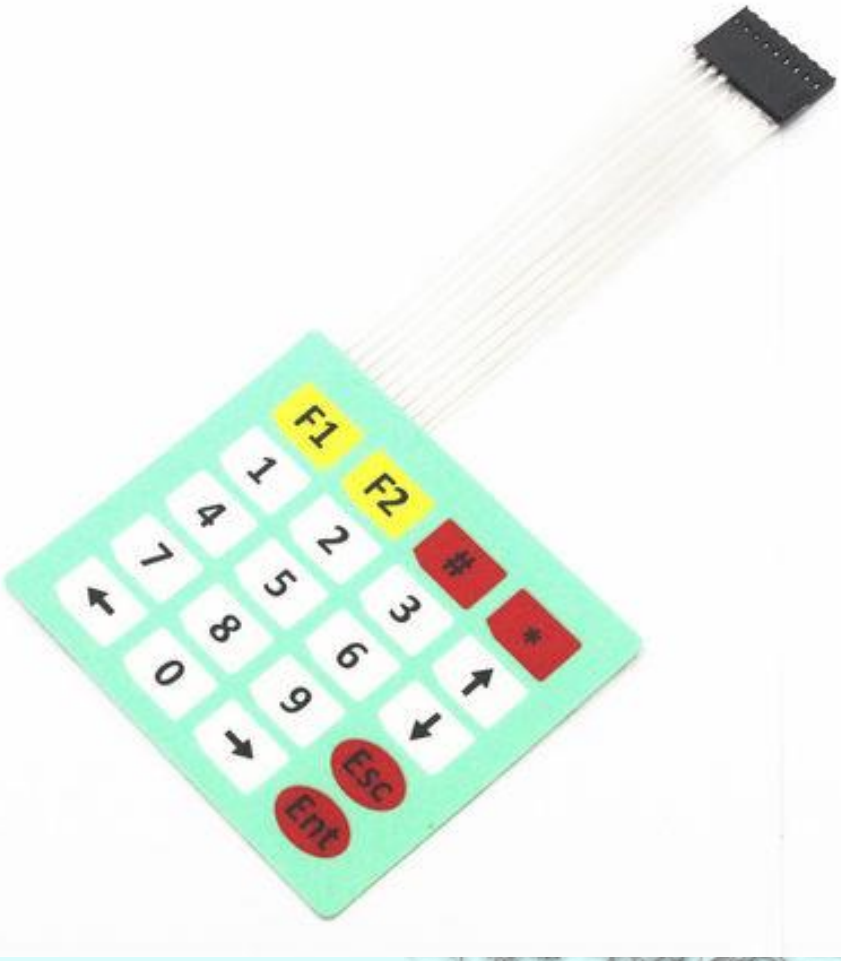
```
u8 word0[]={0x00,0x01,0x00,0x02,0x00,0x04,0x00,0x08,0x00,0x10,0x00,0x20,0x00,0x40,0x00,0x80,0x01,0x00,0x02,0x00
//  为
u8 word1[]={128,0,136,0,144,0,144,16,254,63,64,16,64,16,192,16,32,17,32,18,16,18,8,16,4,9,2,6,0,0,0,0};
//  了
u8 word2[]={0,8,252,31,0,4,0,2,128,1,128,0,128,0,128,0,128,0,128,0,128,0,128,0,160,0,64,0,0,0,0,0};
//  美
u8 word3[]={16,2,32,1,254,15,64,0,252,7,64,0,64,8,255,31,64,0,254,15,160,0,16,1,12,6,3,24,0,0,0,0};
//  好
u8 word4[]={8,0,200,31,8,8,63,4,36,2,36,2,228,63,36,2,34,2,20,2,8,2,20,2,162,2,1,1,0,0,0,0};
//  的
u8 word5[]={8,2,8,2,4,17,62,63,162,16,98,16,34,17,62,18,34,18,34,16,34,16,34,16,62,10,34,4,0,0,0,0};
//  生
u8 word6[]={128,0,136,0,136,0,136,8,252,31,132,0,130,0,129,0,248,15,128,0,128,0,128,0,128,16,255,63,0,0,0,0};
//  活
u8 word7[]={2,8,12,30,232,1,0,1,1,17,242,63,10,1,8,9,228,31,39,8,36,8,36,8,228,15,36,8,0,0,0,0};
//  努
u8 word8[]={8,0,136,31,127,9,36,5,36,2,24,5,164,56,67,16,64,0,252,15,64,8,32,8,144,4,14,3,0,0,0,0};
//  力
u8 word9[]={64,0,64,0,64,0,64,8,252,31,64,8,64,8,64,8,64,8,32,8,32,8,16,8,8,5,6,2,0,0,0,0};
//  !
u8 word10[]={0,0,0,0,24,0,24,0,24,0,24,0,24,0,24,0,0,0,0,0,0,24,0,24,0,0,0,0,0,0,0,0};
```

Keyboard Interfacing

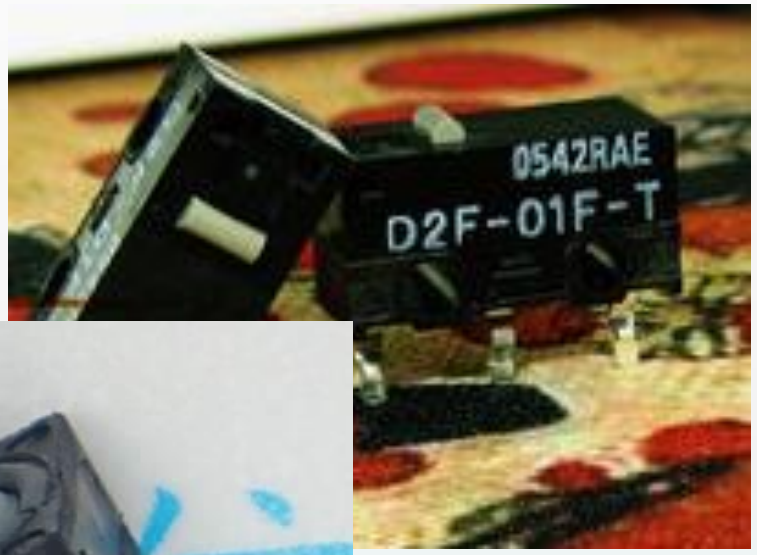
- Keyboards are organized in a matrix of rows and columns
  - The CPU accesses both rows and columns through ports
    - ✓ Therefore, with two 8-bit ports, an 8 x 8 matrix of keys can be connected to a microprocessor
  - When a key is pressed, a row and a column make a contact
    - ✓ Otherwise, there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



key



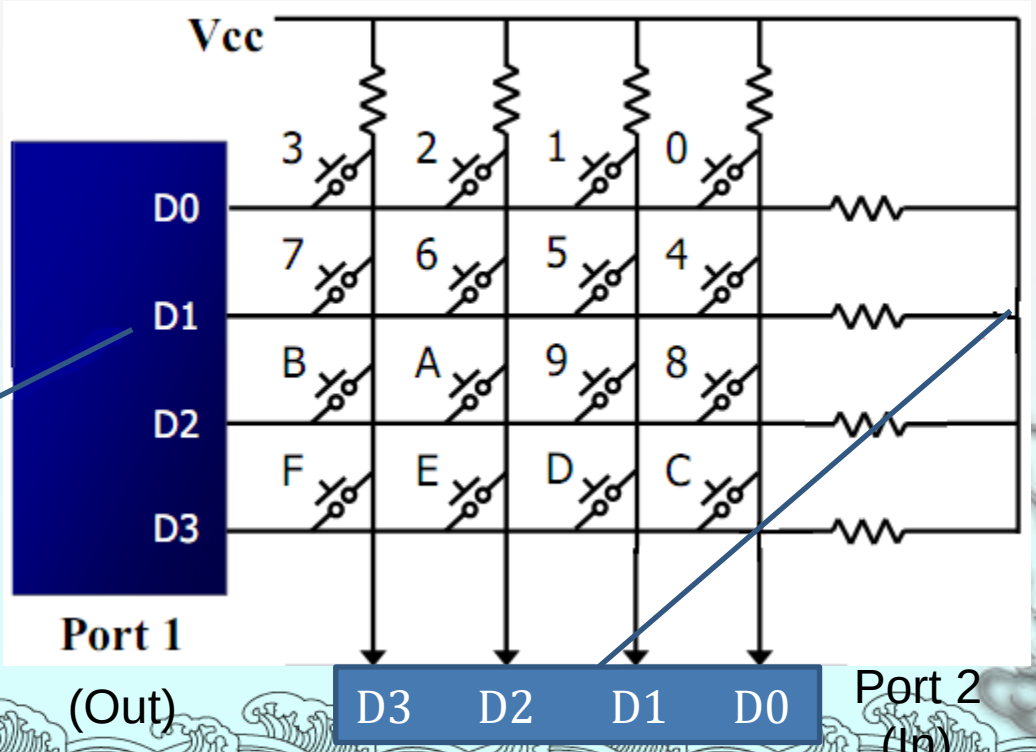
key



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



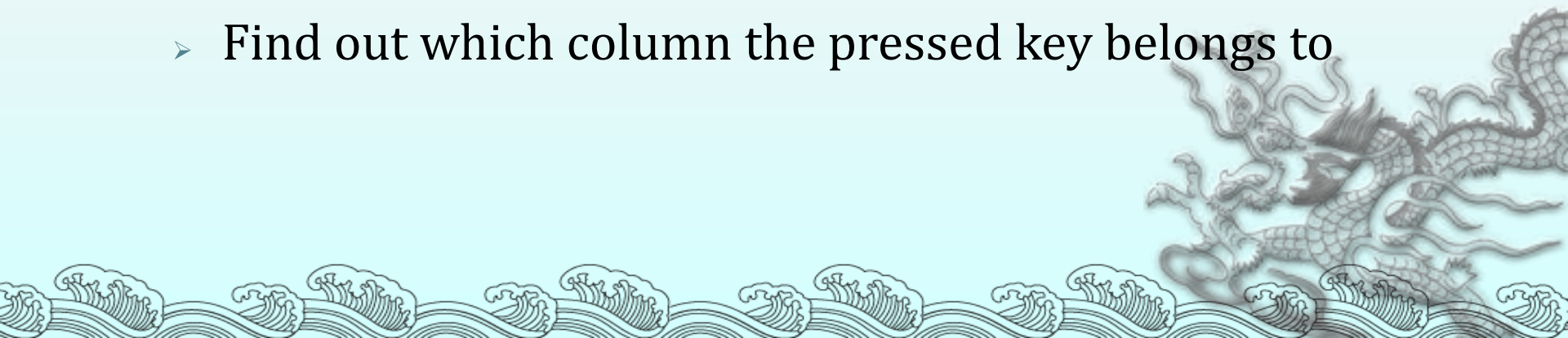
If all the rows are grounded and a key is pressed, one of the columns will have 0 since the key pressed provides the path to ground

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

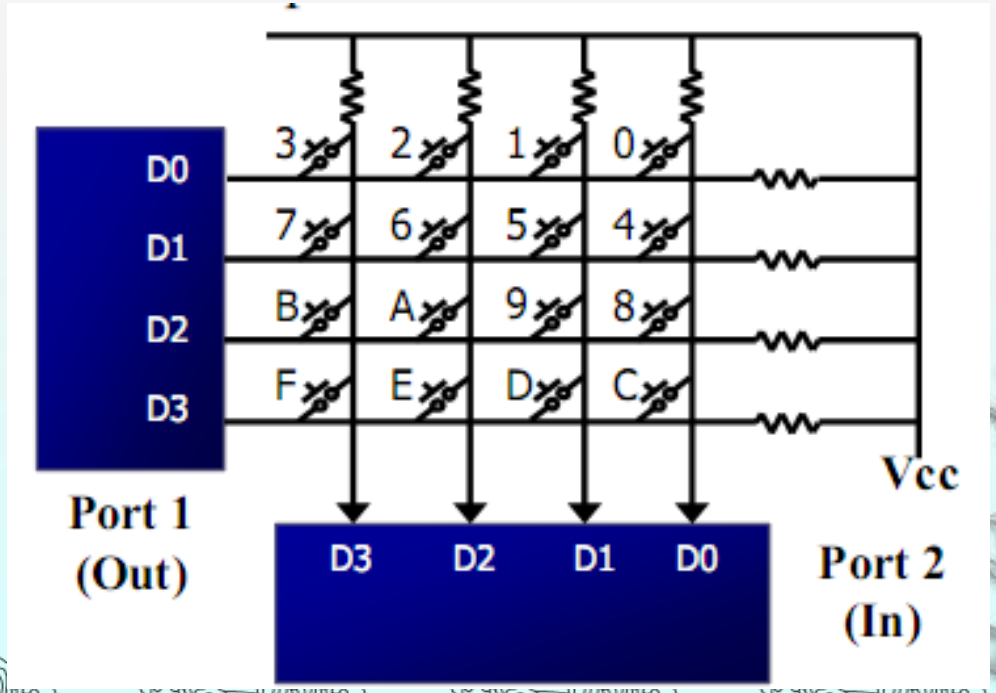
(b) D3 – D0 = 1101 for the row, D3 – D0 = 0111 for the column

Solution :

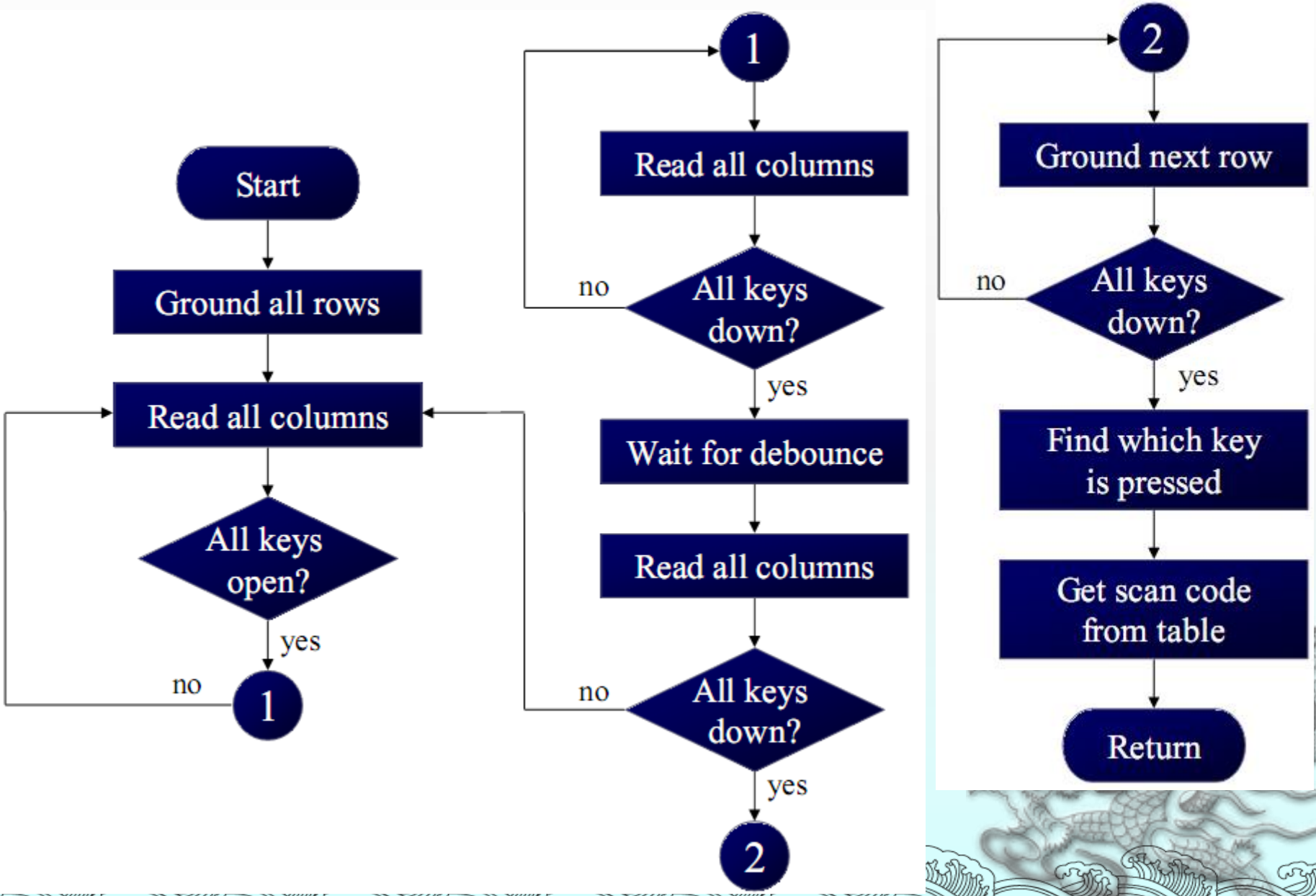
From Figure 13-5 the row and column can be used to identify the key.

(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



STM32矩阵键盘查询程序

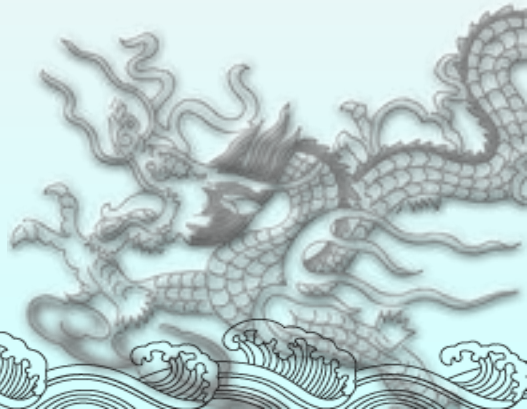
```
void keyscan()
{
  u16 value;
  u8 h1,h2,h3,h4,key;
  GPIO_Write(GPIOB,(u16)(0xfe<<8));  //判断第一行那个按键按下
  value=GPIO_ReadInputData(GPIOB);
  h1=(u8)(value>>8);
  if(h1!=0xfe)
  {
    delayms(200);    //消抖
    if(h1!=0xfe)
    {
      key=h1&0xf0;
      switch(key)
      {
        case 0xe0: GPIO_Write(GPIOA,(u16)(~smg[0]));break;
        case 0xd0: GPIO_Write(GPIOA,(u16)(~smg[1]));break;
        case 0xb0: GPIO_Write(GPIOA,(u16)(~smg[2]));break;
        case 0x70: GPIO_Write(GPIOA,(u16)(~smg[3]));break;
      }
    }
  //  while(h1!=0xfe);
  }

  GPIO_Write(GPIOB,(u16)(0xfd<<8));  //判断第2行那个按键按下
  value=GPIO_ReadInputData(GPIOB);
```

矩阵键盘翻转法

```
unsigned char  Keyboard (void)
{
    unsigned char Rank, Row;

    P0 = 0xf0;                  //置所有行线为低电平
    if(P0 != 0xf0)              //判断所有列线是否全为高电平，若全为高，则无键按下
    {
        delay();                //软件延时消抖
        if(P0 != 0xf0)
        {
            P0 = 0xf0;                          //行线输出低电平，列线作输入
            if(P0_4==0)         Rank=0;         //判断被按下的按键所处的列线
            else if(P0_5==0)    Rank =1;
            else if(P0_6==0)    Rank =2;
            else if(P0_7==0)    Rank =3;

            P0 = 0x0f;                          //列线输出低电平，行线作输入
            if(P0_0==0)         Row =0;         //判断被按下的按键所处的行线
            else if(P0_1==0)    Row =1;
            else if(P0_2==0)    Row =2;
            else if(P0_3==0)    Row =3;

            return  (Rank + Row *4)             //返回按键的键值
          }
    }
}
```

实验安排

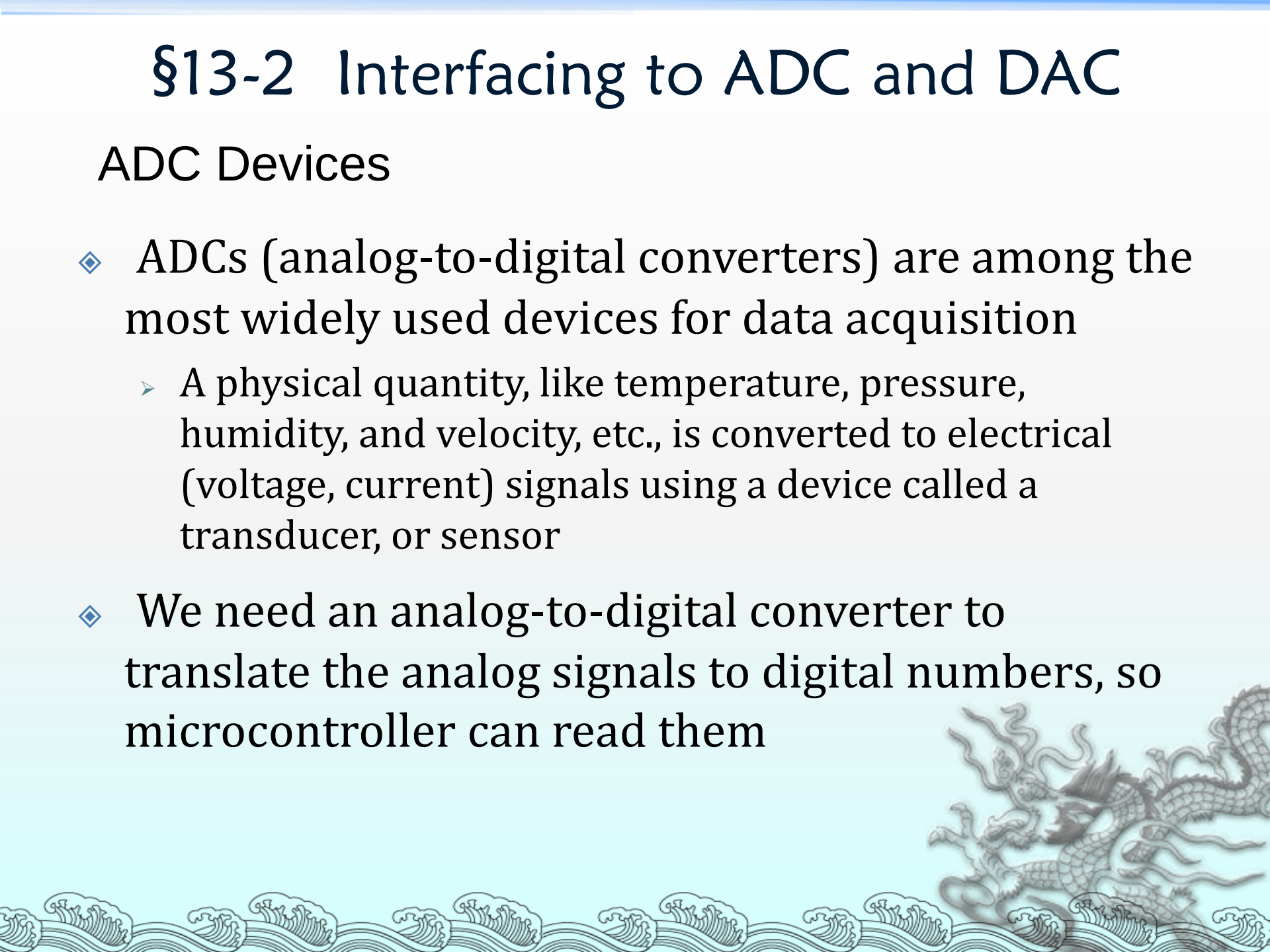
- 为烤箱设置数字按键系统，调节烤箱设定温度范围
- 设定温度范围20~250℃
- 综合使用定时器、按键和显示模块
- 完成按键检测和处理（数字加减）
- 支持长按键（连续按键，键值连续变化）
- 将按键结果显示在LCD上（点阵、1602、数码管任选1）
- 选择自己认为合适的按键数量
- 补充自己认为合理的设定



§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；

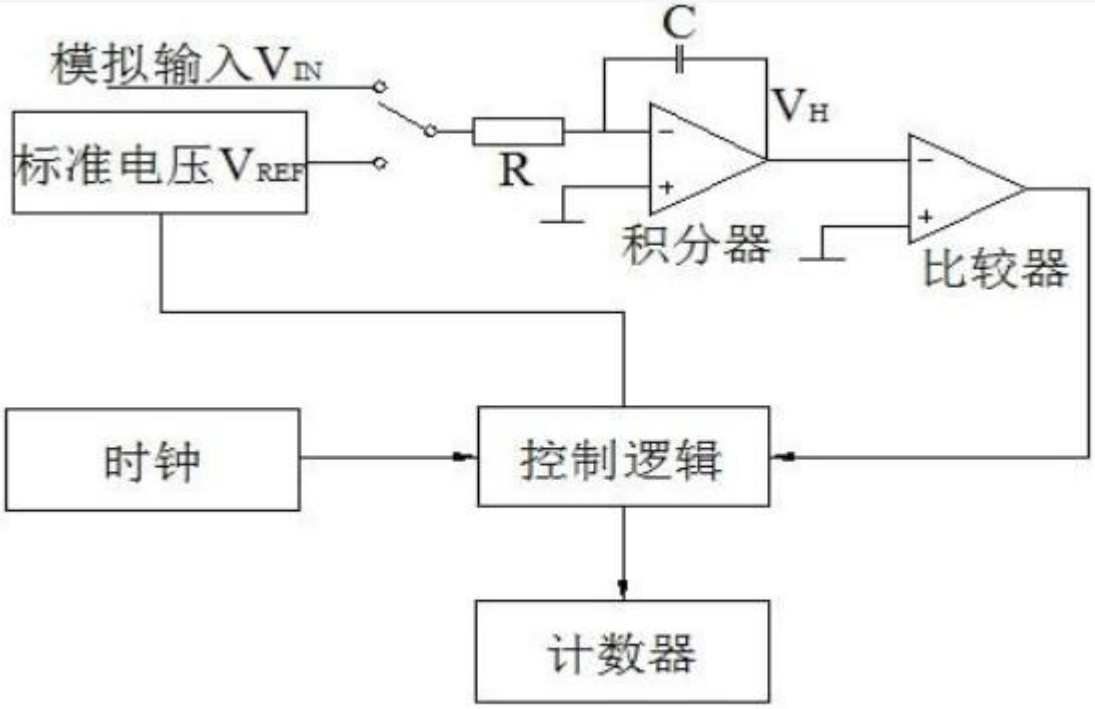
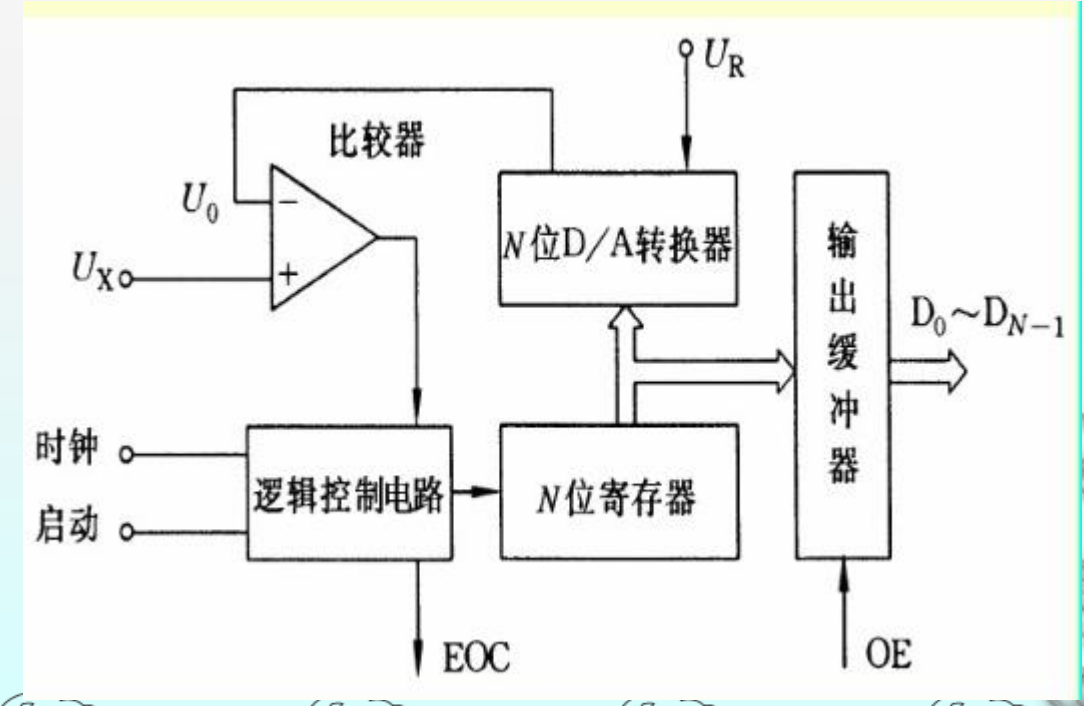


图 2.1 双积分 A/D 转换器电路图

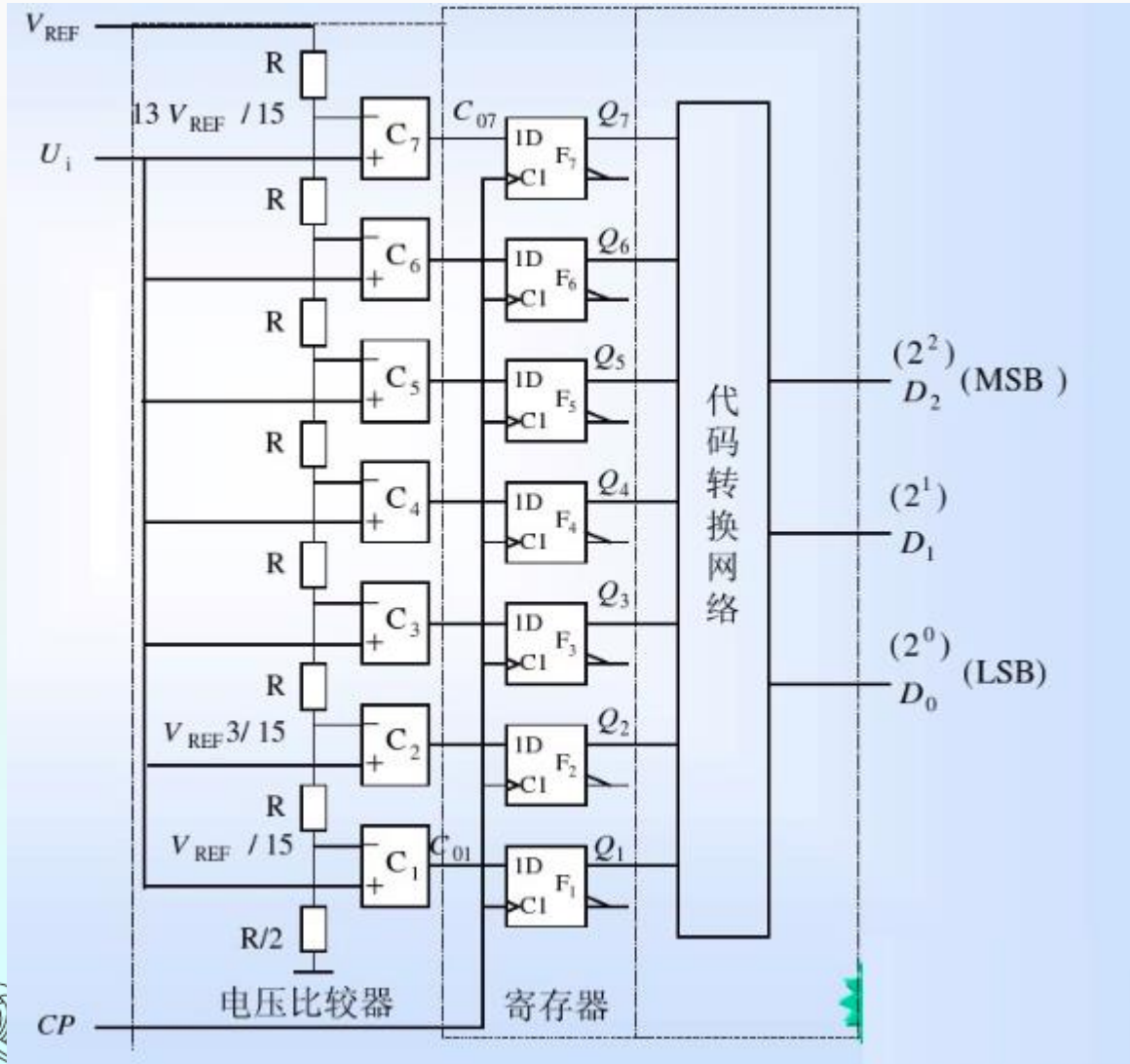
逐次比较型

- 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，低分辨率（<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（4）

- Σ-Δ(Sigma/delta)调制型
  - 由积分器、比较器、1位DA转换器和数字滤波器等组成。
  - 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
  - 间接转换方式实现模数转换的，其原理是首先将输入的模拟信号转换成频率，然后用计数器将频率转换成数字量。
  - 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。

常用A/D转换器芯片ADC0809

（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；

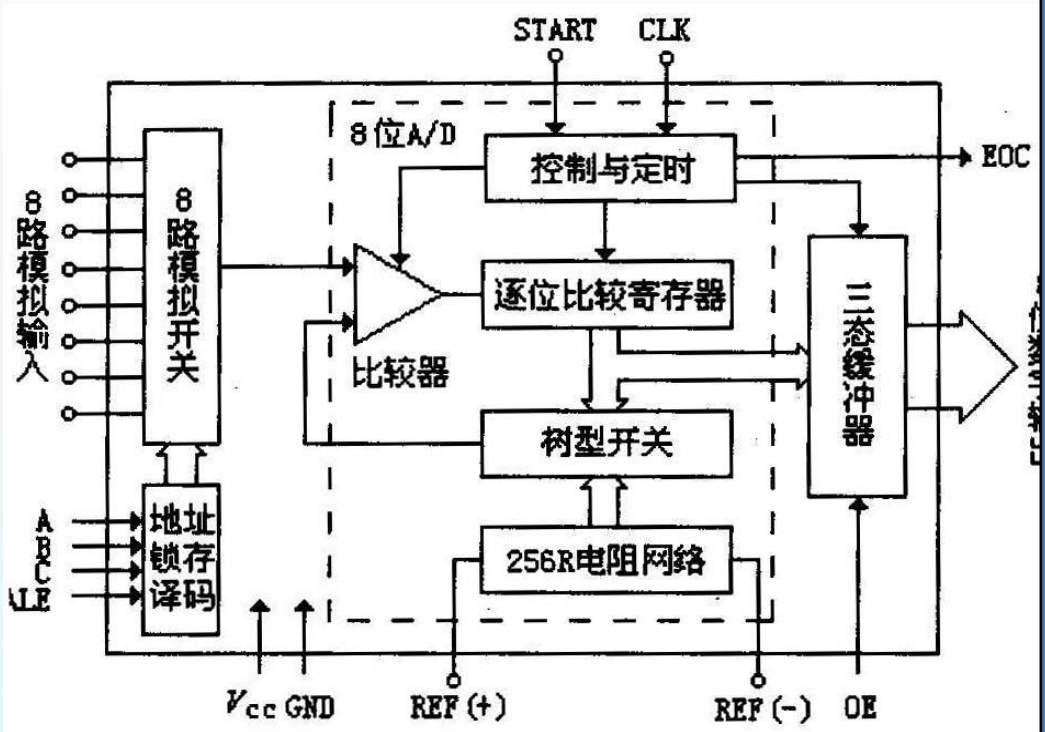
⑧ DIP28封装；

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。

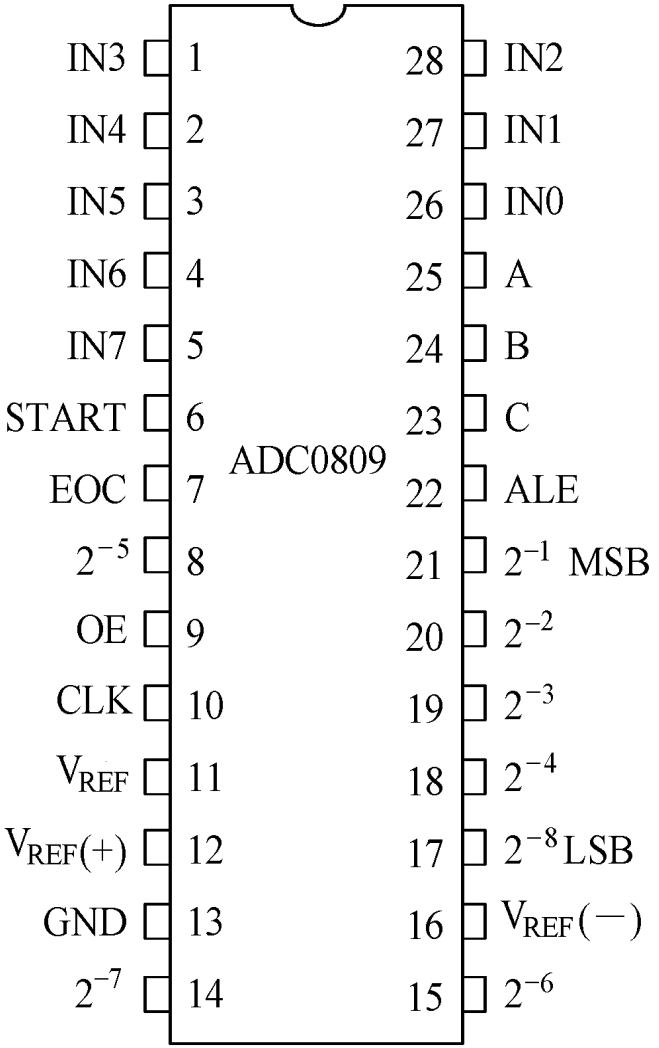
当START端输入一个正脉冲时，立即启动0809进行A/D转换。START端与ALE端连在一起，由80C51WR与0809片选端（例如P2.0）通过或非门相连。

(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

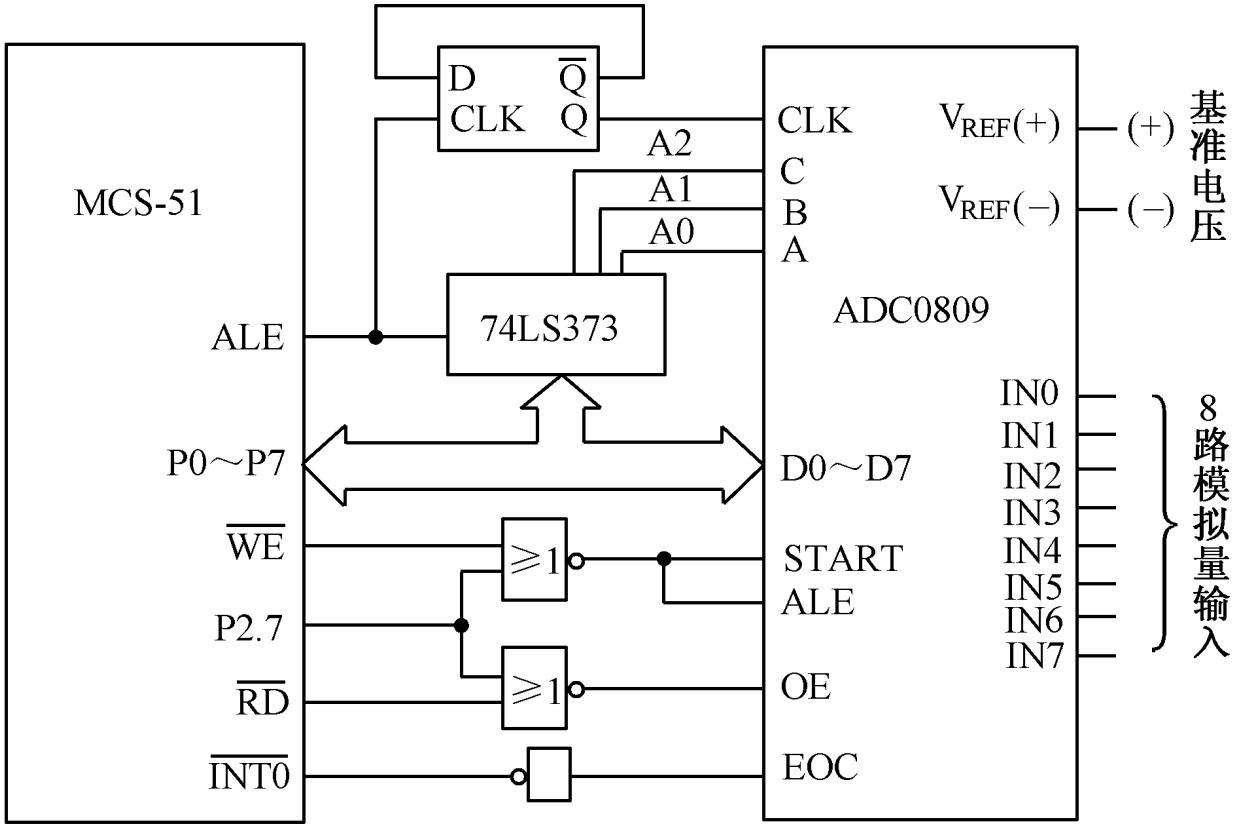
ADC0809与单片机80C51接口

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

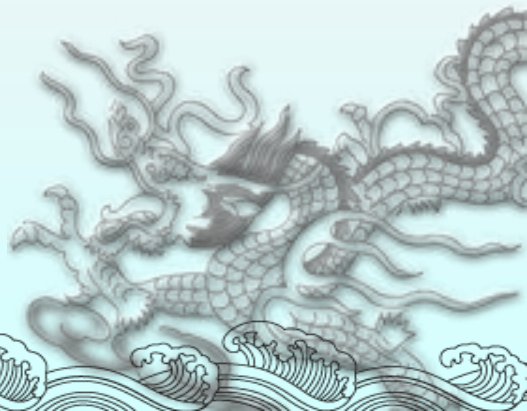
2) 查询方式

3)延时等待方式

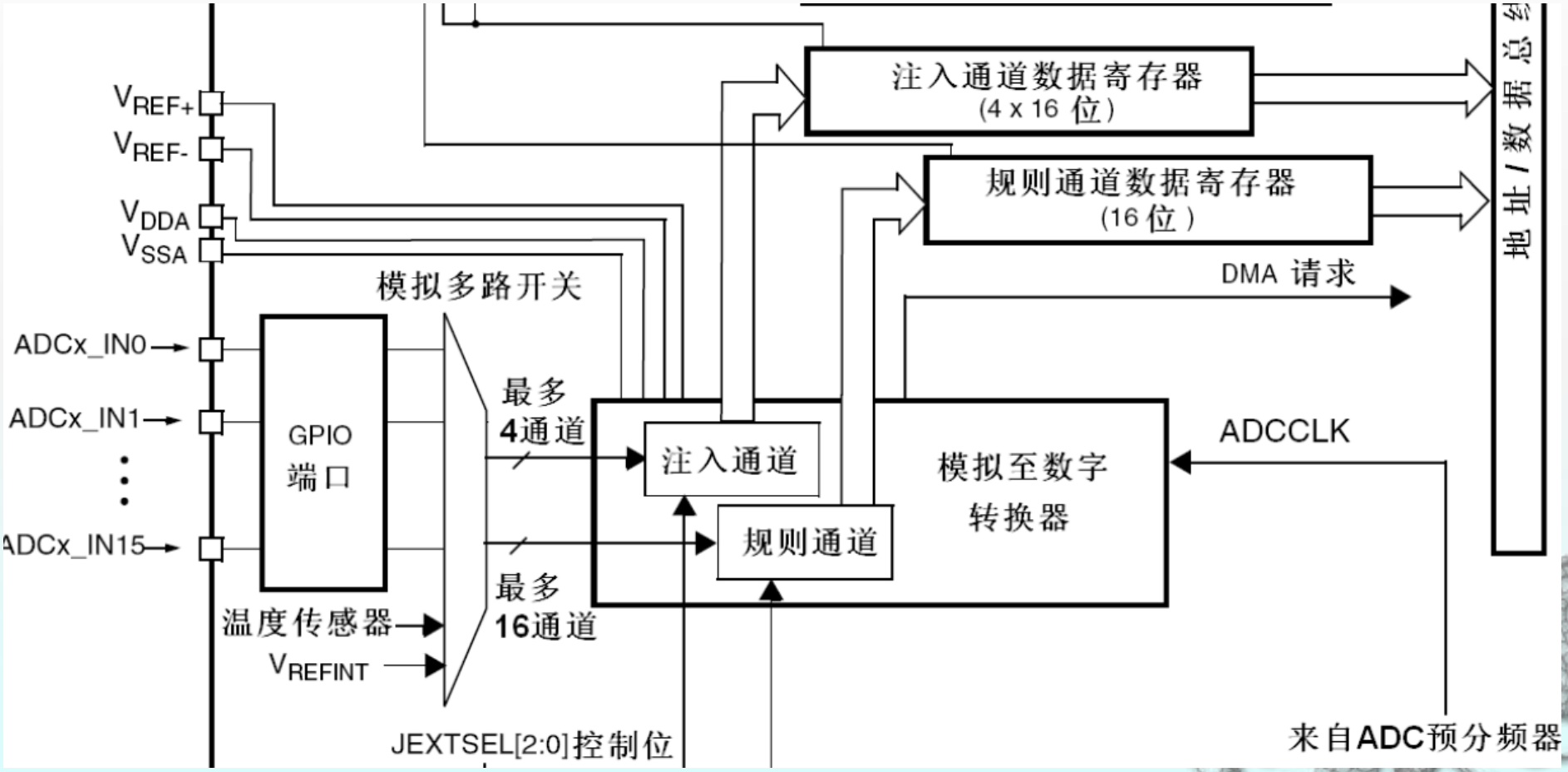


STM32 ADC

- ADC是一个连续近似ADC的转换器，它由18个多路复用通道
- 12位精度，如何计算?
- 单一和连续转化模式
- 扫描模式，采样间隔可以按通道分别编程
- 自校准，双工模式
- 转换频率 1MHz
- 供电范围 2.4-3.6V
- 输入范围 Vref- ≤ Vin ≤ Vref+



ADC模块的框图


注入通道数据寄存器
(4 x 16 位)
规则通道数据寄存器
(16 位)
地址/数据总线
DMA 请求
VREF+
VREF-
VDDA
VSSA
模拟多路开关
ADCx_IN0
ADCx_IN1
ADCx_IN15
GPIO
端口
最多
4通道
注入通道
规则通道
模拟至数字
转换器
ADCCLK
温度传感器
VREFINT
最多
16通道
JEXTSEL[2:0]控制位
来自ADC预分频器

通道选择

- 有16个多路通道。可以把转换组织成两组：规则组和注入组。
- 在任意多个通道上以任意顺序进行的一系列转换构成成组转换。
- 规则组由多达16个转换组成
- 注入组由多达4个转换组成
- 如果设置了DMA位，在每次EOC后，DMA控制器把规则组通道的转换数据传输到SRAM中

ADC program

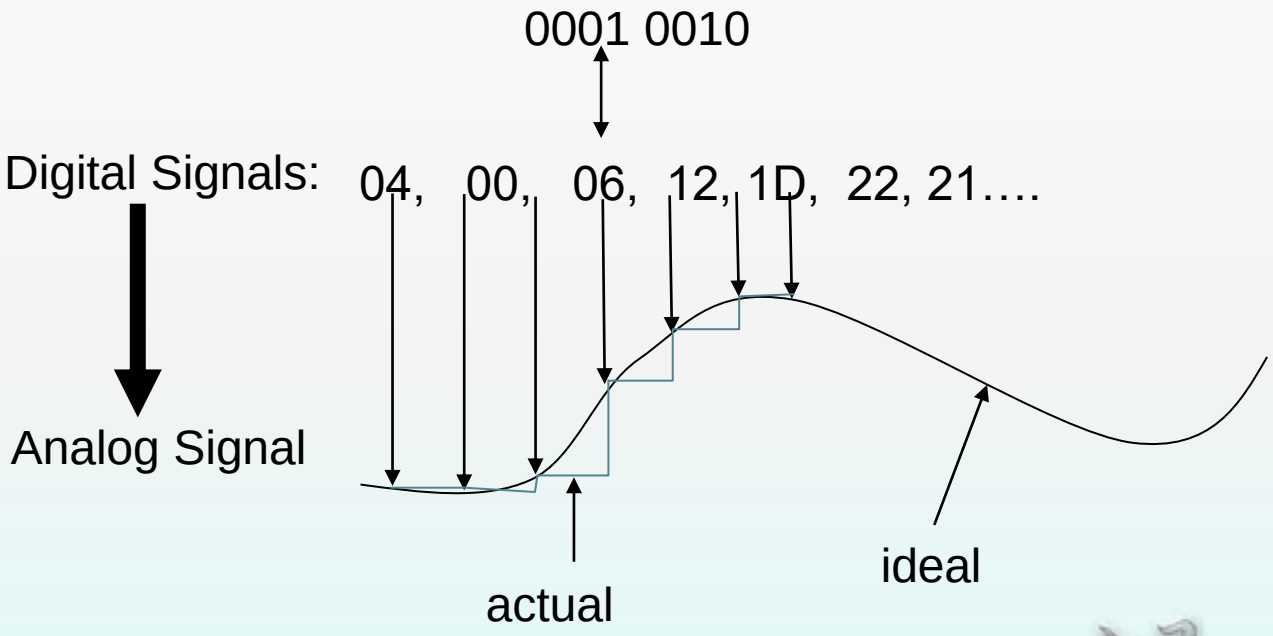
```
#include "public.h"
#include "systick.h"
#include "printf.h"
#include "adc.h"
int main()
{
  u32 ad=0;
  u8 i;
  RCCINIT_PRINTF();     //初始化printf的系统时钟
  RCCINIT_ADC();       //初始化ADC的系统时钟
  GPIOINIT_ADC();      //初始化ADC的端口配置
  GPIOINIT_PRINTF();
  USARTINIT_PRINTF();    //printf串口的初始化配置
  NVICINIT_PRINTF();      //printf中断模式的初始化配置
  ADCINIT_ADC();
  while(1)
  {
    ad=0;
    for(i=0;i<50;i++)//读取50次的AD数值取其平均数较为准确
    {
      ADC_SoftwareStartConvCmd(ADC1, ENABLE);
      while(!ADC_GetFlagStatus(ADC1,ADC_FLAG_EOC));//转换结束标志位
      ad=ad+ADC_GetConversionValue(ADC1);//返回最近一次ADCx规则组的转换结果
    }
    ad=ad/50;
    printf("ad=%f\n",ad*3.3/4096);
    delay_ms(1000);
  }
}
```

ADC 初始化

```
void ADCINIT_ADC()
{
  ADC_InitTypeDef ADC_InitStructure;
  ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
  ADC_InitStructure.ADC_ScanConvMode = DISABLE;
  ADC_InitStructure.ADC_ContinuousConvMode = DISABLE;
  ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
  ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
  ADC_InitStructure.ADC_NbrOfChannel = 1;
  ADC_Init(ADC1, &ADC_InitStructure);

  //设置指定ADC的规则组通道，设置它们的转化顺序和采样时间
  ADC_RegularChannelConfig(ADC1,ADC_Channel_8,1,ADC_SampleTime_239Cycles5);

  ADC_Cmd(ADC1,ENABLE);

  ADC_ResetCalibration(ADC1);//重置指定的ADC的校准寄存器
  while(ADC_GetResetCalibrationStatus(ADC1));//获取ADC重置校准寄存器的状态

  ADC_StartCalibration(ADC1);//开始指定ADC的校准状态
  while(ADC_GetCalibrationStatus(ADC1));//获取指定ADC的校准程序

  ADC_SoftwareStartConvCmd(ADC1, ENABLE);//使能或者失能指定的ADC的软件转换启动功能


}
```

§13-3 Interfacing to DAC

Digital-to-analog convert



DAC Devices

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

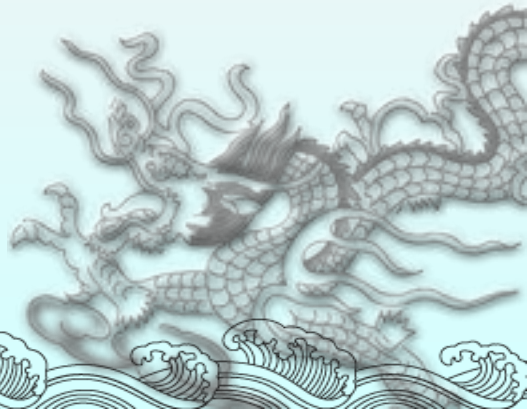
a. Format of digital numbers: binary number
   8 bits, 10 bits, 12 bits, 14 bits, 16 bits

b. Output form : Current output and Voltage output

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

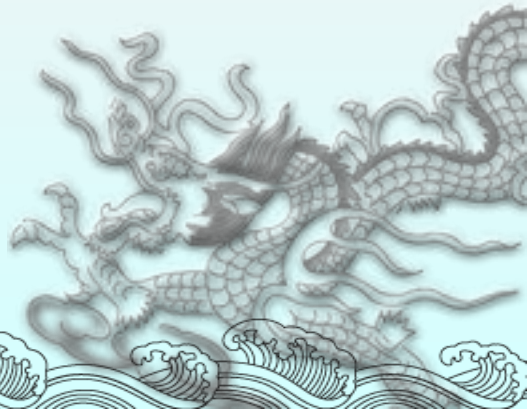
d. Output without latch 、 Output with latch 、 Buffer with two-stage

e. Input form: parallel and serial



STM32 DAC模块

- 2个DAC转换器
- 8位或者12位单调输出
- 12位模式下数据左对齐或者右对齐
- 同步更新功能
- 噪声波形生成，三角波形生成
- 双DAC通道同时或者分别转换
- 每个通道都有DMA功能
- 外部触发转换，触发非常重要
- 输入参考电压VREF+
- DAC输出 = VREF x (DOR / 4095)



DAC编程

```
int main()
{
  u8 i;
  float da;
  RCCINIT();
  GPIOINIT();
  NVICINIT();
  USARTINIT();
  DACINIT();
  while(1)
  {
    da=0;
    for(i=0;i<=10;i++)
    {
      da=i*400;
      DAC_SetChannel1Data(DAC_Align_12b_R,da);//12位 右对齐 PA4 端口输出

      printf("da=%fv\n",3.3*da/4096);
      delayms(1000);
      delayms(1000);
      delayms(1000);
      delayms(1000);
      delayms(1000);//间隔5秒输出一个电压
    }
  }
}
```

THANK YOU!!