单片机的外部结构：

1、 DIP40 双列直插；

2、 P0，P1，P2，P3 四个 8 位准双向 I/O 引脚；（作为 I/O 输入时，要先输出高电平）

3、 电源 VCC（PIN40）和地线 GND（PIN20）；

4、 高电平复位 RESET（PIN9）；（10uF 电容接 VCC 与 RESET，即可实现上电复位）

5、 内置振荡电路，外部只要接晶体至 X1（PIN18）和 X0（PIN19）；（频率为主频的 12 倍）

6、 程序配置 EA（PIN31）接高电平 VCC；（运行单片机内部 ROM 中的程序）

7、 P3 支持第二功能：RXD、TXD、INT0、INT1、T0、T1

单片机内部 I/O 部件：（所为学习单片机，实际上就是编程控制以下 I/O 部件，完成指定任务）

1、 四个 8 位通用 I/O 端口，对应引脚 P0、P1、P2 和 P3；

2、 两个 16 位定时计数器；（TMOD，TCON，TL0，TH0，TL1，TH1）

3、 一个串行通信接口；（SCON，SBUF）

4、 一个中断控制器；（IE，IP）

针对 AT89C52 单片机，头文件 AT89x52.h 给出了 SFR 特殊功能寄存器所有端口的定义。教科书的 160 页给出了针对 MCS51 系列单片机的 C 语言扩展变量类型。

单片机 C 语言编程基础

1、 十六进制表示字节 0x5a：二进制为 01011010B；0x6E 为 01101110。

2、 如果将一个 16 位二进数赋给一个 8 位的字节变量，则自动截断为低 8 位，而丢掉高 8 位。

3、 ++var 表示对变量 var 先增一；var—表示对变量后减一。

4、 x |= 0x0f;表示为 x = x | 0x0f;

5、 TMOD = （ TMOD & 0xf0 ） | 0x05;表示给变量 TMOD 的低四位赋值 0x5，而不改变 TMOD 的高四位。

6、 While（ 1 ）; 表示无限执行该语句，即死循环。语句后的分号表示空循环体，也就是 {;}

在某引脚输出高电平的编程方法：（比如 P1.3（PIN4）引脚）

```
#include //该头文档中有单片机内部资源的符号化定义，其中包含 P1.3
void main（ void ） //void 表示没有输入参数，也没有函数返值，这入单片机运行的复位入口
{
P1_3 = 1; //给 P1_3 赋值 1，引脚 P1.3 就能输出高电平 VCC
While（ 1 ）; //死循环，相当 LOOP: goto LOOP;
}
```

注意：P0 的每个引脚要输出高电平时，必须外接上拉电阻（如 4K7）至 VCC 电源。

在某引脚输出低电平的编程方法：（比如 P2.7 引脚）

```
#include //该头文档中有单片机内部资源的符号化定义，其中包含 P2.7
void main（ void ） //void 表示没有输入参数，也没有函数返值，这入单片机运行的复位入口
```

{

P2_7 = 0; //给 P2_7 赋值 0，引脚 P2.7 就能输出低电平 GND

While（ 1 ）; //死循环，相当 LOOP: goto LOOP;

}

在某引脚输出方波编程方法：（比如 P3.1 引脚）

#include //该头文档中有单片机内部资源的符号化定义，其中包含 P3.1

void main（ void ） //void 表示没有输入参数，也没有函数返值，这入单片机运行的复位入口

{

While（ 1 ） //非零表示真，如果为真则执行下面循环体的语句

{

P3_1 = 1; //给 P3_1 赋值 1，引脚 P3.1 就能输出高电平 VCC

P3_1 = 0; //给 P3_1 赋值 0，引脚 P3.1 就能输出低电平 GND

} //由于一直为真，所以不断输出高、低、高、低……，从而形成方波

}

将某引脚的输入电平取反后，从另一个引脚输出：（ 比如 P0.4 = NOT（ P1.1） ）

#include //该头文档中有单片机内部资源的符号化定义，其中包含 P0.4 和 P1.1

void main（ void ） //void 表示没有输入参数，也没有函数返值，这入单片机运行的复位入口

{

P1_1 = 1; //初始化。P1.1 作为输入，必须输出高电平

While（ 1 ） //非零表示真，如果为真则执行下面循环体的语句

```
{
if（ P1_1 == 1 ） //读取 P1.1，就是认为 P1.1 为输入，如果 P1.1 输入高电平 VCC
{ P0_4 = 0; } //给 P0_4 赋值 0，引脚 P0.4 就能输出低电平 GND
else //否则 P1.1 输入为低电平 GND
//{ P0_4 = 0; } //给 P0_4 赋值 0，引脚 P0.4 就能输出低电平 GND
{ P0_4 = 1; } //给 P0_4 赋值 1，引脚 P0.4 就能输出高电平 VCC
} //由于一直为真，所以不断根据 P1.1 的输入情况，改变 P0.4 的输出电平
}
```

将某端口 8 个引脚输入电平，低四位取反后，从另一个端口 8 个引脚输出：（ 比如 P2 = NOT（ P3 ） ）

```
#include //该头文档中有单片机内部资源的符号化定义，其中包含 P2 和 P3
void main（ void ） //void 表示没有输入参数，也没有函数返值，这入单片机运行的复位入口
{
P3 = 0xff; //初始化。P3 作为输入，必须输出高电平，同时给 P3 口的 8 个引脚输出高电平
While（ 1 ） //非零表示真，如果为真则执行下面循环体的语句
{ //取反的方法是异或 1，而不取反的方法则是异或 0
P2 = P3^0x0f //读取 P3，就是认为 P3 为输入，低四位异或者 1，即取反，然后输出
} //由于一直为真，所以不断将 P3 取反输出到 P2
}
```

注意：一个字节的 8 位 D7、D6 至 D0，分别输出到 P3.7、P3.6 至 P3.0，比如 P3=0x0f，

则P3.7、P3.6、P3.5、P3.4四个引脚都输出低电平，而P3.3、P3.2、P3.1、P3.0四个引脚都输出高电平。同样，输入一个端口P2，即是将P2.7、P2.6至P2.0，读入到一个字节的8位D7、D6至D0。

1、 接电源：VCC（PIN40）、GND（PIN20）。加接退耦电容0.1uF

2、 接晶体：X1（PIN18）、X2（PIN19）。注意标出晶体频率（选用12MHz），还有辅助电容30pF

3、 接复位：RES（PIN9）。接上电复位电路，以及手动复位电路，分析复位工作原理

4、 接配置：EA（PIN31）。说明原因。

发光二极的控控制：单片机I/O输出

将一发光二极管LED的正极（阳极）接P1.1，LED的负极（阴极）接地GND。只要P1.1输出高电平VCC，LED就正向导通（导通时LED上的压降大于1V），有电流流过LED，至发LED发亮。实际上由于P1.1高电平输出电阻为10K，起到输出限流的作用，所以流过LED的电流小于（5V-1V）/10K = 0.4mA。只要P1.1输出低电平GND，实际小于0.3V，LED就不能导通，结果LED不亮。

开关双键的输入：输入先输出高

一个按键KEY_ON接在P1.6与GND之间，另一个按键KEY_OFF接P1.7与GND之间，按KEY_ON后LED亮，按KEY_OFF后LED灭。同时按下LED半亮，LED保持后松开键的状态，即ON亮OFF灭。

#include

```
#define LED P1^1 //用符号 LED 代替 P1_1
#define KEY_ON P1^6 //用符号 KEY_ON 代替 P1_6
#define KEY_OFF P1^7 //用符号 KEY_OFF 代替 P1_7
void main（ void ） //单片机复位后的执行入口，void 表示空，无输入参数，无返回值
{
KEY_ON = 1; //作为输入，首先输出高，接下 KEY_ON，P1.6 则接地为 0，否则输入为 1
KEY_OFF = 1; //作为输入，首先输出高，接下 KEY_OFF，P1.7 则接地为 0，否则输入为 1
While（ 1 ） //永远为真，所以永远循环执行如下括号内所有语句
{
if（ KEY_ON==0 ） LED=1; //是 KEY_ON 接下，所示 P1.1 输出高，LED 亮
if（ KEY_OFF==0 ） LED=0; //是 KEY_OFF 接下，所示 P1.1 输出低，LED 灭
} //松开键后，都不给 LED 赋值，所以 LED 保持最后按键状态。
//同时按下时，LED 不断亮灭，各占一半时间，交替频率很快，由于人眼惯性，看上去为半亮态
}
```

数码管的接法和驱动原理及单片机编程

一支七段数码管实际由 8 个发光二极管构成，其中 7 个组形构成数字 8 的七段笔画，所以称为七段数码管，而余下的 1 个发光二极管作为小数点。作为习惯，分别给 8 个发光二极管标上记号：a,b,c,d,e,f,g,h。对应 8 的顶上一画，按顺时针方向排，中间一画为 g，小数点为 h。

我们通常又将各二极与一个字节的 8 位对应，a（D0），b（D1），c（D2），d（D3），e（D4），f（D5），g（D6），h（D7），相应 8 个发光二极管正好与单片机一个端口 Pn 的 8 个引脚连接，这样单片机就可以通过引脚输出高低电平控制 8 个发光二极的亮与灭，从而显示各种数字和符号；对应字节，引脚接法为：a（Pn.0），b（Pn.1），c（Pn.2），d（Pn.3），e（Pn.4），f（Pn.5），g（Pn.6），h（Pn.7）。

如果将 8 个发光二极管的负极（阴极）内接在一起，作为数码管的一个引脚，这种数码管则被称为共阴数码管，共同的引脚则称为共阴极，8 个正极则为段极。否则，如果是将正极（阳极）内接在一起引出的，则称为共阳数码管，共同的引脚则称为共阳极，8 个负极则为段极。

以单支共阴数码管为例，可将段极接到某端口 Pn，共阴极接 GND，则可编写出对应十六进制码的七段码表字节数据如右图：

16 键码显示的单片机程序

我们在 P1 端口接一支共阴数码管 SLED，在 P2、P3 端口接 16 个按键，分别编号为 KEY_0、KEY_1 到 KEY_F，操作时只能按一个键，按键后 SLED 显示对应键编号。

```
#include
#define SLED P1
#define KEY_0 P2^0
#define KEY_1 P2^1
#define KEY_2 P2^2
```

```
#define KEY_3 P2^3

#define KEY_4 P2^4

#define KEY_5 P2^5

#define KEY_6 P2^6

#define KEY_7 P2^7

#define KEY_8 P3^0

#define KEY_9 P3^1

#define KEY_A P3^2

#define KEY_B P3^3

#define KEY_C P3^4

#define KEY_D P3^5

#define KEY_E P3^6

#define KEY_F P3^7

Code unsigned char Seg7Code[16]= //用十六进数作为数组下标，可直接取得对应的七段
编码字节

// 0 1 2 3 4 5 6 7 8 9 A b C d E F

{0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f, 0x77, 0x7c, 0x39, 0x5e,
0x79, 0x71};

void main ( void )

{

unsigned char i=0; //作为数组下标

P2 = 0xff; //P2 作为输入，初始化输出高
```

```
P3 = 0xff; //P3 作为输入，初始化输出高

While ( 1 )

{

if ( KEY_0 == 0 )  i=0; if ( KEY_1 == 0 )  i=1;

if ( KEY_2 == 0 )  i=2; if ( KEY_3 == 0 )  i=3;

if ( KEY_4 == 0 )  i=4; if ( KEY_5 == 0 )  i=5;

if ( KEY_6 == 0 )  i=6; if ( KEY_7 == 0 )  i=7;

if ( KEY_8 == 0 )  i=8; if ( KEY_9 == 0 )  i=9;

if ( KEY_A == 0 )  i=0xA; if ( KEY_B == 0 )  i=0xB;

if ( KEY_C == 0 )  i=0xC; if ( KEY_D == 0 )  i=0xD;

if ( KEY_E == 0 )  i=0xE; if ( KEY_F == 0 )  i=0xF;

SLED = Seg7Code[ i ]; //开始时显示 0，根据 i 取应七段编码

}

}
```