# 单片机资源扩展方式说明（IO&MM）

***---CT107D 单片机综合训练平台***

CT107D 单片机综合训练平台具有 IO 扩展模式和存储器映射（MM）扩展模式，可以通过调节板上跳线 J13 进行配置。其中 IO 扩展模式较为容易理解，存储器映射扩展模式可以直接通过 XBYTE 关键字来操作部分资源，能够大大简化外设资源程序设计，下面将举例说明两种扩展方式。

## 1. IO 口扩展方式

平台将单片机 P25、P26、P27 接入到 74HC138 译码器的三线输入端口，由此我们可以通过 P25、P26 和 P27 引脚控制 38 译码器的输出端口 Y[0...8]；通过板上的扩展模式配置跳线 J13，将译码器输出端口 Y[0…8]与 GND 经过或非门输出，作为 74HC573 的片选信号，即实现了 IO 扩展方式：

```
#include "reg52.h"

sbit LCD_E = P1^7;
//主函数
void main(void)
{
    LCD_E = 0; //将 LCD 模块禁能，防止 LCD 模块对总线状态的影响
    while(1)
    {
      /* 注释 1：通过此句代码将 P2.7 = 1，P2.6 = 0，P2.5 = 0，不改变 P2 口其它引脚状态。由 138 译码器的真值表可以知道此时译码器输出端 Y4 处于低电平状态；因为为 IO 编程方式，所以 Y4 和 GND 一起经过或非门后，输出高电平，并连接到 573 锁存器的使能端口，将 573 锁存器“打通“，此时锁存器输出端与单片机 P0 口状态一致。*/
      P2 = ((P2&0x1f)|0x80);
      /* 注释 2：因为 573 锁存器已经“打通“，现在通过 P0 口输出 0xff，573 锁存器的输出端也将输出 0xff，从而将所有 LED 熄灭。*/
      P0 = 0xff;
      /* 注释 3：通过此句代码将 P2.7 = 0，P2.6 = 0，P2.5 = 0，不改变 P2 口其它引脚状态。由 138 译码器的真值表可以知道此时译码器输出端 Y4 处于高电平状态；因为为 IO 编程方式，所以 Y4 和 GND 一起经过或非门后，输出低电平，并连接到 573 锁存器的使能端口，将 573 锁存器“锁存“，此时锁存器将输入锁存，输出端口数据不再受到 P0 口影响。*/
      P2 &= 0x1f;
```

```
        /* 一小段延时函数 */
        delay();
        /*与注释 1 相同*/
        P2 = ((P2&0x1f)|0x80);
        /* 注释 2：因为 573 锁存器已经“打通“，现在通过 P0 口输出 0x00，573 锁存器的输
        出端也将输出 0x00，从而将所有 LED 点亮。*/
        P0 = 0x00;
        /*与注释 3 相同*/
        P2 &= 0x1f;
        delay();
    }
}
```

与上例相同，如果想使用执行机构模块（蜂鸣器、继电器、电机等外设），需要做的无非是“打通”与执行结构连接的锁存器，进行操作，操作结束后，“锁存”，程序片段如下：

```
#include "reg52.h"

sbit LCD_E = P1^7;
sbit RELAY = P0^4;
//主函数
void main(void)
{
    LCD_E = 0; //将 LCD 模块禁能，防止 LCD 模块对总线状态的影响
    while(1)
    {
        /* 注释 1：通过此句代码将 P2.7 = 1，P2.6 = 0，P2.5 = 1，不改变 P2 口其它引脚
        状态。由 138 译码器的真值表可以知道此时译码器输出端 Y5 处于低电平状态；因为
        为 IO 编程方式，所以 Y5 和 GND 一起经过或非门后，输出高电平，并连接到 573
        锁存器的使能端口，将 573 锁存器“打通“，此时锁存器输出端与单片机 P0 口状态一
        致。*/
        P2 = ((P2&0x1f)|0xA0);
        /* 注释 2：因为 573 锁存器已经“打通“，现在通过 P0.4 输出高电平，相应的 573 锁
        存器的输出端也将输出高电平，控制继电器 */
        RELAY = 1;
        /* 注释 3：通过此句代码将 P2.7 = 0，P2.6 = 0，P2.5 = 0，不改变 P2 口其它引脚
        状态。由 138 译码器的真值表可以知道此时译码器输出端 Y5 处于高电平状态；因为
        为 IO 编程方式，所以 Y5 和 GND 一起经过或非门后，输出低电平，并连接到 573
        锁存器的使能端口，将 573 锁存器“锁存“，此时锁存器将输入锁存，输出端口数据不
```

```
        再受到 P0 口影响。*/
        P2 &= 0x1f;
        /* 一小段延时函数 */
        delay();
        /*与注释 1 相同*/
        P2 = ((P2&0x1f)|0x80);
        /* 注释 2：因为 573 锁存器已经"打通"，现在通过 P0.4 口输出低电平，相应的 573
        锁存器的输出端也将输出低电平，控制继电器 */
        RELAY = 0;
        /*与注释 3 相同*/
        P2 &= 0x1f;
        delay();
    }
}
```

**2. 存储器映射扩展方式**

51 单片机可以外扩 64K 字节的 RAM 和 ROM 空间，传统的 8051 单片机具有 16 位地址总线和 8 位数据总线，其中 P0 口作为数据和地址低字节的复用端口，P2 口作为高 8 位地址线。CT107D 单片机综合训练平台的存储器映射扩展方式（MM），***是一种可以像操作外部 RAM 存储器一样，操作 LED 指示灯、执行结构（蜂鸣器、继电器等..）、数码管、82C55 等外设资源的扩展方式，当然实现这样的操作，是与 CT107D 单片机综合训练平台的硬件设计具有关联性的***。举例说明，在上面 IO 扩展方式里已经介绍过，如果我们希望通过程序点亮或者熄灭 LED 指示灯需要进行如下操作：

2.1 IO 扩展方式代码片段:

```
P2 = ((P2&0x1f)|0xA0);
P0 = 0x00;   // LED = 0xff;
P2 &= 0x1f;
```

如果，我们将扩展方式配置跳线配置为存储器映射扩展方式(MM)，我们可以通过以下简单代码来实现其操作：

```
XBYTE[0x8000] = 0x00; // XBYTE[0x8000] = 0xff;
```

到现在存在的疑问是，LED 指示灯模块的地址：0x8000 是如何确定的呢？由 CT107D 单片机综合训练平台的硬件电路图我们可以知道，当 P2.7 = 1; P2.6 = 0; P2.5 = 0;(其它地址线不需要关心)，即可将与 LED 指示灯模块连接的 74HC573 "打通"，此时可以通过 P0 口控制 LED 指示灯的状态，因此，LED 指示灯模块的地址为 0x8000；由此类推，我们可以知道执行机构模块的操作地址为 0xA000，数码管段码端的操作地址为 0xE000，数码管位选端口的操作地址为 0xC000 等等。

下面来看一个具体实例：

```
#include "reg52.h"
#include "absacc.h"
sbit LCD_E = P1^7;

//主函数
void main(void)
{
   LCD_E = 0; //将 LCD 模块禁能，防止 LCD 模块对总线状态的影响
   while(1)
   {
       XBYTE[0x8000] = 0x00;  //LED 指示灯全部点亮
       /* 一小段延时函数 */
       delay();

       XBYTE[0x8000] = 0xff;  //LED 指示灯全部熄灭
       delay();
   }
}
```

MM 编程方式，能够简化程序设计，这一点在数码管动态扫描显示的代码部分体现的尤为明显，由于这种扩展方式占用单片机 P3.6 引脚，在使用 4X4 矩阵键盘时，不建议使用这种扩展方式。

数码管显示程序片段

IO 方式

```
void display(void)
{
      //数码管消隐
    P2 = (P2&0x1f|0xe0);
    P0 = 0xff;
    P2 &= 0x1f;

      //位选控制
    P2 = (P2&0x1f|0xc0);
    P0 = (1<<dspcom);
    P2 &= 0x1f;

      //段码输入
    P2 = (P2&0x1f|0xe0);
    P0 = tab[dspbuf[dspcom]];
    P2 &= 0x1f;

    if(++dspcom == 8){
        dspcom = 0;
    }

}
```

MM 方式

```
void display(void)
{
    //数码管消隐
    XBYTE[0xE000] = 0xff

    //位选控制
    XBYTE[0xC000] = (1<<dspcom);

    //段码输入
    XBYTE[0xE000] = tab[dspbuf[dspcom]];

    if(++dspcom == 8){
        dspcom = 0;
    }

}
```