LED灯控制课堂应用总结

前提与约定

1. 硬件连接与编号

- **端口**：所有示例根据题目要求，分别使用`P0`口或`P2`口
- **LED编号**：8个LED灯从右至左编号为 **0号灯** 至 **7号灯**
- **对应关系**：二进制数的最低位(Bit 0)对应0号灯，最高位(Bit 7)对应7号灯
- **点亮方式**：所有代码示例均采用 **高电平(1)点亮LED** 的方式（对应共阴极电路或带驱动芯片的电路）。如果您的硬件是共阳极（低电平点亮），只需将`P_x = value;`修改为`P_x = ~value;`即可

2. 自定义延时函数

```
/*
 * 自定义双参数延时函数
 * 通过改变参数x和y的值来控制延时长度
 */
void delay(unsigned int x, unsigned int y)
{
    unsigned int i, j;
    for (i = 0; i < x; i++)
    {
        for (j = 0; j < y; j++);
    }
}
```

计时规则：固定`x=1000`，通过改变`y`来控制时间：

- `delay(1000, 80);`≈ 1 秒
- `delay(1000, 40);`≈ 0.5 秒
- `delay(1000, 20);`≈ 0.25 秒
- `delay(1000, 10);`≈ 0.125 秒

3. 头文件

- 基础程序需要 `#include <reg52.h>`
- 使用循环移位函数 `_crol_`和 `_cror_`的程序需要额外包含 `#include <intrins.h>`

课堂应用2

1. 点亮偶数、奇数、全部LED

知识点和思路讲解：

- **偶数灯(0,2,4,6)**：对应P0口的P0.0, P0.2, P0.4, P0.6。二进制表示为`01010101B`，转换为十六进制为`0x55`
- **奇数灯(1,3,5,7)**：对应P0口的P0.1, P0.3, P0.5, P0.7。二进制表示为`10101010B`，转换为十六进制为`0xAA`
- **全部灯(0-7)**：对应P0口所有引脚。二进制表示为`11111111B`，转换为十六进制为`0xFF`
- **实现**：直接给P0口赋相应的十六进制值即可

```
#include <reg52.h>

void main()
{
    // P0 = 0x55; // 点亮偶数灯
    // P0 = 0xAA; // 点亮奇数灯
    // P0 = 0xFF; // 点亮全部灯

    P0 = 0x55; // 此处演示点亮偶数灯
    while(1);  // 保持状态
}
```

2. 第5个LED以1Hz频率闪烁

知识点和思路讲解：

- **5号灯**：对应P0.5引脚。8位二进制中只有第5位为1，即`00100000B`，十六进制为`0x20`
- **1Hz频率**：周期为1秒。一个周期内包含"亮"和"灭"两个状态，因此每个状态持续0.5秒
- **延时**：根据约定，0.5秒的延时使用`delay(1000, 40);`
- **实现**：在`while(1)`循环中，先将P0置为`0x20`（点亮5号灯），延时0.5秒；再将P0置为`0x00`（熄灭所有灯），延时0.5秒

```
#include <reg52.h>

// 延时函数声明
void delay(unsigned int x, unsigned int y);

void main()
{
    while(1)
    {
        P0 = 0x20; // 点亮5号灯
        delay(1000, 40); // 延时0.5秒
        P0 = 0x00; // 熄灭所有灯
        delay(1000, 40); // 延时0.5秒
    }
}
```

```
// 延时函数定义
void delay(unsigned int x, unsigned int y)
{
    unsigned int i, j;
    for (i = 0; i < x; i++)
        for (j = 0; j < y; j++);
}
```

3. 高4位与低4位交替闪烁

知识点和思路讲解:

- **高4位(4,5,6,7号灯)**：对应P0.4至P0.7。二进制为`11110000B`，十六进制为`0xF0`
- **低4位(0,1,2,3号灯)**：对应P0.0至P0.3。二进制为`00001111B`，十六进制为`0x0F`
- **时间**：题目要求亮1秒，灭1秒。这里理解为高4位亮1秒，然后低4位亮1秒
- **延时**：1秒对应延时`delay(1000, 80);`
- **实现**：在`while(1)`循环中，先将P0置为`0xF0`（点亮高4位），延时1秒；再将P0置为`0x0F`（点亮低4位），延时1秒

```
#include <reg52.h>

// 延时函数声明
void delay(unsigned int x, unsigned int y);

void main()
{
    while(1)
    {
        P0 = 0xF0; // 点亮高4位
        delay(1000, 80); // 延时1秒
        P0 = 0x0F; // 点亮低4位
        delay(1000, 80); // 延时1秒
    }
}

// 延时函数定义
void delay(unsigned int x, unsigned int y)
{
    unsigned int i, j;
    for (i = 0; i < x; i++)
        for (j = 0; j < y; j++);
}
```

课堂应用3

4. 偶数位与奇数位以2Hz频率闪烁

知识点和思路讲解:

- **偶数灯(0,2,4,6)**：`0x55`
- **奇数灯(1,3,5,7)**：`0xAA`

- **2Hz频率**：周期为0.5秒。一个周期内偶数灯亮、奇数灯亮交替，因此每个状态持续 0.5 / 2 = 0.25秒
- **延时**：0.25秒对应延时`delay(1000, 20);`
- **实现**：在`while(1)`循环中，P0置为`0x55`，延时0.25秒；P0置为`0xAA`，延时0.25秒

```
#include <reg52.h>

// 延时函数声明
void delay(unsigned int x, unsigned int y);

void main()
{
    while(1)
    {
        P0 = 0x55; // 点亮偶数灯
        delay(1000, 20); // 延时0.25秒
        P0 = 0xAA; // 点亮奇数灯
        delay(1000, 20); // 延时0.25秒
    }
}

// 延时函数定义
void delay(unsigned int x, unsigned int y)
{
    unsigned int i, j;
    for (i = 0; i < x; i++)
        for (j = 0; j < y; j++);
}
```

5. P2口高四位与低四位交替亮灭

知识点和思路讲解：

- **逻辑**：同"课堂应用2-3"，高4位(`0xF0`)和低4位(`0x0F`)交替
- **端口**：操作端口由`P0`改为`P2`
- **时间**：题目未指定时间，根据笔记，此处设定为0.5秒（`delay(1000, 40);`）

```
#include <reg52.h>

// 延时函数声明
void delay(unsigned int x, unsigned int y);

void main()
{
    while(1)
    {
        P2 = 0xF0; // P2高四位亮
        delay(1000, 40); // 延时0.5秒
        P2 = 0x0F; // P2低四位亮
        delay(1000, 40); // 延时0.5秒
    }
}

// 延时函数定义
```

```
void delay(unsigned int x, unsigned int y)
{
    unsigned int i, j;
    for (i = 0; i < x; i++)
        for (j = 0; j < y; j++);
}
```

6. P2口偶数位与奇数位交替点亮，各0.25秒

知识点和思路讲解：

- **逻辑**：同"课堂应用3-4"，偶数位(`0x55`)和奇数位(`0xAA`)交替
- **端口**：操作端口由`P0`改为`P2`
- **时间**：题目明确指定为0.25秒，对应延时`delay(1000, 20);`

```
#include <reg52.h>

// 延时函数声明
void delay(unsigned int x, unsigned int y);

void main()
{
    while(1)
    {
        P2 = 0x55; // P2偶数位亮
        delay(1000, 20); // 延时0.25秒
        P2 = 0xAA; // P2奇数位亮
        delay(1000, 20); // 延时0.25秒
    }
}

// 延时函数定义
void delay(unsigned int x, unsigned int y)
{
    unsigned int i, j;
    for (i = 0; i < x; i++)
        for (j = 0; j < y; j++);
}
```

课堂应用4

1. 走马灯、流水灯、左右移界定

知识点和思路讲解：

- **走马灯**：通常指**只有一个**亮点在8个LED中循环移动，其余灯保持熄灭
- **流水灯**：通常指灯**依次点亮并保持**，直到所有灯都亮（有"充满"效果）。广义上也泛指各种动态的花样闪烁模式
- **左移 (Left Shift)**：*程序操作*：对二进制数执行左移操作（如`a << 1`或`_crol_(a, 1)`） *物理现象*：亮点从低位向高位移动，即从 **0号灯 -> 7号灯**（从右向左）

- **右移 (Right Shift)**： *程序操作*：对二进制数执行右移操作（如 `a >> 1` 或 `_cror_(a, 1)`） *物理现象*：亮点从高位向低位移动，即从 **7号灯 -> 0号灯**（从左向右）

2 & 3. 实现左右移走马灯（三种方法）

知识点和思路讲解：

实现走马灯（单个亮点移动）有三种主流方法，均以P2口为例，延时0.25秒（delay(1000, 20);）：

1. **位移运算符 (`<<`, `>>`)**： *左移*：`a = 0x01; a <<= 1;`。当`a`移位到`0x00`（即`0x80 << 1`）时，需要将其重置回`0x01` *右移*：`a = 0x80; a >>= 1;`。当`a`移位到`0x00`（即`0x01 >> 1`）时，需要将其重置回`0x80`
2. **循环移位函数 (`_crol_`, `_cror_`)**： 需包含 `#include <intrins.h>` *左移*：`a = _crol_(a, 1);` (Circular Rotate Left)。当`0x80`左移时，会自动循环到`0x01`，无需重置 *右移*：`a = _cror_(a, 1);` (Circular Rotate Right)。当`0x01`右移时，会自动循环到`0x80` *注意*：笔记中右移代码的初始值是`0x01`，使用`_cror_`会使其变为`0x80` -> `0x40` ... 效果是从右向左再从最左边开始，这与物理右移（7->0）的起始点不同。如果严格要求7->0，初始值应为`0x80`
3. **数组法**： 将走马灯的每一步状态（P2口的值）预先定义在一个数组`table[]`中 *左移*：`table[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80};` *右移*：`table[] = {0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01};` 使用`for`循环遍历数组，依次将数组元素赋值给P2口

方法一：位移运算符 (`<<`, `>>`)

左移 (0 -> 7)

```
#include <reg52.h>
void delay(unsigned int x, unsigned int y);
void main(){
    unsigned char a = 0x01; // 从0号灯开始
    while(1){
        P2 = a;
        delay(1000, 20);
        a <<= 1;
        if (a == 0) a = 0x01; // 移出后重置
    }
}
void delay(unsigned int x, unsigned int y){
    unsigned int i,j;
    for(i=0;i<x;i++)
        for(j=0;j<y;j++);
}
```

右移 (7 -> 0)

```
#include <reg52.h>
void delay(unsigned int x, unsigned int y);
void main(){
    unsigned char a = 0x80; // 从7号灯开始
    while(1){
        P2 = a;
        delay(1000, 20);
        a >>= 1;
        if (a == 0) a = 0x80; // 移出后重置
```

```
    }
}
void delay(unsigned int x, unsigned int y){
    unsigned int i,j;
    for(i=0;i<x;i++)
        for(j=0;j<y;j++);
}
```

方法二：循环移位函数 (`_crol_`, `_cror_`)

循环左移 (0 -> 7 -> 0 ...)

```
#include <reg52.h>
#include <intrins.h> // 包含循环移位头文件
void delay(unsigned int x, unsigned int y);
void main(){
    unsigned char a = 0x01;
    while(1){
        P2 = a;
        delay(1000, 20);
        a = _crol_(a, 1); // 循环左移1位
    }
}
void delay(unsigned int x, unsigned int y){
    unsigned int i,j;
    for(i=0;i<x;i++)
        for(j=0;j<y;j++);
}
```

循环右移 (0 -> 7 -> 6 ... -> 0)

```
#include <reg52.h>
#include <intrins.h> // 包含循环移位头文件
void delay(unsigned int x, unsigned int y);
void main(){
    unsigned char a = 0x01; // 初始状态为0号灯
    while(1){
        P2 = a;
        delay(1000, 20);
        a = _cror_(a, 1); // 循环右移1位 (0x01 -> 0x80 -> 0x40 ...)
    }
}
void delay(unsigned int x, unsigned int y){
    unsigned int i,j;
    for(i=0;i<x;i++)
        for(j=0;j<y;j++);
}
```

方法三：数组法

左移 (0 -> 7)

```
#include <reg52.h>
void delay(unsigned int x, unsigned int y);
```

```
void main(){
    // 定义左移模式表
    unsigned char table[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80};
    char i;
    while(1){
        for(i=0; i<8; i++){
            P2 = table[i];
            delay(1000, 20);
        }
    }
}
void delay(unsigned int x, unsigned int y){
    unsigned int i,j;
    for(i=0;i<x;i++)
        for(j=0;j<y;j++);
}
```

右移 (7 -> 0)

```
#include <reg52.h>
void delay(unsigned int x, unsigned int y);
void main(){
    // 定义右移模式表
    unsigned char table[] = {0x80, 0x40, 0x20, 0x10, 0x08, 0x04, 0x02, 0x01};
    char i;
    while(1){
        for(i=0; i<8; i++){
            P2 = table[i];
            delay(1000, 20);
        }
    }
}
void delay(unsigned int x, unsigned int y){
    unsigned int i,j;
    for(i=0;i<x;i++)
        for(j=0;j<y;j++);
}
```

4. 使用数组法实现左右来回移动走马灯

知识点和思路讲解:

- **逻辑**：将"左移"和"右移"的模式合并
- **数组**：先定义从`0x01`到`0x80`（7步），再定义从`0x40`回到`0x02`（6步） `0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80`(向左) `0x40, 0x20, 0x10, 0x08, 0x04, 0x02`(向右返回)
- **数组大小**：共 8 + 6 = 14个状态
- **实现**：定义包含14个元素的数组，在`while(1)`中使用`for(i=0; i<14; i++)`循环遍历

```
#include <reg52.h>
void delay(unsigned int x, unsigned int y);
void main(){
    // 定义左右来回模式表
```

```
    unsigned char table[] = {0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80, //
向左
                             0x40, 0x20, 0x10, 0x08, 0x04, 0x02};           // 向
右
    char i;
    while(1){
        for(i=0; i<14; i++){ // 循环遍历14个状态
            P2 = table[i];
            delay(1000, 20);
        }
    }
}
void delay(unsigned int x, unsigned int y){
    unsigned int i,j;
    for(i=0;i<x;i++)
        for(j=0;j<y;j++);
}
```

5. 使用数组法实现各种流水灯

知识点和思路讲解：

- **优势**：数组法（查表法）的最大优势是实现任意复杂的LED模式，只需将每一步的P口状态值填入数组即可
- **示例1 (填充式)**：实现从右向左（0->7）依次点亮并保持，直到全亮 `0x01`(0号亮) `0x03`(0,1号亮) `0x07`(0,1,2号亮) ... `0xFF`(全亮) 延时设为0.5秒（`delay(1000, 40);`），全亮后保持1秒（`delay(1000, 80);`）
- **示例2 (中间向两边)**：实现从中间(3,4号)向两边(0,7号)展开 `0x18`(3,4号亮) `0x24`(2,5号亮) `0x42`(1,6号亮) `0x81`(0,7号亮)

示例1：从右向左填充式流水灯

```
#include <reg52.h>
void delay(unsigned int x, unsigned int y);
void main(){
    // 定义填充模式表
    unsigned char table[] = {0x01, 0x03, 0x07, 0x0F, 0x1F, 0x3F, 0x7F, 0xFF};
    char i;
    while(1){
        for(i=0; i<8; i++){
            P2 = table[i];
            delay(1000, 40); // 延时0.5秒
        }
        delay(1000, 80); // 全亮后保持1秒
    }
}
void delay(unsigned int x, unsigned int y){
    unsigned int i,j;
    for(i=0;i<x;i++)
        for(j=0;j<y;j++);
}
```

示例2：中间向两边展开式流水灯

```
#include <reg52.h>
void delay(unsigned int x, unsigned int y);
void main(){
    // 定义中间向两边展开模式表
    unsigned char table[] = {0x18, 0x24, 0x42, 0x81}; // 3&4 -> 2&5 -> 1&6 -> 0&7
    char i;
    while(1){
        for(i=0; i<4; i++){
            P2 = table[i];
            delay(1000, 40); // 延时0.5秒
        }
    }
}
void delay(unsigned int x, unsigned int y){
    unsigned int i,j;
    for(i=0;i<x;i++)
        for(j=0;j<y;j++);
}
```