Arduino 数码管数字循环显示程序详解

本实验目标：一位数码管循环显示数字

目标分析：为了完成这个实验，我们首先对于它进行架构的分析

传感器：无
控制器：UNO/ESP32
执行器：一位数码管

接着是效果分析：

有一个数码管，他会从0到9显示数字，进行数字的循环显示

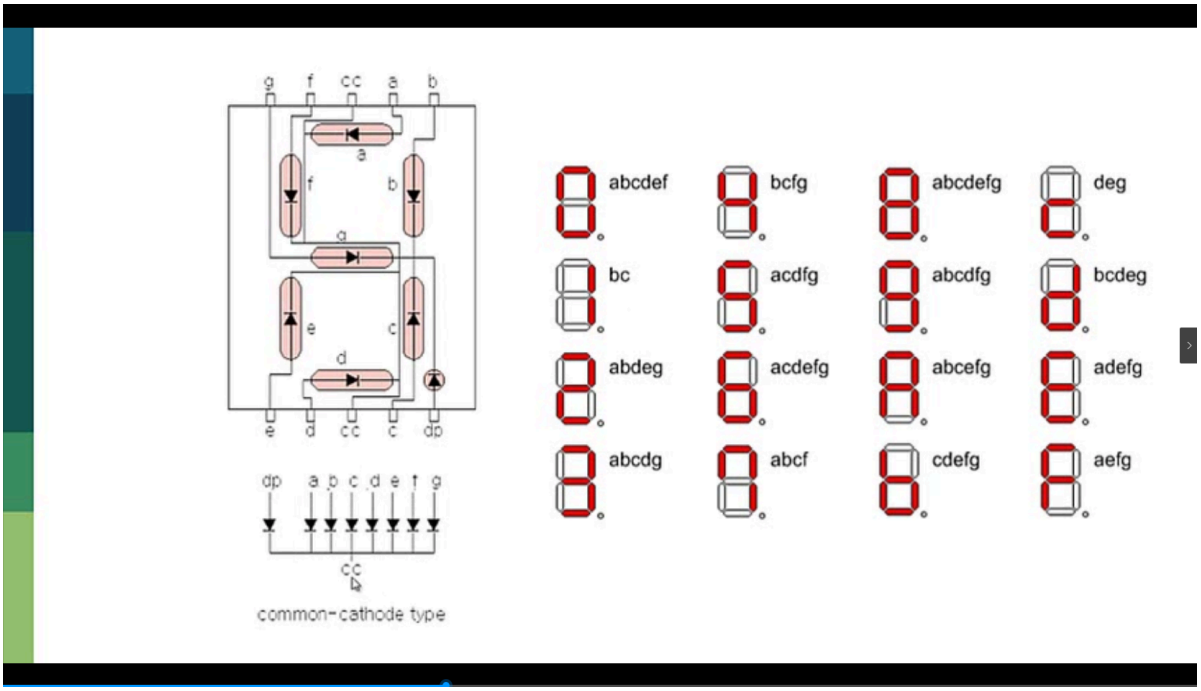
明确目标系统与效果，开始正式学习---

# 代码全貌

```
volatile int x; // 全局变量，存储当前显示数字

void 数码管显示数字(int x) {
  // 控制各段引脚电平（根据数字x）
  if (x == 0) { /* 0的段码 */ }
  if (x == 1) { /* 1的段码 */ }
  // ...其他数字类似
}

void setup(){
  x = 0; // 初始化显示数字为0
  // 配置所有段引脚为输出模式
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
}

void loop(){
  数码管显示数字(x); // 显示当前数字
  delay(1000);       // 保持1秒
  x = x + 1;         // 数字递增
  if (x == 10) {     // 达到10时归零
    x = 0;
  }
}
```

# 核心概念解析

## 1. 数码管工作原理




| 术语 | 说明 |
| --- | --- |
| 七段数码管 | 由7个LED段（a-g）组成，可显示0-9数字 |
| 共阴/共阳 | 公共端接GND（共阴）或VCC（共阳），决定各段亮灭逻辑 |
| 段码 | 控制各段亮灭的组合，不同数字对应不同的段码组合。 |

## 2. 引脚与段对应关系（假设共阴数码管）

| 引脚 | 对应段 | 功能说明 |
| --- | --- | --- |
| 4 | a | 顶部横段 |
| 5 | b | 右上竖段 |
| 6 | c | 右下竖段 |
| 7 | d | 底部横段 |
| 9 | e | 左下竖段 |
| 10 | f | 左上竖段 |
| 11 | g | 中间横段 |

# 代码逐行详解

## 1. 全局变量声明

```
volatile int x;
```

- `volatile`：告知编译器该变量可能被意外修改（常用于中断场景）
- **实际作用**：在本程序中可省略，保留为兼容后续扩展

## 2. 数码管显示函数

```
void 数码管显示数字(int x) {
  if (x == 0) {
    digitalWrite(4,HIGH); // a段亮
    digitalWrite(5,HIGH); // b段亮
    digitalWrite(6,HIGH); // c段亮
    digitalWrite(7,LOW);  // d段灭（异常，应为HIGH）
    digitalWrite(9,HIGH); // e段亮
    digitalWrite(10,HIGH);// f段亮
    digitalWrite(11,HIGH);// g段灭（异常，应为LOW）
  }
  // 其他数字类似...
}
```

**问题分析**：

- **段码错误**：示例中数字0的d段（引脚7）和g段（引脚11）电平设置与标准段码不符
- **标准共阴数码管段码表**：

| 数字 | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| ... | | | | | | | |

**建议修正**：

```
if (x == 0) {
  digitalWrite(4, HIGH); // a
  digitalWrite(5, HIGH); // b
  digitalWrite(6, HIGH); // c
  digitalWrite(7, HIGH); // d
  digitalWrite(9, HIGH); // e
  digitalWrite(10,HIGH); // f
  digitalWrite(11, LOW); // g
}
```

## 3. 初始化函数（setup）

```
void setup(){
  x = 0;
  pinMode(4, OUTPUT); // 配置a段引脚
  pinMode(5, OUTPUT); // b段
  pinMode(6, OUTPUT); // c段
  pinMode(7, OUTPUT); // d段
  pinMode(9, OUTPUT); // e段
  pinMode(10,OUTPUT); // f段
  pinMode(11,OUTPUT); // g段
}
```

## 4. 主循环（loop）

```
void loop(){
  数码管显示数字(x); // 调用显示函数
  delay(1000);     // 保持显示1秒
  x++;             // 等效于x = x + 1
  if (x == 10) {  // 检测是否超过9
    x = 0;         // 复位到0
  }
}
```

**执行流程**:

1. 显示当前数字
2. 等待1秒
3. 数字递增
4. 当数字达到10时重置为0
5. 重复循环

---

# 硬件连接指南

## 共阴数码管标准接法

```
Arduino引脚 → 220Ω电阻 → 数码管各段（a-g）
数码管公共端（COM） → GND
```

## 本程序对应电路

| 数码管段 | Arduino引脚 | 限流电阻 |
| --- | --- | --- |
| a | 4 | 220Ω |
| b | 5 | 220Ω |
| c | 6 | 220Ω |
| d | 7 | 220Ω |

| 数码管段 | Arduino引脚 | 限流电阻 |
| --- | --- | --- |
| e | 9 | 220Ω |
| f | 10 | 220Ω |
| g | 11 | 220Ω |
| COM | GND | - |

//这里是共阴数码管，各段都有一个共同的GND，因此他们的电阻实际上是同一个电阻。

---

# 关键问题解析

## Q1：显示数字残缺或错误

- **排查步骤**：
  1. 检查所有段引脚连接是否正确
  2. 验证数码管公共端是否接地
  3. 核对段码表是否与硬件匹配

## Q2：如何加快/减慢切换速度

- **修改延时参数**：

```
delay(500);  // 改为0.5秒切换
delay(2000); // 改为2秒切换
```

## Q3：显示多个数码管

- **扩展方案**：

```
// 使用数组存储段码
byte segCodes[10] = {
  0b1111110, // 0
  0b0110000, // 1
  // ...其他数字
};

void 显示数字(int num, int digit){
  // 增加位选控制
  digitalWrite(digitPin[digit], LOW);
  // 输出段码
  // ...
}
```

---

# 程序优化建议

## 1. 使用段码数组

```
const byte segCodes[10] = {
  // gfedcba（低位到高位）
  0b00111111, // 0
  0b00000110, // 1
  0b01011011, // 2
  0b01001111, // 3
  0b01100110, // 4
  0b01101101, // 5
  0b01111101, // 6
  0b00000111, // 7
  0b01111111, // 8
  0b01101111  // 9
};

void 数码管显示数字(int x){
  byte code = segCodes[x];
  digitalWrite(4, code & 0b00000001);
  digitalWrite(5, code & 0b00000010);
  // 其他段类似...
}
```

## 2. 增加消隐处理

```
void 数码管显示数字(int x){
  digitalWrite(4, LOW); // 先关闭所有段
  // ...其他段同理
  // 再设置需要点亮的段
  // ...
}
```

---

# 扩展实验

## 实验1：倒计时功能

```
void loop(){
  for(int i=9; i>=0; i--){
    数码管显示数字(i);
    delay(1000);
  }
}
```

## 实验2：按键控制切换

```
void loop(){
  if(digitalRead(2) == HIGH){ // 按钮接D2
    x = (x+1)%10;
    delay(200); // 防抖
  }
  数码管显示数字(x);
}
```

## 实验3：亮度调节

```
void loop(){
  int brightness = analogRead(A0)/4; // 电位器调光
  analogWrite(3, brightness); // PWM控制公共端
  数码管显示数字(x);
}
```

---

# 总结

本程序通过直接控制各段电平实现数码管显示，核心知识点包括：

1. **数码管结构**：理解共阴/共阳类型与段码关系
2. **数字编码**：掌握各数字对应的段亮灭组合
3. **循环控制**：实现数字自动递增显示
4. **硬件连接**：正确配置限流电阻与引脚对应

通过优化段码存储方式和增加控制功能，可以扩展出更复杂的显示效果，比如不局限于数字，还有各种字符的形成。

---

有问题找柯萌(⊙_⊙)
mingyoufhh@outlook.com