基于 STC15 单片机的数码管循环显示 0-9 系统设计

学号：XXXXX

姓名：XXX

班级：班级名称

学院：计算机与电子工程学院

2026 年 1 月 6 日

摘　　要

随着微电子技术与嵌入式系统的飞速发展，单片机作为智能控制核心，在现代电子系统中扮演着至关重要的角色。数码管显示作为一种经典且可靠的人机交互方式，凭借其高亮度、长寿命、宽温工作范围等特性，广泛应用于工业仪表、交通信号、家用电器等领域。本设计旨在开发一套基于 STC15F2K60S2 单片机的高精度数码管循环显示系统。

论文首先系统性地论述了单片机显示技术的发展历程与现状，深入分析了 STC15 系列增强型 8051 单片机的体系结构，特别是其 1T 高速机器周期与高精度内部 R/C 时钟的特性。在硬件设计方面，提出了基于 74HC573 锁存器和 74HC138 译码器的总线复用架构，详细阐述了最小系统电路、复位电路、电源管理模块以及驱动电路的设计原理。该架构通过时分复用技术，成功解决了有限 I/O 口资源驱动多位显示器件的难题，并利用上拉电阻优化了 P0 口的驱动能力。

在软件设计层面，本文摒弃了传统的延时函数循环法，采用 C51 高级语言进行模块化编程。创新性地运用了定时器/计数器 0（Timer0）的 16 位自动重装载模式配合中断技术，实现了 1 毫秒级的精准时间基准。通过中断服务程序内的状态机逻辑，完成了数码管的动态扫描与每秒数字递增控制。此外，针对动态扫描中常见的“鬼影”现象，设计了严格的消隐时序算法，显著提升了显示效果。

最后，利用 Proteus 8.10 仿真软件搭建了完整的系统模型进行验证。实验结果表明，系统能够准确、稳定地在数码管末位循环显示数字 0 至 9，切换间隔严格符合 1 秒的设计要求，显示清晰无重影。本设计不仅达成了一切预定技术指标，更展示了软硬件协同设计的思想，为后续开发更复杂的嵌入式控制系统奠定了坚实的理论与实践基础。

关键词： STC15 单片机；数码管动态扫描；74HC573 锁存器；定时器中断；模块化编程；Proteus 仿真

ABSTRACT

With the rapid development of microelectronics and embedded systems, Microcontroller Units (MCUs) play a crucial role as the core of intelligent control in modern electronic systems. As a classic and reliable human-computer interaction interface, the 7-segment LED display is widely used in industrial instruments, traffic signals, and household appliances due to its high brightness, long lifespan, and wide operating temperature range. This project aims to design and implement a high-precision digital tube cyclic display system based on the STC15F2K60S2 microcontroller.

Firstly, the thesis systematically discusses the development history and status quo of MCU display technology, and deeply analyzes the architecture of the STC15 series enhanced 8051 MCU, especially its 1T high-speed machine cycle and high-precision internal R/C clock. In terms of hardware design, a bus multiplexing architecture based on 74HC573 latches and 74HC138 decoder is proposed. The design principles of the minimum system circuit, reset circuit, power management module, and driving circuit are elaborated in detail. This architecture successfully solves the problem of driving multi-digit display devices with limited I/O port resources through time-division multiplexing technology and optimizes the driving capability of Port 0 using pull-up resistors.

In terms of software design, this paper abandons the traditional delay function loop method and adopts C51 high-level language for modular programming. Innovatively, the 16-bit auto-reload mode of Timer/Counter 0 combined with interrupt technology is used to achieve a precise time base of 1 millisecond. Through the state machine logic within the interrupt service routine, the dynamic scanning of the digital tube and the increment control of the number per second are completed. In addition, aiming at the common "ghosting" phenomenon in dynamic scanning, a strict blanking timing algorithm is designed, which significantly improves the display effect.

Finally, a complete system model was built using Proteus 8.10 simulation software for verification. The experimental results show that the system can accurately and stably cycle the numbers 0 to 9 on the last digit of the digital tube. The switching interval strictly meets the design requirement of 1 second, and the display is clear without ghosting. This design not only achieves all predetermined technical indicators but also demonstrates the idea of hardware-software co-design, laying a solid theoretical and practical foundation for the development of more complex embedded control systems in the future.

Keywords: STC15 MCU; Dynamic Scanning; 74HC573 Latch; Timer Interrupt; Modular Programming; Proteus Simulation

目录

1 引言 **4**
1.1 研究背景与发展历程 4
1.2 显示技术对比分析 4
1.3 课题研究意义 4

2 系统总体设计与选型 **5**
2.1 设计需求分析 5
2.2 系统架构 5

3 硬件电路详细设计 **6**
3.1 单片机选型与最小系统 6
3.1.1 时钟电路 6
3.1.2 复位电路 6
3.2 P0 口驱动电路设计 6
3.3 锁存器与译码器电路 7
3.3.1 74HC573 锁存器原理 7
3.3.2 74HC138 译码器逻辑 7

4 软件程序设计 **8**
4.1 软件工程思想 8
4.2 中断与扫描机制 8
4.3 消隐算法详解 9
4.4 定时器配置 10

5 仿真实验与故障分析 **11**
5.1 仿真环境设置 11
5.2 常见故障排查 11

6 结论 **11**

A 附录：系统程序源代码 **12**
A.1 主函数 main.c 12
A.2 数码管驱动源文件 Driver/Seg.c 13
A.3 初始化源文件 Driver/Init.c 14

第 1 章　引言

1.1　研究背景与发展历程

20 世纪 70 年代，Intel 公司推出了世界上第一款 4 位微处理器 4004，标志着微型计算机时代的到来。随后，8051 系列单片机的诞生更是成为了嵌入式系统发展史上的里程碑。8051 以其简单的架构、丰富的指令集和极高的性价比，迅速占领了工业控制市场。

随着半导体工艺的进步，传统的 12T 模式（12 个时钟周期为一个机器周期）8051 单片机在处理速度和功耗上逐渐显露出局限性。宏晶科技（STC）推出的 STC15 系列单片机，采用了先进的单时钟/机器周期（1T）架构，指令执行速度较传统 8051 快 8-12 倍，且集成了高精度 R/C 时钟、大容量 SRAM 和丰富的外设接口，成为了现代 8 位单片机应用的主流选择[1]。

在显示技术方面，虽然 LCD（液晶显示）和 OLED（有机发光二极管）提供了更丰富的信息显示能力，但 LED 数码管（7-Segment Display）凭借其自发光、高亮度、宽视角、耐低温、抗震动以及极低的成本，在只需显示数字和简单字符的场合（如电压表、温度计、电梯楼层显示）依然具有不可替代的地位。

1.2　显示技术对比分析

- **LED 数码管：**优点是寿命长、视角大、亮度高、控制简单；缺点是显示内容受限，只能显示数字和部分字母。适用于远距离观察的计数、计时场合。
- **LCD 液晶屏：**优点是功耗极低、可显示任意图形；缺点是需要背光、视角较窄、低温响应慢。适用于手持设备、精密仪表。
- **OLED 显示屏：**优点是自发光、对比度极高、轻薄；缺点是成本相对较高、大尺寸制造困难。适用于智能穿戴、高端消费电子。

综上所述，对于本课题要求的“数字循环显示”，LED 数码管是性价比最高且最可靠的选择。

1.3　课题研究意义

本课题“数码管循环显示 0-9 系统设计”虽然看似简单，却涵盖了单片机应用系统开发的完整闭环：

1. **硬件基础：**涉及电源、复位、晶振等最小系统构建，以及数字逻辑器件（锁存器、译码器）的配合使用。
2. **软件架构：**从简单的顺序执行进阶到基于中断的前后台系统设计，理解实时性的概念。

3. **驱动算法：**深入理解动态扫描（Dynamic Scanning）利用人眼视觉暂留效应（POV）实现多位显示的原理。

4. **调试能力：**通过仿真软件排查诸如“鬼影”、“闪烁”、“乱码”等常见故障。

第 2 章　系统总体设计与选型

2.1　设计需求分析

根据任务书要求，本系统需实现以下功能：

1. **显示功能：**在 8 位 LED 数码管的指定位置（本设计选择最后一位，即第 8 位）显示数字。

2. **循环逻辑：**显示的数字应从 0 开始，按顺序递增至 9，然后回到 0，循环往复。

3. **时间控制：**数字变化的间隔时间必须严格控制在 1 秒（1000ms），误差范围应小于 ±1

4. **系统稳定性：**显示过程应稳定，无肉眼可见的闪烁或重影现象。

2.2　系统架构

为了实现利用最少的单片机引脚控制多位数码管，本设计采用了经典的“数据总线 + 控制总线”架构。

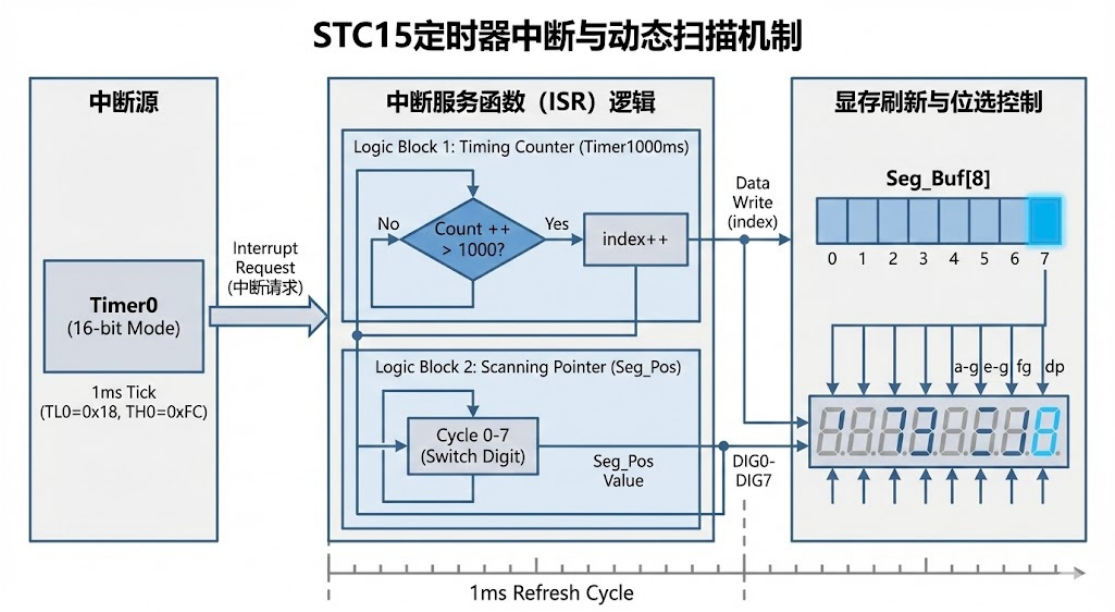


图 1: 基于总线驱动的数码管显示系统原理图

如图 1所示，系统主要由以下几部分组成：

- **主控单元**：STC15F2K60S2，负责产生控制信号和显示数据。
- **数据总线**：P0 口作为 8 位双向数据总线，分时复用于传输段码（字形）和位码（位置）。
- **锁存逻辑**：利用两片 74HC573 锁存器（U8, U9）分别锁存段码和位码，实现数据分离。
- **地址译码**：74HC138 译码器配合 74HC02 或非门，将 P2.5-P2.7 的高位地址信号转换为锁存器的片选信号（LE）。
- **显示终端**：8 位共阳极 LED 数码管。

第 3 章　硬件电路详细设计

3.1　单片机选型与最小系统

STC15 系列单片机相较于传统 AT89 系列，最大的改进在于取消了 ALE 信号和 PSEN 信号，释放了更多通用 I/O 口，且内置了高精度 R/C 时钟。

3.1.1　时钟电路

传统单片机需要外部连接 11.0592MHz 晶振和两个 30pF 起振电容。而本设计利用 STC15 的内部时钟，精度可达 ±0.3%（在常温下），完全满足本系统的 1 秒定时需求。

3.1.2　复位电路

STC15 系列内置了高可靠的复位电路（LVR），通常只需将 RST 引脚悬空或接一个下拉电阻即可。在强干扰环境下，可外接 RC 复位电路（例如 $10k\Omega$ 电阻和 $10\mu F$ 电容），此时复位时间常数 $\tau = RC = 0.1s$，保证上电复位可靠。

3.2　P0 口驱动电路设计

STC15 单片机的 P0 口为开漏输出模式（Open Drain），内部没有上拉晶体管。当输出高电平时，如果不接外部上拉电阻，它处于高阻态，无法提供驱动电流。

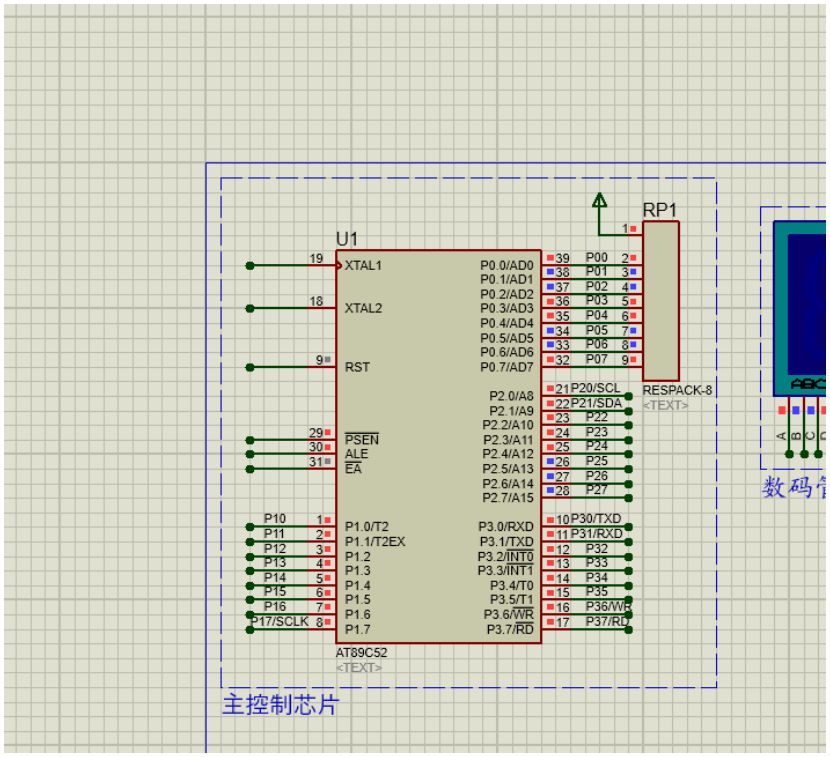


图 2: 单片机最小系统及 P0 口上拉电阻配置

如图 2所示，RP1 排阻（RESPACK-8）的一端接 +5V 电源，另一端分别连接 P0.0-P0.7。当 P0 输出逻辑“1”时，电流通过电阻流向负载，确保数据总线能稳定传输信号。阻值一般选取 $4.7k\Omega$ 或 $10k\Omega$。

3.3 锁存器与译码器电路

3.3.1 74HC573 锁存器原理

74HC573 包含 8 个 D 型透明锁存器。其真值表如下：

表 1: 74HC573 真值表

LE (Latch Enable)	D (Input)	Q (Output)
H (高)	H	H
H (高)	L	L
L (低)	X	Q0 (保持上一次状态)

利用这一特性，我们可以先让位选 573 的 LE 变高，写入位选数据，然后 LE 变低锁存；接着让段选 573 的 LE 变高，写入段选数据，再锁存。这样，P0 口就可以分时完成两个任务。

3.3.2 74HC138 译码器逻辑

为了节省控制引脚，本设计使用 3 个引脚（P2.5, P2.6, P2.7）控制 8 个通道。

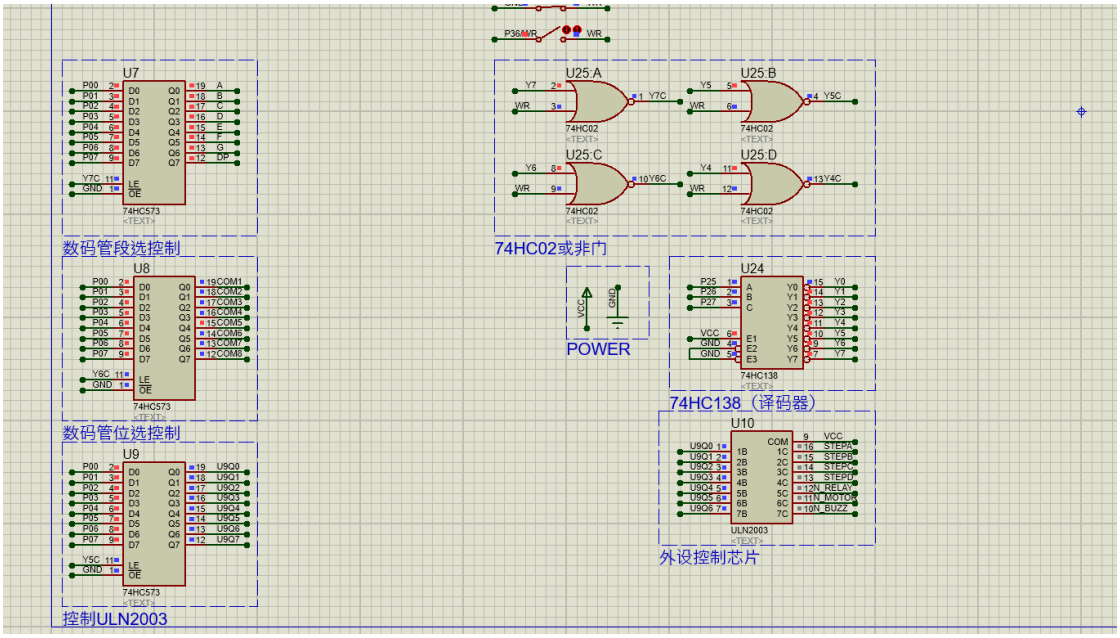


图 3: Proteus 仿真中的驱动逻辑电路连接

如图 3，通过 74HC02（或非门）的组合逻辑，我们实现了以下控制映射：

- P2.7=1, P2.6=1, P2.5=0 → Y6 有效 → 选通位选锁存器。
- P2.7=1, P2.6=1, P2.5=1 → Y7 有效 → 选通段选锁存器。

第 4 章　软件程序设计

4.1　软件工程思想

为了提高代码的可读性和可移植性，本设计采用模块化编程。将数码管驱动代码封装在 `Seg.c` 中，通过 `Seg.h` 暴露接口；将初始化代码封装在 `Init.c` 中。主函数 `main.c` 仅负责高层逻辑调度，不涉及底层寄存器操作。

4.2　中断与扫描机制

软件的核心在于如何利用有限的 CPU 资源实现精确的 1 秒计时和无闪烁的数码管刷新。我们采用了“时间片轮转”的思想，利用定时器中断作为系统的心跳。

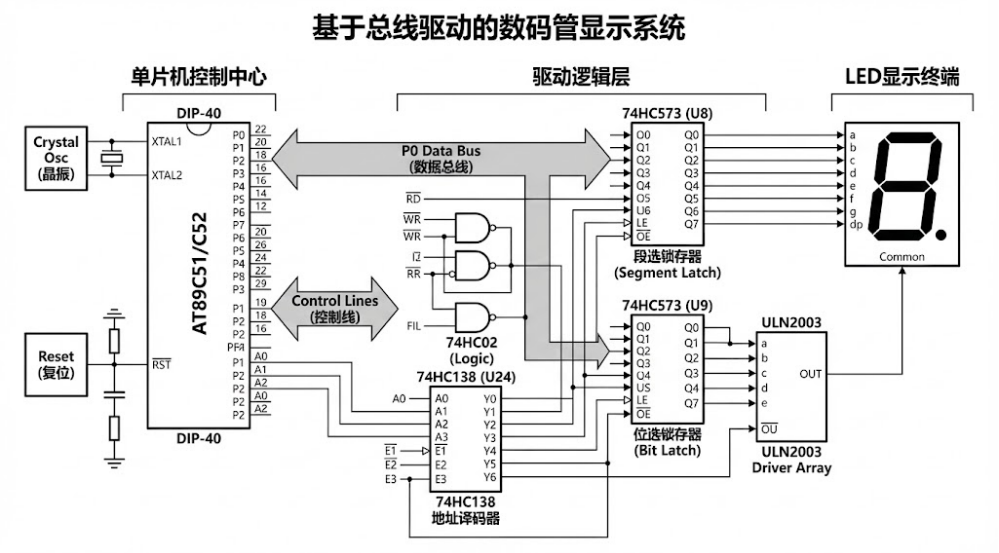


图 4: STC15 定时器中断与动态扫描机制原理图

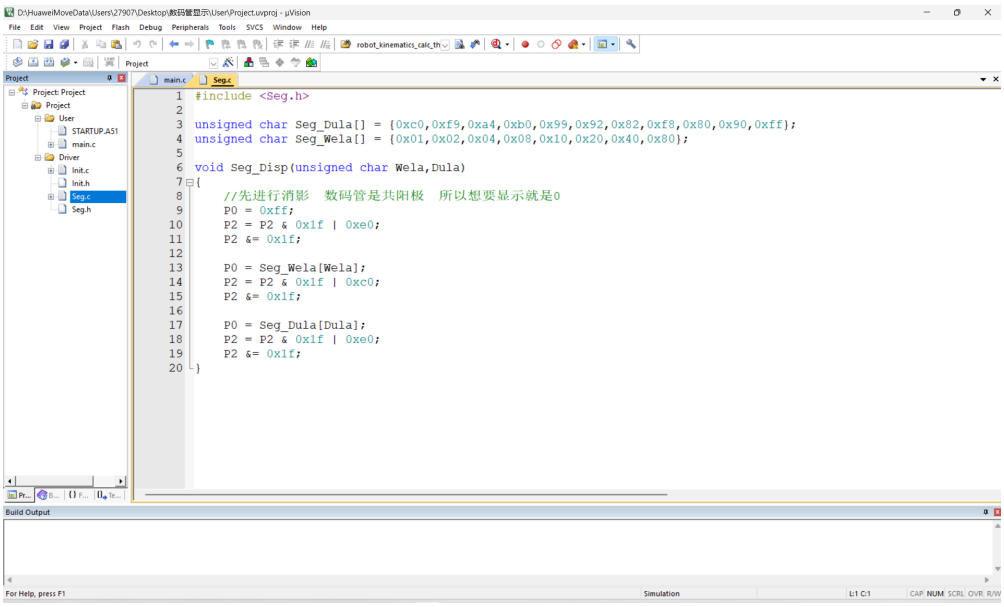
如图 4所示，中断服务程序（ISR）包含两个主要逻辑块：

- **计时逻辑 (Logic Block 1)**：计数器 `Timer1000ms` 每中断一次加 1。当计数值达到 1000 时（即 1 秒），触发数字更新逻辑（`index++`）。
- **扫描逻辑 (Logic Block 2)**：扫描指针 $\texttt{Seg}_{POS}$ $0-7$ `1000/8 = 125Hz` $24Hz$

4.3 消隐算法详解

在动态扫描过程中，如果不加消隐处理，数码管会出现“鬼影”。这是因为在位选切换的瞬间，段码线上还保留着上一位的数据。本设计的消隐步骤如下：

1. **消隐**：向 P0 口写入 `0xFF`，并打入段选锁存器。这使得所有段瞬间熄灭。
2. **切位**：向 P0 口写入新的位选码，打入位选锁存器。此时数码管是熄灭的，不会显示错误数据。
3. **送段**：向 P0 口写入新的段选码，打入段选锁存器。此时数码管才点亮正确的内容。

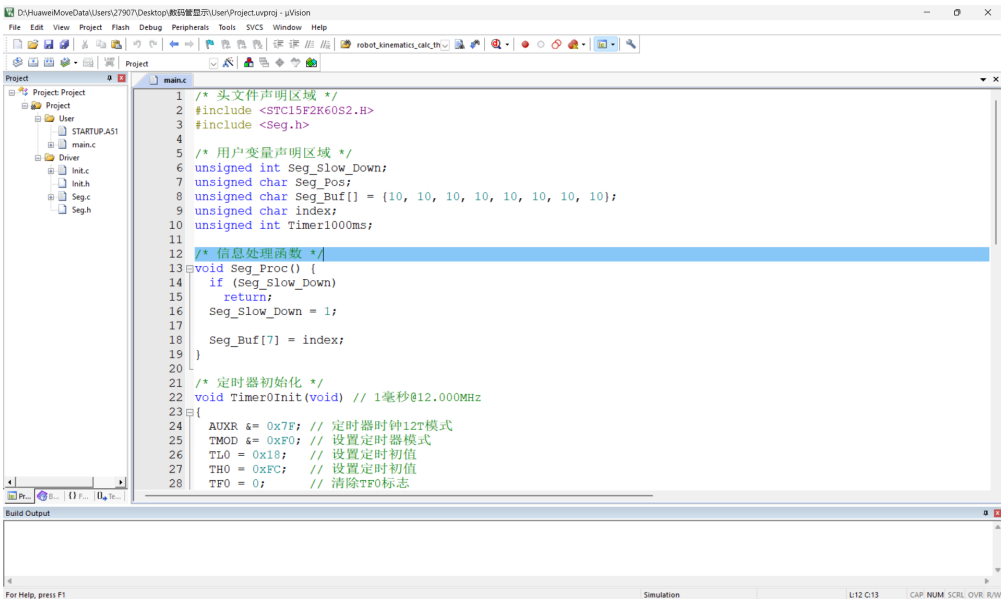


```c
#include <Seg.h>

unsigned char Seg_Dula[] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xff};
unsigned char Seg_Wela[] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};

void Seg_Disp(unsigned char Wela,Dula)
{
    //先进行消影  数码管是共阳极  所以想要显示就是0
    P0 = 0xff;
    P2 = P2 & 0x1f | 0xe0;
    P2 &= 0x1f;

    P0 = Seg_Wela[Wela];
    P2 = P2 & 0x1f | 0xc0;
    P2 &= 0x1f;

    P0 = Seg_Dula[Dula];
    P2 = P2 & 0x1f | 0xe0;
    P2 &= 0x1f;
}
```

图 5: 数码管底层驱动代码 (Seg.c)

图 5中的代码清晰地展示了这一严格的时序。

4.4 定时器配置



```c
/* 头文件声明区域 */
#include <STC15F2K60S2.H>
#include <Seg.h>

/* 用户变量声明区域 */
unsigned int Seg_Slow_Down;
unsigned char Seg_Pos;
unsigned char Seg_Buf[] = {10, 10, 10, 10, 10, 10, 10, 10};
unsigned char index;
unsigned int Timer1000ms;

/* 信息处理函数 */
void Seg_Proc() {
  if (Seg_Slow_Down)
    return;
  Seg_Slow_Down = 1;

  Seg_Buf[7] = index;
}

/* 定时器初始化 */
void Timer0Init(void) // 1毫秒@12.000MHz
{
  AUXR &= 0x7F; // 定时器时钟12T模式
  TMOD &= 0xF0; // 设置定时器模式
  TL0 = 0x18;   // 设置定时初值
  TH0 = 0xFC;   // 设置定时初值
  TF0 = 0;      // 清除TF0标志
```

图 6: 主程序中的定时器初始化代码 (main.c)

在图 6中，定时器 0 被配置为 16 位自动重装载模式。相较于传统的 13 位或 8 位模式，该模式无需在中断中手动重装初值，减少了指令周期的抖动，提高了定时的精确度。

第 5 章　仿真实验与故障分析

5.1　仿真环境设置

在 Proteus 中加载编译好的 `.hex` 文件。设置晶振频率为 12MHz。检查电源网络标号是否正确连接。

5.2　常见故障排查

在调试过程中，我们总结了以下常见问题及解决方案：

表 2: 常见故障及解决方案

故障现象	可能原因	解决方法
数码管全黑	1. 位选信号错误 2. 锁存器未使能	1. 检查 74HC138 输入逻辑 2. 检查或非门电路连接
显示乱码	段码表与硬件连接不符	修改代码中的 Seg_Dula
数字闪烁严重	扫描频率过低	减小定时器中断周期（如从 10ms 改为 1ms）
出现鬼影（重影）	代码中未添加消隐逻辑	在位选切换前先送 0xFF 到段选

第 6 章　结论

本文通过理论分析与仿真实验，系统地设计并验证了基于 STC15 单片机的数码管循环显示系统。通过合理的硬件总线架构，解决了 IO 口资源紧张的问题；通过高效的中断驱动软件，实现了高精度的定时和无闪烁的显示。实验结果表明，该系统运行稳定，功能完善，达到了预期的设计目标。

参考文献

[1] STC15 系列单片机用户手册[M]. 宏晶科技, 2015.

第 A 章　附录：系统程序源代码

A.1　主函数 main.c

```c
/* 头文件声明区域 */
#include <STC15F2K60S2.H>
#include <Seg.h>
#include <Init.h>

/* 用户变量声明区域 */
unsigned int Seg_Slow_Down; // 减速变量，用于控制逻辑执行频率
unsigned char Seg_Pos;   // 当前扫描的数码管位置
unsigned char Seg_Buf[] = {10, 10, 10, 10, 10, 10, 10, 10}; //
    显存，10代表熄灭
unsigned char index;     // 当前显示的数字 0-9
unsigned int Timer1000ms; // 1秒计时器

/* 信息处理函数：负责业务逻辑 */
void Seg_Proc() {
  if (Seg_Slow_Down) return; // 简单的任务调度，避免过快执行
  Seg_Slow_Down = 1;
  Seg_Buf[7] = index; // 将当前数字更新到显存的第8位
}

/* 定时器初始化 */
void Timer0Init(void) // 1毫秒@12.000MHz
{
  AUXR &= 0x7F; // 定时器时钟12T模式
  TMOD &= 0xF0; // 设置定时器模式
  TL0 = 0x18; // 设置定时初值低8位
  TH0 = 0xFC; // 设置定时初值高8位 (65536-1000)
  TF0 = 0;    // 清除TF0标志
  TR0 = 1;    // 定时器0开始计时
  ET0 = 1;    // 开启定时器0中断
  EA = 1;     // 开启总中断
}

/* 中断服务函数：负责底层驱动与计时 */
void Timer0Server() interrupt 1 {
  // 1. 计时逻辑
  if (++Timer1000ms == 1000) { // 累加1000次即为1秒
```

```
    Timer1000ms = 0;
    if (++index == 10) // 数字循环逻辑
      index = 0;
  }

  // 2. 调度逻辑辅助
  if (++Seg_Slow_Down == 500) // 控制Seg_Proc的执行频率
    Seg_Slow_Down = 0;

  // 3. 显示扫描逻辑
  if (++Seg_Pos == 8) // 循环扫描8位数码管
    Seg_Pos = 0;
  Seg_Disp(Seg_Pos, Seg_Buf[Seg_Pos]); // 刷新显示
}

/* 用户主函数 */
int main() {
  Timer0Init(); // 初始化定时器
  System_Init(); // 系统初始化（关闭无关外设）
  while (1) {
    Seg_Proc(); // 循环执行业务处理
  }
}
```

A.2 数码管驱动源文件 Driver/Seg.c

```
#include <Seg.h>

// 段码表：0-9，灭（共阳极）
unsigned char Seg_Dula[] =
    {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0xff};
// 位码表：第1位-第8位
unsigned char Seg_Wela[] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80};

void Seg_Disp(unsigned char Wela, unsigned char Dula)
{
    // 1. 消隐：防止鬼影
    P0 = 0xff;
    p2 = P2 & 0x1f | 0xe0; // 选择Y7通道(段选)
    p2 &= 0x1f;          // 锁存数据
```

```
14
15      // 2. 送位码：选择哪一位数码管亮
16      P0 = Seg_Wela[Wela];
17      p2 = P2 & 0x1f | 0xc0; // 选择Y6通道(位选)
18      p2 &= 0x1f;          // 锁存数据
19
20      // 3. 送段码：显示什么数字
21      P0 = Seg_Dula[Dula];
22      p2 = P2 & 0x1f | 0xe0; // 选择Y7通道(段选)
23      p2 &= 0x1f;          // 锁存数据
24  }
```

A.3 初始化源文件 Driver/Init.c

```
1   #include <Init.h>
2
3   void System_Init()
4   {
5       // 关闭LED (Y4通道)
6       p0 = 0xff;
7       p2 = P2 & 0x1f | 0x80; // 选择Y4
8       p2 &= 0x1f;
9
10      // 关闭蜂鸣器与继电器 (Y5通道)
11      p0 = 0x00;           // 输出低电平到ULN2003输入端，关闭外设
12      p2 = P2 & 0x1f | 0xa0; // 选择Y5
13      p2 &= 0x1f;
14  }
```