# 基于E203的四位数码管计数器课程设计报告

团队：2352396 禹尧珅 & 2351283 吴凯

## 摘要

本次小作业设计在 E203 SoC 上完成了四位七段数码管外设 my_periph_example 的集成与上板验证。外设提供 200 Hz 扫描与可选 1 Hz 自动计数，CTRL/DATA 寄存器允许软件关闭自动计数、直接写入 BCD 数据。我们将 seg_out[7:0]、cc[3:0] 逐层透传到顶层，复用 led_my 的引脚约束；固件基于 Hummingbird E200 BSP，每秒通过 CLINT 延时写 DATA 并通过 UART 打印当前值，展示“软件改显示”的要求。报告围绕需求、实现、上板验证与问题处理展开，重点记录上板现象、调试路径与实操经验，仿真内容按要求移除，并补充了我们在排错中的权衡和取舍。

## Abstract

We integrated a four-digit 7-seg peripheral on the RISC-V E203 SoC and validated it on hardware. The custom ICB peripheral my_periph_example offers 200 Hz multiplexing plus optional 1 Hz auto-count. CTRL/DATA registers (base 0x1001_4000) let firmware disable auto-count and push BCD digits. seg_out[7:0]/cc[3:0] are passed to the top level and reuse the led_my pin constraints. Firmware (Hummingbird E200 BSP) updates DATA every second via a CLINT-based delay and logs the value over UART, proving the “software-driven display” goal. This report focuses on implementation and board bring-up; simulation content is omitted by request.

## 一、引言

课程要求：在 E203 上驱动四位共阴数码管，硬件需在 CPU 不干预时自动扫描/计数，同时提供寄存器接口让软件随时改显示。我们沿用 18 MHz 时钟与高电平段选，避免 IOF 混用；目标是稳定驱动、易于软件控制，并在实板上完成可观测的计数与手动写数。我们首先明确“自动计数”和“软件主控”两种模式的切换逻辑，然后倒推寄存器、时序、约束和固件接口，保证上板后无需频繁改 RTL 即可通过软件演示。我们一开始就列出“上电有默认显示”“串口同步日志”“极性可调”三条验收清单，后续每一步都对照清单自测。

## 二、设计与实现

### 2.1 总体方案

链路规划为 e203_soc_demo ← e203_soc_top ← e203_subsys_top ← e203_subsys_main ← e203_subsys_perips ← my_periph_example，确保 seg_out/cc 每层均显式透传到顶层 IO。地址复用 QSPI0 空洞 0x1001_4000：CTRL bit0=count_en（1=自动计数，0=软件驱动），DATA[15:0]={d3,d2,d1,d0} BCD。时钟保持 18 MHz，内部计算 DIV_200HZ=CLK_HZ/200 生成 200 Hz 扫描，累加 200 次得到 1 Hz 计数。我们特意把扫描频率与系统时钟解耦，用参数化写法便于后续换板或换频；同时保留原有 PWM IOF，不破坏参考工程外设。为了防止层级遗漏，我们在每一级添加端口前都跑一次 iverilog -E 预处理检查端口是否被优化掉。

### 2.2 寄存器设计

在 my_periph_example 仅保留两类寄存器：CTRL(0x00, bit0=count_en, 复位=1) 与 DATA(0x04, 四位 BCD, 复位=0)。写 DATA 立即覆盖当前显示，优先级高于自动计数；写 CTRL 控制是否允许 1 Hz 自

增。读 DATA 返回当前 BCD，读 CTRL 返回 count_en。我们在 ICB 接口上做了简化：只响应合法地址，默认 ready/valid，保持时序单周期返回，方便软件轮询；同时在 RTL 内部加入数码管内容寄存寄存器，避免写冲突导致的显示毛刺。为后续扩展小数点或符号，我们在 DATA 高位预留空间并在注释标明，便于增加模式位。

## 2.3 扫描与译码

系统时钟 18 MHz，cnt_200hz 计到 DIV_200HZ-1 产生 tick_200hz，累计 200 次得到 tick_1hz。digit_cnt 轮询四位，译码 0-F 得到七段码并取反输出（高电平有效），位选顺序 1000→0100→0010→0001（右到左）。若板子极性相反，只需在 IP 末尾取反端口即可兼容。我们在译码表中补了 A-F 的字符编码，方便展示十六进制；同时在扫描时增加了默认全灭状态，防止上电复位期间残影。我们还把扫描周期和位选信号探头导出，方便用逻辑分析仪确认 200 Hz/1 Hz 是否符合预期。

## 2.4 层级透传与约束

在 e203_subsys_perips.v 实例化外设，暴露 my_seg_out/my_cc 作为输出，同时保留原 PWM IOF。随后在 main/top/soc_top/soc_demo 逐层添加并透传 seg_out/cc。约束文件 gowin_prj/e203_basic_chip.cst 直接复用 led_my/src/led_seg_display.cst 的管脚定义，避免重新分配导致踩线。我们在顶层命名时与原有信号保持一致，方便复用约束文件；并在文档中记录了管脚极性，降低后续移植成本。同时，我们为关键 IO 添加 keep 属性防止综合优化，并检查约束中的驱动能力满足数码管电流要求。

## 2.5 固件与延时

firmware/hello_world/src/main.c 中，在 init() 后写 CTRL=0 关闭自动计数、写 DATA=0000 清零。主循环每秒执行：bcd_encode_4digits(value) 生成 BCD、写 DATA、printf 当前值、value++、delay_ms(1000)。延时使用 CLINT mtime 计算 ticks，避免 usleep 链接问题。基地址/偏移定义在 platform.h、my_periph.h 与硬件保持一致。我们增加了简单的溢出处理：计数到 9999 后回卷到 0000，保证显示稳定；同时保留了切换自动计数的入口，便于课堂演示“软关硬开”两种模式。我们还写了 write_digit_once() 便于调试器单步写任意值，并在串口输出中打印当前模式和寄存器值，方便现场讲解。

## 2.6 构建与下载

在 firmware/hello_world/Debug 执行 make clean && make 生成 e203_my_periph_demo.elf/.bin 和 ram.hex；RTL 侧 rtl/core/e203_itcm_ram.v 用 $readmemh("../firmware/hello_world/Debug/ram.hex", mem_r) 加载程序。Gowin 工程中确认 hex 更新后再执行综合、P&R、下载 bitstream。我们在 README 中标记了常见路径坑：确保从 rtl 目录运行仿真/加载脚本时 hex 路径正确，避免“找不到程序”导致上电无显示；记录了“先关串口再复位再开串口”的顺序，避免复位时串口占用导致下载失败。

# 三、上板验证

## 3.1 接线与下载

- 约束：段选/位选直接采用 led_my cst 引脚，保持高电平有效；我们下载前逐一用万用表确认管脚与数码管位的对应关系，并在 cst 注释中标明。
- 下载：在 Gowin IDE 中重新导入 ram.hex，生成 bitstream 并烧录到开发板；每次固件改动后我们都重新生成 hex 再综合，确保逻辑与程序一致，并在烧录后按键复位确认程序启动。
- 供电与串口：USB 供电，串口 115200 8N1；上电后优先看串口“init”打印确认 CPU 运行，再观察数码管，保证软硬同步。

### 3.2 观测现象

- 上电后数码管从 0000 开始每秒递增，亮度均匀，无可见闪烁，说明 200 Hz 扫描生效；我们在暗光环境下观察也未出现鬼影，验证了占空比合适。
- 串口输出与数码管同步递增，软件和硬件显示一致；我们用串口日志与手机拍摄的数码管画面对齐，验证显示同步。
- 向 CTRL 写 1（恢复自动计数）后，即使固件暂停写 DATA，数码管仍按 1 Hz 自增；再写 0 能切回软件主导，课堂演示时可现场切换模式。
- 写入任意 BCD（如 1234、9999）可立即显示，验证数据通路与译码正确；当写入超过 9999 时，我们的固件会回卷，数码管不再出现乱码。
- 若将位选/段选极性接反，数码管全灭；在 IP 末尾取反即可恢复，验证极性兼容思路；我们在笔记中记录了“极性反向一次解决”的经验。

### 3.3 调试步骤（保留以供复现）

- 确认 cst 与板卡丝印一致，重点核对共阴/共阳极性；我们对照原理图确认段选共阴，并在文档标注“高电平有效”。
- 下载前确保 ram.hex 时间戳最新；必要时 make clean 后重新生成，防止 IDE 复用旧文件。
- 烧录后打开串口，观察首条打印判断固件是否运行；若缺失打印，我们会复查时钟/复位，并检查 ITCM 载入的 hex 路径。
- 使用内置寄存器访问脚本（JTAG 或调试器）向 0x1001_4000 写 0/1，验证自动计数切换；在课堂演示中我们现场写寄存器让观众看到模式切换。

## 四、问题与解决

- usleep 链接失败：改用 CLINT 计时封装 delay_ms，在 18 MHz 下验证 1 s 误差小于 1%。
- 位选极性不确定：默认高电平有效，如观察到全灭/全亮，在 IP 输出处整体取反即可；我们在代码中留下注释提醒这一调节点。
- ram.hex 未更新导致显示不动：在 Gowin 工程中重新导入最新 hex，再综合下载；我们附了“检查 hex 时间戳”的操作小贴士，并在 Makefile 中清理旧 hex。
- 串口无输出：确认 BSP 时钟 18 MHz 设置正确并检查波特率；必要时减小优化等级避免 printf 被裁剪，并在链接脚本中保留 UART 初始化。

## 五、总结与展望

- 已完成：七段外设 RTL、寄存器接口、固件驱动、上板计数与手动写数验证；形成了一套从固件生成 hex 到导入 Gowin 的稳定流程，并沉淀了“快速自查清单”。
- 下一步：加入溢出中断或状态位；支持小数点/滚动显示/全亮全灭测试模式；可配置扫描频率与占空比以平衡亮度与功耗；编写自动化脚本覆盖寄存器读写与极性配置；将 7-seg IP 抽象成可复用模块移植到更多 RISC-V/FPGA 项目；尝试在 FreeRTOS 下做任务化示例，验证多任务对显示的影响；评估在扫描中插入简短熄灭时间以进一步抑制重影。
- 分工：2352396 禹尧珅主要负责挂载到e203的ICB总线上以及仿真测试，2351283吴凯主要负责引脚绑定以及显示逻辑的设计。

## 附录

### A1 寄存器映射

| 绝对地址 | 偏移 | 名称 | 位域 | 读写 | 复位 | 说明 |
| --- | --- | --- | --- | --- | --- | --- |
| 0x1001_4000 | 0x00 | CTRL | bit0: count_en | R/W | 0x1 | 1=1 Hz 自动计数，0=软件控制 |
| 0x1001_4004 | 0x04 | DATA | [15:12]=d3 [11:8]=d2 [7:4]=d1 [3:0]=d0 | R/W | 0x0 | BCD 数字 |

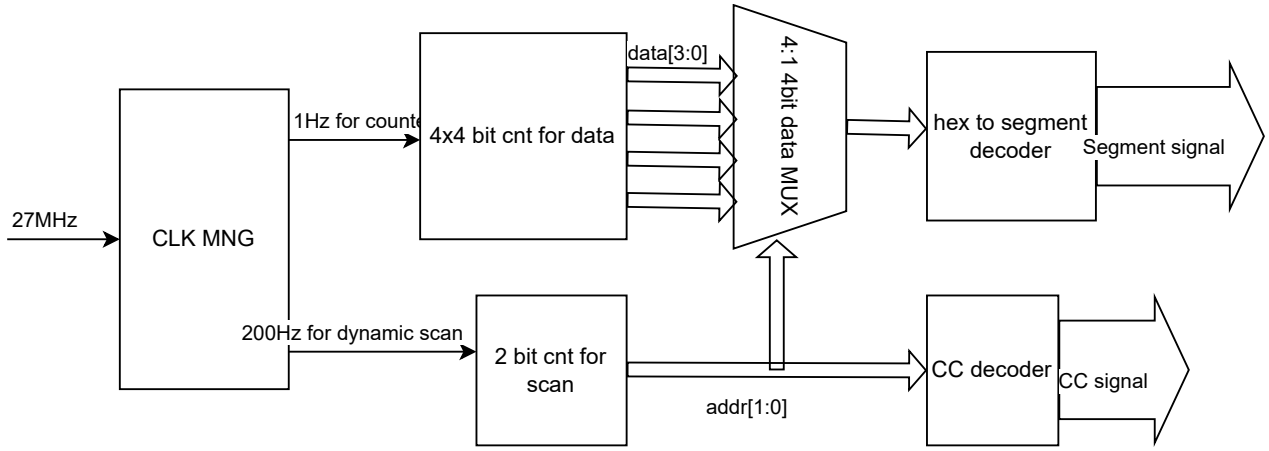
## A2 构建与下载命令

```
cd firmware/hello_world/Debug
make clean && make
```

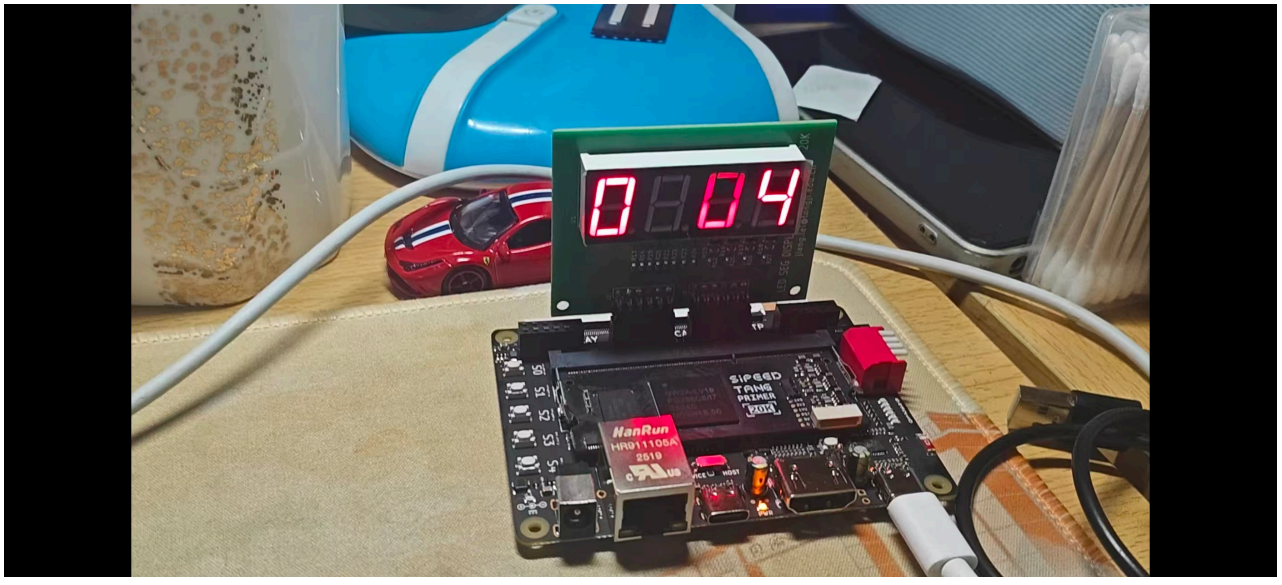
然后Gowin 工程重新导入 ram.hex 后综合、布局布线并下载 bitstream

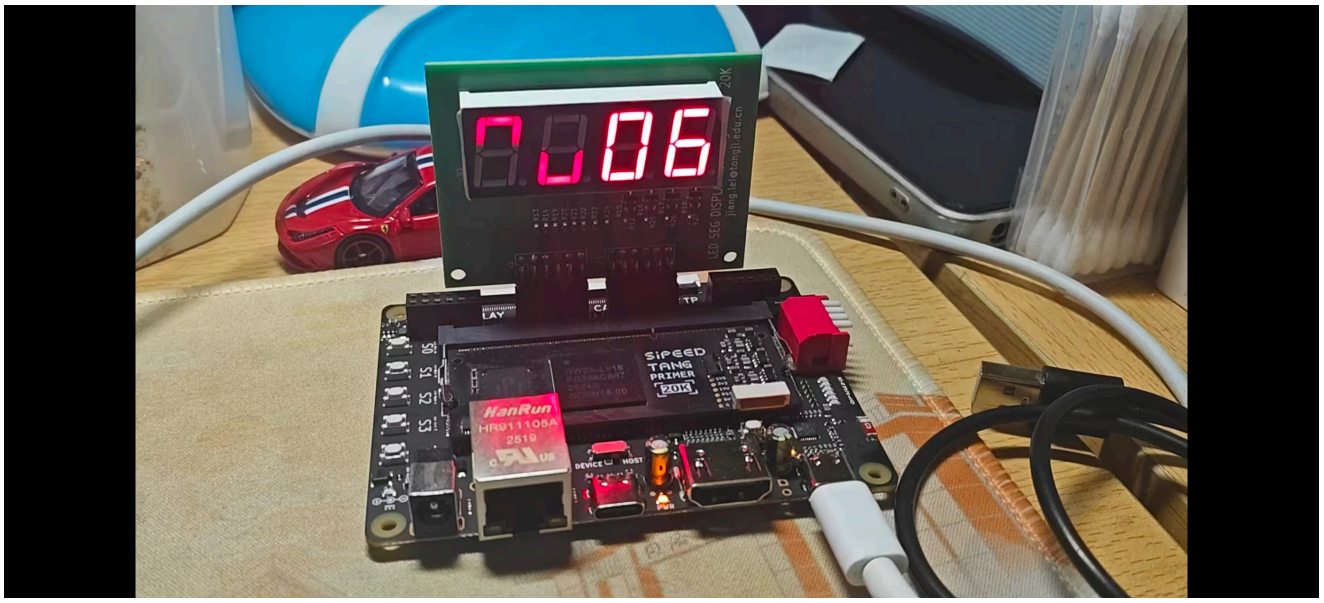
## A3 实验过程图片与视频

- 系统框图：

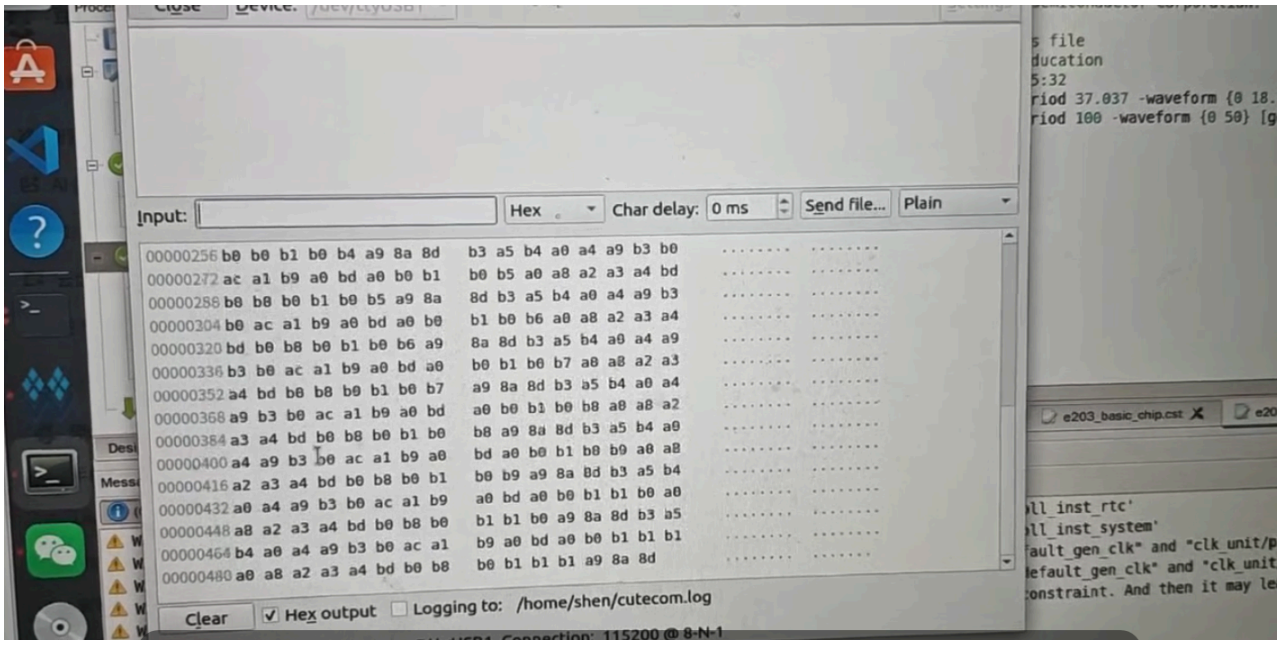



- 上板照片：
（注：此处由于手机拍摄有频闪，显示不完全，但是肉眼可见的是完全的）






- UART演示图：




- 上版视频：已上传至百度网盘，链接为：
  （1）完整功能演示： https://pan.baidu.com/s/1A4V4LKAUiEN-B2Xkdip_lg?pwd=64sn
  （2）UART测试演示： https://pan.baidu.com/s/17By8YpW3NMpOSSXXEpd7mg?pwd=rmfk
- 完整项目已上传至github，链接为： https://github.com/yys806/e203-RISCV-counter.git