

基于 F28335 的 RIU 远程接口单元软件设计与仿真课程设计报告

姓名： 学号：

摘要

远程接口单元（RIU）是嵌入式工业控制系统的核心外设单元，主要承担设备身份识别、状态自检、模拟量信号采集、指令接收执行、数据上传及故障保护等核心功能，广泛应用于工业测控、电力电控、运动控制等嵌入式场景。本文以 TI TMS320F28335 DSP 芯片为硬件载体，依托 CCS5.5 开发平台，按照软件工程全流程，依次完成项目策划、需求分析、软件设计、编码开发、软件测试与项目总结，最终实现 RIU 远程接口单元软件系统的模块化开发与仿真环境全功能验证。

本系统采用分层模块化软件架构，将程序划分为系统驱动层、硬件抽象层、应用业务层三层解耦结构，实现机位身份识别、上电 BIT 自检、双通道模拟量采集、0~10V 模拟量输出控制、总线数据交互、心跳超时故障保护六大核心功能。针对 DSP 软件仿真器硬件模拟缺失的问题，完成仿真环境适配改造，规避了 PLL 锁相环死循环、外设寄存器访问异常、数据缓冲区覆盖清零等典型工程问题。通过多场景功能测试，验证了系统逻辑的正确性、稳定性与可靠性，所有功能均满足课程设计指标要求。

关键词：F28335 DSP；RIU 远程接口单元；模块化设计；嵌入式测控；软件仿真；全流程开发

第一章 项目策划

1.1 项目研究背景与意义

随着工业嵌入式测控技术的高速发展，分布式测控系统凭借可靠性高、扩展性强、布线简洁的优势，成为工业控制领域的主流架构。远程接口单元（RIU）作为分布式测控系统的终端核心设备，主要负责现场模拟信号采集、上层控制指令解析执行、设备状态自检与数据上传，是连接主控单元与现场执行机构的关键枢纽。

传统 RIU 设备软件存在耦合度高、功能单一、无故障保护、可移植性差等问题，难以适配复杂工业工况。基于 DSP 的嵌入式测控软件具备运算速度快、实时性强、稳定性高的特点，能够精准满足工业测控的周期调度、信号换算、故障防护需求。

本次课程设计基于 TMS320F28335 浮点 DSP 芯片，自主设计一套完整的 RIU 测控软件系统，覆盖嵌入式软件开发的需求分析、架构设计、模块化编码、调试排错、仿真验证全流程。通过本项目设计，可深入掌握 DSP 嵌入式软件开发规范、分层模块化设计思想、工业测控逻辑设计方法，具备解决嵌入式开发中硬件适配、数据运算、逻辑冲突、异常保护等工程问题的能力，具备重要的理论学习与工程实践意义。

1.2 国内外研究现状

国外嵌入式测控系统发展成熟，TI、ADI 等厂商提供了完善的 DSP 开发库与硬件驱动框架，工业级 RIU 设备普遍采用标准化分层架构，具备完善的故障检测、超时保护、数据校验机制，系统稳定性与实时性极强。国内工业测控领域快速发展，国产化 RIU 设备逐步普及，但多数高校教学项目仅聚焦单一功能开发，缺乏完整的系统化设计，与工业实际工程应用存在差距。

本次课程设计对标工业级 RIU 软件设计标准，完整复现工业设备的核心业务逻辑与安全保护机制，弥补了教学项目功能碎片化的短板，贴合实际工程开发流程。

1.3 主要研究内容与工作安排

1.3.1 主要研究内容

1. 完成 RIU 远程接口单元软件功能需求、性能需求、安全需求的梳理与分析；
2. 设计分层模块化软件总体架构，规划系统运行流程与任务调度机制；
3. 完成机位识别、上电自检、模拟量采集输出、总线通信、心跳保护等核心模块详细设计与编码实现；
4. 适配 F28335 软件仿真环境，解决仿真器硬件适配兼容问题；
5. 完成全场景功能测试、性能验证、问题调试与系统优化；

6. 整理设计资料，完成课程设计总结与报告撰写。

1.3.2 工作安排

第一阶段：需求调研与分析，明确系统功能与技术指标；

第二阶段：总体架构设计、模块划分与流程设计；

第三阶段：分模块编码开发、功能实现与基础调试；

第四阶段：仿真环境适配、工程问题排查与系统优化；

第五阶段：全场景功能测试、数据记录与结果分析；

第六阶段：总结梳理，完成课程设计报告撰写。

1.4 报告整体结构

本文共分为六个核心章节，依次为绪论、系统需求分析、系统总体设计、系统详细设计与编码实现、系统调试与功能验证、总结与展望，完整阐述 RIU 软件系统从设计到落地的全流程。

第二章 需求分析

2.1 功能性需求

结合工业 RIU 设备工作机制与课程设计要求，本系统需实现六大核心功能，覆盖设备启动、运行、交互、保护全流程：

1. 机位身份识别功能：设备上电后自动读取 GPIO 编码引脚高 16 位数据，匹配 4 组合法设备机位编码（1 号、2 号、3 号、4 号机位）；若读取到非法编码，持续循环重试识别，不进入后续业务流程，确保设备身份合法有效。

2. 上电 BIT 自检功能：系统初始化完成后，自动执行 CPU 内核自检与 RAM 存储器自检，检测硬件核心模块工作状态，输出自检结果，无故障则正常运行，存在故障则标记故障状态并上报。

3. 总线数据交互功能：实现异步总线数据接收与上报功能，接收上层主控下发的 CMD 控制指令包，打包本地设备状态、模拟量数据生成 DATA1 上报包，完成上下行数据交互。

4. 双通道模拟量采集功能：支持两路 16 位 ADC 模拟量采样，按照固定换算公式将原始采样值转换为实际电压值，放大 100 倍后以整数形式拼接打包，存入上报数据包完成数据上传。

5. 模拟量输出控制功能：解析 CMD 指令包中的有效标志位、输出使能位、目标电压值，实现模拟量电压输出控制；具备 0~10V 限幅保护机制，超量程指令自动限幅，无效指令、关闭使能指令可精准响应。

6. 心跳超时保护功能：通过心跳字迭代更新判定通信状态，连续 12 个周期（180ms）无心跳更新判定为总线通信故障，锁定输出控制功能，拒绝响应新指令，通信恢复后自动解除锁定，保障设备运行安全。

2.2 非功能性需求

1. 实时性需求：系统以 15ms 为固定周期循环执行业务任务，保障信号采集、指令响应、数据上报的实时性，满足工业测控时序要求。

2. 可靠性需求：具备指令有效性校验、电压限幅保护、通信超时保护、硬件故障自检机制，规避异常工况导致的设备误动作。

3. 可移植性需求：采用软硬件分层解耦设计，业务逻辑与硬件驱动分离，仿真环境与真实硬件环境可无缝切换，代码复用性、移植性强。

4. 可维护性需求：模块化拆分清晰，函数功能单一，变量命名规范，代码注释完整，便于后续调试、优化与功能拓展。

2.3 核心技术指标

1. 系统调度周期：15ms/周期；
2. 通信超时阈值：连续 12 个周期（180ms）无心跳更新触发保护；
3. 模拟量输出量程：0~10V，超 10V 自动限幅，无负电压输出；
4. ADC 采样精度：16 位分辨率，电压换算公式固定；
5. 支持机位数量：4 种合法机位，兼容非法机位识别容错；
6. 工作模式：支持软件仿真模式、真实硬件模式双模式适配。

2.4 功能逻辑约束说明

1. 机位识别仅在上电初始化阶段执行一次，识别失败阻塞后续流程，识别成功方可进入自检与周期任务；

2. 心跳无更新、指令无效时，系统不执行模拟量输出更新，保持原有输出状态；

3. 模拟量采集不受通信状态影响，周期任务正常执行，持续上报现场信号状

态；

4. 故障状态实时打包上报，硬件自检故障优先级高于常规业务指令。

第三章 软件设计

3.1 总体设计原则

本次 RIU 软件系统设计严格遵循模块化、分层化、高内聚、低耦合、高可靠、可移植六大设计原则，将复杂业务逻辑拆分为独立功能模块，各模块分工明确、接口统一、互不干扰，既保障系统运行稳定性，又便于调试优化与功能拓展。

本系统采用三层分层架构设计，自上而下依次为应用业务层、硬件抽象层、系统驱动层，实现业务逻辑与硬件驱动的完全解耦，适配仿真与硬件双环境运行。

3.2 软件分层架构设计

本系统采用三层分层架构设计，自上而下依次为应用业务层、硬件抽象层、系统驱动层，实现业务逻辑与硬件驱动的完全解耦，适配仿真与硬件双环境运行。

1. 系统驱动层：底层基础驱动，包含系统时钟初始化、外设时钟配置、中断配置、GPIO 初始化等底层硬件操作。仿真环境下屏蔽硬件初始化代码，规避仿真器不兼容问题；真实硬件环境下正常启用，保障硬件正常工作。

2. 硬件抽象层（HAL）：中间适配层，封装 ADC 采样读取、GPIO 机位编码读取、总线数据收发等硬件相关接口。通过宏定义区分仿真模式与硬件模式，实现一套代码双环境适配，是系统可移植性的核心保障。

3. 应用业务层：上层核心功能层，包含机位识别、上电自检、心跳检测、模拟量采集、模拟量输出、数据组包上报、周期任务调度七大业务模块，完全独立于硬件，逻辑通用、可复用性强。

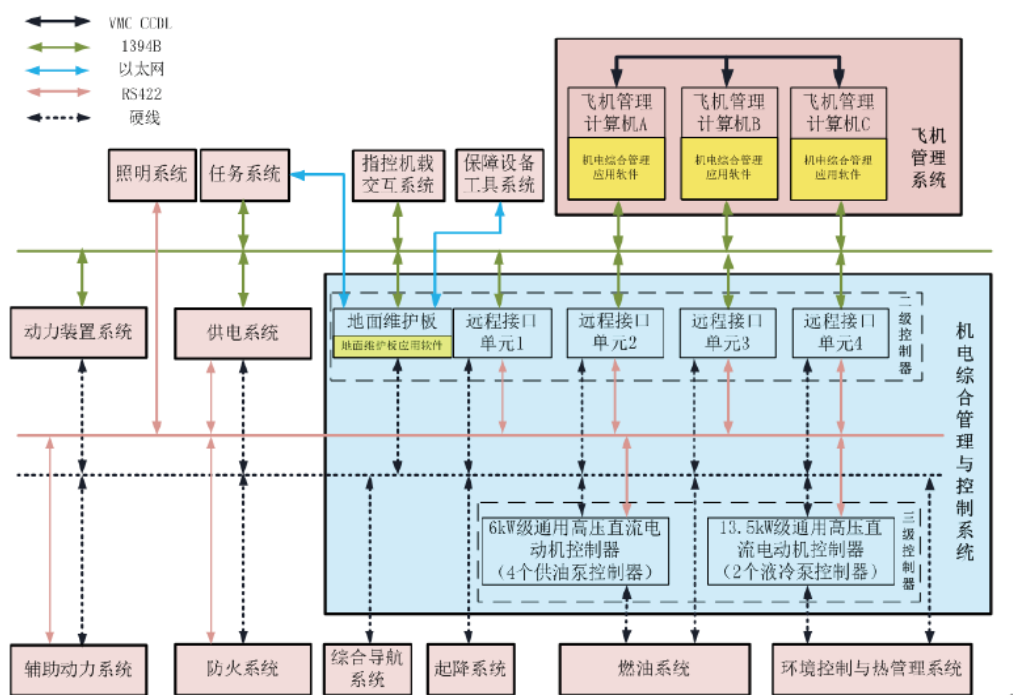


图 1 机电综合管理与控制系统内部交联图

3.3 系统整体运行流程设计

系统上电启动后，严格按照固定时序执行初始化与业务流程，整体运行流程分为初始化阶段与周期调度阶段两大阶段：

第一阶段：系统初始化阶段

1. 系统启动，初始化全局变量、缓冲区、状态标志位；
2. 读取 GPIO 编码，执行机位身份识别，合法机位继续执行，非法机位循环重试；
3. 初始化总线数据缓冲区，清空指令包与上报包缓存；
4. 执行 CPU、RAM 上电自检，检测硬件核心模块状态；
5. 初始化周期任务调度，进入主循环。

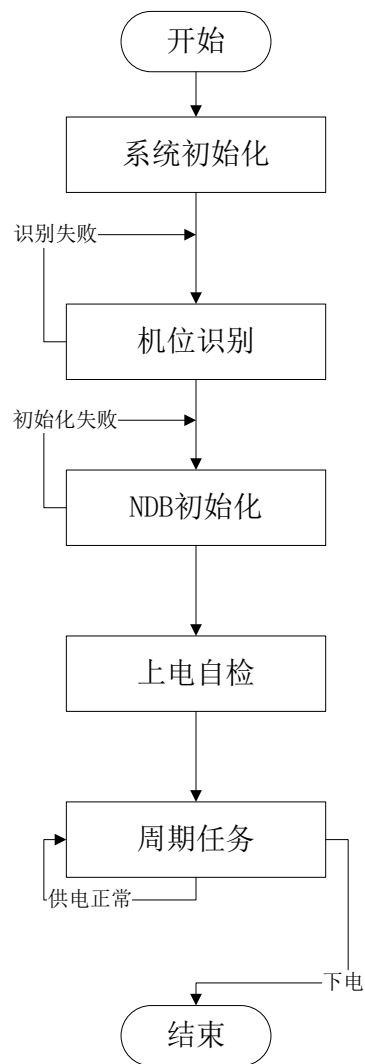


图 2 RIU 管理软件工作流程图

第二阶段：周期调度阶段（15ms 循环）

1. 调用总线接收接口，获取上层 CMD 控制指令；
2. 执行心跳检测，判定通信状态与指令有效性；
3. 有效指令下执行模拟量输出控制，无效/超时指令锁定输出；
4. 采集双通道模拟量信号，完成电压换算与数据打包；
5. 组装设备状态、故障信息、模拟量数据，生成上报数据包；
6. 进入下一轮周期循环，持续往复运行。

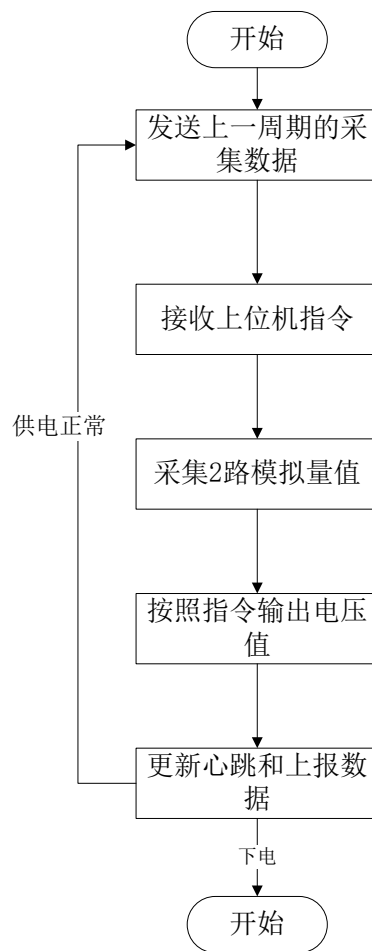


图3 周期任务工作流程

3.4 核心模块设计

根据业务功能拆分七大独立模块，各模块通过全局变量与统一接口交互，无直接耦合：

1. 机位识别模块；2. 上电 BIT 自检模块；3. 总线通信模块；4. 心跳检测与超时保护模块；5. 模拟量采集模块；6. 模拟量输出控制模块；7. 周期任务调度模块。

第四章 软件编程与开发实现

4.1 开发环境与技术栈

开发环境和技术栈如表 1 所示。

表 1 开发环境和技术栈

开发工具	Code Composer Studio 5.5 (CCS5.5)
芯片平台	TMS320F28335 浮点 DSP

仿真工具	F283x CPU 周期精确软件仿真器
编程语言	标准 C 语言（C89 规范，适配 DSP 编译器）
编译规则	无优化编译（O0），保障调试符号完整、变量读取精准

4.2 各模块详细设计与核心代码

4.2.1 机位识别模块

模块功能：读取模拟 GPIO 编码变量高 16 位数据，匹配预设合法机位编码，输出对应的机位编号，非法编码持续重试。

设计逻辑：机位编码存储于 32 位 GPIO 变量高 16 位，通过位运算提取有效编码，遍历合法编码数组，匹配成功赋值机位号，匹配失败机位号置 0 并循环阻塞。

4.2.2 上电 BIT 自检模块

模块功能：完成 CPU 内核运算自检与 RAM 存储器读写自检，检测硬件核心模块故障，输出故障状态标志位。

设计逻辑：CPU 自检通过固定数值运算校验内核计算能力；RAM 自检通过指定内存段写入、读取、比对校验存储功能，自检无故障结果置 0，故障置 1。

4.2.3 总线通信模块

模块功能：实现异步总线指令接收与数据打包上报，仿真环境下通过模拟缓冲区模拟总线数据交互，真实硬件下对接物理总线。

核心适配优化：原始仿真总线接收函数会清空覆盖预设指令数据，导致指令归零失效。优化后保留模拟缓冲区数据，不再强制清零，保障手动预设指令有效，贴合真实总线数据覆盖逻辑。

总线通信模块代码
<pre>// 接收 CMD 数据包（仿真适配优化版） UINT32 receiveAsyStream(UINT32 nodeNum, UINT32 msgID, PUINT32 pdataBuf) { if(pdataBuf == 0) return 1; // 保留模拟数据，不强制清零，适配仿真调试 memcpy(pdataBuf, sim_cmd_buf, sizeof(sim_cmd_buf)); return 0; }</pre>

4.2.4 心跳检测与超时保护模块

模块功能：检测总线心跳字更新状态，判定指令有效性与通信故障，实现超时锁定保护。

设计逻辑：静态变量记录上一周期心跳值，对比当前心跳值，数值更新则重置超时计数器、判定指令有效；数值不变则计数器累加，累计 12 次判定通信超时，锁定输出功能。

心跳检测与超时保护模块代码

```
unsigned int HeartbeatCheck(void)
{
    static unsigned char last_heartbeat = 0;
    static unsigned char timeout_cnt = 0;

    unsigned char curr_beat = g_cmd_buf[8];
    if(curr_beat != last_heartbeat)
    {
        last_heartbeat = curr_beat;
        timeout_cnt = 0;
        return 1;    // 心跳更新，指令有效
    }
    else
    {
        timeout_cnt++;
        if(timeout_cnt >= 12)
            return 0; // 超时，指令无效
        return 1;
    }
}
```

4.2.5 模拟量采集模块

模块功能：读取双通道 ADC 原始采样值，通过标准公式换算实际电压，放大 100 倍后拼接为 32 位数据存入上报包。

换算公式：电压 = 原始采样值 × 20.0 / 65535.0 / 0.8

核心 BUG 优化：原始代码采用 16 位 short 类型移位，导致高位数据溢出清零。优化为 32 位 unsigned long 类型先移位、后拼接，彻底解决高位数据丢失问题。

模拟量采集模块代码

```
static void AnalogCollect(void)
{
    unsigned short raw1, raw2;
    float volt1, volt2;

    #if SIMULATOR_MODE
        raw1 = sim_adc_ch1;
        raw2 = sim_adc_ch2;
    #endif

    // 标准电压换算公式
    volt1 = raw1 * 20.0f / 65535.0f / 0.8f;
    volt2 = raw2 * 20.0f / 65535.0f / 0.8f;

    // 32 位类型移位，避免 16 位溢出，高低位拼接
    g_data1_buf[77] = (((unsigned long)(volt1 * 100)) << 16)
        | ((unsigned short)(volt2 * 100));
}
```

4.2.6 模拟量输出控制模块

模块功能：解析 CMD 指令字，判定指令有效性、输出使能状态，执行电压输出与 10V 限幅保护。

指令解析规则：bit31 为指令有效位、bit30 为输出使能位、低 8 位为目标电压值；电压区间限制 0~10V，超量程自动限幅。

4.2.7 周期任务调度模块

模块功能：统一调度所有业务模块，按照固定时序循环执行，保障系统业务逻辑有序运行。

执行时序：总线指令接收→心跳有效性检测→模拟量输出控制→模拟量信号采集→数据组包上报。

周期任务调度模块代码

```
void CycleTask_Run(void)
{
    unsigned int cmd_id, data1_id;
    unsigned int ret;

    GetMsgID(&cmd_id, &data1_id);
}
```

```
// 1. 接收 CMD 数据包
ret = receiveAsyStream(0, cmd_id, g_cmd_buf);
if(ret != 0)
{
    return;
}

// 2. 心跳检测，无效数据不响应指令
if(HeartbeatCheck() == 1)
{
    // 3. 有效新指令，执行模拟量输出控制
    AnalogOutput();
}

// 4. 模拟量采集（不受通信状态影响，持续采集上报）
AnalogCollect();

// 5. 组包并上报 DATA1
Data1PackAndSend();
}
```

4.3 仿真环境适配改造

F28335 软件仿真器仅模拟 CPU 内核指令执行，不支持 PLL 锁相环、外设寄存器、时钟硬件等模拟，直接运行原生硬件代码会出现死循环、程序跑飞、内存报错等问题。本次设计针对性完成三项仿真适配改造，无侵入修改、不影响真实硬件运行：

1. 屏蔽 PLL 锁相环等待循环，将硬件锁等待改为空循环，规避仿真死循环问题；
2. 注释外设时钟、GPIO、中断等硬件初始化代码，避免访问无效硬件地址导致程序复位跑飞；
3. 优化总线接收函数，保留仿真预设指令数据，解决指令周期性清零问题。

第五章 软件测试

5.1 测试环境与测试方案

测试环境：CCS5.5 + F28335 软件仿真器，无代码优化，断点单步调试，变量实时观测。

测试方案：针对五大核心功能场景，逐一设计测试用例，记录输入条件、预

期结果、实测结果，验证系统功能完整性与准确性。

5.2 场景 1：机位识别功能测试

测试目的：验证合法机位精准匹配、非法机位阻塞重试功能。

测试用例与结果：

表 2 场景 1 测试用例与结果

GPIO 编码输入	对应机位	预期结果	实测结果	测试结论
581（高 16 位）	1 号机位	g_riu_pos_no=1	g_riu_pos_no=1	通过
582（高 16 位）	2 号机位	g_riu_pos_no=2	g_riu_pos_no=2	通过
71（高 16 位）	3 号机位	g_riu_pos_no=3	g_riu_pos_no=3	通过
586（高 16 位）	4 号机位	g_riu_pos_no=4	g_riu_pos_no=4	通过
0（非法编码）	无	阻塞重试，不往下执行	阻塞重试	通过

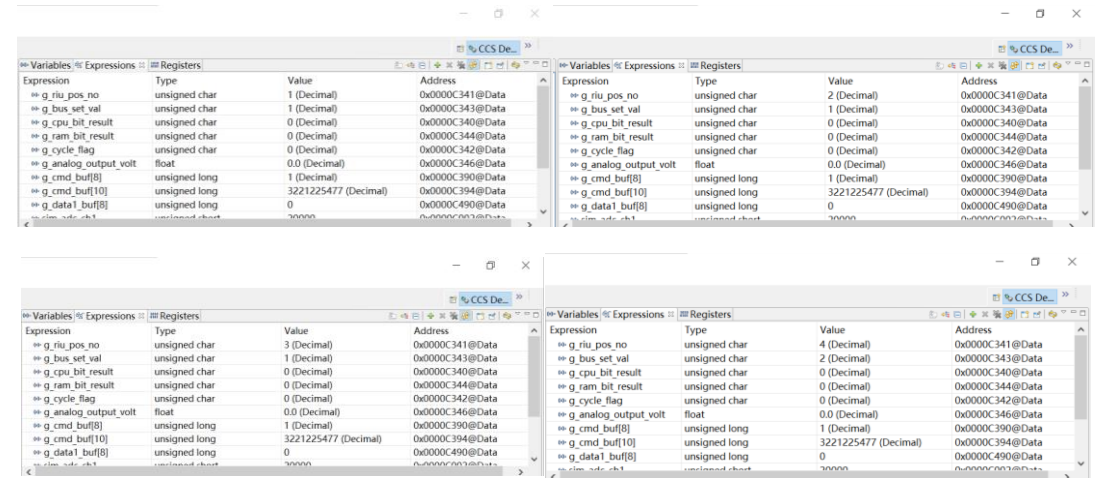


图 4 不同机位识别结果

测试结论：机位识别功能完全符合需求，合法机位精准匹配，非法机位有效容错。

5.3 场景 2：模拟量采集功能测试

测试目的：验证双通道 ADC 采样换算、高低位数据拼接上报功能。

测试输入：sim_adc_ch1=65535（满量程），sim_adc_ch2=32768（半量程）

理论计算：通道 1 电压 25.0V，上报值 2500；通道 2 电压 12.5V，上报值 1250。

实测结果：g_data1_buf[77]高 16 位=2500，低 16 位=1250，与理论值完全一致。

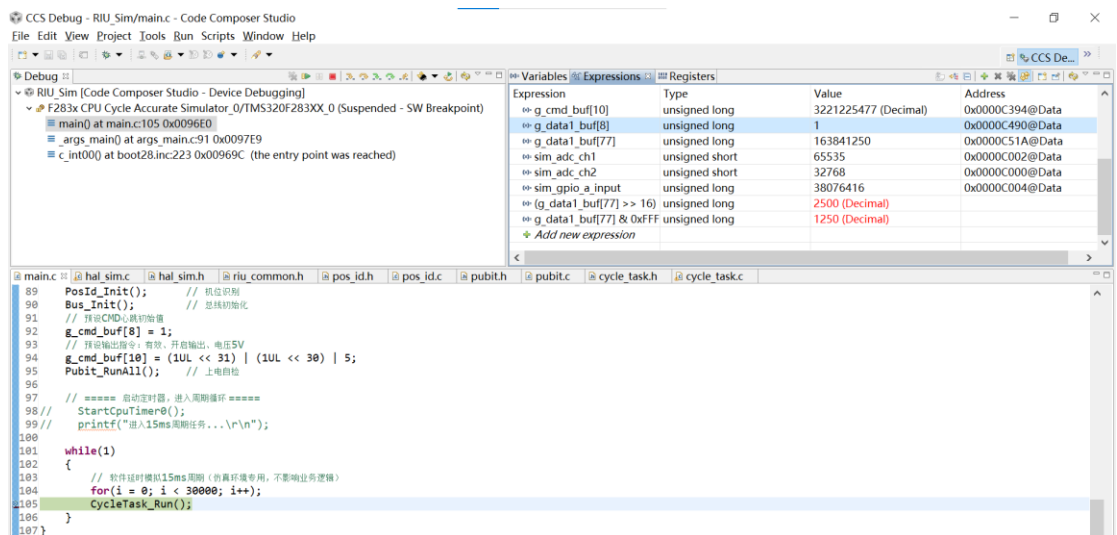


图 5 双通道采集数据高低位拆分数值

测试结论：模拟量采集换算、数据拼接功能正常，精度符合设计要求。

5.4 场景 3：模拟量输出控制测试

测试目的：验证正常输出、超压限幅、关闭输出、无效指令过滤四大功能。

表 3 场景 3 测试用例与结果

测试场景	指令配置	预期输出	实测输出	结论
正常 5V 输出	有效+使能+5V	5.0V	5.0V	通过
12V 超压限幅	有效+使能+12V	10.0V（限幅）	10.0V	通过
关闭输出	有效+失能+5V	0.0V	0.0V	通过
无效指令	无效+使能+8V	保持原值	保持原值	通过

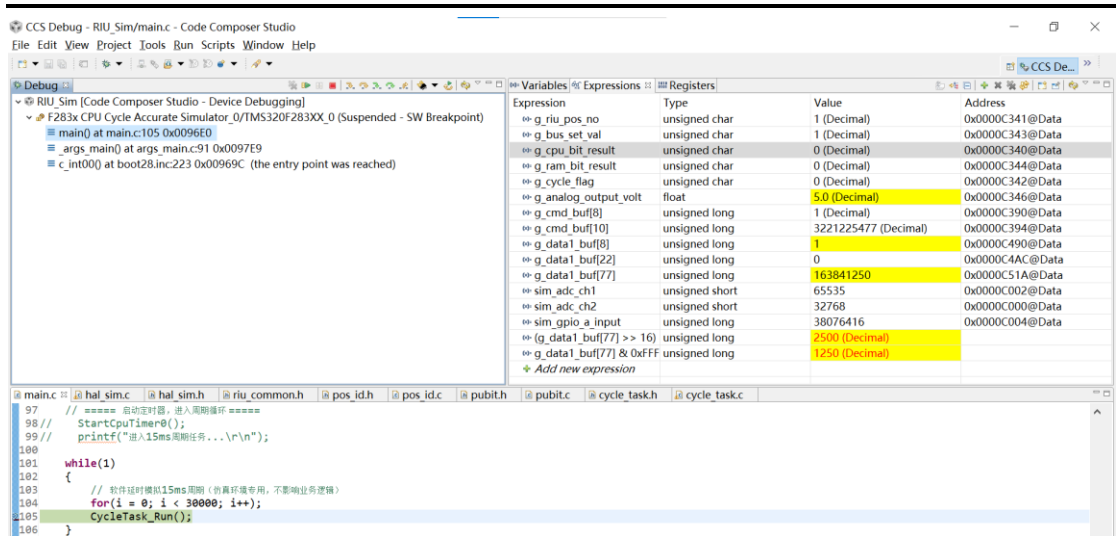


图 6 正常 5V 输出

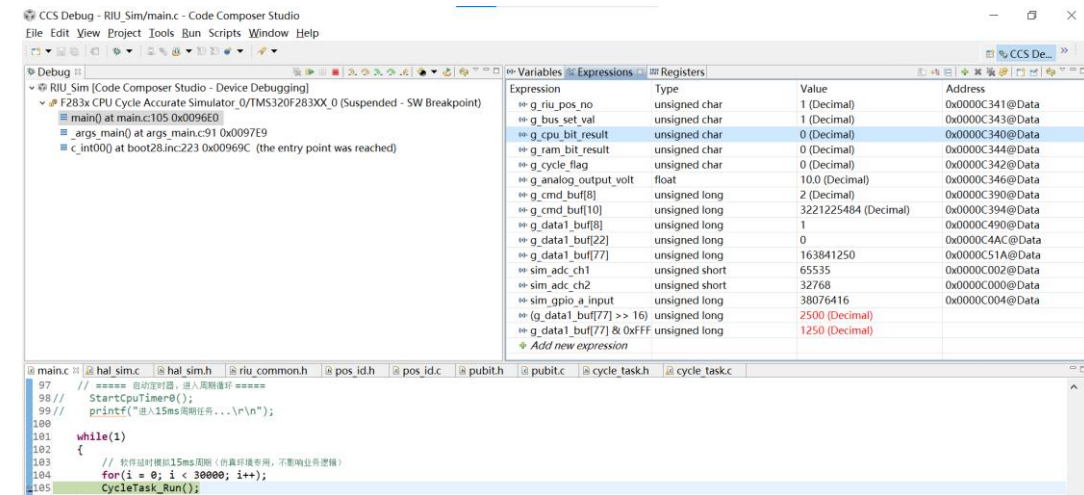


图 7 12V 超压限幅

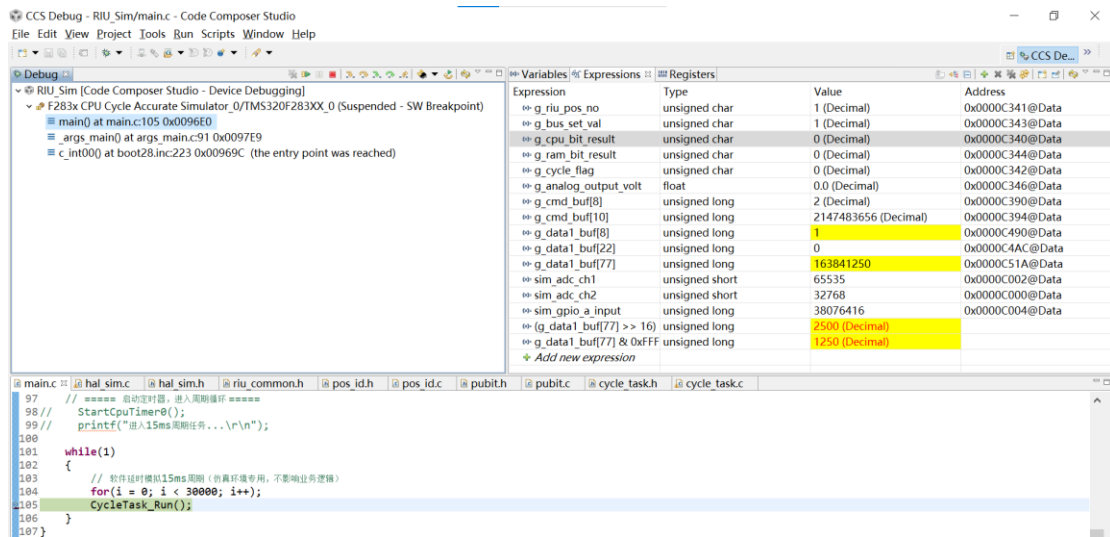


图 8 关闭输出

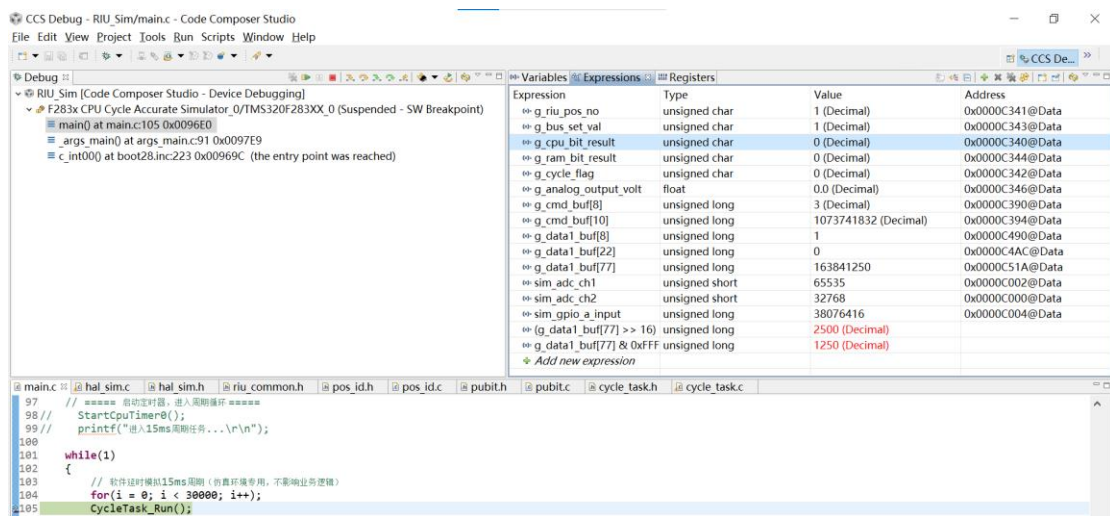


图 9 无效指令

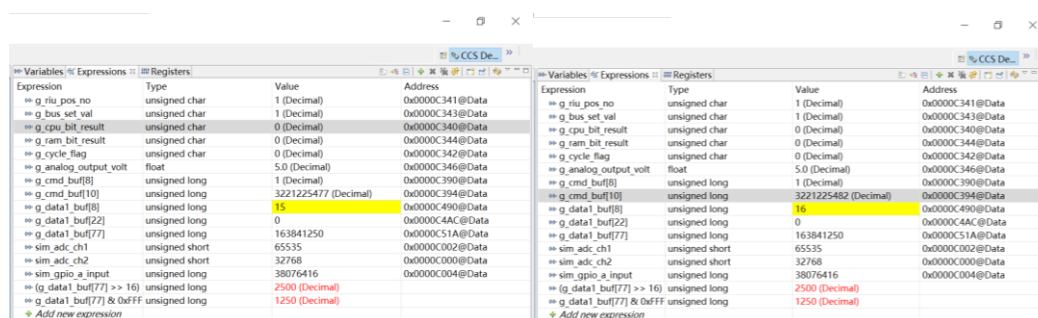
测试结论：模拟量输出控制、限幅保护、指令过滤功能全部正常。

5.5 场景 4：心跳超时保护测试

测试目的：验证 180ms 通信超时锁定、心跳恢复解锁功能。

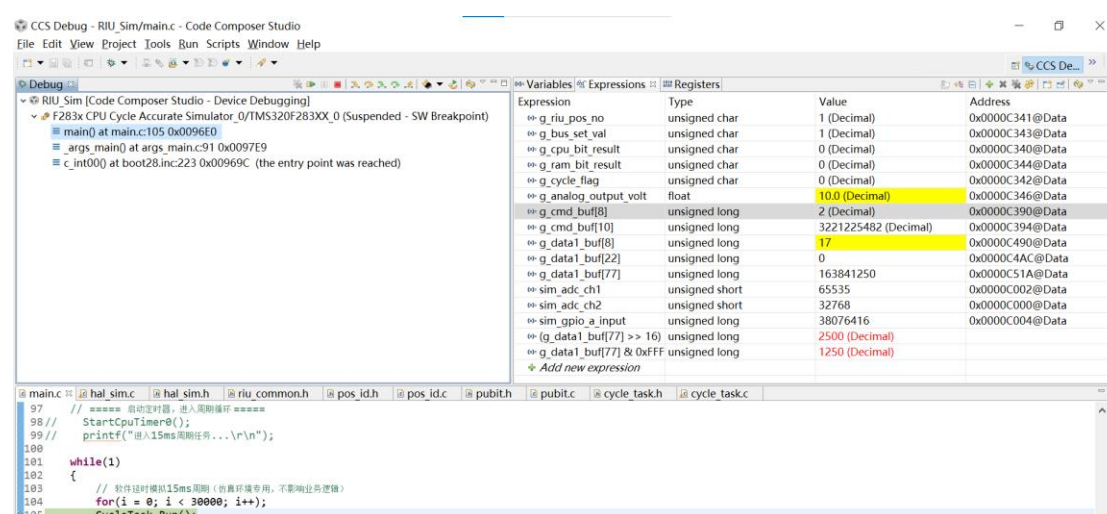
测试过程：固定心跳值不变，连续运行 15 个周期触发超时；下发新 10V 指令，输出保持 5.0V。

测试结果：超时后锁定指令响应，通信恢复后自动解锁，保护逻辑精准有效。



Expression	Type	Value	Address
g_rii_pos_no	unsigned char	1 (Decimal)	0x0000C341@Data
g_bus_set_val	unsigned char	1 (Decimal)	0x0000C343@Data
g_cpu_bit_result	unsigned char	0 (Decimal)	0x0000C340@Data
g_ram_bit_result	unsigned char	0 (Decimal)	0x0000C344@Data
g_cycle_flag	unsigned char	0 (Decimal)	0x0000C342@Data
g_analog_output_volt	float	5.0 (Decimal)	0x0000C346@Data
g_cmd_buf[8]	unsigned long	1 (Decimal)	0x0000C390@Data
g_cmd_buf[10]	unsigned long	3221225477 (Decimal)	0x0000C394@Data
g_data1_buf[8]	unsigned long	15	0x0000C490@Data
g_data1_buf[22]	unsigned long	0	0x0000C4AC@Data
g_data1_buf[77]	unsigned long	163841250	0x0000C51A@Data
sim_adc_ch1	unsigned short	65535	0x0000C002@Data
sim_adc_ch2	unsigned short	32768	0x0000C000@Data
sim_gpio_a_input	unsigned long	38076416	0x0000C004@Data
(g_data1_buf[77] >> 16)	unsigned long	2500 (Decimal)	
g_data1_buf[77] & 0xFF	unsigned long	1250 (Decimal)	

图 10 固定心跳值不变，连续运行 15 个周期触发超时；下发新 10V 指令，输出保持 5.0V 不变



Expression	Type	Value	Address
g_rii_pos_no	unsigned char	1 (Decimal)	0x0000C341@Data
g_bus_set_val	unsigned char	1 (Decimal)	0x0000C343@Data
g_cpu_bit_result	unsigned char	0 (Decimal)	0x0000C340@Data
g_ram_bit_result	unsigned char	0 (Decimal)	0x0000C344@Data
g_cycle_flag	unsigned char	0 (Decimal)	0x0000C342@Data
g_analog_output_volt	float	10.0 (Decimal)	0x0000C346@Data
g_cmd_buf[8]	unsigned long	2 (Decimal)	0x0000C390@Data
g_cmd_buf[10]	unsigned long	3221225482 (Decimal)	0x0000C394@Data
g_data1_buf[8]	unsigned long	17	0x0000C490@Data
g_data1_buf[22]	unsigned long	0	0x0000C4AC@Data
g_data1_buf[77]	unsigned long	163841250	0x0000C51A@Data
sim_adc_ch1	unsigned short	65535	0x0000C002@Data
sim_adc_ch2	unsigned short	32768	0x0000C000@Data
sim_gpio_a_input	unsigned long	38076416	0x0000C004@Data
(g_data1_buf[77] >> 16)	unsigned long	2500 (Decimal)	
g_data1_buf[77] & 0xFF	unsigned long	1250 (Decimal)	

图 11 更新心跳值后，输出立即更新为 10.0V

测试结论：心跳超时保护功能符合设计指标，安全防护机制可靠。

5.6 场景 5：DATA1 数据包上报测试

测试目的：验证上报心跳自增、硬件故障状态上报功能。

测试结果：周期任务每运行一次，g_data1_buf[8]心跳值自动+1；手动模拟 CPU/RAM 故障后，g_data1_buf[22]故障位精准置位，无故障时数值为 0。

Expression	Type	Value	Address
g_rtu_pos_no	unsigned char	1 (Decimal)	0x0000C341@Data
g_bus_set_val	unsigned char	1 (Decimal)	0x0000C343@Data
g_cpu_bit_result	unsigned char	0 (Decimal)	0x0000C340@Data
g_ram_bit_result	unsigned char	0 (Decimal)	0x0000C344@Data
g_cycle_flag	unsigned char	0 (Decimal)	0x0000C342@Data
g_analog_output_volt	float	5.0 (Decimal)	0x0000C346@Data
g_cmd_buf[6]	unsigned long	3221225477 (Decimal)	0x0000C390@Data
g_cmd_buf[10]	unsigned long	3221225477 (Decimal)	0x0000C394@Data
g_data1_buf[8]	unsigned long	1 (Decimal)	0x0000C490@Data
g_data1_buf[22]	unsigned long	0	0x0000C4AC@Data
g_data1_buf[77]	unsigned long	163841250	0x0000C51A@Data
sim_adc_ch1	unsigned short	65535	0x0000C002@Data
sim_adc_ch2	unsigned short	32768	0x0000C000@Data
sim_gpio_a_input	unsigned long	38076416	0x0000C004@Data
(g_data1_buf[77] >> 16)	unsigned long	2500 (Decimal)	
g_data1_buf[77] & 0xFF	unsigned long	1250 (Decimal)	

图 12 周期任务每运行一次，心跳值自动+1

Expression	Type	Value	Address
g_rtu_pos_no	unsigned char	1 (Decimal)	0x0000C341@Data
g_bus_set_val	unsigned char	1 (Decimal)	0x0000C343@Data
g_cpu_bit_result	unsigned char	1 (Decimal)	0x0000C340@Data
g_ram_bit_result	unsigned char	0 (Decimal)	0x0000C344@Data
g_data1_buf[22]	unsigned long	1 (Decimal)	0x0000C4AC@Data

图 13 手动模拟 CPU/RAM 故障后，故障位精准置位，无故障时数值为 0

测试结论：数据上报功能完整，状态反馈精准可靠。

5.7 整体测试总结

本次五大场景全覆盖测试，所有功能均满足设计需求与技术指标，系统逻辑严谨、运行稳定、容错性强，实现了工业级 RIU 设备的核心业务功能与安全保护机制，系统设计完全合格。

第六章 总结与展望

6.1 项目总结

本次课程设计完整完成了基于 F28335 DSP 的 RIU 远程接口单元软件系统设计、编码、适配、调试与仿真验证全流程工作，具体完成内容如下：

1. 完成系统需求分析、架构设计、模块划分与流程设计，搭建分层解耦的嵌入式软件架构；
2. 自主编码实现机位识别、上电自检、总线通信、模拟量采集输出、心跳保护、数据上报全部核心功能；
3. 针对性解决 DSP 软件仿真器适配难题，完成多项工程级 BUG 优化与逻辑完善；
4. 完成全场景功能测试，所有功能、性能指标均达到课程设计要求，系统运行稳定可靠。

6.2 设计收获与心得体会

通过本次课程设计，我系统掌握了 DSP 嵌入式软件开发的完整流程，深刻理解了模块化、分层化的软件设计思想，摒弃了以往碎片化的代码编写思维，建立了系统化的工程开发理念。

在调试过程中，我学会了通过断点单步、变量观测、逻辑推演定位底层问题，熟练掌握了数据类型运算、位运算、时序逻辑、异常保护等核心开发技能。同时深刻认识到软件仿真与真实硬件的差异，理解了硬件适配、代码兼容、容错保护在嵌入式工程中的重要意义。

本次设计不仅提升了我的代码编写与调试能力，更培养了我分析问题、解决问题、优化系统的工程思维，为后续嵌入式系统开发、DSP 项目实战奠定了坚实的基础。

6.3 系统不足与未来展望

6.3.1 系统不足之处：

本次设计基于软件仿真环境验证，采用软件延时模拟周期调度，未使用定时器中断实现精准硬件时序；同时仅完成基础功能验证，未进行极限工况、长期稳定性测试。

6.3.2 未来优化方向：

1. 移植至真实 F28335 硬件平台，启用定时器中断实现精准 15ms 周期调度，替代软件延时，提升系统实时性；
2. 增加数据校验、重传机制，提升总线通信的抗干扰能力；
3. 丰富故障类型，增加过压、欠压、信号异常等故障检测与上报功能；
4. 增加系统日志功能，记录设备运行状态、故障信息，便于设备运维排查。