文章编号:1671-4598(2024)04-0248-09 DOI:10.16526/j.cnki.11-4762/tp.2024.04.035 中图分类号:TP311.52 文献标识码:A

基于 GD32 的网络在线烧写技术研究

宋书龙，葛 露
（中国西南电子技术研究所，成都 610036）

摘要：针对国产微控制器终端设备程序升级需要拆卸设备的问题，提出了基于 GD32 网络在线烧写固件的方案，通过手持控制器实现对终端设备的应用程序软件升级，达到无须仿真器更新固件程序目的；该方案详细论述了在线烧写系统的组成，在应用编程技术的工作原理，以及上位机程序、手持控制器和终端设备模块各功能设计与实现，并给出了各个模块的使用步骤和方法；终端设备采用多分区操作，保证系统在升级过程中即使发生异常，也不会导致产品变砖，充分提高了系统的可维护性与容错性，手持控制器在外场升级携带方便，维护简单、方便、快捷，实验测试表明，在线更新后的固件程序能够正确运行，从而解决外场拆卸产品和挂载仿真器更新程序的困难。

关键词：GD32；IAP；在线升级；手持控制器；以太网

Research on Network Online Updating Technology of Based on GD32

SONG Shulong，GE Lu
（Southwest China Institute of Electronic Technology，Chengdu 610036，China）

Abstract：Aimed at the program upgrading of domestic micro controller terminal equipment needs to be disassembled，a scheme based on GD32 network online updating firmware is proposed，the application software of terminal devices is upgraded using the handheld controller，so as to achieve the purpose of updating the firmware program without the emulator. The scheme discusses in detail the compositions of the online updating system，principles of programming techniques，and design and implementation of various functions of the upper computer program，handheld controller and terminal equipment module，and provides the usage steps and methods for each module. The terminal equipment software adopts the multi-partition operation to ensure that the system remains functional even in the event of an exception during the upgrading process，which greatly enhances the maintainability and fault tolerance of the online updating system. The handheld controller has the advantages of easy to upgrade and carry in the field，simple，convenient，and fast maintenance. Experimental testing results show that the firmware program after online updating is successful，thereby resolving the difficulties of disassembling products in the field and updating the program with the emulator.

Keywords：GD32；in-application programming (IAP)；online update；handheld controller；Ethernet

0 引言

嵌入式设备在实际使用中可能会遇到各种软件问题和用户需求的变化，这就需要对软件进行升级以提供更好的功能和性能，尤其是交付到外场的设备，需要拆卸设备链接仿真器进行软件升级，对于大型复杂的设备，甚至需要多名专业人员到现场，就会导致软件升级过程比较繁琐，增加了维护成本。软件维护人员对嵌入式终端设备的软件进行维护升级，常用的方法是拆卸设备，使用仿真器工具进行烧写，而采用在线更新软件的方式，可以解决不拆设备和挂载仿真器烧写的问题。国外的公司 PHILIPS 的 P89C51RX2xx 系列单片机同时支持两种功能在系统中编程和在应用中编程，PHILIPS 为了使在线更新技术得以推广，还在单片机上提供了免费的 BOOT 固件，使在线烧写功能实现更简单，意法半导体公司的 PSD 系列微控制器的片内 Flash 大容量存储器，也同时支持在系统中编程和在应用中编程。国内的华大半导体有限公司 HC32L136 系列 32 位微控制器支持在线 APP 更新，在芯片上电复位的状态下，BOOT0 脚为高电平时，芯片会进入在线编程模式，通过上位机可以进行在线编程。兆易创新科技集团股份有限公司的 GD32 系列片上 FLASH 高达 3 072 K 字节用于存储指令或数据，支持从系统存储器启动，在应用编程则可由用户在应用中实现。本文采用 GD32F450 系列 ARMCortex-M4 内核的 32 位通用微控制器（MCU，microcontroller unit），处理器主频高达 200 MHz，控制器内部集成了一个以太网外设，支持 100 Mbps 数据传输速率，借助以太网外设，GD32F450 控制器可以通过 ETH 外设按照 IEEE 802.3-2002 和 IEEE 1588-2008 标准发送和接收数据包。文献［1-5］同类的研究大都是采用串口、CAN 等总线在线烧写，本设计是利用在应用编程技术进行免拆设备，在外场通过网络在线快速升级 GD32 国产微控制器固件。

1 烧写技术原理

1.1 仿真器烧写

通过仿真器烧写有两种接口：一种是 JTAG 模式，另

收稿日期:2023-09-06； **修回日期**:2023-10-06。
作者简介:宋书龙(1989-)，男，硕士研究生，工程师。
引用格式:宋书龙，葛 露.基于 GD32 的网络在线烧写技术研究[J].计算机测量与控制，2024，32(4)：248-256.

一种是 SWD 模式，JTAG（Joint Test Action Group）国际标准测试协议被广泛应用于各种基于嵌入式系统的开发、测试和编程任务中。它提供了一种通用接口，可以方便地访问芯片内部电路和功能，可以实现与目标设备的通信，读取和写入芯片内部寄存器或存储器中的内容，以及监控和控制芯片的状态，JTAG 接口通常由 TMS、TCK、TDI 和 TDO 共 4 条线组成，TMS 用于控制时序和模式选择，TCK 线则提供定时脉冲，TDI 和 TDO 分别用于输入和输出数据。SWD（Serial Wire Debug）一种和 JTAG 不同的模式，使用的协议与 JTAG 也不一样，SWD 只需要 4 个引脚，结构简单，需要的硬件接口为：GND、RST、SWDIO 和 SWDCLK，分别为地线、复位、数据线及时钟，调试阶段的终端模块大都使用仿真器仿真调试和烧写，除了仿真器可以烧写以外，还有一些通过总线方式在线烧写[6-8]。

1.2　在线烧写

在嵌入式技术中，在线编程方式主要有两种：IAP（在应用中编程）和 ISP（在系统中编程）。

ISP（In-system Programming）在系统中编程，用于嵌入式系统或芯片程序的更新和烧录，跟传统的更新和烧录方式相比，ISP 允许直接通过系统的通信接口如 SPI、I2C、UART 等对芯片内部进行编程操作，无需拆卸芯片，在系统上对程序进行更新、配置或修改，使用专门的编程器连接到目标设备进行编程操作，只要留出上位机的通信口，使用上位机软件就可以实现对芯片内部存储器的数据、配置文件、参数设置等进行修改，由于 ISP 技术可以对芯片内部直接访问，在系统编程中，需要确保编程操作的准确性和安全性，避免意外写入错误的代码而损坏目标设备[9]。

IAP（In-application Programming）在应用中编程，它与 ISP 相似，但 ISP 通常涉及到整个系统的编程，而 IAP 是在特定应用程序中对固件进行更新，目标设备开机上电，直接运行应用程序完成固件更新，而无需重新启动目标设备，需要把目标设备上的存储器划分为 3 个独立的区域：一个区域作为引导程序，一个区域用于更新程序，最后一个区域用于存储新固件的升级文件，通过对程序存储器的指定段进行读和写操作，从而实现对目标设备上的程序的更新和烧录[10]。

2　系统结构及原理

在线烧写系统主要由 3 部分组成：上位机软件、手持控制器和终端设备，上位机软件主要通过网络把数据发送给手持控制器，手持控制器把数据发送给终端设备，终端设备功能是运行点亮流水灯，系统如图 1 所示。

2.1　系统工作流程

1）上位机软件通过网络，把编译链接生成最终的 BIN 格式文件发送给手持控制器。

2）终端设备上电时候，会进入 BOOT 引导区，此时如果手持控制器没有给终端设备发送“启动烧写”指令，BOOT 引导区的程序等待 10 s 以后，自动跳转到 APP 运行区执行；此时如果手持控制器给终端设备发送“启动烧写”指令，BOOT 引导区的程序收到正确的指令会跳转到 UPDATE 升级区，手持控制器发送需要待烧写的 BIN 文件给终端设备，由 UPDATE 升级区运行的程序把收到的数据烧写到指定 Flash 分区上，烧写完成后自动跳转到 APP 运行区执行。

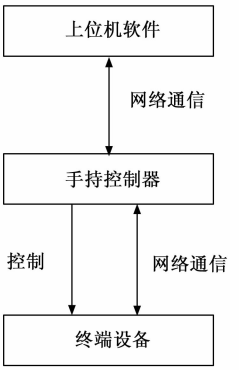

图 1　系统框图

3）终端设备程序在 APP 运行区运行，手持控制器可以给终端设备发送复位命令，此时终端设备程序会进入 BOOT 引导区运行[11-12]。

2.2　系统功能设计

2.2.1　上位机模块设计

如图 2 所示，上位机软件基于 PC 机开发设计，读取本次编译的 BIN 格式文件，通过网络发送给手持控制器，网络收发数据使用 UDP 协议，每次传输数据的大小 1 024 字节，数据传输完成会给手持控制器发送传输完成标志。

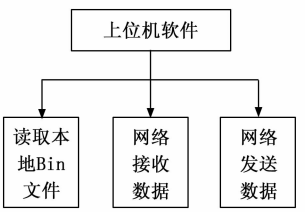

图 2　上位机模块图

2.2.2　手持控制器模块设计

手持控制器的片上 Flash 不需要分区，程序运行于片上 Flash 的开始区域 0x08000000，收到数据存储在片外 Flash 上，外挂的 Flash 存储容量 512 MB。

手持控制器由自检、数据读取、启动烧写及数据注入等命令组成，手持控制器发送自检命令给终端设备模块，终端设备模块收到命令将自检结果发送给手持控制器，如果手持控制器未收到自检数据，会重传自检命令给终端设备，重传 3 次以后仍未收到数据或收到数据异常，都会显示自检超时，只有收到正确的自检数据，会把自检结果信息显示，通过自检命令判断手持控制器与终端设备网络链接是否正常[13-14]。

如图 3 所示，手持控制器会主动发送数据读取命令给上位机软件，接收到上位机软件发送的数据会写入到外挂的片外 Flash 进行存储，当手持控制器会发送数据注入命令给终端设备模块时，会从外挂的片外 Flash 中读取数据，发送给终端设备模块。

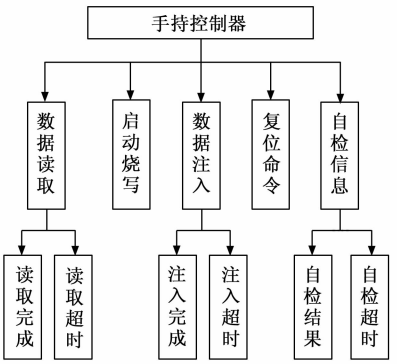

图 3 手持控制器模块图

2.2.3 终端设备模块设计

终端设备模块的程序分区运行，防止在更新程序时候失败，导致应用程序无法执行，终端设备模块分为 3 个区加载程序，分别是 BOOT 区、UPDATE 区和 APP 应用区，如图 4 所示。

1）BOOT 区：终端模块设备开机上电 BOOT 区的程序先加载运行，根据接收到的网络命令进行程序跳转。

2）UPDATE 区：固件程序更新区，主要是更新程序，下载完成新固件后，自动跳转到 APP 应用区运行。

3）APP 应用区：主要用来实现所需要的业务的程序。

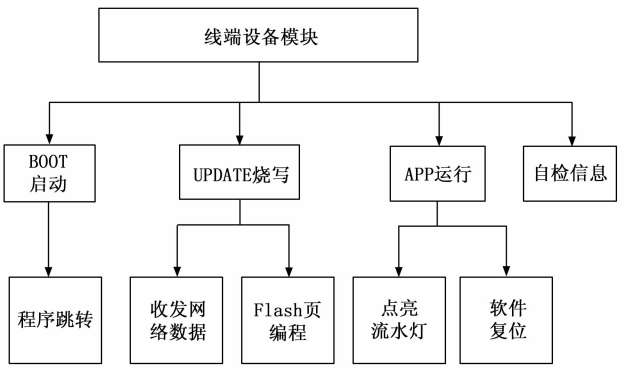

图 4 终端设备模块图

终端设备模块开机上电时候，程序会在 BOOT 区运行，首先会初始化网络，此时手持控制器发送“启动烧写”命令，终端设备模块 BOOT 区的程序就会跳转到 UPDATE 区域去运行，手持控制器发送“数据注入”命令，会把待烧写文件通过网络发送给终端设备模块，终端设备模块收到正确数据后会把数据写入到片上 Flash，数据写入完成后，终端设备模块的程序会跳转到 APP 运行区去运行，就会看到流水灯点亮，此时手持控制器发送复位命令，终端设备模块就会进行软件复位，程序会从 BOOT 区重新运行，APP 程序为用户功能代码，执行用户功能操作。

2.3 终端设备模块 Flash 分区设计

GD32F450ZI 上使用了两片闪存，前 1 024 kB 容量在第 0 片闪存中，后续的容量在第 1 片闪存中，主存储闪存总共容量为 3 072 kB，包含 8 个 16 kB 的扇区、2 个 64 kB 的扇区、14 个 128 kB 的扇区、4 个 256 kB 的扇区[15]。主存储闪存的每个扇区都可以单独擦除，支持 32 位整字或 16 位半字、字节编程，4 kB 页擦除，扇区擦除和整片擦除操作，Bank0 的总大小为 1 024 kB，闪存的地址范围为 0x08000000-0x080FFFFF。根据固件在线升级的需要，将其划分为 3 个作用不同的区域，具体划分如表 1 所示。

表 1 终端设备 Flash 地址区域划分

地址范围	分区大小/kB	分区名字	分区作用
0x08000000～0x0802FFFF	192	BOOT 引导区	根据指令跳转不同分区
0x08030000～0x0805FFFF	192	UPDATE 升级区	烧写应用程序
0x08060000～0x080DFFFF	512	APP 运行区	运行应用程序

终端设备模块在系统上电后或复位后立即进入 BOOT 引导区执行，即系统上电或复位后，软件从地址为 0x8000000 处执行，因此在 MDK 上需设置 BOOT 引导区片上 Flash 起始地址为 0x08000000，BOOT 引导区生成的 BIN 文件大小为 52 kB 设置分区大小为 192 kB，同样原理，UPDATE 升级区生成的 BIN 文件大小 100 kB，因此在 MDK 上需要设置 UPDATE 升级区片上 Flash 起始地址为 0x08030000，APP 运行区生成的 BIN 文件大小 150 kB，在 MDK 上需要设置 APP 运行区片上 Flash 起始地址为 0x08060000，BOOT 引导区、UPDATE 升级区和 APP 运行区的存储区域相互独立，只存在简单的跳转关系[16]。

终端模块 BOOT 引导区的程序必须通过仿真器烧入，运行于片上 Flash 的开始区域 0x08000000，UPDATE 升级区的程序也必须通过仿真器烧入，运行于片上 Flash 的开始区域 0x08030000，APP 运行区的程序可以通过手持控制器烧入，运行于片上 Flash 的开始区域 0x08060000，以后需要更新终端模块的程序，只需要更新手持控制器存储内容，无需通过仿真器进行更新，即使在烧写的过程中突然掉电，只会导致 APP 运行区的程序会破坏，BOOT 引导区和 UPDATE 烧写区的程序仍然可以正确运行。

3 系统硬件设计

3.1 启动模式

ARM 结构 CM4 手册规定芯片复位时要从 0x0000 0000 地址开始取出中断向量，GD32 的片上 Flash 在 MDK 里被设置为起始地址 0x0800 0000，因为启动模式决定了向量表的位置，GD32 有 3 种启动模式：

1）片上 Flash 存储器。当选择从片上 Flash 启动模式后，使用 JTAG 或 SWD 模式烧写程序，一般会将程序下载

到设备内置的片上 Flash 存储器中的 0x0800 0000 地址处，设备重启直接从这个地址启动程序，片上 Flash 的 0x0800 0000 地址被映射到 0 地址处，这个时候 CM4 既可以在 0 地址处访问中断向量表，也可以在 0x0800 0000 地址处访问中断向量表，而代码存放在地址 0x0800 0000。GD32 通过做了一个启动映射，NVIC 中有一个寄存器，称为向量表偏移量寄存器，在文件 GD32f4xx _ misc. c 通过修改该寄存器的值就能重新定位向量表，向量表在地址空间中的位置是通过重定位寄存器设置的值来指出向量表的地址。在复位后，该寄存器的值为 0[17]。

2）系统存储器启动。从系统存储器启动，通过设置引导引脚 BOOT1＝0 BOOT0＝1，芯片能够从特定区域的固化程序开始执行，特定区域的内存地址从 0x1FFFF000 开始，存放了厂家预置的程序，也称为 ISP 程序，通常是只读的，在出厂后，这个区域的数据不可以被修改或擦除，因此被认为是 ROM 区域。它存放着芯片出厂时预置的固化程序，主要用于初始化系统和提供下载功能，我们可以使用厂商提供的 ISP 程序，通过串口总线将用户的应用程序下载到芯片的 Flash 存储器中，下载完成后，为了使芯片能够正常启动运行已下载的程序，需要将 BOOT0 引脚和 BOOT1 引脚都重新设置为低电平。这种方式的好处是在不使用仿真器的情况下，直接通过串口进行程序更新和调试[18]。

3）片上 SRAM 启动。从片上 SRAM 启动时，确保程序正确执行，需要在应用程序的初始化代码中重新设置向量表的位置，从特定的起始地址 0x2000 0000 进行访问，与前面提到的系统存储器启动不同，SRAM 无法永久存储程序，且在掉电后数据将丢失，如果需要长期保存的数据，必须采取其他方式进行存储，以防止掉电导致数据丢失，因此这种启动模式主要用于程序的调试和临时存储数据[19]。

在许多 GD32 系列芯片上，存在两个引脚 BOOT0 和 BOOT1，它们在芯片复位时的电平状态决定了芯片复位后从哪个区域开始执行程序。多数正常启动情况下 BOOT0 和 BOOT1 引脚都被拉低，电平都为 0 时，芯片从片上 Flash 存储器的起始地址处开始执行程序。当芯片从系统存储器启动时候，BOOT0 引脚被设置为高电平，才允许外部串口下载新的程序到芯片的 Flash 存储器中，下载完成后，将 BOOT0 引脚重新设置为低电平，以便芯片正常启动已下载的程序，而 BOOT1 引脚通常保持为低电平。当芯片从 SRAM 启动，BOOT0 和 BOOT1 两个引脚都需要拉高。启动模式如表 2 所示。

表 2　启动模式

启动模式选择引脚		引导源选择
BOOT1	BOOT0	
X	0	片上 Flash 存储器
0	1	系统存储器启动
1	1	片上 SRAM

通过设置 BOOT1 和 BOOT0 两个引脚的电平状态，就可以灵活切换芯片的启动模式，本文启动模式设计为 BOOT1＝0 和 BOOT0＝0 从片上 Flash 存储器启动，如图 5 所示。

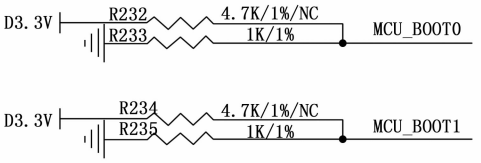

图 5　启动模式原理图

3.2　PHY 芯片原理

DP83848 是一种由德州仪器（Texas Instruments）公司生产的集成以太网控制芯片，用于实现 100 Mbit/s 单路物理层以太网的收发功能，支持 100 M 的以太网通信，集成度高，具有低功耗等性能，同时支持 MII（媒体独立接口）与 RMII（简化的媒体独立接口）两种与物理层（PHY）通讯的标准接口，实现以太网数据帧的发送与接收，遵守 IEEE 802.3-2002 标准和 IEEE 1588-2008 标准，还集成了串行管理接口（SMI，serial management interface），专门用于访问 PHY 芯片寄存器[20-21]，SMI 接口包括 MDC 和 MDIO 两条信号线，MDIO 是用来读写 PHY 的寄存器，以获取 PHY 的状态，MDC 为 MDIO 提供时钟。SMI 接口可以支持最多 32 个 PHY，但在任意时刻只能访问一个 PHY 的一个寄存器。

本文采用 RMII 接口模式，需要 7 根线通信，MII 模式需要 16 根通信线，RMII 模式在功能上与 MII 是相同的，在保持物理层器件现有特性的前提下减少了 PHY 芯片引脚数量，RMII 接口模式主要包括以下引脚：参考时钟、发送使能、接收使能、发送数据及接收数据等。PHY 芯片原理如图 6 所示。

ETH _ 50M：在 RMII 模式下使用，提供给 PHY 芯片的 50 MHz 参考时钟，这个外部时钟非常重要，需要稳定、高精度，让系统各个组件时序同步。

MCU _ PHY _ TXEN：数据发送使能，高电平有效。在整个数据发送过程保存有效电平，表示 MAC 层正在将要传输的数据放到 MCU _ PHY _ TXD0 和 MCU _ PHY _ TXD1 上。

MCU _ PHY _ TXD0：发送数据，MCU _ PHY _ TXEN 有效后，作为发送端。

MCU _ PHY _ TXD1：在 RMII 模式下只有 2 位，只有在 MCU _ PHY _ TXEN 有效数据线才有效。

MCU _ PHY _ CRS _ DV：数据接收使能，用于数据接收。对于 RMII 模式，这个 MCU _ PHY _ CRS _ DV 引脚有效，则认为在 MCU _ PHY _ RXD0 和 MCU _ PHY _ RXD1 上的数据是有效的。

MCU _ PHY _ RXD0：接收数据，与 ETH _ 50M 时钟同步，在 MCU _ PHY _ CRS _ DV 有效后的每个时钟周期里，MCU _ PHY _ RXD0 和 MCU _ PHY _ RXD1 接收

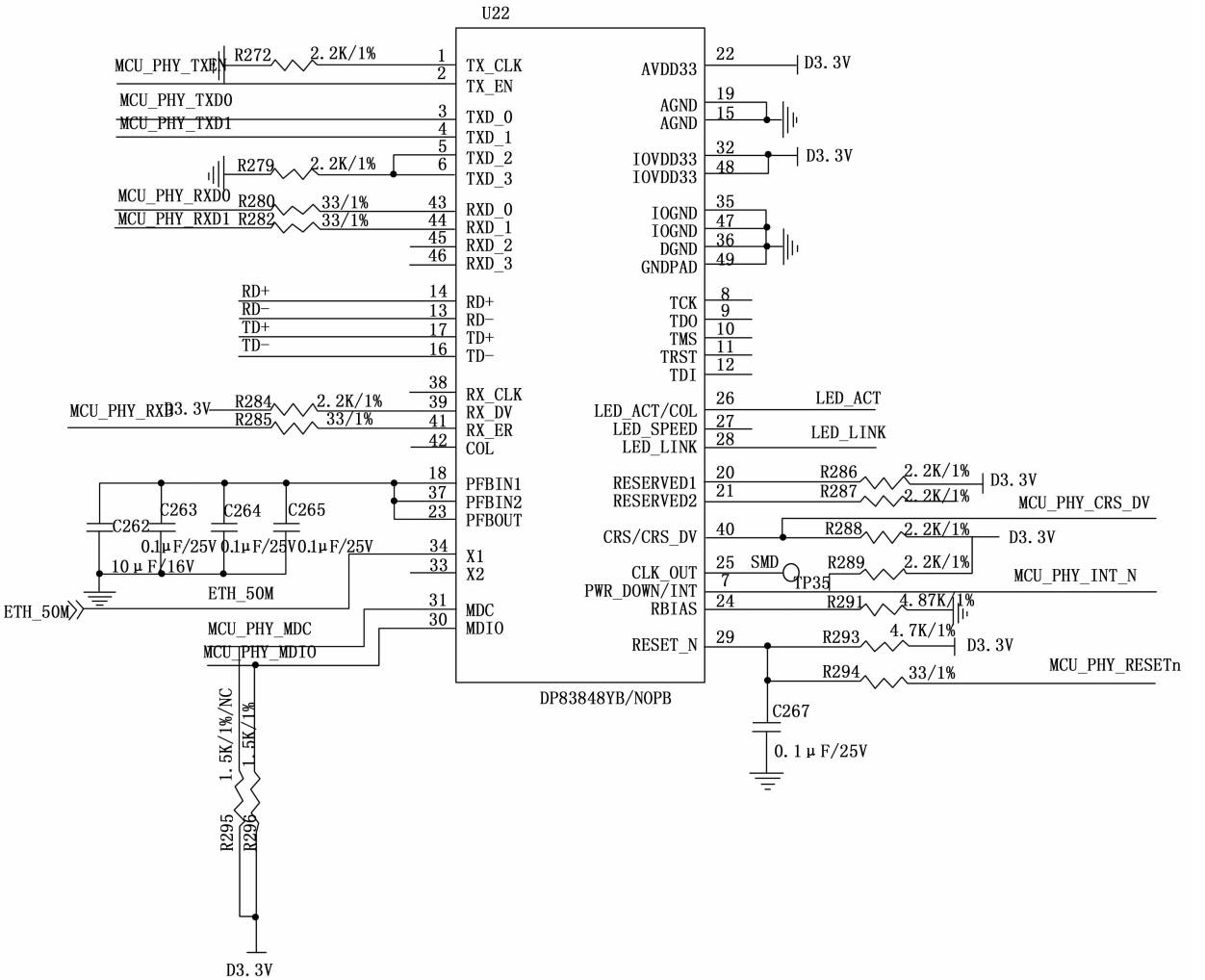

图 6 PHY 芯片原理图

PHY 芯片的两位数据。

MCU _ PHY _ RXD1：接收数据，MCU _ PHY _ CRS _ DV 有效后，作为接收端。

MCU _ PHY _ MDC：为 MCU _ PHY _ MDIO 提供时钟，最高频率为 2.5 MHz 的时钟信号，在空闲状态下该引脚保持为低电平状态。

MCU _ PHY _ MDIO：用于与 PHY 之间的数据传输，与 MCU _ PHY _ MDC 时钟线配合，接收和发送数据。

3.3 网络变压器原理

网络变压器原理如图 7 所示。在 PHY 芯片与 RJ45 网络插座之间还需要一个网络变压器，PHY 芯片工作时送出的上行数据信号从网络变压器的 Pin1 和 Pin3 进入，再由 Pin16 和 Pin14 输出，经 RJ45 水晶头连接网线发送出去，外部设备送来的下行数据信号经网线和 RJ45 水晶头，由 Pin9 和 Pin11 进入网络变压器，再由 Pin6 和 Pin8 输出，送到 PHY 芯片的收发器上。

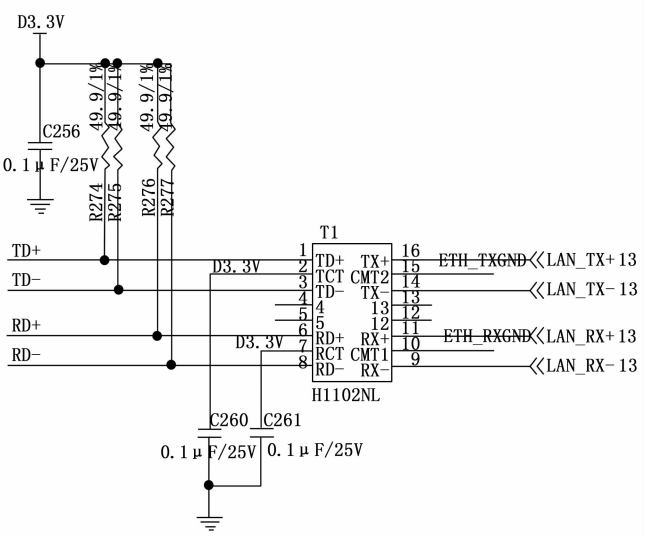

图 7 网络变压器原理图

理论上来说，可以不接网络变压器，从 PHY 芯片出来信号直接连到 RJ45 上，也是能正常工作的，但传输距离就会受到限制，因为 PHY 芯片驱动器的功率有限，当网线较长时，到接收端 RJ45 上的信号会逐渐衰减，导致工作异

常，添加网络变压器后，通过变压器输出信号显著提高，传输距离进一步增加。增加网络变压器相当于将 PHY 芯片与 RJ45 线路隔离，减少 PHY 芯片受到各种干扰，输出信号更加稳定，还可以确保不同型号 PHY 芯片信号正常传输，增强整个网络的兼容性。

4　系统软件设计

4.1　上位机软件设计

上位机软件通过网络与手持控制器进行数据通信，手持控制器是固件更新操作的发起端，手持控制器发送数据读取命令给上位机，上位机把编译生成的 BIN 文件发送手持控制器存储。其具体实现流程如下：

1）上位机软件初始化，读取本地更新 BIN 文件，如果收到手持控制器发送的数据读取命令后，会把文件大小通过网络发送给手持控制器，手持控制器接收到正确的文件大小后擦除本地片外 Flash；

2）上位机根据读取数据帧长度判断是否需要拆帧发送，数据超过 1 024 字节需要进行拆分为若干帧，没有超过 1 024 字节的数据直接发送，上位机发送的每帧数据 1 024 字节，发送完成都要等待手持控制器回复，否则对该帧数据进行重传 3 次，重传 3 次仍没有收到表示传输失败；

3）上位机下发的每帧数据都收到了手持控制器回复，当传输完最后一帧数据，手持控制器会显示数据读取成功，如果传输过程中出现异常，手持控制器会显示数据读取超时。

4.2　手持控制器软件设计

手持控制器操作按键选择相应功能执行，跟设备进行命令式交互，通过以太网接口与终端设备模块通信，交互信息显示在屏上显示，手持控制器起到了中转数据的作用，手持控制器设计流程如图 8 所示。

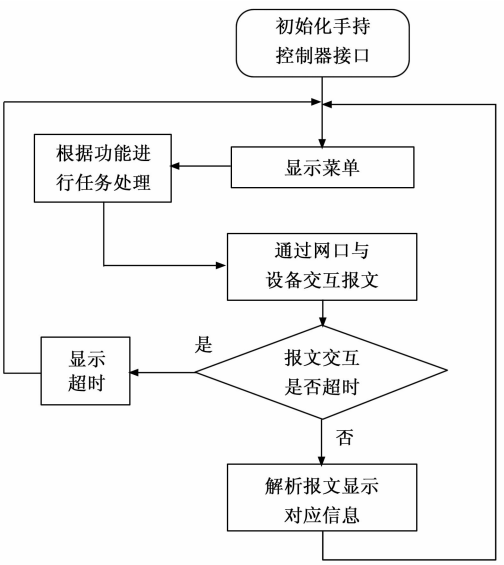

图 8　手持控制器设计流程

手持控制器交互具体实现流程如下：

1）上电初始化手持控制器，主要包括显示模块、按键模块、网络传输模块、Flash 模块等；

2）初始化完成进入主菜单，通过查询判断按键是否被按下，如果按下就会发送对应的命令给终端设备模块，终端设备模块收到命令回复确认消息，超过设置的时间未收到数据，则显示超时；

3）如果按下数据注入命令，就会读取存储在片外 Flash 中的数据，通过网络发送给终端设备模块，发送完成后显示发送成功；如果按下数据读取命令，就会接收上位机软件发送的数据，并将数据存储到片外 Flash，接收完成后显示接收成功；

4）如果按下自检命令，就会通过网络发送给终端设备模块，终端设备模块收到自检命令就会把自检结果回传给手持控制器。

4.3　终端设备模块软件设计

终端设备模块的设计需要创建 3 个工程，分别是 BOOT 工程、UPDATE 工程和 APP 点亮流水灯工程，如图 9 所示，终端模块上电运行 BOOT 区程序，等待手持控制器发送升级命令，10 s 没有升级收到命令，程序跳转到 APP 区运行。如果终端设备模块上电以后，手持控制器发送升级命令，程序跳转到 UPDATE 区运行，将收到的数据写入到 0x8060000 开始的地址，数据全部烧写完成后，程序会跳转到 APP 区运行。

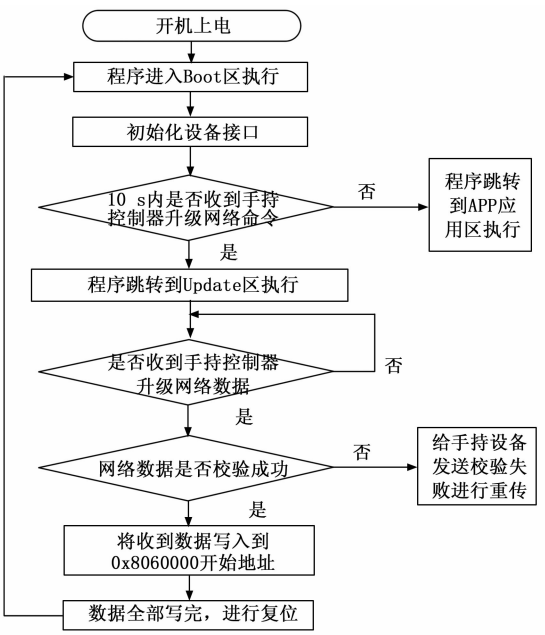

图 9　终端模块的图

4.3.1　BOOT 启动区的软件设计

终端模块上电复位后在 0x0800 0000 地址处访问中断向量表，如果收到手持控制器升级命令，跳转到 UPDATE 区，10 s 没有收到手持控制器命令，跳转到 APP 区，下面程序是收到升级命令从 BOOT 区跳转到 UPDATE 区。

```
define FLASH_UPDATE_ADDR 0x08003000 / * UPDATE 区中断向量表 * /
void iap_jump_to_update(void)
```

```
{
    if (((*(__IO uint32_t*)FLASH_UPDATE_ADDR) &
0x2FFE0000 )  0x20000000) /*检查栈顶地址是否合法 */
    {
__disable_irq();/*屏蔽所有中断,否则可能导致跳转的失
败*/
    JumpAddress = *(__IO uint32_t*)(FLASH_UPDATE_
ADDR + 4); /*设置 UPDATE 区堆栈首地址即跳转到新的程序
的起始地址 */
    Jump_To_Update = (pFunction) JumpAddress; /*将跳转
到达 UPDATE 区程序的首地址赋值给一个函数指针*/
    __set_MSP(*(__IO uint32_t*) FLASH_UPDATE_AD-
DR); /*初始化堆栈指针*/
    Jump_To_Update (); /*函数指针完成跳转*/
    }
}
```

4.3.2 UPDATE 烧写区的软件设计

UPDATE 区的程序能够运行，需要在程序中再次被映射中断向量表，根据表 1 终端设备模块片上 Flash 地址区域划分，UPDATE 区中断向量表被映射到 0x0803 0000，修改中断向量表函数使用 nvic _ vector _ table _ set（NVIC _ VECTTAB _ FLASH，0x30000），UPDATE 区程序存储更改为 0x0803 0000，如图 10 所示。

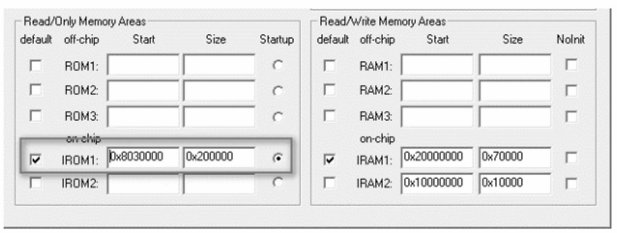

图 10 UPDATE 区程序存储位置

手持控制器负责将上位机软件下发的数据帧内容通过网络总线转发到对应的终端设备模块，UPDATE 区的程序按照表 3 数据帧消息内容进行解析，每帧前 8 个字节分别表示帧头、长度和帧序号，后面接着 1 024 字节真实 BIN 文件

表 3 数据帧消息内容

字节	消息数据	内容
BYTE1	帧头	0xA5
BYTE2		0xA5
BYTE3	消息长度	8+n 字节，其中 n≤1024
BYTE4		
BYTE5	帧序号	帧序号从 1 开始
BYTE6		
BYTE7		
BYTE8		
BYTE9	固件数据	十六进制数
· · ·	· · ·	· · ·
BYTE8+n	固件数据	十六进制数
BYTE8+n+1	CRC 数据	十六进制数

数据，为避免传输出错会进行 CRC 校验，终端设备模块收到每帧数据都正确，就会写入到片上 Flash 存储。

下面程序是 UPDATE 区核心程序，主要功能是烧写 BIN 文件到片上 Flash 中，烧写完成程序会自动跳转到 APP 区。

```
define FLASH_APP2_ADDR (0x08060000) /* APP 区起始
地址*/
define READ_LEN (1024) /*一帧固件数据大小*/
void Write_MCU_To_Flash(void)/*把固件数据写到 flash*/
{
uint32_t data_buff[512]={0};
if(JTAG_flag  1)
{
g_cn = 0;/*计数清 0*/
}
else if(JTAG_flag  2)/*收到一帧数据 1024 字节*/
{
memcpy(data_buff,g_UdpRxBuf+8,READ_LEN);
fmc_write_32bit_data(FLASH_APP2_ADDR+g_cn,READ_
LEN/4, data_buff); /*写入片上 FLASH*/
g_cn+=READ_LEN;
}
else if(JTAG_flag  3) /*收到最后一帧数据不足 1024 字节
*/
{
memcpy(data_buff,g_UdpRxBuf+8,data_len);
fmc_write_32bit_data(FLASH_APP2_ADDR+g_cn,data_
len/4, data_buff); /*写入片上 FLASH*/
g_cn = g_cn + data_len;
}
}
```

4.3.3 APP 运行区的软件设计

1）同样原理 APP 区的程序能够运行，根据表 1 终端设备模块片上 Flash 地址区域划分，APP 区中断向量表被映射到 0x08060000，修改中断向量表映射位置 nvic _ vector _ table _ set（NVIC _ VECTTAB _ FLASH，0x60000）。

2）默认情况下 MDK 编译链接是不会生成 BIN 文件，需要配置工程选项，在 MDK 的“Options For Target->Users”中加入 fromelf 指令，编译生成 BIN 格式文件如图 11 所示，K \ ARM \ ARMCC \ bin \ fromelf. exe --bin-out-

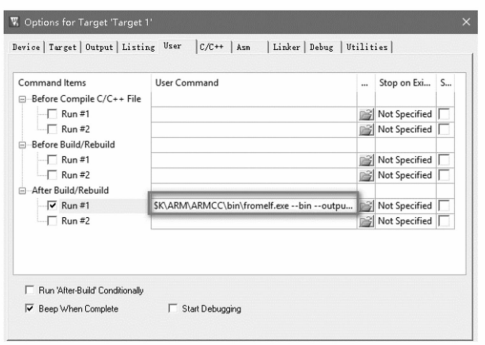

图 11 配置工程

```
00000000 C8 71 01 20 79 02 06 08  9D 60 06 08 5B 31 06 08  C7 4C 06 08 19 30 06 08  39 67 06 08 00 00 00 00
00000020 00 00 00 00 00 00 00 00  00 00 00 00 C1 01 06 08  8D 02 06 08 00 00 00 00  15 02 06 08 79 62 06 08
00000040 93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08
00000060 93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08
00000080 93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  1D 30 06 08 93 02 06 08  93 02 06 08 45 31 06 08
000000A0 93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  F1 63 06 08 93 02 06 08  93 02 06 08 93 02 06 08
000000C0 93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08
000000E0 93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08
00000100 93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08
00000120 93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08
00000140 93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 51 66 06 08
00000160 93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  00 00 00 00 00 00 00 00
00000180 93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  93 02 06 08 93 02 06 08  00 00 00 00 00 00 00 00
000001A0 93 02 06 08 93 02 06 08  93 02 06 08 DF F8 0C D0  00 F0 58 FC 00 48 00 47  F9 11 07 08 C8 71 01 20
000001C0 2A 4B 19 68 08 69 B0 E8  F0 4F 80 F3 09 88 BF F3  6F 8F 4F F0 00 00 80 F3  11 88 70 47 00 00 00 00
000001E0 06 48 00 68 00 68 80 F3  08 88 62 B6 61 B6 BF F3  4F 8F BF F3 6F 8F 00 DF  00 BF 00 BF 08 ED 00 E0
```

图 12　BIN 文件的原始数据

put＝@L. bin ！ L。原始数据如图 12 所示，其中 BIN 文件是最直接的代码映像，它记录的内容就是要存储到片上 FLASH 的二进制数据，在片上 FLASH 中烧写的内容与 BIN 文件内容完全相同[22]。

3）程序在 UPDATE 区运行时候，才能更新固件，在 APP 区运行无法更新，此时，手持控制器给终端设备模块发送复位命令，通过 NVIC _ SystemReset（）函数执行软件复位。软件复位后程序就会重新从 BOOT 区运行，手持控制器给终端设备模块发送烧写命令，终端设备模块程序跳转到 UPDATE 区运行，固件更新完成后会自动运行新的应用程序。

5　实验结果与分析

本文是把生成固件通过上位机发送给手持控制器，通过手持控制器，对终端设备模块的应用程序进行烧写，通过实验对固件升级过程进行模拟测试验证，并对终端设备模块的环境实验进行测试验证，最后对实验结果进行分析和总结。

5.1　固件升级

1）按照 3.3 节完成终端设备模块设计，把片上 Flash 全部擦除，BOOT 区的可执行程序和 UPDATE 区的可执行程序通过仿真器烧录到片上 Flash，将 APP 应用程序生成的 BIN 文件，通过上位机软件发送到手持控制器；

2）终端设备模块上电复位，通过手持控制器把 APP 应用程序生成的 BIN 文件烧写到片上 Flash 指定地址，通过读取 0x0806 0000 地址数据与 BIN 源文件比对，如果两者一样，说明烧写到片上 Flash 成功，BIN 源文件的部分数据如图 12 所示，片上 Flash 部分数据如图 13 所示；

3）烧写完成后，手持控制器会显示烧写完成，观察终端设备模块上的流水灯会闪烁，表明此次升级成功。有多种情况会造成固件升级失败，比如升级过程中网络断开、突然断电都会导致终端设备模块烧写失败，此时终端

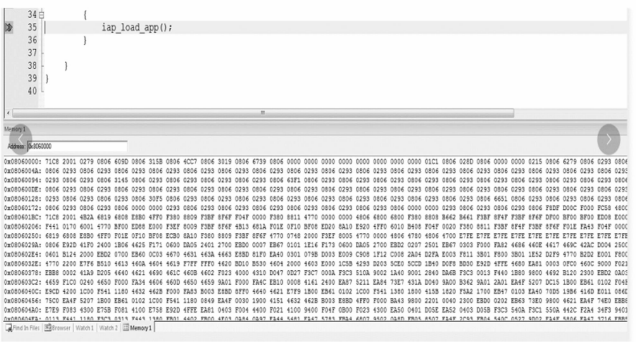

图 13　片上 Flash 读出来的数据

模块上电复位会无法工作，流水灯无法点亮。由于本设计分为 BOOT 区、UPDATE 区和 APP 区，异常造成固件升级失败，只是 APP 区的程序无法正常运行，断电后终端设备模块重新上电烧写，仍可以将固件写入 APP 区完成升级。

5.2　环境实验测试

通过手持控制器烧写完成的终端设备模块放到温箱进行环境实验测试，分别在高温正 52 ℃、低温负 55 ℃和常温正 30 ℃进行功能验证，观察流水灯是否点亮，在此范围温度上电断电各 30 次，测试记录如图 14 所示，流水灯全部点亮。

5.3　实验结果

通过上述实验表明，在－55～52 ℃流水灯都点亮，固件都能正常工作，上下电各 30 次全部点亮，成功率均达到了 100％，实验结果与预期一致，通过在线烧写固件方法，可以让终端设备模块程序正确运行，方案是稳定可靠的。

6　结束语

本文是通过免拆终端模块及无须挂载仿真器对其在线烧写，并对其功能进行验证，通过实验测试验证流水灯都能正常工作，说明此方案可行性，达到了预期的目的，基

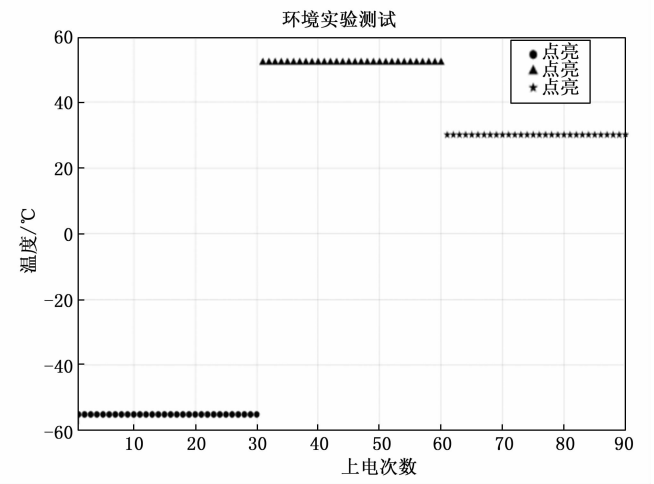

图 14 环境实验图

于 GD32 芯片网络在线烧写固件技术可靠，为外场升级免拆设备提供依据，在企业的实际生产和维护过程中使用在线烧写技术，可以降低设备维护人力成本，大大地提高了企业经济效益。

参考文献：

[1] 刘 浩，李荣冰，刘建业，等. 基于串口通信的 DSP 在线烧写技术研究 [J]. 电子测量技术，2017，40 (7)：184 - 187.

[2] 李光龙，刘振威，乔海强，等. 基于 CAN 总线的 STM32F107 程序在线升级 [J]. 测控技术，2018，37 (9)：156 - 158.

[3] 孙秀芳，姜 凯. 基于 DSP 的 SPI 接口自举引导程序的实现 [J]. 信息与电脑，2021 (20)：116 - 118.

[4] 周洋洋，赵昶宇. 基于 CAN 总线的 DSP28335 在线烧写方法研究 [J]. 科技与创新，2019 (6)：58 - 59.

[5] 严建龙. 基于串口的 F2811 程序在线升级 [J]. 微处理机，2019，6 (3)：61 - 64.

[6] 郭少雷，王玲玲，郭 磊. 基于 GD32 的在线升级系统设计与实现 [J]. 电路与控制，2021，36 (3)：43 - 46.

[7] 沈 全，唐明军. 一种基于 IAP 技术的批量升级 GD32 微控制器芯片程序的方法 [J]. 电子技术，2022，51 (7)：15 - 17.

[8] JUAN L I，WANG J H，XUE M，et al. Design and research of remote debugger and upgrade system for DSP [J]. 半导体光子学与技术：英文版，2009，15 (1)：63 - 68.

[9] WANG J Y，LI S Z. ISP Technology’s application in remote upgrade for the intelligent instruments [J]. Advanced Materials Research，2012，7：711 - 716

[10] YU Y，KURNIANGGORO L，WAHYONO，et al. Online programming design of distributed system based on multi-level storage [J]. International Conference on Intelligent Computing，2016 (7)：745 - 752.

[11] ZHANG T，LI J Z，LUO H，et al. A method for remotely upgrade C8051F041 program based on CAN bus network [C] //MATEC Web of Conferences，2020.

[12] ZHANG Q Y，CHEN X W，ZHANG S L，et al. Design and implementation of remote upgrade system for vehicle terminal based on GPRS [C] //IEEE Asia-Pacific Conference on Image Processing，Electronics and Computers，2021：636 - 640.

[13] 蔡晓乐，屈盼让，李 军，等. 基于微控制器的程序在线升级设计 [J]. 计算机应用，2023，26 (2)：109 - 112.

[14] 李 正，刘晓源，曹云侠，等. IAP 远程升级技术在空间应用 [J]. 计算机科学与应用，2017，7 (9)：886 - 893.

[15] 曹玉保. 基于双备份的兆易创新 GD32 程序升级方案研究 [J]. 中国集成电路，2021，30 (z1)：23 - 26.

[16] 韩兆渊，王晓东，黄国勇. 基于 IAP 的北斗终端程序远程升级技术的研究 [J]. 计算机与数字工程，2017，45 (5)：844 - 848.

[17] 姜晓道，赵紫君. 基于 IAP 的通用嵌入式系统在线升级功能设计 [J]. 电子技术，2022 (3)：20 - 23.

[18] 唐鹏程，汪旭明，胡 力. 用 IAP 技术在线升级 STM32 单片机固件 [J]. 吉首大学学报，2019，40 (1)：21 - 26.

[19] 吕春艳，靳占军，张乐君，等. STM32 单片机在线升级设计及实现 [J]. 信息通信，2017，6：110 - 111.

[20] 谢昌伟，顾瀚戈，钟洪念，等. HTTP 模式下 STM32 程序远程升级设计 [J]. 电子与封装，2023，23 (5)：501 - 505.

[21] 胡 权，张宏宽，周伯涛，等. 面向城乡公交的嵌入式系统远程升级设计方案 [J]. 电子元器件与信息技术，2022，6：204 - 208.

[22] 文 丰，温 倩，武慧军. 基于 IAP 的嵌入式系统在线编程设计 [J]. 单片机与嵌入式系统应用，2022，12：37 - 41.

(上接第 247 页)

[17] 肖 游，智小琦，王 琦. 基于 FDS 与 CFD 组合的快速烤燃数值模拟 [J]. 火炸药学报，2022，45 (4)：536 - 543.

[18] 刘 斌. ANSYS Fluent 2020 综合应用案例详解 [M]. 北京：清华大学出版社，2021.

[19] 刘 斌. Fluent 2020 流体仿真从入门到精通 [M]. 北京：清华大学出版社，2021.

[20] 唐家朋. ANSYS fluent 16.0 超级学习手册 [M]. 北京：人民邮电出版社，2016.

[21] 于巧燕，侯 磊，柴 冲，等. 高压天然气管道喷射火焰长度和伤害范围研究 [J]. 石油科学通报，2022，7 (4)：593 - 603.

[22] CCHRISTO F，DALLY B B. Modeling turbulent reacting jets issuing into a hot and diluted coflow [J]. Combustion and Flame，2005，142 (1/2)：117 - 129.

[23] 李英杰. 基于 CFD 的管道流动控制设备流动特性的研究 [D]. 杭州：浙江科技学院，2022.

[24] RODAK P S. Numerical simulation of different combustion regimes in a laboratory combustor [D]. Lisbon：TéCNICO LISBOA，2017：35 - 38.

[25] 王 竞. 基于 CFD 模拟污水厂污泥管道内速度、浓度及密度分布研究 [D]. 长春：吉林建筑大学，2023.