

产品名称：modbus模块	密级：机密
产品版本：V1.0	Total 9 pages 共 9 页

modbus模块软件详细设计说明书

Prepared by		Date	2014-03-22
拟制		日期	
Reviewed by		Date	
评审人		日期	
Approved by		Date	
批准		日期	

Revision Record 修订记录

Date 日期	Revision Version 修订 版本	Sec No. 修改 章节	Change Description 修改描述	Author 作者

目录

1	引言	3
1.1	编写目的	3
1.2	背景	3
1.3	参考资料	3
1.4	Modbus模块使用框图	3
2	设计概述	4
2.1	任务和目标	4
2.1.1	需求概述	4
2.1.2	运行环境概述	4
2.1.3	条件与限制	4
2.1.4	开发环境及调试工具	4
3	系统详细设计	5
3.1	系统结构设计及子系统划分	5
3.2	系统功能模块详细设计	5

1 引言

1.1 编写目的

本说明书编制的目的是说明modbus模块软件设计的总体思想及各个任务的设计思想，为程序员编码提供依据。

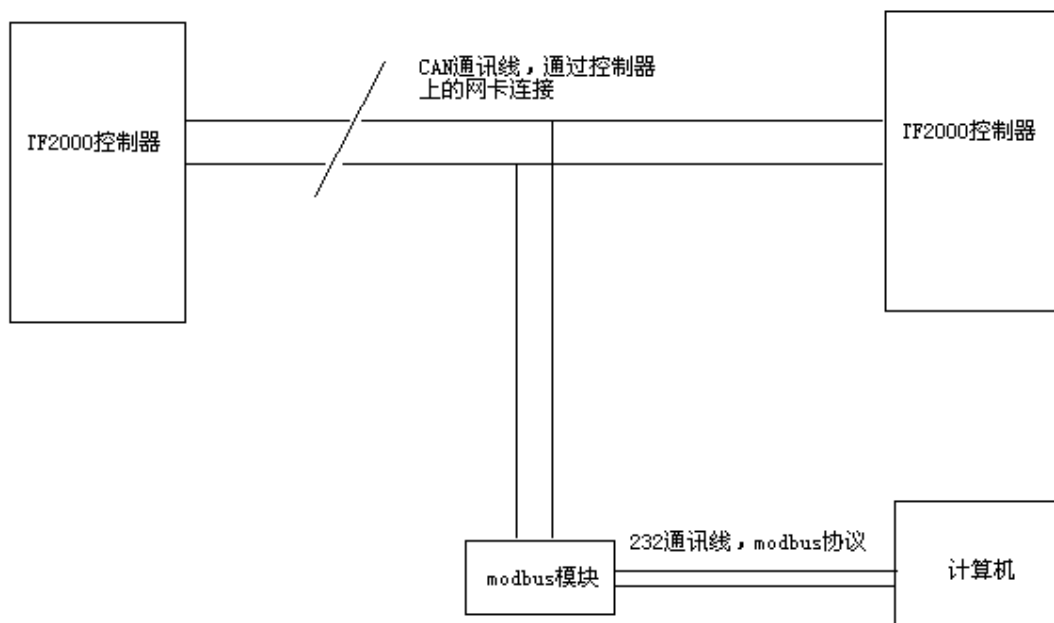
1.2 背景

- A. 待开发模块系统名称：modbus模块
- B. 该模块基本概念：该模块为计算机的从设备，用于从火灾报警控制器的网卡取得火灾报警控制器的报警信息并存储，与计算机通讯遵循modbus协议，等待计算机查询报警信息的模块。
- C. 开发项目组名称：modbus开发项目组

1.3 参考资料

- MODBUS通讯协议中文版. pdf
- tyco_Modbus协议. doc
- 20120720通讯协议. doc （网卡通讯协议）
- Modbus模块1.0（源程序）（原modbus模块源程序，主要参考modbus通讯部分）
- Tf2000网络卡源程序 （主要参考CAN通讯部分）

1.4 Modbus模块使用框图



2 设计概述

2.1 任务和目标

达到能够从火灾报警控制器的网络线上取得火灾报警控制器的报警信息并能够通过modbus协议正常响应计算机的查询命令（03指令）得目标。

2.1.1 需求概述

- 输入：通过火灾报警控制器的网络获取火灾报警控制器的报警信息
- 输出：遵循modbus协议，通过计算机的查询命令，将火灾报警控制器的报警信息发送给计算机。

2.1.2 运行环境概述

使用LPC2292 MCU进行模块的控制，本模块使用UCOSII实时内核，采用任务+中断的方式进行系统设计。

2.1.3 条件与限制

无

2.1.4 开发环境及调试工具

开发环境：KEIL MDK 进行开发。

调试工具：使用J-LINK仿真器进行调试。

3 系统详细设计

3.1 系统结构设计及子系统划分

UCOSII为一实时性很高的操作系统，按照具体功能可划分为各个任务执行。

根据modbus模块的功能，可将modbus模块的设计划分为以下几个任务。

- uart0_rcv_task: 用于解析从计算机接收的数据，根据解析的数据发送读取存储器的信号量
- Can1_Rcv_Task: 用于解析从火灾报警控制器联网的网络线上获取的报警信息，根据解析的数据发送写存储器的信号量。
- Exter_Ram_Write_Task: 外部Ram写任务，因外部Ram的读和写属于共享资源，有可能存在优先级反转的问题，故不能将外部Ram的读和写并如通讯数据的解析，需单独作为任务处理。
- Exter_Ram_Read_Task: 外部Ram读任务。
- Wait_Time_Task: 时间控制任务，用于执行需频繁检测的代码，完成CAN数据发送及LED灯控制

中断服务程序：

- CAN中断：用于接收火灾报警控制器的报警或控制命令。
- UART中断：用于接收计算机数据及发送回应数据给计算机。
- TIMERO中断：用于处理系统的嘀嗒定时器。

3.2 系统功能模块详细设计

3.2.1 数据结构设计

```
#define TaskStkLengh 192 //任务栈长度

typedef unsigned char BOOLEAN;           /* 布尔变量 */
typedef unsigned char U8;                 /* 无符号8位整型变量 */
typedef signed char S8;                   /* 有符号8位整型变量 */
typedef unsigned short U16;               /* 无符号16位整型变量 */
typedef signed short S16;                 /* 有符号16位整型变量 */
typedef unsigned int U32;                 /* 无符号32位整型变量 */
typedef signed int S32;                   /* 有符号32位整型变量 */
typedef float FP32;                       /* 单精度浮点数（32位长度） */
typedef double FP64;                      /* 双精度浮点数（64位长度） */

#define CAN_FIFO_LEN 64
```

CAN接收缓冲区结构，用于缓存从CAN接收到的数据及标志

```
typedef struct
{
    INT32U buff[CAN_FIFO_LEN][4];

    INT8S in;

    INT8S out;
}CAN_FIFO;

typedef union          //CAN总线接收数据结构，用于分析CAN总线的数据
{
    Struct
    {
        INT32U rfs;
        INT32U rid;
        INT32U rda;
        INT32U rdb;
    }1;
    INT8U tab[16];
}CANRX;
```

UART发送缓冲区结构，用于缓存未发送完的数据

```
typedef struct          uart_send_list
{
    INT8U buff[256];
    INT8U buff_len;
    INT8U buff_send_len;
    //struct uart_send_list *uart_list_next;
}UART_SEND_LIST;
```

外部RAM操作结构体，用于控制外部RAM的读写

读结构体，读取外部RAM时按照16位读取，不区分高低字节

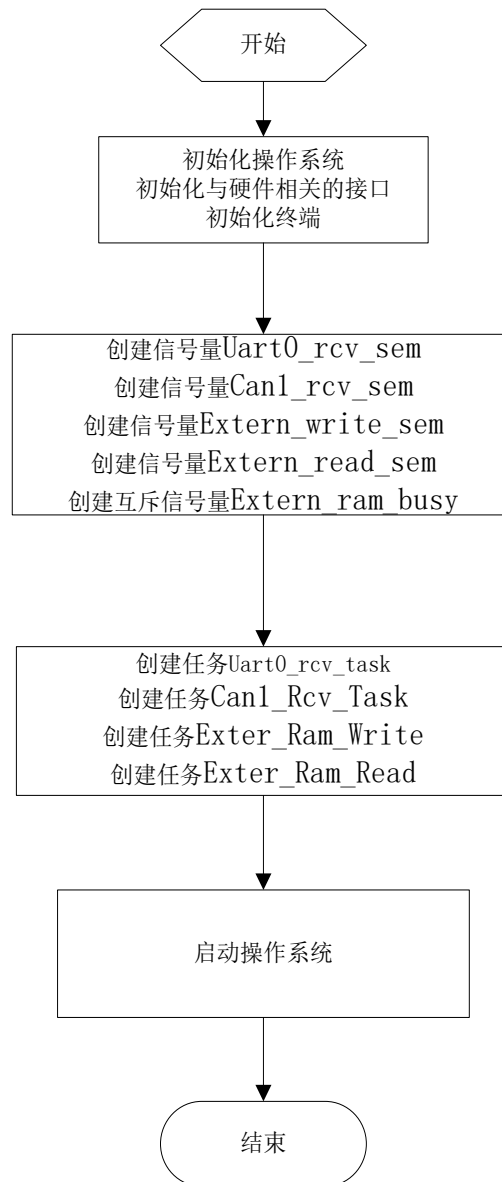
```
typedef struct
{
    U32 start_addr;          //读起始地址
```

```
    U8 Len;                //读长度
}EX_RAM_READ_COM;
写结构体，写外部RAM时需区分高低字节
typedef struct
{
    U32 Write_Inf[EX_WRI_MAX_LEN+1][2]; //[][0]要写的地址,[][1]高16位表示要
写的的数据，低16位表示要写的类型1表示写低字节，2表示高字节、0表示要写全字、如果
只写一个字节，则高字节为需要写的位，
    U8 In;                //结构的输入地址，分析通讯帧所得

    U8 Out;              //结构的输出地址，写外部RAM使用
}EX_RAM_WRI_COM;
信号量说明：
Uart0_rcv_sem：UART0接收信号量，当UART中断完整接收到一包数据时，通知
                Uart0_Rcv_Task任务进行数据解析。
Can1_rcv_sem：CAN接收信号量，当CAN中断完整接收到一包数据时，通知Can1_Rcv_Task
                任务进行数据解析。
Extern_write_sem：外部RAM写信号量，用于通知Exter_Ram_Write进行写操作
Extern_read_sem：外部RAM读信号量，用于通知Exter_Ram_Read进行读操作。
Extern_ram_busy：互斥型信号量，用于标示外部Ram正在操作。

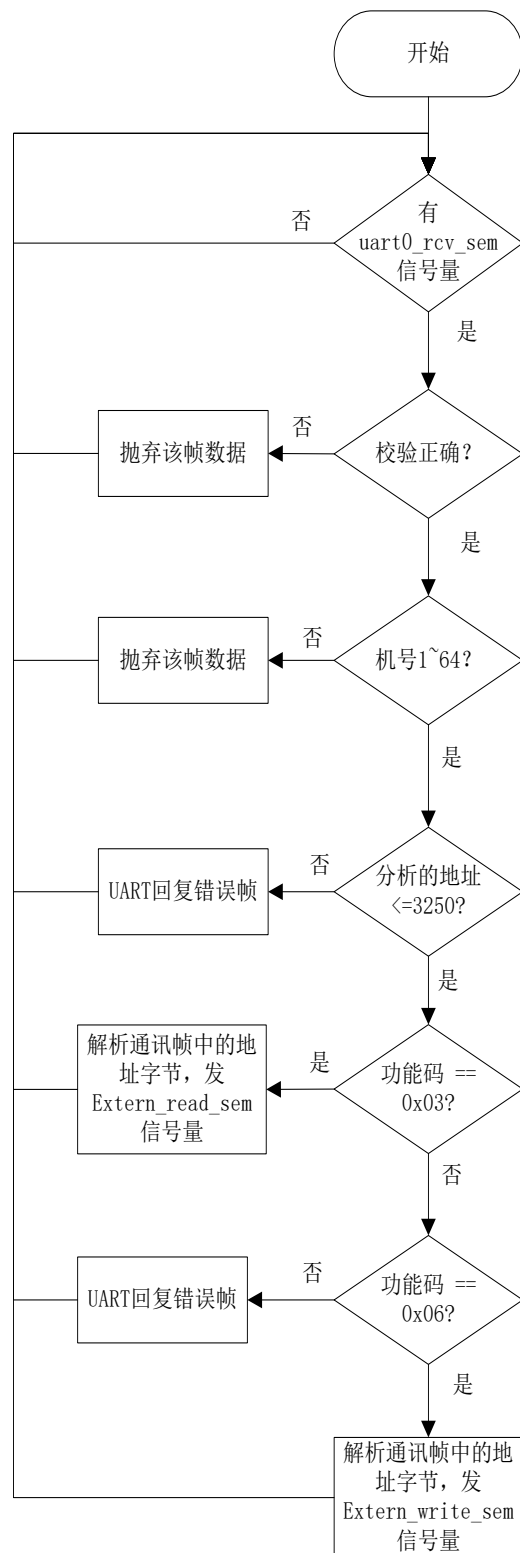
部分变量说明：
INT8U uart0_buff[256];    //uart接收缓冲区
INT8U uart0_buff_len;     //uart 接收缓冲区长度
```

3.2.2 主程序完成任务的创建及端口中断等的初始化工作流程图如下所示



3.2.3 uart0_rcv_task接收任务

用于解析从uart接收的按照modbus协议发送的数据帧，根据命令确定要读取的寄存器地址及读取的数量。流程图如下所示：



3.2.4 can1_rcv_task接收任务

功能：对接收的can数据按照主机与网卡的通讯协议进行分析，获取要写入的外部ram地址
流程如下所示：

