Document Title: PM_FW communication protocol module design description of Safety Control System

Document Number: 16-Q04-000120

Project Number: CT-RD-1601

Project Name: First phase of Safety Control System Development Project

Material Number: N/A

Document Version: A

Classification Level: Highly secret

Document Status: CFC

Controlled Status: Under control

Prepared by: Li Qi 2016-12-26

Checked by: Zhu Genghua 2016-12-30

Countersigned by: Liu Yang, Wang Dong

Approved by: Wen Yiming 2016-12-30

Revision History

No.	Relevant Chapter	Change Description	Date	Version Before Change	Version After Change	Prepared by	Checked by	Approved by
1		Document created	2016-12-26	None	A	Li Qi	Zhu Genghua	Wen Yiming
2								
3								
4								
5								

Relationship between this version and old versions: None.

文件名称：安全控制系统 PM_FW 通讯协议模块设计说明书

文件编号：16-Q04-000120

项目编号：CT-RD-1601

项目名称：安全控制系统开发项目一期

物料编号：

版本号/修改码：A

文件密级：机密

文件状态：CFC

受控标识：受控

拟制：李琦 2016 年 12 月 26 日

审核：朱耿华 2016 年 12 月 30 日

会签：刘阳、王东

批准：温宜明 2016 年 12 月 30 日

修订页

编号	章节名称	修订内容简述	修订日期	订前版本	订后版本	拟制	审核	批准
1		创建	2016-12-30		A	李琦	朱耿华	温宜明
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								

本版本与旧文件（版本）的关系：

Content 目录

1 Document overview 文档概述

1.1 Introduction 综述

This document describes the design description of communication protocol function of PM_FW of Safety Control System. The document describes the overall concept of the function of the module, and then the sub-function of the modules are described in detail.

This document is the output of module design phase of PM_FW, and is the input for the follow-up coding phase.

本文档描述了安全控制系统中 PM_FW 内部通讯模块的设计方案。文档首先描述了模块功能的总体设计思路，然后将模块功能划分为若干子功能并进行详细说明。

本文档是 PM_FW 模块设计的输出，也是后续编码的输入。

1.2 Reference 参考文档

1.2.1 Project documents 内部参考文档

[1] Embedded software safety concept of Safety Control System [505], 15-Q02-000059

[1] 安全控制系统嵌入式软件安全概念说明书 [505], 15-Q02-000059

[2] PM_FW software overall design description of safety control system [506], 15-Q02-000074

[2] 安全控制系统 PM_FW 总体设计说明书 [506], 15-Q02-000074

1.3 Terms and abbreviations 术语和缩略语

1.3.1 Terms 术语

Table 1-1 Terms

表 1-1 术语

No. 序号	Term 术语	Description 解释
1.	IP_BUS	Communication between PM and IO modules. PM 与 IO 模块之间的通讯总线。
2.	CM_BUS	Communication between PM and CM. PM 与 CM 之间的通讯总线。
3.	PM_BUS	Communication between PMs. PM 之间的通讯总线。
4.	System Net	Communication between control station and PC. 控制站与上位机之间的通讯网络。
5.	Safety Net	Safe communication between control stations.

		控制站之间的安全通讯。
6.	Control station 控制站	A set of triple redundant control system, which includes triple redundant PMs and IO modules under control. 一套三冗余的控制系统，包含三冗余 PM 和 PM 控制的各种 IO 模块。
7.	System response time 系统响应时间	Time interval from the moment that transition of demand signal generated at input ETP to the moment that transition of response signal generated at output ETP. 从系统输入端子板上产生需求信号跳变的时刻到输出端子板上产生相应的响应信号跳变之间的时间。
8.	Control cycle 控制周期	Time interval between adjacent two runs of user program execution. PM 两次执行用户程序间隔时间。
9.	Project 工程	Files which contain configuration information for control station and generated by IEC 61131 configuration software. These files contain all the information required by control station to implement control, including user control program (binaries) to be loaded and executed as well as configuration information of task, CM, PM and IO modules. IEC 61131 组态软件在完成编译后，为控制站生成的组态信息文件，该文件包含可加载执行的用户控制程序（二进制程序）、任务配置信息、CM 配置信息、PM 配置信息和 IO 模块配置信息等各种控制站完成控制所需的信息。
10.	Source project 源工程文件	Source file of the project before compiling. 工程在编译前的源文件。
11.	User program 用户程序	Part of project which contain user control program (binaries) to be loaded and executed and configuration information of task. 工程中的一部分：可加载执行的用户控制程序（二进制程序）和任务配置信息。

1.3.2 缩略语

Table 1-2 Abbreviations

表 1-2 缩略语

No. 序号	Abbreviation 缩略语	English description 英文	Chinese description 中文
1.	PM	Processor Module	主处理器模块
2.	CM	Communication Module	通讯模块
3.	BI	Bus Interface Module	总线接口模块
4.	AI	Analog Input Module	模拟量输入模块
5.	AO	Analog Output Module	模拟量输出模块

6.	DI	Digital Input Module	数字量输入模块
7.	DO	Digital Output Module	数字量输出模块
8.	OSP	Over Speed Protect Module	超速保护模块
9.	SOE	Sequence Of Events	SOE 事件
10.	SIL	Safety Integrity Level	安全完整等级
11.	PW	Power Module	电源模块
12.	OPC	OLE for Process Control	用于过程控制的对象链接与嵌入式技术
13.	UP	User Program	用户程序

2 Module overview 模块概述

The location of the communication protocol module (marked red) in the software hierarchy is shown below.

通讯协议模块（标红）在软件层次中的位置如下图所示。

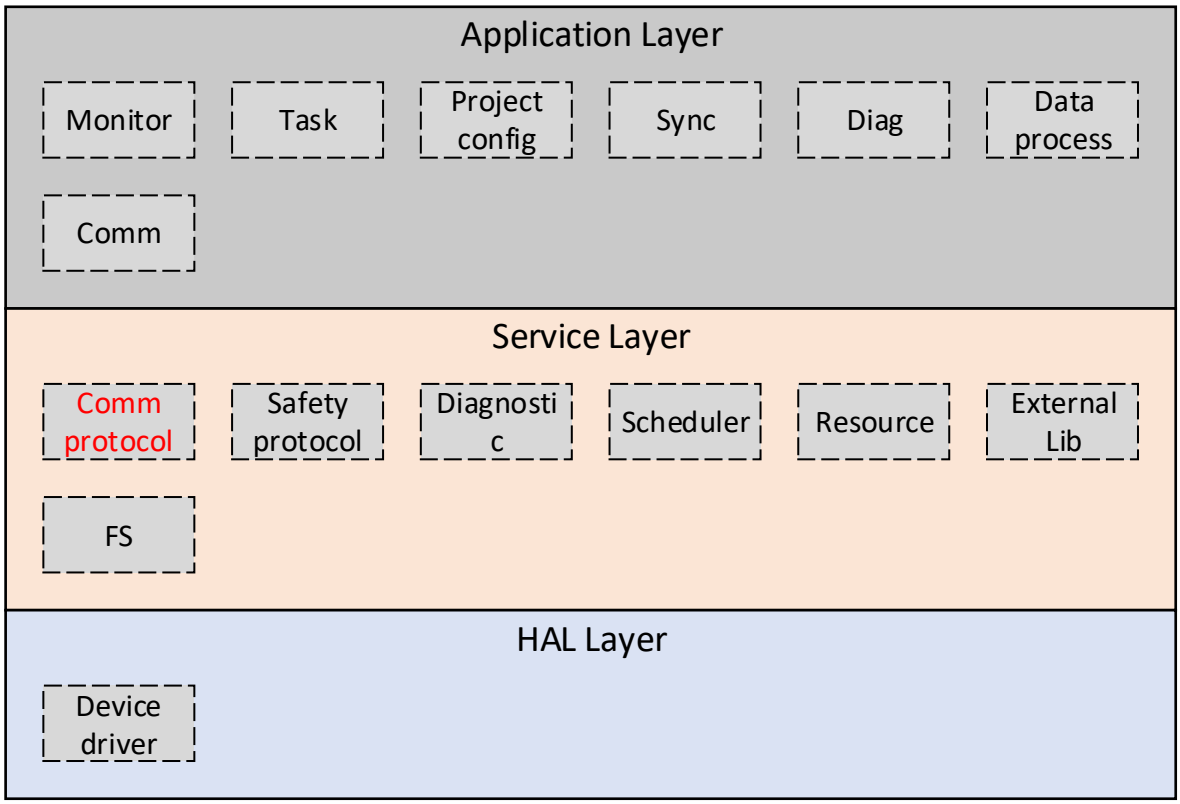

Figure 2-1 the location of the module

图 2-1 模块位置

3 Module design 模块设计

3.1 Function description 功能描述

This module implements the communication protocol between PM and CM, between PM, and between PM and IO.

该模块实现 PM 和 CM 之间，PM 之间，PM 和 IO 之间的通讯协议。

3.2 Design concept 设计思路

3.2.1 Communication data flow between PM and CM PM和CM通讯数据流

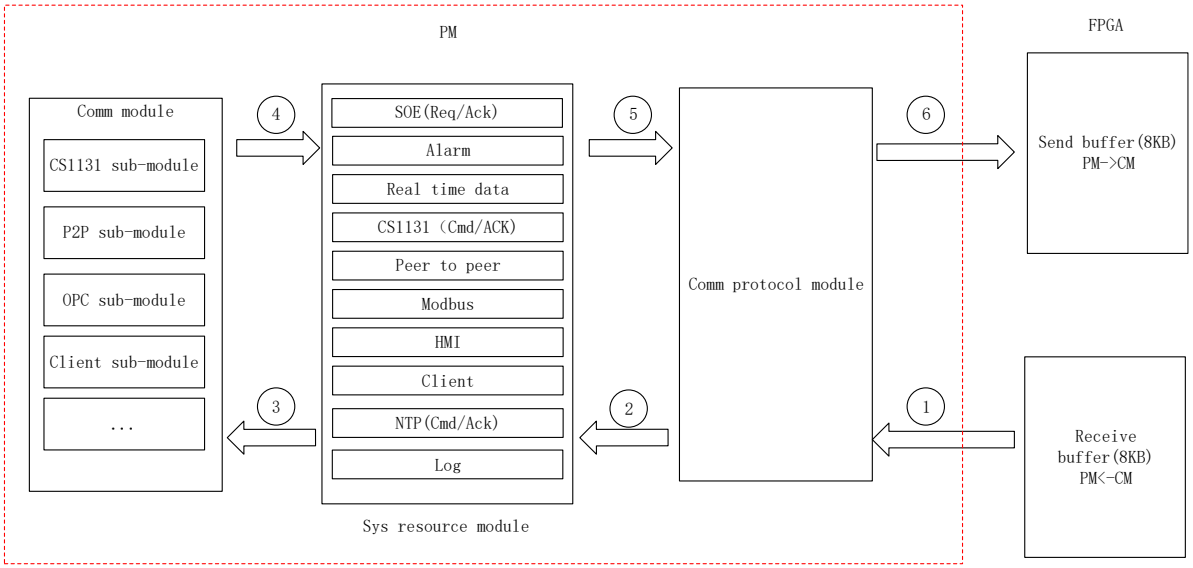

图 3- 1 PM 通讯数据流

Figure 3-1 PM communication data flow

① Communication protocol module receive data from CM;

①通讯协议模块接收来自 CM 的数据；

②Communication protocol module parse data and set the data to system resource module;

②通讯协议模块对数据进行初步解析后将数据分类放入资源管理模块；

③ Communication protocol module parse the command from CM;

③通讯模块对收到的命令进行解析；

④ Communication protocol module set the response to system resource module;

④通讯模块生成应答信息，发送到资源管理模块；

⑤ Communication protocol module get the response from system resource module and pack

the response;

⑤通讯协议模块从资源管理模块中获取通讯模块的应答数据并进行打包；

⑥Communication protocol module send the response to CM.

⑥通讯协议模块将打包后数据发送出去。

3.2.2 PM send detection message to CM PM向CM发送探测消息

①PM send detection message to CM periodically;

①PM 周期性发送探测数据包给 CMx；

②send the package to PM via FPGA;

②将探测包从 PM 发送出去；

③PM receive the response from CM;

③PM 接收来自 CMx 的数据；

④Parse the response, and set the response to the buffer; if PM received the response it means PM established the communication with the related CM, and the communication state shall be set to normal.

④对接收到的数据进行解析，将探测包应答信息放入对应缓冲区；当 PM 检测到来自 CMx 的应答时表明 PM 与 CMx 成功建立通讯链接，此时 PM 与 CMx 的通讯链接设置为正常。

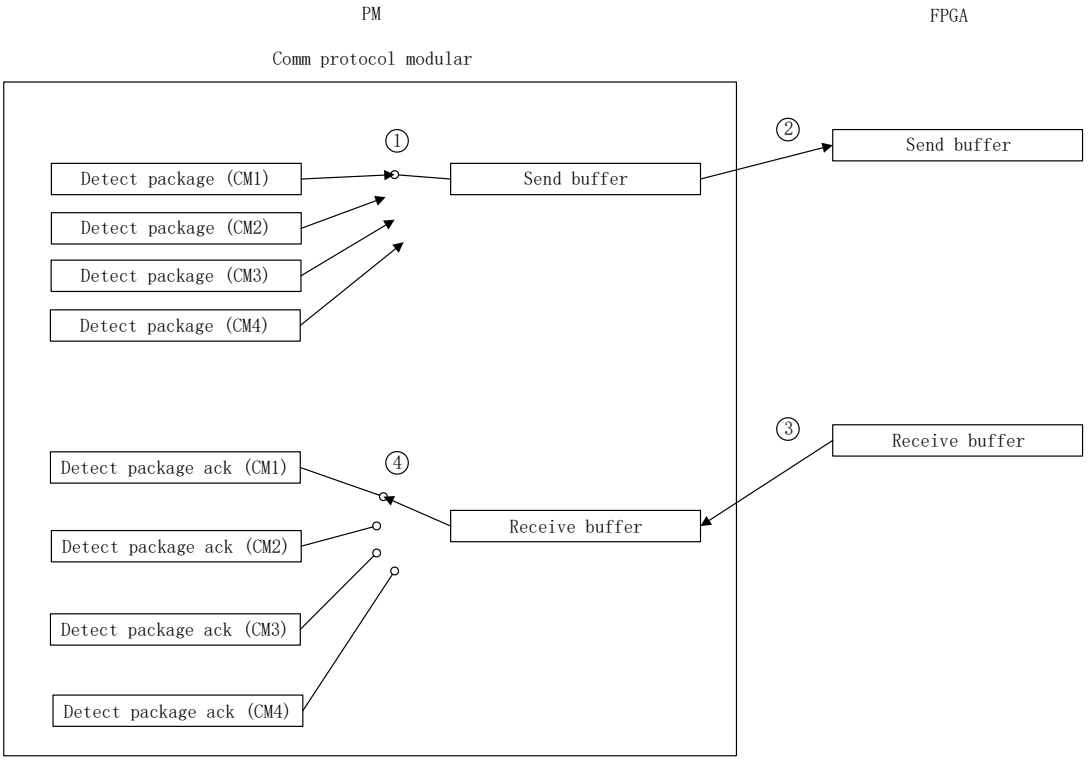

Figure 3-2 send detection message

图 3- 2 发送探测消息

3.2.3 PM send real time data to CM PM向CM广播实时数据

①Pack the real time data, which may be divided into several packages;

①将实时数据分批打包；

②send the real time data to CM;

②通讯协议模块将打包数据广播出去；

③receive the response from CM;

③通讯协议模块接收来自 CM 的数据；

④parse the response, and set the response to receiver-buffer.

④对接收数据进行解析，将实时数据应答放入对应缓冲区。

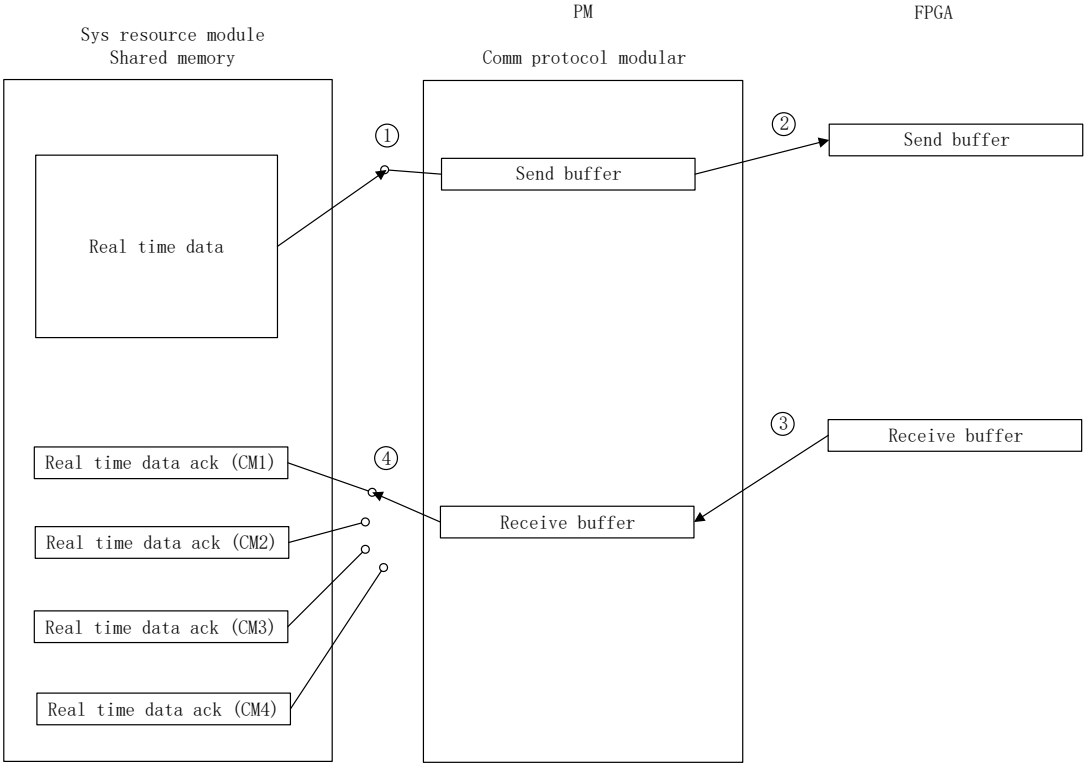

Figure 3-3 send real time data to CM

图 3- 3 上报实时数据

3.2.4 PM send configuration files PM发送配置文件

①Receive command from CM;

①接收来自 CM 的命令；

②Parse the command, and set it into system resource module;

②通讯协议模块对命令进行解析，将 CM 获取配置信息请求的命令放入资源管理模块；

③Pack the configuration files;

③通讯协议模块将 CM 配置信息打包；

④Send the configuration files to CM.

④将配置信息发送给对应的 CM。

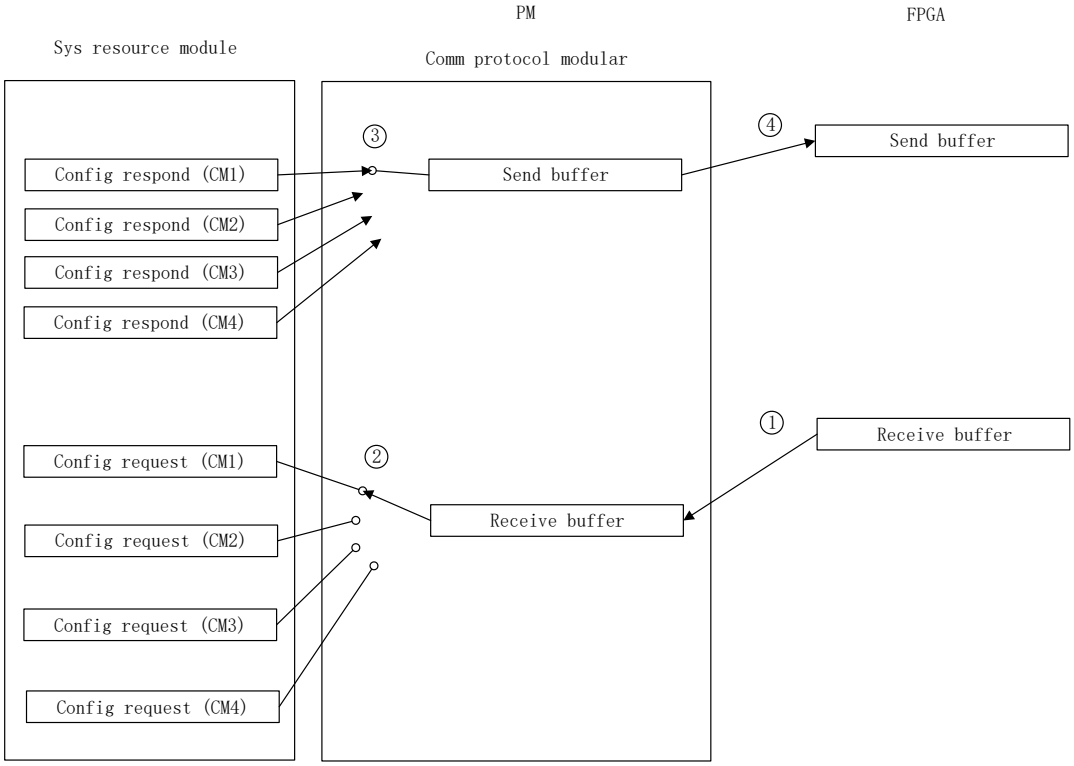

图 3- 4 PM send the configure files

图 3- 5 PM 发送配置文件

3.2.5 PM send P2P data PM发送P2P数据

①Pack the P2P data;

①将待发送的 P2P 数据打包；

②Send the P2P data to CM;

②通讯协议模块将数据发送出去；

③Receive the data from CM;

③通讯协议模块接收来自 CM 的数据；

④Parse the data, and set the response to the buffer.

④通讯协议模块对数据进行解析，将应答放入相应的缓冲区。

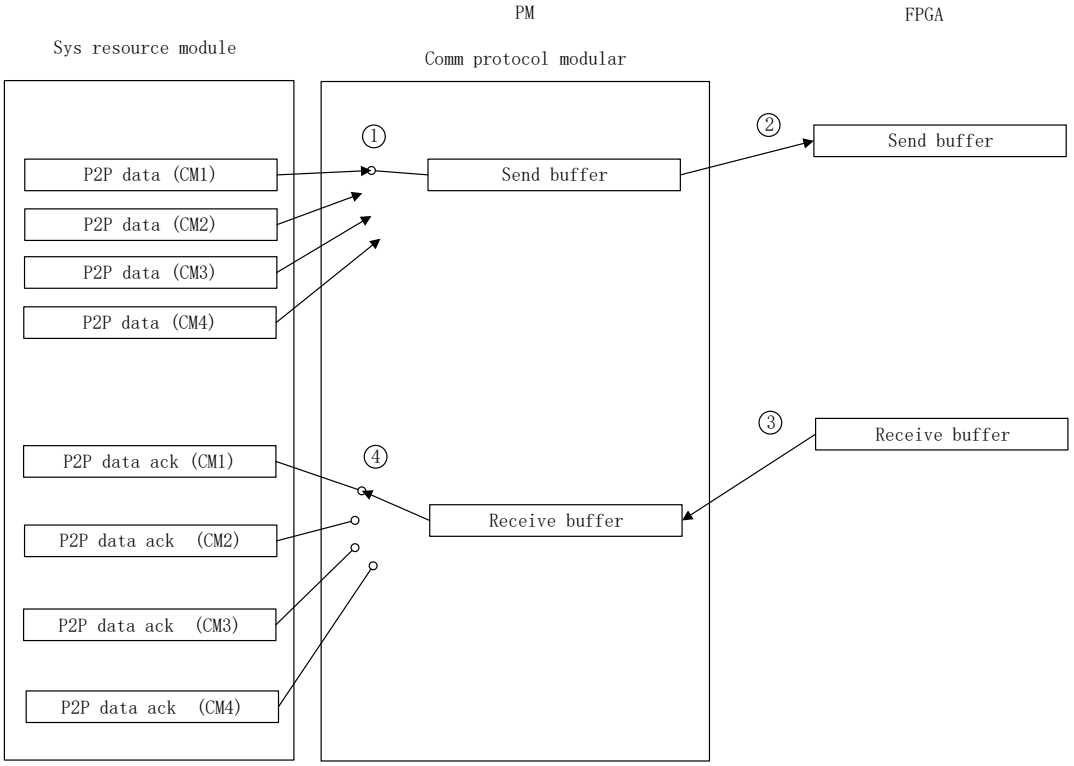

Figure 3-5 PM send P2P data

图 3- 6 PM 发送 P2P 数据

3.2.6 PM receive P2P data PM接收P2P数据

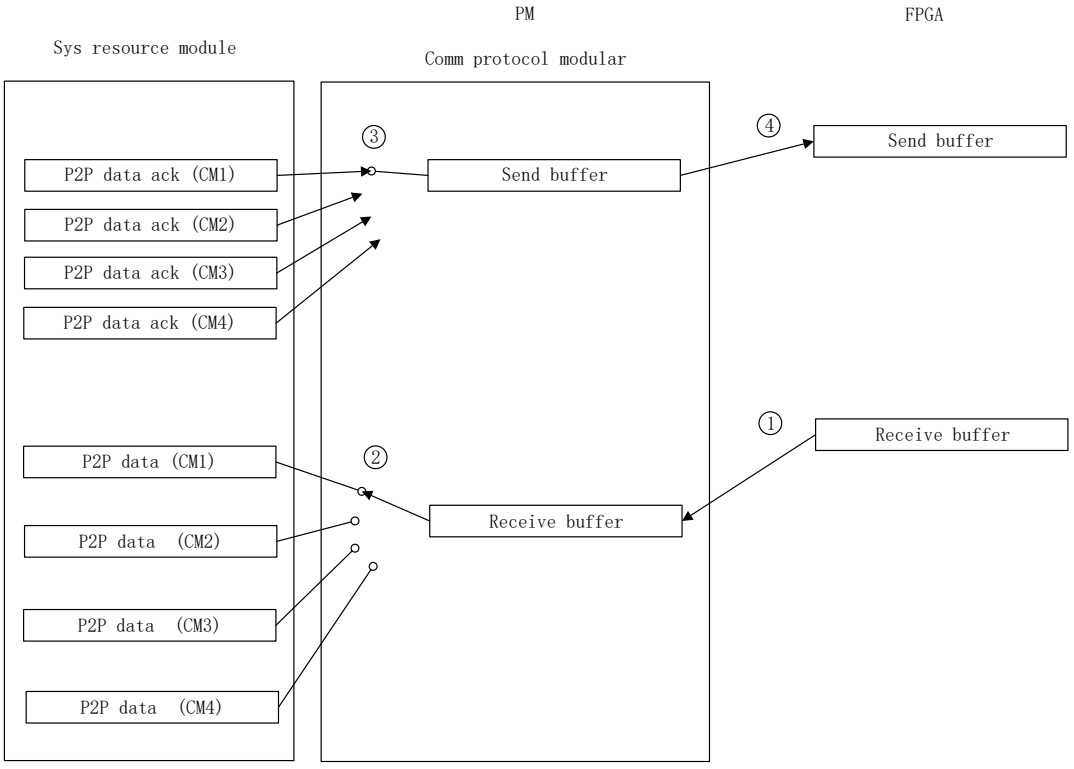

Figure 3-6 PM receive P2P data

图 3- 6 PM 接收 P2P 数据

3.2.7 SOE software data handling SOE软件数据处理

①Receive data from CM;

①通讯协议模块接收来自 CM 的数据；

②Parse data, and set the SOE request to the related receive-buffer;

②通讯协议模块对接收数据进行解析，将 SOE 请求放入对应缓冲区；

③pack the response;

③将 SOE 应答打包；

④send the response to CM;

④将打包后数据发送出去。

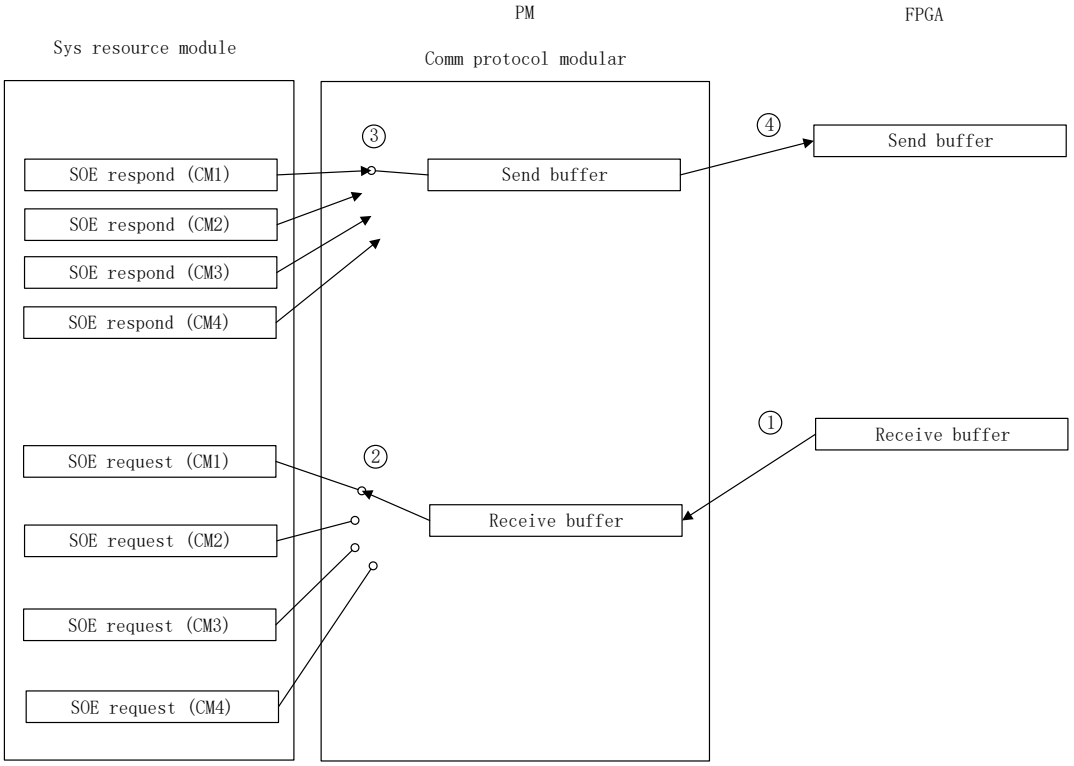

Figure 3-7 communicate with SOE software

图 3- 7 与 SOE 软件通讯

3.2.8 CS1131 configuration software data handling CS1131组态软件数据处理

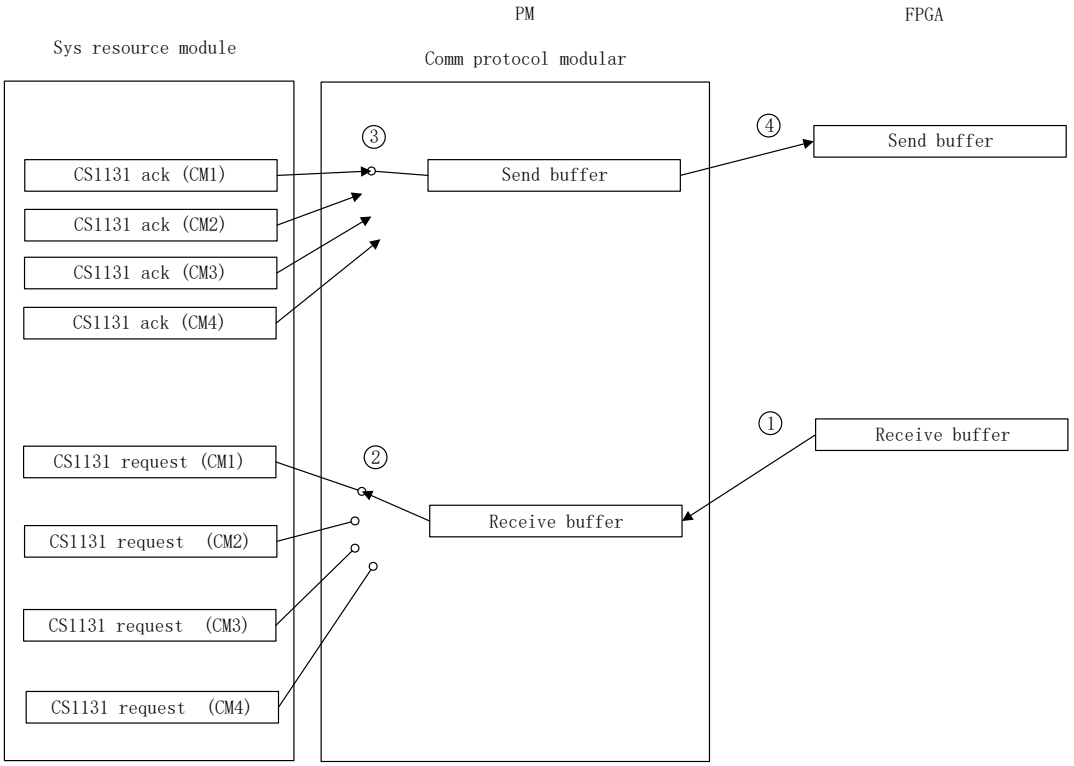

Figure 3- 7 communicate with CS1131

图 3- 8 与 CS1131 组态软件通讯

3.2.9 Diagnostic software data handling 诊断软件数据处理

This module get and process the request from diagnostic software, and it used the similar strategy as CS1131 software.

该模块获取并处理诊断软件发送来的请求，与处理 CS1131 软件发送来的请求采用相同的处理策略。

3.2.10 AMS software data handling AMS软件数据处理

This module get and process the request from AMS software, and it used the similar strategy as CS1131 software.

该模块获取并处理 AMS 软件发送来的请求，与处理 CS1131 软件发送来的请求采用相同的处理策略。

3.2.11 OPC Server data handling OPC SERVER数据处理

This module get and process the request from OPC Server, and it used the similar strategy as CS1131 software.

该模块获取并处理 OPC SERVER 发送来的请求，与处理 CS1131 软件发送来的请求采用相同的处理策略。

3.2.12 Modbus master data handling Modbus主站数据处理

This module get and process the request from Modbus master, and it used the similar strategy as CS1131 software.

该模块获取并处理 Modbus 主站发送来的请求，与处理 CS1131 软件发送来的请求采用相同的处理策略。

3.2.13 Communication data flow between PMs PM之间通讯

This module provides the read and write interface for communication between PMs.

本模块提供 PM 之间通讯的读写接口。

3.2.14 Communication data flow between PM and IO PM和IO之间通讯

This module provides read and write interfaces for communications between PM and IO. Including parameter setting interface, real-time data interface, mailbox interface, etc.

本模块提供 PM 和 IO 之间通讯的读写接口。包括参数设置接口，邮箱接口等等。

3.3 Interface function 接口函数

The interface functions provided by this module is shown as follows:

模块提供的接口函数如下：

3.3.1 CM_BUS

1. void CMBusProtocolInit (void)

Input argument 输入参数	Output argument 输出参数	Description 描述
No. 无。	No. 无。	Module initialization. 模块初始化。

2. void CMBusProtocolCycle (void)

Input argument 输入参数	Output argument 输出参数	Description 描述
No. 无。	No. 无。	Communication protocol interface function. 内部通讯周期运行函数

3.3.2 PM_BUS

3. int16_t PMBUS_Receive(int8_t bus_no, int8_t buf_no, uint8_t *buf)

Input argument 输入参数	Output argument 输出参数	Description 描述
bus_no: bus No. buf_no: buffer type buf: buffer address	return: data length	PM_BUS receive. PM_BUS 接收。

4. (void) int16_t PMBUS_Send(int8_t bus_no, int8_t buf_no, uint8_t *buf, int32_t len, int32_t timeout)

Input argument 输入参数	Output argument 输出参数	Description 描述
bus_no: bus No. buf_no: buffer type buf: buffer address timeout: time out parameter	return: data length	PM_BUS send. PM_BUS 发送。

3.3.3 IP_BUS

5. uint8_t IPBUSMailSend(uint8_t ucMailSend[], IPBUSAPPMailHeader_t *pstMailHeader)

Input argument 输入参数	Output argument 输出参数	Description 描述

ucMailSend: data buf. pstMailHeader: mailbox header	return: send state	Mailbox send. 邮箱发送。

6. uint8_t IPBUSMailRecv(uint8_t ucMailRecv[], IPBUSAPPMailHeader_t *pstMailHeader)

Input argument 输入参数	Output argument 输出参数	Description 描述
ucMailRecv: data buf. pstMailHeader: mailbox header	return: receive state	Mailbox receive. 邮箱发送。

7. void IPBUSReleaseDataRAMRight(void)

Input argument 输入参数	Output argument 输出参数	Description 描述
No. 无。	No. 无。	Release data right. 释放权限

8. void IPBUSSetAPPState(uint16_t usState)

Input argument 输入参数	Output argument 输出参数	Description 描述
usState: PM state of IP_BUS	No. 无。	Set the PM state of IP_BUS 设置 IP_BUS 中 PM 状态

9. uint16_t IPBUSGetAPPState(void)

Input argument 输入参数	Output argument 输出参数	Description 描述
No. 无。	Return: PM state of IP_BUS	Get the PM state of IP_BUS 读取 IP_BUS 中 PM 状态

Other functions for parameters setting of IP_BUS are similar with the function 8/9, and they are not repeated here.

IP_BUS 通讯协议中其他对参数的设置和接口 8/9 类似，不再赘述。

3.4 Global variable 全局变量

Table 3-1 Global variable list

表 3-1 全局变量列表

No.	Type	Name	Description

序号	变量类型	名称	描述
1.	PMRecvBuffer_t	s_stRecvBuffer	Receive buffer 接收数据缓冲区
2.	PMSendBuffer_t	s_stSendBuffer	Send buffer 发送数据缓冲区
3.	PMCommHandling_t	s_stCommRecv	Receive data handling 接收数据处理
4.	PMCommHandling_t	s_stCommSend	Send data handling 发送数据处理
5.	bool_t s	s_bSendDetectMsgFlag [NUM_OF_CM]	Send detect message flag 发送探测消息标识
6.	uint16_t	s_usSdDetectSn [NUM_OF_SLOT]	Send detect message serial number 发送探测消息序列号
7.	uint16_t	s_usRvDetectSn [NUM_OF_SLOT]	Receive detect message serial number 接收探测消息序列号

3.5 Data structure 数据结构

```
#pragma pack(1)
typedef struct PMCommHandlingTag
{
    uint16_t usUsed;                /*used buffer 已使用空间 */
    uint16_t usRest;                /*reset buffer 剩余空间 */
    uint16_t usOffset;              /*offset in buffer 可以使用的偏移 */
} PMCommHandling_t;

typedef struct PMInfoStructTag
{
    uint16_t usLen;                 /*data length 数据长度 */
    uint16_t usOffset;              /*offset in buffer 在缓冲区中的偏移地址 */
} PMInfoStruct_t;

/*send out data structure   PM 发送数据格式*/
typedef struct PMSendDataHeaderTag
{
    uint16_t usCMAddr;              /*CM address:1~4 CM 地址：1~4 */
    uint16_t usPackageSerialNum;    /*serial number 数据包序列号  */
    uint16_t usDataLen;             /*data length 数据内容长度 */
    PMInfoStruct_t stSysStateBlock;  /*system state 系统状态信息（PM 状态、CM 状态及校
```

```
时信息）*/
    PMInfoStruct_t stDetectMsgBlock;   /*detect message PM 发送的探测消息 */
    PMInfoStruct_t stSOEResp;          /*SOE response SOE 应答 */
    PMInfoStruct_t stRealTimeData;     /*real time data 实时数据 */
    PMInfoStruct_t stCS1131Resp;       /*CS1131 response 组态软件应答 */
    PMInfoStruct_t stP2PReq;           /*P2P request P2P 请求 */
    PMInfoStruct_t stP2PResp;          /*P2P response P2P 应答 */
    PMInfoStruct_t stOPCResp;          /*OPC response OPC 应答 */
    PMInfoStruct_t stClientReq;        /*Client request 客户端请求 */
    PMInfoStruct_t stClientResp;       /*Client response 客户端应答 */
    PMInfoStruct_t stAMSResp;          /*AMS response AMS 应答 */
    PMInfoStruct_t stCfgTableBlock;    /*configure table 配置表 */
    PMInfoStruct_t stConfigRespBlock; /*configure response 应答 CM 配置信息 */
    uint16_t usReserved[16];           /*reserved 保留 32B */
    uint32_t uiCRC32;                  /*CRC 32 位 CRC 校验码 */
} PMSendHeader_t;

typedef struct PMSendDataBodyTag
{
    uint8_t ucDataBuffer[PM_SEND_BUFF_BODY_SIZE];
} PMSendBody_t;

typedef struct PMSendBufferTag
{
    PMSendHeader_t stSendHeader;       /* Header 发送的头部信息 */
    PMSendBody_t stSendBody;           /*Body 发送的实际信息 */
} PMSendBuffer_t;

/* receive data structure PM 接收数据格式 */
typedef struct PMRecvDataHeaderTag
{
    uint16_t usCMAddr;                 /*CM address:1~4 CM 地址：1~4 */
    uint16_t usPackageSerialNum;       /*serial number 数据包序列号 */
    uint16_t usDataLen;                /*data length 数据内容长度 */
    PMInfoStruct_t stCMStateBlock;     /*CM state 系统状态信息（CM 状态及校时信息）*/
    PMInfoStruct_t stDetectMsgAckBlock;/*detect message response 对探测消息的应答 */
    PMInfoStruct_t stSOEReq;           /*SOE request SOE 请求 */
    PMInfoStruct_t stRealTimeDataAck; /*real time data response 实时数据应答 */
    PMInfoStruct_t stCS1131Req;        /*CS1131 request 组态软件请求 */
    PMInfoStruct_t stP2PReq;           /*P2P request P2P 请求 */
    PMInfoStruct_t stP2PResp;          /*P2P response P2P 应答 */
    PMInfoStruct_t stModbus;           /*Modbus data Modbus 数据 */
    PMInfoStruct_t stOPCReq;           /*OPC request OPC 请求 */
    PMInfoStruct_t stClientReq;        /*client request 客户端请求 */
    PMInfoStruct_t stClientResp;       /*client response 客户端应答 */
```

```
    PMInfoStruct_t stAMSReq;              /*AMS response AMS 请求 */
    PMInfoStruct_t stConfigReqBlock;   /*configure response 请求 CM 配置信息 */
    PMInfoStruct_t stInterCmd;            /*internal command 内部命令信息 */
    uint16_t usReserved[14];           /*reserved 保留 28B */
    uint32_t uiCRC32;                     /*CRC 32 位 CRC 校验码 */
} PMRecvHeader_t;

typedef struct PMRecvDataBodyTag
{
    uint8_t ucDataBuffer[PM_RECV_BUFF_BODY_SIZE];
} PMRecvBody_t;

typedef struct PMRecvBufferTag
{
    PMRecvHeader_t stRecvHeader;        /*header 接收的头部信息 */
    PMRecvBody_t stRecvBody;             /*body 接收的实际内容信息 */
} PMRecvBuffer_t;

/*send the detection message to CM 发送到 CM 的探测消息 */
typedef struct PMDetectMsgTag
{
    uint32_t uiFlag;                    /*detect flag: 0x12345678 探测标识：  0x12345678-探测
数据包 */
} PMDetectMsg_t;

/*send the detection package block to CM 发送到 CM 的探测消息数据块 */
typedef struct PMDetectMsgBlockTag
{
    /*start flag 起始标识 */
    uint16_t usStartFlag;
    /*data length 数据长度  sizeof(DetectMsg_t)*/
    uint16_t usLen;
    /*detect message 探测消息 */
    PMDetectMsg_t stDetectMsg;
    /*CRC 校验码 */
    uint32_t uiCrc32;
} PMDetectMsgBlock_t;

/*receive the response from CM 接收的来自 CM 的应答消息 */
typedef struct PMDetectMsgAckTag
{
    uint32_t uiFlag;                    /*detect flag: 0x87654321 探测标识：  0x87654321-探测
数据包应答 */
} PMDetectMsgAck_t;
```

```
/*receive the detect message response data block from CM  接收的来自 CM 的探测消息应答数据块  */
typedef struct PMDetectMsgAckBlockTag
{
    /*start flag  起始标识  */
    uint16_t usStartFlag;
    /*data length  数据长度   sizeof(PMDetectMsgAck_t)*/
    uint16_t usLen;
    /*response  应答  */
    PMDetectMsgAck_t stDetectMsgAck;
    /*CRC  校验码  */
    uint32_t uiCrc32;
} PMDetectMsgAckBlock_t;

/*configure command from CM  来自 CM 的获取配置信息命令  */
typedef struct PMConfigReqTag
{
    uint32_t  uiRequest;                /*request  flag:  0x00000000-no  request  0xFFFFFFFF-all configure files   请求标识：   0x00000000-无请求   0xFFFFFFFF-请求所有配置信息  */
} PMConfigReq_t;

/*configure request data block from CM  来自 CM 的配置信息请求数据块  */
typedef struct PMConfigReqBlockTag
{
    /*start flag  起始标识  */
    uint16_t usStartFlag;
    /*data length  数据长度   sizeof(PMConfigReq_t)*/
    uint16_t usLen;
    /*request  请求命令  */
    PMConfigReq_t stConfigReq;
    /*CRC  校验码  */
    uint32_t uiCrc32;
} PMConfigReqBlock_t;

/*send the configure files data block to CM  发送的 CM 的配置信息数据块  */
typedef struct PMConfigRespBlockTag
{
    /*start flag  起始标识  */
    uint16_t usStartFlag;
    /*data length  数据长度   sizeof(CMConfigInfo_t)*/
    uint16_t usLen;
    /*configuration  配置信息  */
    CMConfigInfo_t stCMConfigInfo;
    /*CRC  校验码  */
    uint32_t uiCrc32;
```

```
} PMConfigRespBlock_t;

/*send the configure table to CM  发送的 CM 的配置表数据块  */
typedef struct CfgFileTableBlockTag
{
    /*start flag  起始标识  */
    uint16_t usStartFlag;
    /*data length  数据长度    sizeof(SysCfgFileTable_t)*/
    uint16_t usLen;
    /*configure table  配置表  */
    SysCfgFileTable_t stCfgTable;
    /*CRC  校验码  */
    uint32_t uiCrc32;
} CfgFileTableBlock_t;

#pragma pack()
```

3.6 List of sub-function 子功能列表

The sub-functions list is shown as follows:

子功能列表如下。

Table 3-2 Sub function list

表 3-2 子功能列表

Sub function No. 子功能编号	Function description 功能描述
SWDD-PM-CP_NSafR_SecR_A_001	Module initialization. 模块初始化
SWDD-PM-CP_NSafR_SecR_A_002	CM_BUS communication protocol cycle function. 内部通讯周期运行函数
SWDD-PM-CP_NSafR_SecR_A_003	PM_BUS communication interface PM_BUS 通讯接口
SWDD-PM-CP_NSafR_SecR_A_004	IP_BUS communication interface IP_BUS 通讯接口

4 Design of sub-function 子功能设计

4.1 Module initialization 模块初始化

SWDD-PM-CP_NSafR_SecR_A_001

4.1.1 CMBusProtocolInit

4.1.1.1 Function Description 功能描述

Communication protocol module initialization function.

内部通讯模块初始化函数。

4.1.1.2 Argument Description 参数说明

- Definition 函数定义

void CMBusProtocolInit (void);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

No.

无。

4.1.1.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.2 Module cycle operation 模块周期运行

SWDD-PM-CP_NSafR_SecR_A_002

This sub-function is used to process the internal data between CM and PM cyclically.

本子功能用于周期性处理 CM 和 PM 间的内部通讯数据。

4.2.1 CMBusProtocolCycle

4.2.1.1 Function Description 功能描述

This function is used to process the request or response from SOE software, CS1131 software, AMS software, OPC Server, Diagnostic Software, NTP/SNTP timing server, Modbus, P2P etc.

本函数用于周期处理来自 SOE 软件、CS1131 软件、AMS 软件、OPC SERVER、诊断软件、NTP/SNTP SERVER、Modbus、P2P 等的请求或应答。

4.2.1.2 Argument Description 参数说明

- Function Definition 函数定义

void CMBusProtocolCycle (void);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

No.

无。

4.2.1.3 Processing flow 处理流程

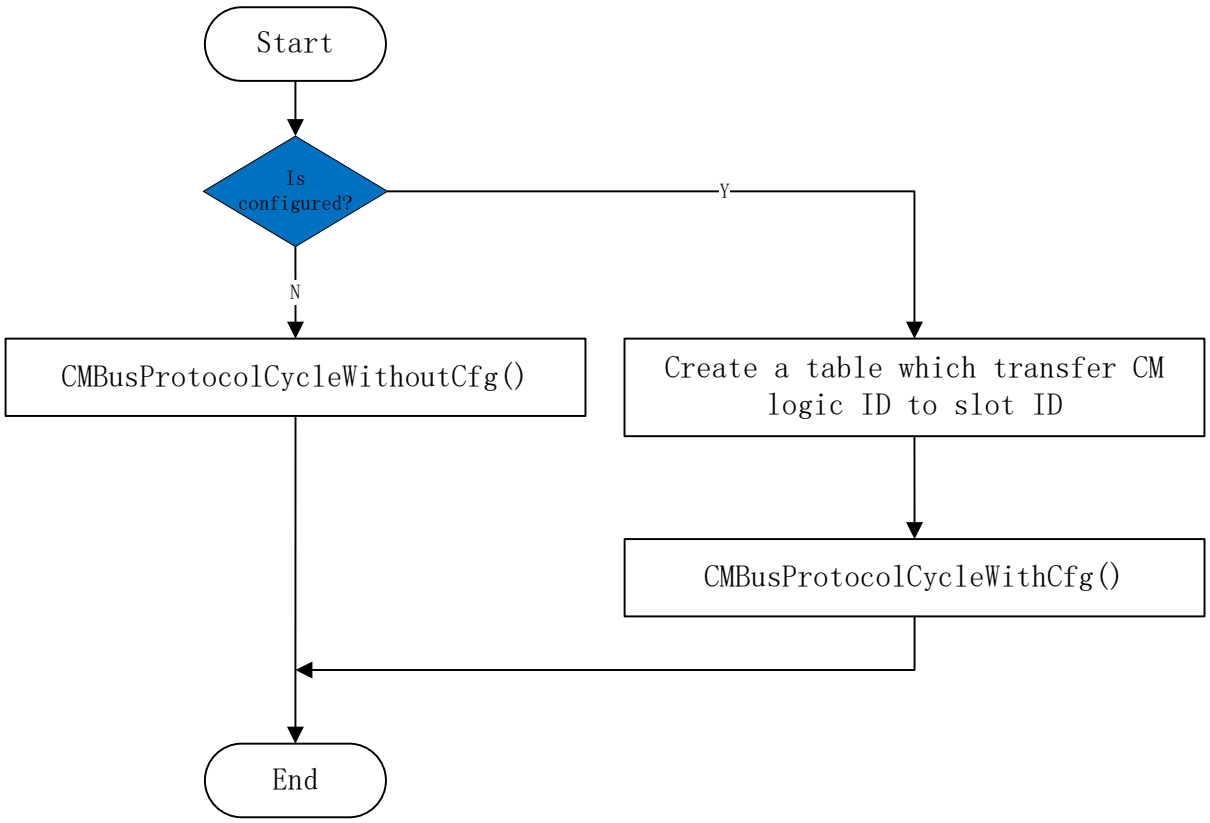

Figure 4-1 internal communication cycle processing flow

图 4-1 内部通讯周期运行流程图

4.2.2 CMBusProtocolCycleWithCfg

4.2.2.1 Function Description 功能描述

This function is used to receive the request or response from SOE software, CS1131 software, AMS software, OPC Server, Diagnostic Software, NTP/SNTP timing server, Modbus, P2P etc.

本函数用于周期接收来自 SOE 软件、CS1131 软件、AMS 软件、OPC SERVER、诊断软件、NTP/SNTP SERVER、Modbus、P2P 等的请求或应答。

4.2.2.2 Argument Description 参数说明

- Function Definition 函数定义

void CMBusProtocolCycleWithCfg(void);

- Input argument 输入参数

No

无

- Output argument 输出函数

No.

无。

4.2.2.3 Processing flow 处理流程

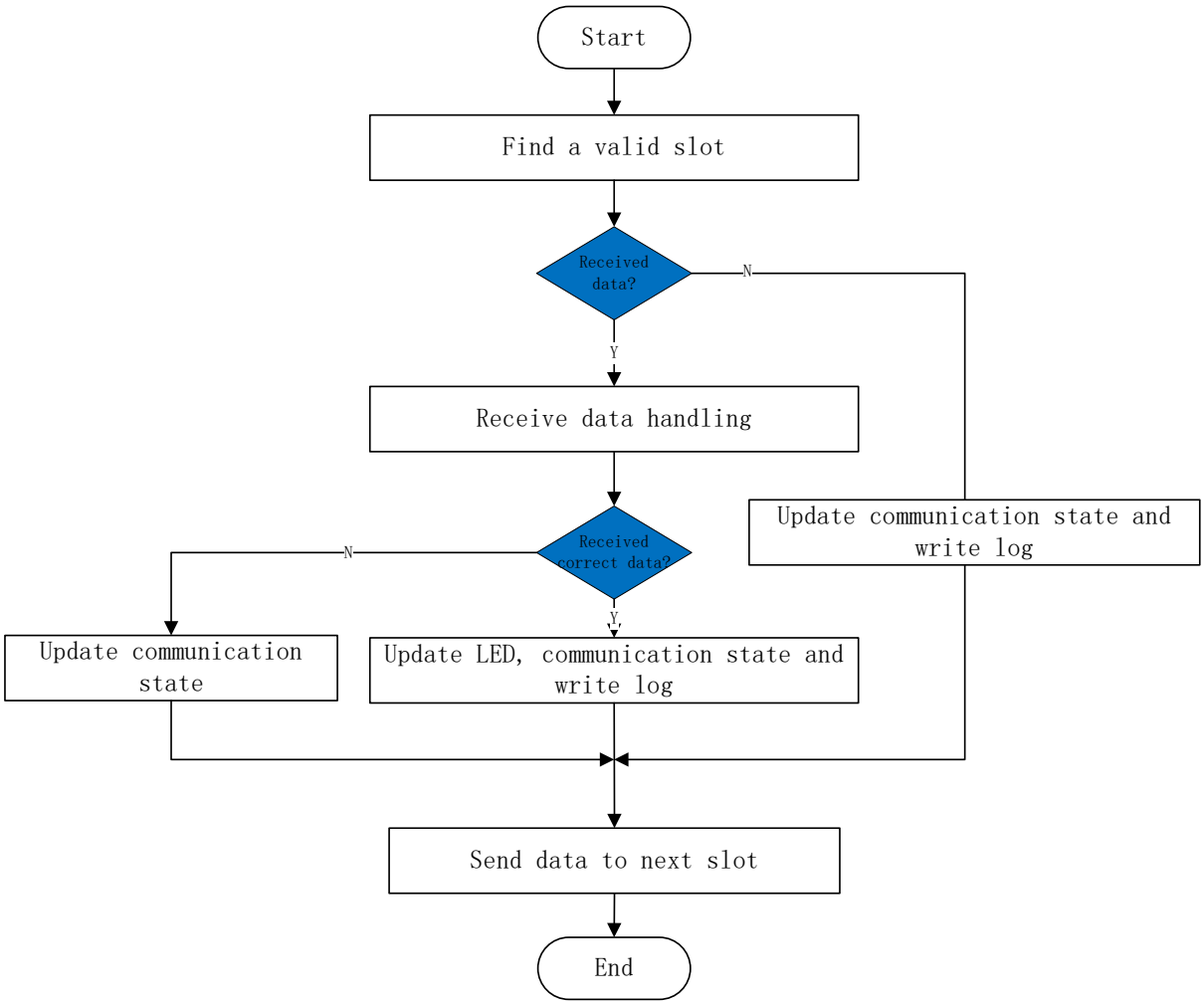

Figure 4-2 internal communication cycle processing flow (configured)

图 4-2 内部通讯周期运行流程图（有配置信息）

4.2.3 CMBusProtocolCycleWithoutCfg

4.2.3.1 Function Description 功能描述

This function is used to process the received data from CS1131.

本函数用于周期处理来自 CS1131 的数据。

4.2.3.2 Argument Description 参数说明

- Function Definition 函数定义

static void CMBusProtocolCycleWithoutCfg(void);

- Input argument 输入参数

No

无

- Output argument 输出函数

No.

无。

4.2.3.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.2.4 CMBusProtocolReceive

4.2.4.1 Function Description 功能描述

This function is used to receive the request or response from SOE software, CS1131 software, AMS software, OPC Server, Diagnostic Software, NTP/SNTP timing server, Modbus, P2P etc. which is transferred by CM.

本函数用于周期接收由 CM 转发的 SOE 软件、CS1131 软件、AMS 软件、OPC SERVER、诊断软件、NTP/SNTP SERVER、Modbus、P2P 等的请求或应答。

4.2.4.2 Argument Description 参数说明

- Function Definition 函数定义

static bool_t CMBusProtocolReceive (PMRecvBuffer_t *pstRecvBuf, CMSlot_t emCMSlot);

- Input argument 输入参数

PMRecvBuffer_t *pstRecvBuf

Point to receive buffer

指向接收缓冲区

CMSlot_t emCMSlot

CM’s slot

CM 的槽号。

- Output argument 输出函数

No.

无。

4.2.4.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.2.5 CMBusProtocolSend

4.2.5.1 Function Description 功能描述

This function is used to send the response from SOE software, CS1131 software, AMS software, OPC Server, Diagnostic Software, NTP/SNTP timing server, Modbus, P2P etc. to CM.

本函数用于周期发送对 SOE 软件、CS1131 软件、AMS 软件、OPC SERVER、诊断软件、NTP/SNTP SERVER、Modbus、P2P 等的应答到 CM。

4.2.5.2 Argument Description 参数说明

- Function Definition 函数定义

static void CMBusProtocolSend(PMSendBuffer_t *pstSendBuf, CMSlot_t emCMSlot);

- Input argument 输入参数

PMSendBuffer_t *pstSendBuf

Point to send buffer

指向发送缓冲区

CMSlot_t emCMSlot

CM's slot

CM 的槽号

- Output argument 输出函数

No.

无。

4.2.5.3 Processing flow 处理流程

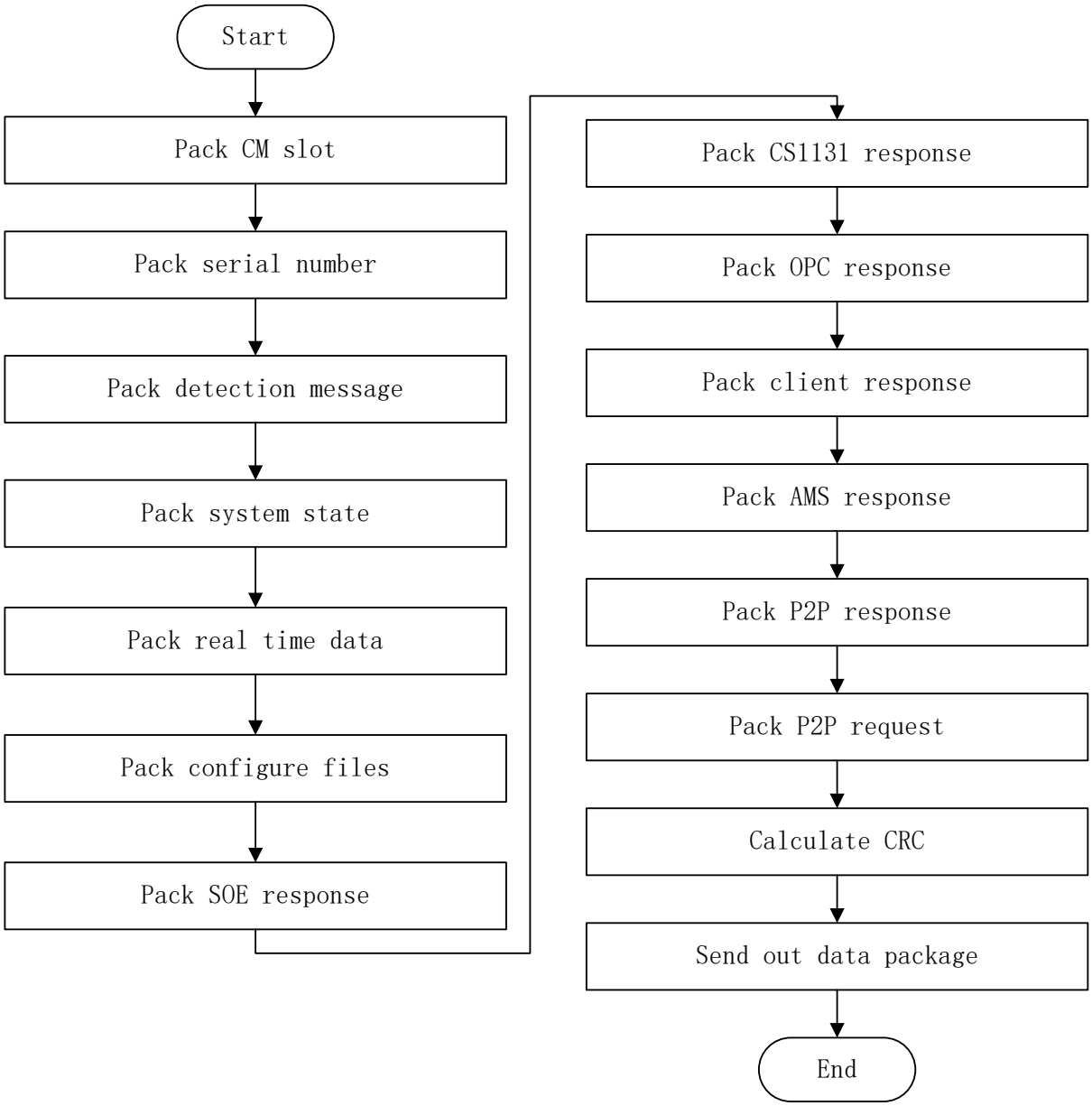

Figure 4-3 send data processing flow

图 4-3 发送数据处理流程图

4.3 PM_BUS communication interface PM_BUS 通讯接口

SWDD-PM-CP_NSafR_SecR_A_003

4.3.1 PMBUS_Receive

4.3.1.1 Function Description 功能描述

PM_BUS receive function.

PM_BUS 接收函数。

4.3.1.2 Argument Description 参数说明

- Definition 函数定义

int16_t PMBUS_Receive(int8_t bus_no, int8_t buf_no, uint8_t *buf)

- Input argument 输入参数

bus_no: bus No.

buf_no: buffer type

buf: buffer address

- Output argument 输出函数

return: data length

4.3.1.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.3.2 PMBUS_Send

4.3.2.1 Function Description 功能描述

PM_BUS send function.

PM_BUS 发送函数。

4.3.2.2 Argument Description 参数说明

- Definition 函数定义

int16_t PMBUS_Send(int8_t bus_no, int8_t buf_no, uint8_t *buf, int32_t len, int32_t timeout)

- Input argument 输入参数

bus_no: bus No.

buf_no: buffer type

buf: buffer address

timeout: time out parameter

- Output argument 输出函数

return: data length

4.3.2.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.4 IP_BUS communication interface IP_BUS 通讯接口

SWDD-PM-CP_NSafR_SecR_A_004

4.4.1 IPBUSMailSend

4.4.1.1 Function Description 功能描述

Mailbox send function.

邮箱发送函数。

4.4.1.2 Argument Description 参数说明

- Definition 函数定义

uint8_t IPBUSMailSend(uint8_t ucMailSend[], IPBUSAPPMailHeader_t *pstMailHeader);

- Input argument 输入参数

ucMailSend: data buf.

pstMailHeader: mailbox header

- Output argument 输出函数

return: send state

4.4.1.3 Processing flow 处理流程

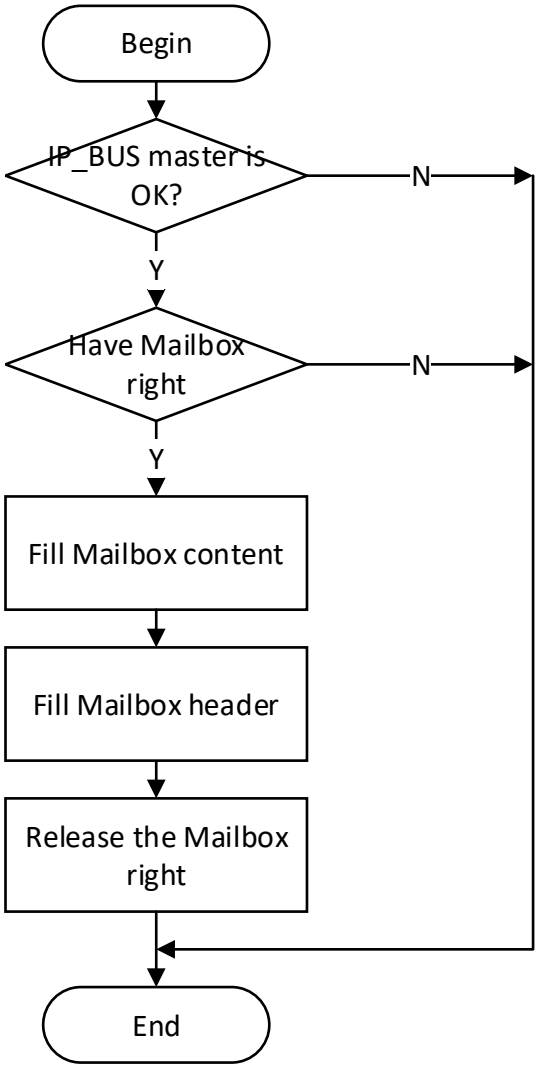

Figure 4-4 mailbox send data processing flow

图 4-4 邮箱发送处理流程图

4.4.2 IPBUSMailRecv

4.4.2.1 Function Description 功能描述

Mailbox receive function.

邮箱接收函数。

4.4.2.2 Argument Description 参数说明

- Definition 函数定义

uint8_t IPBUSMailRecv(uint8_t ucMailRecv[], IPBUSAPPMailHeader_t *pstMailHeader);

- Input argument 输入参数

ucMailRecv: data buf.

pstMailHeader: mailbox header

- Output argument 输出函数

return: receive state

4.4.2.3 Processing flow 处理流程

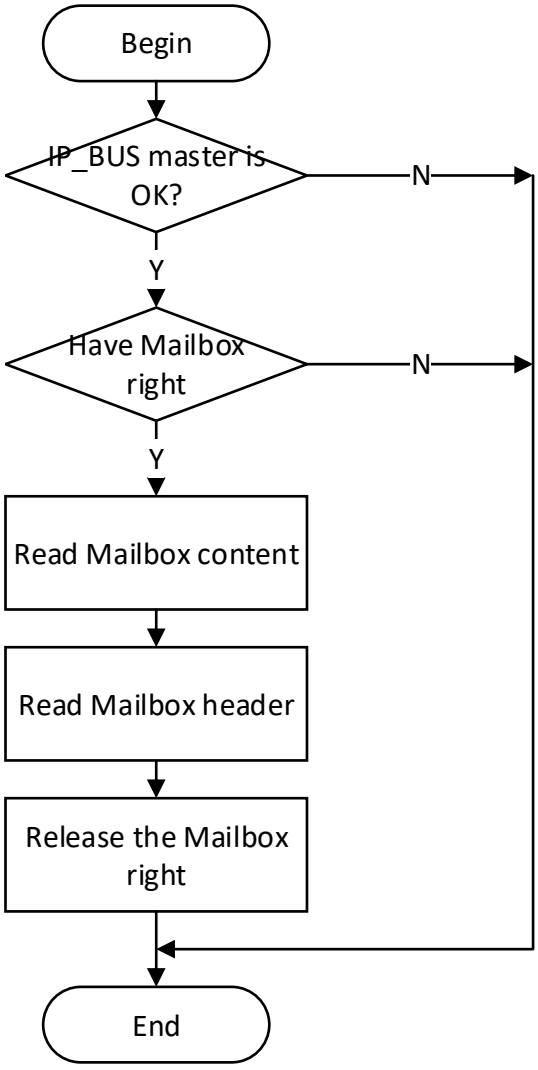

Figure 4-5 mailbox receive data processing flow

图 4-5 邮箱接收处理流程图

4.4.3 IPBUSReleaseDataRAMRight

4.4.3.1 Function Description 功能描述

Release data right.

释放权限

4.4.3.2 Argument Description 参数说明

- Definition 函数定义

void IPBUSReleaseDataRAMRight(void);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

No.

无。

4.4.3.3 Processing flow 处理流程

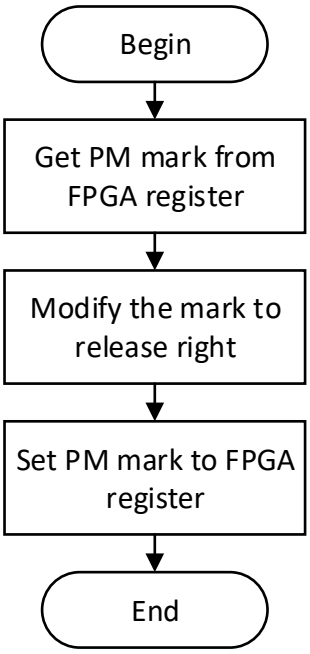

Figure 4-6 release right processing flow

图 4-6 释放权限流程图

4.4.4 IPBUSSetAPPState

4.4.4.1 Function Description 功能描述

Set the PM state of IP_BUS.

设置 IP_BUS 中 PM 状态。

4.4.4.2 Argument Description 参数说明

- Definition 函数定义

void IPBUSSetAPPState(uint16_t usState);

- Input argument 输入参数

usState: PM state of IP_BUS

➢ Output argument 输出函数

No.

无。

4.4.4.3 Processing flow 处理流程

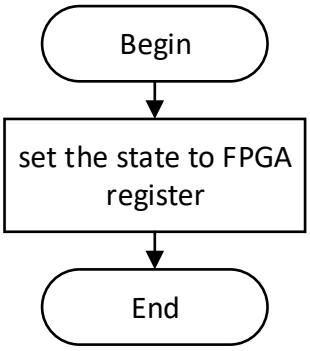

Figure 4-7 set PM state processing flow

图 4-7 设置 PM 状态流程图

4.4.5 IPBUSGetAPPState

4.4.5.1 Function Description 功能描述

Get the PM state of IP_BUS

读取 IP_BUS 中 PM 状态

4.4.5.2 Argument Description 参数说明

➢ Definition 函数定义

uint16_t IPBUSGetAPPState(void);

➢ Input argument 输入参数

No.

无。

➢ Output argument 输出函数

Return: PM state of IP_BUS

4.4.5.3 Processing flow 处理流程

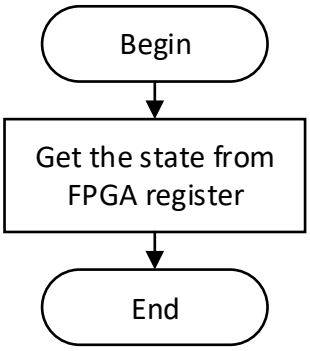

Figure 4-8 get PM state processing flow

图 4-8 读取 PM 状态流程图

——以下无正文