Document Title: PM_FW system resource management module design description of Safety Control System

Document Number: 16-Q04-000135

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 系统资源管理模块设计说明书

文件编号：16-Q04-000135

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 system resource 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 系统资源模块的设计方案。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 system resource module (marked red) in the software hierarchy is shown below.

系统资源模块（标红）在软件层次中的位置如下图所示。

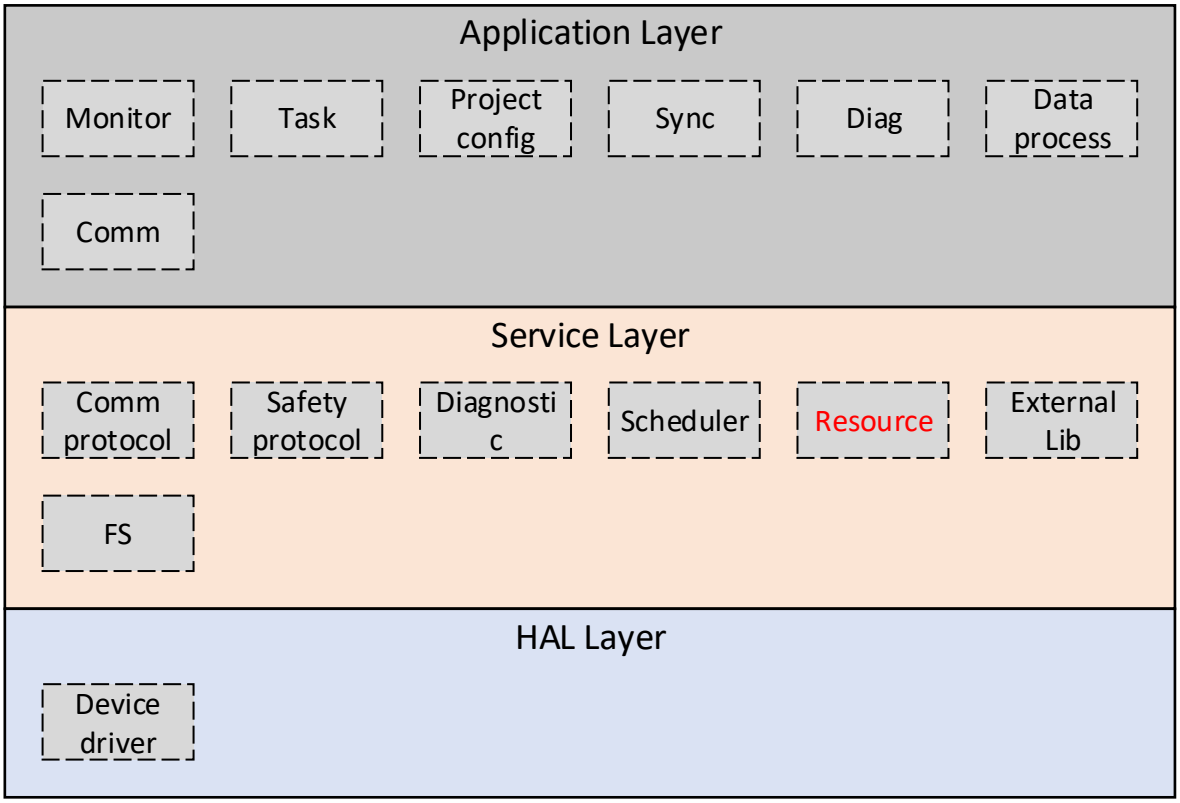

Figure 2-1 the location of the system resource module

图 2-1 模块位置

The system resource management module defines massive set and get interface, and other modules access the global variables via set/get interface.

系统资源模块中定义了大量的 set 和 get 接口函数，其它模块通过这些接口函数实现对全

局变量的访问操作。

3 Module design 模块设计

3.1 Function description 功能描述

1) Common interface (core0/core1) 公共接口（core0/core1）

   It provides the following function: read/write log, read/write shared memory, read/write configuration information, read/write system management state, read/write sync command, and PM's version related function.

   其提供如下功能：读写日志，读写共享内存，读写配置信息，读写系统管理状态，读写同步命令，以及 PM 版本相关接口函数。

2) Real time data (core0) 实时数据（core0）

   Real time data operation interface.

   操作实时数据的接口函数

3) File manager (core1) 文件管理（core1）

   Read or write file.

   读写文件

4) LED manager (core1) LED 状态灯管理（core1）

5) internal communication resource (core1) 内部通讯资源

   Read or write SOE/ CS1131 (Configuration Software) /OPC/client/AMS/internal command request or response

   读写 SOE/ CS1131（组态软件）/OPC/client/AMS/internal command 的请求或应答。

6) Real time data (core1) 实时数据（core1）

   Read/write real time data

   读写实时数据

7) Configuration file (core1) 配置文件（core1）

   Read/write configuration file

   读写配置文件

8) State management (core1) 状态管理（core1）

9) Sync (core1) 同步（core1）

3.2 Design concept 设计思路

The system resource module defines massive set and get interface, and other modules access the system resource (static variables) via set/get interface.

系统资源管理模块中定义了大量的 set 和 get 接口函数，其它模块通过这些接口函数实现对系统资源的访问操作。

3.3 Interface function 接口函数

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

1. void LogInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Log initial 日志初始化

2. void LogWrite(uint32_t usLogID);

Input argument 输入参数	Output argument 输出参数	Description 描述
uint32_t usLogID Log's ID 日志的 ID		Write log 写日志

3. int32_t LogRead(LogInfo_t *pstLogInfo, uint16_t usLogicPos, uint32_t uiLogType, puint32_t puiNextIndex);

Input argument 输入参数	Output argument 输出参数	Description 描述
uint16_t usLogicPos Log's position 日志位置 uint32_t uiLogType Log's type 日志类型	LogInfo_t *pstLogInfo Point to log information 指向日志信息 puint32_t puiNextIndex Point to next log's index 指向下一条日志的索引号	Read log 读日志

4. void LEDInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	LED initialization 状态灯初始化

5. LEDState_t GetLEDState(LEDName_t emLED);

Input argument 输入参数	Output argument 输出参数	Description 描述
LEDName_t emLED The name of the LED LED 灯的名称	No 无	Get the state of the LED 获取 LED 灯的状态

6. void SetLEDState(LEDName_t emLED, LEDState_t emLEDState);

Input argument 输入参数	Output argument 输出参数	Description 描述
LEDName_t emLED The name of the LED LED 灯的名称 LEDState_t emLEDState The state of the LED LED 灯的状态	No 无	Set the state of the LED 设置 LED 灯的状态

7. void UpdateLEDRunningState(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Update the running state of LEDs 更新正在运行 LED 灯的状态

8. void SharedCommInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No	No	Communication initialization

无	无	between core0 and core1. Core0 与 core1 间的通讯初始化

9. bool_t SysGetSMMsg (SMMsgType_t emMsgType, CMController_t emCMIndex,puint8_t pucContent, puint16_t pusLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
SMMsgType_t emMsgType Message type 消息类型 CMController_t emCMIndex CM's ID CM 的 ID	puint8_t pucContent Point to the content of the message 指向消息内容 puint16_t pusLen Point to the length of the message 指向消息长度	Get message in shared memory 从共享内存中获取消息

10. bool_t SysSetSMMsg (SMMsgType_t emMsgType , CMController_t emCMIndex,puint8_t pucContent, uint16_t usLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
SMMsgType_t emMsgType Message type 消息类型 CMController_t emCMIndex CM's ID CM 的 ID puint8_t pucContent Point to the content of the message 指向消息内容 uint16_t usLen The length of the message 消息长度	No 无	Set message in shared memory 向共享内存中写入消息

11. void SharedConfigInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Configuration initialization in shared memory 共享内存中的配置信息初始化

12. CtrlStationConfigInfo_t* SharedGetCtrlStationInfo(Task_ID_t emTaskID);

Input argument 输入参数	Output argument 输出参数	Description 描述
Task_ID_t emTaskID Task ID	No 无	Get controller station configure information from shared memory 从共享内存获取控制站配置信息

13. CMConfigInfo_t* SharedGetCMConfigInfo(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Get the configuration of CM from shared memory 从共享内存获取 CM 的配置信息

14. void SharedRetainInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Retain data initialization in shared memory 掉电保持数据初始化

15. bool_t SharedSetTask0RetainData(puint8_t pucBuf, uint16_t usLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
puint8_t pucBuf Point to the content of retain data 指向掉电保持的内容	No 无	Write retained data to SRAM 将掉电保持数据写入 SRAM

uint16_t usLen the length of retain data 掉电保持数据长度		

16. bool_t SharedGetTask0RetainData(puint8_t pucBuf, uint16_t usLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
uint16_t usLen the length of retain data 掉电保持数据长度	puint8_t pucBuf Point to the content of retain data 指向存储掉电保持的缓冲区	Read retained data from SRAM 从掉电保持数据读取 SRAM

17. void SharedRtDataInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Real time data initialization in shared memory 初始化共享内存中的实时数据

18. puint8_t SysGetSharedMemRtDataStartAddr(SysRtDataType_t emDataType);

Input argument 输入参数	Output argument 输出参数	Description 描述
SysRtDataType_t emDataType The type of real time data 实时数据类型	No 无	Get the start address of real time data in shared memory 获取共享内存中的实时数据起始地址

19. void SharedSOEInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	SOE initialization in shared memory 初始化共享内存中的 SOE

20. bool_t SharedSetSOE(uint8_t ucQueueIndex, SoftSOE_t stSOE);

Input argument	Output argument	Description

输入参数	输出参数	描述
uint8_t ucQueueIndex The index of SOE queue SOE 队列索引号 SoftSOE_t stSOE The content of SOE SOE 内容	No 无	Write SOE to the related queue 向相应的队列中写入 SOE

21. int32_t SharedGetSOE(uint8_t ucQueueIndex, SoftSOE_t* pstSOE, uint16_t usPos, puint32_t puiNextIndex);

Input argument 输入参数	Output argument 输出参数	Description 描述
uint8_t ucQueueIndex the index of SOE queue SOE 队列索引号 uint16_t usPos the position of the SOE in the queue SOE 在队列中的位置	SoftSOE_t* pstSOE Point to SOE 指向 SOE puint32_t puiNextIndex Point to the next index of SOE 指向下一条 SOE	Get SOE from SOE queue 从 SOE 队列中读取 SOE

22. PMController_t SysGetLocalPMID(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Get local PM's ID 获取本地 PM 的 ID 号

23. uint8_t SysGetIECShedCtrlFlg(Task_ID_t emTaskID);

Input argument 输入参数	Output argument 输出参数	Description 描述
Task_ID_t emTaskID Task ID	No 无	Get user task control flag 获取用户任务控制标志

24. void SysSetIECShedCtrlFlg(Task_ID_t emTaskID, uint8_t ucIECShedCtrlFlg);

Input argument 输入参数	Output argument 输出参数	Description 描述
Task_ID_t emTaskID, Task ID uint8_t ucIECShedCtrlFlg control flag 控制标志	No 无	Set user task control flag 设置用户任务控制标志

25. void CommSysSyncInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Sync initialization 同步初始化

26. void SysGetSyncTime(SyncTime64_t* punSyncTime);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	SyncTime64_t* punSyncTime Point to sync time 指向同步时间	Get sync time 获取同步时间

27. void SysSetSyncTime(SyncTime64_t* punSyncTime);

Input argument 输入参数	Output argument 输出参数	Description 描述
SyncTime64_t* punSyncTime Point to sync time 指向同步时间	No 无	Set sync time 设置同步时间

28. uint16_t GetSwitchKeyPos(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Get switch key position 获取钥匙开关位置

29. uint16_t GetLocalPMAddr(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Get local PM's address 获取本地 PM 的地址

30. void SysInit (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。

31. uint16_t SysGetPMState(PMController_t emPMID);

Input argument 输入参数	Output argument 输出参数	Description 描述
emPMID. PM's ID PM 的 ID 号。	No 无	Get the related PM's state 获取相关 PM 的状态

32. bool_t SysSetPMState(PMController_t emPMID, uint16_t usPMState);

Input argument 输入参数	Output argument 输出参数	Description 描述
emPMID. PM's ID PM 的 ID 号。 usState. PM's state PM 的状态。	No 无	Set the related PM's state 设置相关 PM 的状态

33. void SysGetSysCMState(SysCMState_t *pstSysCMState);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	SysCMState_t *pstSysCMState Point to CM's state 指向 CM 的状态	Get CM's state 获取 CM 的状态

34. bool_t SysSetCMState(CMController_t emCMID, uint16_t usCMState);

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

emCMID. PM 的 id 号。 PM's ID usCMState. CM's state CM 的状态。	 No 无	 Set the related CM's state 设置相关 CM 的状态

35. bool_t SysSetSysState(SysState_t *pstSysState);

Input argument 输入参数	Output argument 输出参数	Description 描述
SysState_t *pstSysState Point to system state 指向系统状态	 No 无	 Set system state 设置系统状态

36. uint16_t SysGetCMBusCommState(CMController_t emCMID, PMController_t emPMID);

Input argument 输入参数	Output argument 输出参数	Description 描述
emCMID. CM's ID CM 的 ID 号。 emPMID. PM's ID PM 的 ID 号。	 No 无	 Get internal communication state between CM and PM 获取 PM 与 CM 间的内部通讯 状态

37. bool_t SysSetCMBusCommState(CMController_t emCMID, PMController_t emPMID, uint16_t usInterState);

Input argument 输入参数	Output argument 输出参数	Description 描述
emCMID. CM's ID CM 的 ID 号。 emPMID. PM's ID PM 的 ID 号。 usInterState Internal communication state	 No 无	 Set internal communication state between CM and PM 设置 PM 与 CM 间的内部通讯 状态

38. SysSystemTime_t SysGetPMSystemTime(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Get system time in PM 获取 PM 中的系统时间

39. bool_t SysSetCMSystemTime(CMController_t emCMID, SysSystemTime_t *pstSysTime);

Input argument 输入参数	Output argument 输出参数	Description 描述
emCMID. CM's ID CM 的 ID 号。 pstSysTime point to system time 指向系统时间	No 无	Set system time in CM 设置 CM 中的系统时间

40. SysSystemTime_t SysGetCMSystemTime(CMController_t emCMID);

Input argument 输入参数	Output argument 输出参数	Description 描述
CMController_t emCMID CM's ID CM 的 ID 号	No 无	Get CM's system time 获取来自 CM 的系统时间

41. void FileManagerInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	File management initialization 文件管理初始化

42. void FileManagerCycle(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	File management operation 文件管理操作

43. void LEDManagerInit(void);

Input argument	Output argument	Description

输入参数	输出参数	描述
No 无	No 无	LED management initialization LED 管理初始化

44. void LEDManagerCycle(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	LED management operation LED 管理操作

45. void SysCfgFileInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Configuration file initialization 配置文件初始化

46. uint16_t SysGetCfgFileInfo (SysCfgFileType_t emCfgFileType, uint32_t uiRdOffset, puint8_t pucContent, uint16_t usLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
SysCfgFileType_t emCfgFileType Configure file type 配置文件类型 uint32_t uiRdOffset Read offset 读偏移地址 uint16_t usLen	puint8_t pucContent Point to the content buffer 指向读取的内容缓冲区	Get the content from configure file 从配置文件中读取内容

47. uint16_t SysSetCfgFileInfo(SysCfgFileType_t emCfgFileType, puint8_t pucContent, uint16_t usLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
SysCfgFileType_t emCfgFileType Configure file type	No 无	Write content to configuration file 向配置文件中写入内容

配置文件类型 uint32_t uiRdOffset Read offset 读偏移地址 puint8_t pucContent Point to the content buffer 指向写入的内容缓冲区 uint16_t usLen		

48. bool_t SysDeleteCfgFile(SysCfgFileType_t emCfgFileType);

Input argument 输入参数	Output argument 输出参数	Description 描述
SysCfgFileType_t emCfgFileType Configure file type 配置文件类型	No 无	Delete configuration file 删除配置文件

49. void SysCMBusInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	CM bus initialization CM bus 初始化

50. bool_t SysGetMsg (CMController_t emCMID, SysMsgType_t emMsgType, puint8_t pucContent, puint16_t pusLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
emCMID CM's ID CM 的 ID 号 emMsgType Message type 消息类型	pucContent point to message content 指向消息内容 pusLen point to message length 指向消息长度	Get message 获取消息

51. bool_t SysSetMsg (CMController_t emCMID, SysMsgType_t emMsgType , puint8_t

pucContent, uint16_t usLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
emCMID CM's ID CM 的 ID 号 emMsgType Message type 消息类型 pucContent point to message content 指向消息内容 pusLen point to message length 指向消息长度	No 无	Set message 存储消息

52. void SysFileManagerInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	File management initialization 文件管理初始化

53. bool_t SysWriteFile(int32_t iFd, uint8_t ucContent[], uint16_t usLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
int32_t iFd, file descriptor 文件描述符 uint8_t ucContent[], the content will be written 待写入的内容 uint16_t usLen the length of	No 无	Write file 写文件

content 内容长度		

54. bool_t SysReadFile(int32_t iFd, uint16_t usLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
int32_t iFd, file descriptor 文件描述符 uint16_t usLen the length of content 内容长度	No 无	Read file 读文件

55. void SysModbusInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Modbus initialization Modbus 初始化

56. bool_t SysGetModbusMsgQueue (CMController_t emCMID, puint8_t pucContent, puint16_t pusLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
CMController_t emCMID CM's ID CM 的 ID	puint8_t pucContent Point to the content buffer 指向读出内容缓冲区 puint16_t pusLen Point to the length of content 指向内容长度	Get Modbus message from queue 从 Modbus 队列中读取出 Modbus 消息

57. bool_t SysSetModbusMsgQueue(CMController_t emCMID, puint8_t pucContent, uint16_t usLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
CMController_t	No	Set Modbus message to queue

emCMID CM's ID CM 的 ID puint8_t pucContent Point to the content buffer 指向写入内容缓冲区 puint16_t pusLen Point to the length of content 指向内容长度	无	向 Modbus 队列中写入 Modbus 消息

58. void SysSwitchPosInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Switch key position initialization 钥匙开关初始化

59. uint32_t SysReadSwitchPos(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Get switch key position

60. void SysP2PInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	P2P initialization P2P 初始化

61. bool_t SysGetP2PMsg (CMController_t emCMID, SysP2PMsgType_t emMsgType, puint8_t pucContent, puint16_t pusLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
CMController_t	puint8_t pucContent	Get P2P message from queue

emCMID CM's ID CM 的 ID SysP2PMsgType_t emMsgType Message type 消息类型	Point to the content buffer 指向读出内容缓冲区 puint16_t pusLen Point to the length of content 指向内容长度	从队列中读取出 P2P 消息

62. bool_t SysSetP2PMsg (CMController_t emCMID, SysP2PMsgType_t emMsgType , puint8_t pucContent, uint16_t usLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
CMController_t emCMID CM's ID CM 的 ID SysP2PMsgType_t emMsgType Message type 消息类型 puint8_t pucContent Point to the content buffer 指向写入内容缓 冲区 uint16_t pusLen the length of content 内容长度	No 无	Set P2P message to queue 向队列中写入 P2P 消息

63. void SysRTDataInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Real time data initialization 实时数据初始化

64. bool_t SysReadRTData(SysRtDataAreaType_t emDataArea, puint8_t pucContent, uint16_t

usLen, puint16_t pusRdLen);

Input argument 输入参数	Output argument 输出参数	Description 描述
SysRtDataAreaType_t emDataArea The type of data area 数据区类型 uint16_t usLen pre-read data length 预读取数据长度	puint8_t pucContent Point to the readout data buffer 指向读出的数据缓冲区 puint16_t pusRdLen Point to the length of readout data 指向读出的数据长度	Read real time data 读取实时数据

65. void SysStateMngtInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	System state management initialization 系统状态管理初始化

66. void SysSetState(sys_state_t emSysState);

Input argument 输入参数	Output argument 输出参数	Description 描述
sys_state_t emSysState System state 系统状态	No 无	Set system state 设置系统状态

67. sys_state_t SysGetState(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No 无	No 无	Get system state 获取系统状态

68. void SysSyncInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No	No	Synchronize initialization

无	无	同步初始化

3.4 Global variable 全局变量

Table 3-1 Global variable list

表 3-1 全局变量列表

No. 序号	Type 变量类型	Name 名称	Description 描述
1.	LogQueue_t	s_stQueueInfo	Log queue information. 日志队列信息
2.	LEDState_t	s_emLEDStatus [NUM_OF_LED]	LED state 记录状态灯状态
3.	SMMsgBox_t	s_stSMMsgBox [NUM_OF_CM] [NUM_OF_SM_MSGTYPE]	Core0-core1 communication message box Core0 和 core1 间通讯的信箱
4.	CodeArea_t	s_stCodeArea	It is used to store user code 用于存储用户代码
5.	SharedRTDataArea_t	g_stSharedRtData	Real time data in shared memory 共享区实时数据
6.	SOEQueue_t	stSOEQueue [MAX_SOE_QUEUE_NUM]	SOE queue SOE 队列
7.	SoftSOE_t	stSoftSOEArray [MAX_SOE_NUM_DDR]	Soft SOE 软 SOE
8.	PRG_RTDataArea_t	s_stPRGRtData	Real time data in core0 Core0 所使用的实时数据
9.	SysFMReq_t	s_stSysFMReq	Request from RTS task 来自 RTS 任务的请求
10.	SysFMAck_t	s_stSysFMAck	Response to RTS task 发送给 RTS 任务的应答
11.	SysCfgDB_t	s_stSysCfgDB	Configuration files are

		[NUM_OF_CFG_DB]	stored in database 用于存储配置文件的数据库
12.	SysMsgBox_t	s_stSysMsgBox [NUM_OF_CM] [NUM_OF_MSGTYPE];	Store SOE/CS1131 message etc. 存储 SOE/CS1131 信息等
13.	SysState_t	s_stSysState	System state 系统状态
14.	ProjVerInfo_t	s_stProjVer[NUM_OF_CM]	Project version information 工程版本信息
15.	SysCM2PM_t	s_stCM2PMInfo [NUM_OF_CM]	CM to PM information CM 传输到 PM 的信息
16.	SysFMReqMsgBox_t	s_stFMReqMsgBox	Store request message 存放请求消息
17.	SysFMAckMsgBox_t	s_stFMAckMsgBox	Store acknowledge message 存放应答消息
18.	SysModbusMsgQueue_t	s_stSysModbusMsgQueue [NUM_OF_CM]	Modbus message is stored in the queue Modbus 消息存放在该队列中
19.	float_t	s_fTemperature	Temperature 温度
20.	Alarm_t	s_emSupplyAlarmBit [NUM_OF_SUPPLY_TYPE]	Alarm bit 报警位
21.	uint32_t	s_uiPMHardStatus [NUM_OF_PM]	PM's hardware status PM 的硬件状态
22.	SysP2PMsgBox_t	s_stSysP2PMsgBox [NUM_OF_CM] [NUM_OF_P2PMSGTYPE]	P2P message box 存放 P2P 消息
23.	RTDataArea_t	s_stRTDataArea [NUM_OF_RTDATA_AREA]	Real time data area 实时数据区
24.	SharedAppend_t	s_stAppendInfo	Appended information area 附加数据区

25.	SysPMStateInfo_t	s_stPMStateInfo [NUM_OF_PM]	PM's state information PM 的状态信息
26.	sys_state_t	s_emSysState	System state 系统状态
27.	SysSyncSrvQueue_t	s_stSysSyncSrvQueue	The queue is used to synchronize service data 服务同步队列
28.	SysSyncPrjQueue_t	s_stSysSyncPrjQueue	The queue is used to synchronize project 工程同步队列
29.	SysSrvInfo_t	s_stSysSrvInfo [NUM_OF_CM] [NUM_OF_SRVTYPE]	Store service data information 同步服务数据存放结构

3.5 Data structure 数据结构

1. Log 日志

```
typedef struct LogInfoTag
{
    uint32_t uiSec;       /*second start time:1970.01.01 00:00:00 */
    uint32_t uiMicroSec; /*millisecond 时间戳 -微秒 */
    uint32_t uiLogID;   /*log id 不同的ID代表不同的含义*/
}LogInfo_t;
typedef struct LogQueueTag
{
    uint16_t usMagic1;   /* magic 幻数*/
    uint16_t usSize;       /*size 队列大小 */
    uint16_t usHeader;     /*header 队头*/
    uint16_t usRetainHeader;     /*retain header 掉电保持队头*/
    uint16_t usTail;       /*tail 队尾 */
    uint16_t usQueueState;/*queue state: empty or full 队列空满状态标识 */
    uint16_t usMagic2; /* magic 幻数 */
}LogQueue_t;
```

2. LED 状态灯

```
typedef enum LEDNameTag
{
    LED_NAME_FORCE = 0,                    /* FORCE灯*/
    LED_NAME_SYNC,                         /* SYNC灯 */
    LED_NAME_ACTIVE,                       /* ACTIVE灯 */
```

```
    LED_NAME_FAULT,                          /* FAULT灯  */
    LED_NAME_PASS,                           /* PASS灯  */

    LED_NAME_INIT,                           /* INIT灯  */
    LED_NAME_PROG,                           /* PROG灯  */
    LED_NAME_RUN,                            /* RUN灯  */
    LED_NAME_STOP,                           /* STOP灯  */

    LED_NAME_CMBUS,                          /* CM_BUS灯  */
    LED_NAME_PMBUS,                          /* PM_BUS灯  */
    LED_NAME_IPBUS,                          /* IP_BUS灯  */

    NUM_OF_LED,                              /*  状态灯个数  */
    LED_NAME_ILLEGAL
} LEDName_t;

typedef enum LEDStateTag
{
    LED_OFF = 0,                             /*off  灭*/
    LED_ON,                                  /*on  亮  */
    LED_BLINK,                               /*blink  闪烁  */
    NUM_OF_LED_STATE,                        /*led state  状态  */
    LED_ILLEGAL_STATE                        /*invalid state  非法状态  */
} LEDState_t;
```

3. Core0-core1 communication core0-core1 通讯

```
/* message  消息*/
typedef struct SMMsgTag
{
    uint16_t usLen;                          /*message length  消息内容长度  */
    uint16_t usReserved;
    uint8_t ucContent[MAX_SM_MSG_LEN];       /*message content  存放消息内容
*/
} SMMsg_t;

/* message box  信箱  */
typedef struct SMMsgBoxTag
{
    uint32_t uiLock;                         /* box is locked or not  信箱是否可以
被访问  */
    bool_t bMsgBoxStatus;                    /* box state: empty or full  信箱的两种
状态：空或满  */
    uint8_t ucReserved[3];
    SMMsg_t stMsg;                           /* message  消息  */
```

```
} SMMsgBox_t;

/* message type (request/response)消息类型（请求或应答）*/
typedef enum SMMsgTypeTag
{
    SM_SOE_REQ = 0,                          /* request  请求  */
    SM_SOE_RESP = 1,                         /* response  应答  */
    SM_CS1131_REQ = 2,                       /* request  请求  */
    SM_CS1131_RESP = 3,                      /* response  应答  */
    SM_OPC_REQ = 4,                          /* request  请求  */
    SM_OPC_RESP = 5,                         /* response  应答  */
    SM_CLIENT_REQ = 6,                       /* request  请求  */
    SM_CLIENT_RESP = 7,                      /* response  应答  */
    SM_AMS_REQ = 8,                          /* request  请求  */
    SM_AMS_RESP = 9,                         /* response  应答  */
    SM_P2P_REQ = 10,                         /* request  请求  */
    SM_P2P_RESP = 11,                        /* response  应答  */
    SM_MODBUS_REQ = 12,                      /* request  请求  */
    SM_MODBUS_RESP = 13,                     /* response  应答  */
    SM_CTRLSRV_REQ = 14,                     /* request  请求  */
    SM_CTRLSRV_RESP = 15,                    /* response  应答  */
    NUM_OF_SM_MSGTYPE
} SMMsgType_t;
```

4. Real time data 实时数据

```
typedef struct SharedRTDataAreaTag
{
    bool_t bLock;                            /*data area is locked or not  是否可以
被访问:true-不允许访问  false-允许访问  */
    uint8_t ucReserved1[3];
    bool_t bStatus;                          /*state: read only/write only  两种状态：
只读或只写: true-只读  false-只写*/
    uint8_t ucReserved2[3];
    uint8_t ucG[MAX_RTDATA_G_AREA_LEN];      /*Area-G content G区表决后数
据内容  */
    uint8_t ucI[MAX_RTDATA_I_AREA_LEN];      /* Area-I content I区表决后数据
内容  */
    uint8_t ucQ[MAX_RTDATA_Q_AREA_LEN];      /* Area-Q content Q区表决后
数据内容  */
    SharedAppend_t stAppend;                 /*Append information  附加内容  */
}SharedRTDataArea_t;
```

5. SOE

```
/*Soft SOE structure  软SOE结构体定义*/
```

```
typedef struct SoftSOETag
{
    uint32_t uiSec;       /*second 时间戳 相对起点走过的秒 起点：1970.01.01 00:00:00 */
    uint32_t uiMicroSec; /*micro-second 时间戳 -微秒 */
    uint16_t usIndex;   /*index 序号 */
    uint8_t ucBlockNo; /*block number 块号 */
    uint8_t ucValue;     /*value 跳变信息 */
}SoftSOE_t;

/*Hard SOE structure 硬SOE结构体定义*/
typedef struct HardSOETag
{
    uint32_t uiSec;        /*second 时间戳 相对起点走过的秒 起点：1970.01.01 00:00:00 */
    uint32_t uiMicroSec;   /*micro-second 时间戳 -微秒*/
    uint8_t ucModNo;       /*module number 模块号 */
    uint8_t ucChannelNo;/*channel number 通道号*/
    uint8_t ucValue;       /*value 跳变信息 */
    uint8_t ucReserved; /*reserved 保留 */
}HardSOE_t;

#pragma pack (4)
/*SOE queue   SOE队列结构体定义*/
typedef struct SOEQueueTag
{
    uint32_t uiLockFlg;   /*Is it locked or not 共享内存锁定标识*/
    uint16_t usType;       /*type:1-cycle 2-history 类型：1循环覆盖   2不覆盖 */
    uint32_t uiBase;       /*base address 队列基地址 */
    uint16_t usSize;       /*queue size 队列大小 */
    uint16_t usHeader;     /*header 队头*/
    uint16_t usTail;       /*tail 队尾 */
    uint8_t ucQueueState;/*state: empty/full 队列空满状态标识 */
    uint8_t ucStorage;     /*storage: SRAM/DDR */
    bool_t bConfiged;      /*Is is configured or not 队列是否被配置 */
}SOEQueue_t;

typedef struct SOESRAMQueueTag
{
    uint16_t usHeader;     /*Header 队头*/
    uint16_t usTail;       /*Tail 队尾 */
    uint16_t usQueueState;/*state: empty/full 队列空满状态标识 */
    uint16_t usStorage;     /*storage: SRAM/DDR */
}SOESRAMQueue_t;

typedef struct SOESRAMQueuesInfoTag
{
```

```
    uint32_t uiFlag;
    SOESRAMQueue_t stSOESRAMQueue[MAX_SOE_QUEUE_NUM];
}SOESRAMQueuesInfo_t;
#pragma pack ()
```

6. Internal communication message 内部通讯消息

```
/*message 消息*/
typedef struct SysMsgTag
{
    uint16_t usLen;                          /*message length 消息内容长度 */
    uint8_t ucContent[MAX_MSG_LEN];          /*message content 存放消息内容 */
} SysMsg_t;

/* 信箱 */
typedef struct SysMsgBoxTag
{
    bool_t bLock;                            /*box is locked or not 信箱是否可以被访
问 */
    bool_t bMsgBoxStatus;                    /*state: empty/full 信箱的两种状态：空或
满 */
    SysMsg_t stMsg;                          /*message 消息 */
} SysMsgBox_t;

/*message type (request/response) 消息类型（请求或应答）*/
typedef enum SysMsgTypeTag
{
    SOE_REQ = 0,                             /*request 请求 */
    SOE_RESP = 1,                            /* response 应答 */
    CS1131_REQ = 2,                          /* request 请求 */
    CS1131_RESP = 3,                         /* response 应答 */
    OPC_REQ = 4,                             /* request 请求 */
    OPC_RESP = 5,                            /* response 应答 */
    CLIENT_REQ = 6,                          /* request 请求 */
    CLIENT_RESP = 7,                         /* response 应答 */
    AMS_REQ = 8,                             /* request 请求 */
    AMS_RESP = 9,                            /* response 应答 */
    INTER_CMD_REQ = 10,                      /* request 请求 */
    INTER_CMD_RESP = 11,                     /* response 应答 */
    DIAG_REQ = 12,                           /* request 请求 */
    DIAG_RESP = 13,                          /* response 应答 */
    NUM_OF_MSGTYPE
} SysMsgType_t;
```

7. PM /CM state PM/CM 状态

```
/* PM state PM状态  */
typedef struct SysPMStateTag
{
    /*PM state PM状态  */
    uint16_t usPMState[NUM_OF_PM];
    /*system time PM设置的系统时间  */
    SysSystemTime_t stPMtime[NUM_OF_PM];
} SysPMState_t;

/*CM state CM状态  */
typedef struct SysCMStateTag
{
    /*CM state CM状态  */
    uint16_t usCMState[NUM_OF_CM];
    /*internal communication state  内部通讯状态  */
    uint16_t usInterCommState[NUM_OF_CM][NUM_OF_PM];
    /*external communication state  各通讯口的外部通讯状态  */
    uint16_t usExterCommState[NUM_OF_CM][NUM_OF_NETWORK_PORT];
    /*system time CM设置的系统时间  */
    SysSystemTime_t stCMtime[NUM_OF_CM];
} SysCMState_t;
```

8. File management 文件管理

```
#pragma pack(1)
typedef struct SysFMReqTag
{
    uint8_t ucFileName[MAX_FILE_NAME_LEN];        /*file name  文件名字符串  */
    uint32_t uiCtrlWord;                          /*control word  控制字  */
    int32_t iFd;                                  /*file descriptor  文件描述符  */
    uint16_t usLen;                               /*file length  写文件内容长度  */
    uint8_t ucContent[MAX_FILE_MSG_LEN];          /*file content  存放待写入的文件内容
*/
} SysFMReq_t;

typedef struct SysFMReqMsgBoxTag
{
    bool_t bLock;                                 /*box is locked or not  信箱是否可以被访
问  */
    bool_t bStatus;                               /*state: empty/full  信箱的两种状态：空或
满  */
    SysFMReq_t stReqMsg;                          /*request message  请求消息  */
} SysFMReqMsgBox_t;

typedef struct SysFMAckTag
```

```
{
    uint32_t uiStatusWord;                          /*status word  状态字*/
    uint32_t uiErrCode;                             /*error code  错误码  */
    int32_t iFd;                                    /*file descriptor  文件描述符  */
    uint16_t usLen;                                 /*file length  读文件内容长度  */
    uint8_t ucContent[MAX_FILE_MSG_LEN];            /*file content  存放读出的文件内容  */
} SysFMAck_t;

typedef struct SysFMAckHeaderTag
{
    uint32_t uiStatusWord;                          /*status word  状态字*/
    uint32_t uiErrCode;                             /*error code  错误码  */
    int32_t iFd;                                    /*file descriptor  文件描述符  */
} SysFMAckHeader_t;

typedef struct SysFileMngtAckTag
{
    bool_t bLock;                                   /*message box is locked or not  信箱是否
可以被访问  */
    bool_t bStatus;                                 /*state: empty/full  信箱的两种状态：空或
满  */
    SysFMAck_t stAckMsg;                            /*responsed message  应答消息  */
} SysFMAckMsgBox_t;

#pragma pack()
```

9. Modbus information Modbus 信息

```
/* Modbus message  消息*/
typedef struct SysModbusMsgTag
{
    uint16_t usLen;                                 /*message length  消息内容长度  */
    uint8_t ucContent[MAX_MODBUS_MSG_LEN];          /*message content  存放消息内容
*/
} SysModbusMsg_t;

/*message queue  消息队列*/
typedef struct SysModbusMsgQueueTag
{
    uint16_t usHead;                                /*header  队列头部*/
    uint16_t usTail;                                /*tail  队列尾部  */
    uint16_t usNum;                                 /*element number  队列中元素个数  */
    SysModbusMsg_t stMsg[MAX_MODBUS_MSG_NUM];       /*message  消息  */
} SysModbusMsgQueue_t;
```

10. P2P information P2P 信息

```
/*P2P message 消息*/
typedef struct SysP2PMsgTag
{
    uint16_t usLen;                                  /*message length 消息内容长度 */
    uint8_t ucContent[MAX_P2P_MSG_LEN];              /*message content 存放消息内容 */
} SysP2PMsg_t;

/*P2P message box 信箱 */
typedef struct SysP2PMsgBoxTag
{
    bool_t bLock;                                    /*message box is locked or not 信箱是否
可以被访问 */
    bool_t bMsgBoxStatus;                            /*state: empty/full 信箱的两种状态：空或
满 */
    SysP2PMsg_t stMsg;                               /*P2P message 消息 */
} SysP2PMsgBox_t;

/*message type(request /response) 消息类型（请求或应答）*/
typedef enum SysP2PMsgTypeTag
{
    P2P_PM_REQ = 0,                                  /*PM send request PM发请求 */
    P2P_PM_RESP = 1,                                 /*PM receive response PM收应答 */
    P2P_EXT_REQ = 2,                                 /*receive external request 收来自外部
请求 */
    P2P_EXT_RESP = 3,                                /*send external response 发应答给外
部 */
    NUM_OF_P2PMSGTYPE
} SysP2PMsgType_t;
```

11. Real time data area(core1) 实时数据区 （core1）

```
/*real time data area 实时数据区 */
typedef struct RTDataAreaTag
{
    bool_t bLock;                                    /*data area is locked or not 是否可以
被访问:true-不允许访问 false-允许访问 */
    bool_t bDataAreaStatus;                          /*state: empty/full 两种状态：只读或
只写: true-只读 false-只写*/
    bool_t bReadFlag[MAX_RTDATA_AREA_NUM];           /*read flag: true-finished
false-not finished true:读完 false 未读完 */
    uint32_t uiReadOffset[MAX_RTDATA_AREA_NUM];      /*read offset 当前读操作的偏
移地址 */
    uint32_t uiLen[MAX_RTDATA_AREA_NUM];             /*G/I/Q area length G/I/Q区数
```

```
据长度 */
    uint8_t ucContent[MAX_RTDATA_G_AREA_LEN];          /*Area-G content G区数据内容
*/
    uint8_t ucInputContent[MAX_RTDATA_I_AREA_LEN];   /*Area-I content I区数据内容 */
    uint8_t ucOutputContent[MAX_RTDATA_Q_AREA_LEN]; /*Area-Q content Q区数据内容
*/
    uint8_t ucAppendInfo[MAX_RTDATA_APPEND_INFO_SIZE]; /*Append information 存放
SharedAppend_t结构体信息 */
} RTDataArea_t;
```

12. State management 状态管理

```
/*system state 系统状态 */
typedef enum
{
    SYS_STABLE_STATE = 0x00000000U,                    /*stable state 系统状态:
稳态 = 0 */
    SYS_UNSTABLE_STATE,                                /*unstable state 系统
状态:非稳态 = 1 */
    MAX_SYS_STATE
}sys_state_t;

/*PM hardware diagnosis state PM 硬件诊断状态 */
typedef enum
{
    HARD_OFFLINE = 0x00000000U,                        /*offline 硬件状态: 离
线 */
    HARD_ONLINE                                        /*online 硬件状态:
在线 */
}hard_status_t;

#pragma pack (1)


typedef struct SysPMStateInfoTag
{
    uint32_t ucPackNum;
    ActiveFlag_t emActiveFlag;
    poweron_flg_t emPoweronFlg;
    Key_Switch_Type_t emSwitchKeys;
    work_status_t emWorkStatus;
    sync_status_t emSyncStatus;
    download_flag_t emDownloadFlg;
    task_status_t emTaskStatus[MAX_TASK_NUM];
    PrjInfo_t stPrjInfo;                               /*current project information 当前使用工程
```

```
的工程信息*/
    PrjInfo_t stDLPrjInfo;                          /*download project information(not used yet)
下装工程（未生效），工程信息*/
    SysAjustSyncTime_t stAjustSyncTime;
}SysPMStateInfo_t;
pragma pack ()
```

13. sync 同步

```
typedef struct SysSyncTaskTag
{
    /*Source PM's ID 同步任务目的PM ID号 */
    PMController_t emPMId;
    /*destination PM's ID 同步任务来源CM ID号*/
    CMController_t emCMId;
    /*user task id (used for real time data sync)UP任务的ID号，用于实时数据同步*/
    Task_ID_t emUPTaskId;
    /*task id 任务号 */
    sync_task_id_t emTaskId;
    /*task state 任务状态 */
    sync_task_stat_t emTaskStatus;
    /*task wait flag 任务等待标识 */
    sync_task_wait_t emTaskWait;

}SysSyncTask_t;

typedef struct SysSyncSrvQueueTag
{
    /*task number 同步任务队列任务数 */
    uint32_t uiTaskNum;
    /*header 队列头 */
    uint16_t usHead;
    /*tail 队列尾 */
    uint16_t usTail;
    /*sync task element 任务队列元素 */
    SysSyncTask_t stSysSyncTask[SYNC_SRV_TASKS_NUM];

}SysSyncSrvQueue_t;

typedef struct SysSyncPrjQueueTag
{
    /*task number 同步任务队列任务数 */
    uint32_t uiTaskNum;
    /*header 队列头 */
    uint16_t usHead;
```

```
    /*tail 队列尾 */
    uint16_t usTail;
    /*sync task element 任务队列元素 */
    SysSyncTask_t stSysSyncTask[SYNC_PRJ_TASKS_NUM];
}SysSyncPrjQueue_t;

typedef struct SysJoinReqStatusTag
{
    /*new PM is joined 新PM请求加入标识 */
    Join_req_status_t emJoinReqFlag;
    /*active time 激活时间 */
    SyncTime64_t unActiveTime;
    /*sync timeout 同步超时时间 */
    SyncTime64_t unTimeOutTime;
}SysJoinReqStatus_t;

typedef struct SysAjustSyncTimeTag
{
    /*sync master clock flag 同步时钟主从标识 */
    uint32_t uiMaster;
    /*adjust time enable flag 校时使能标识 */
    ajust_status_t emAjustEn;
    /*adjust time 同步时钟校时时间 */
    SyncTime64_t unSyncAjustTime;
}SysAjustSyncTime_t; /* 12 bytes */
```

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-SS_SafR_SecR_A_001	Module initialization. (core0/core1) 模块初始化
SWDD-PM-SS_SafR_SecR_A_002	Common resource management (core0/core1) 公共资源管理（core0/core1）
SWDD-PM-SS_SafR_SecR_A_003	Real time data management (core0/core1)

	实时数据操作接口（core0/core1）
SWDD-PM-SS_SafR_SecR_A_004	File management (core0/core1) 文件管理（core0/core1）
SWDD-PM-SS_NSafR_SecR_A_005	LED management (core0/core1) LED 管理（core1）
SWDD-PM-SS_SafR_SecR_A_006	Internal communication management (core1) 内部通讯管理（core1）
SWDD-PM-SS_SafR_SecR_A_007	State management (core0/core1) 状态管理
SWDD-PM-SS_SafR_SecR_A_008	Sync management (core0/core1) 同步管理
SWDD-PM-SS_NSafR_SecR_A_009	Log 日志
SWDD-PM-SS_SafR_SecR_A_010	CRC algorithm CRC 算法
SWDD-PM-SS_SafR_SecR_A_011	Shared memory 共享内存
SWDD-PM-SS_NSafR_SecR_A_012	Version 版本

4 Design of sub-function 子功能设计

4.1 Module initialization 模块初始化

SWDD-PM-SS_SafR_SecR_A_001

4.1.1 SysInit

4.1.1.1 Function Description 功能描述

System resource initialization (core1).

系统资源管理模块初始化（core1）。

4.1.1.2 Argument Description 参数说明

- Definition 函数定义

int SysInit (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

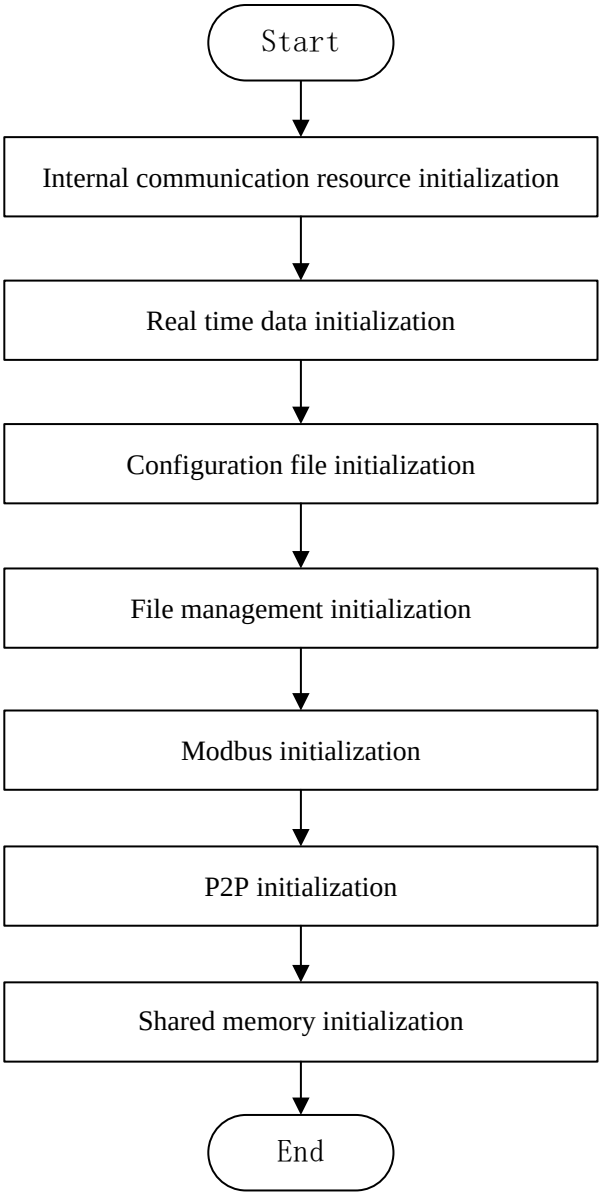

Figure4-1 System resource initialization processing flow
图 4-1 系统资源初始化流程图

4.2 Common resource management 公共资源管理

SWDD-PM-SS_SafR_SecR_A_002

This sub-function is used for other modules to access the system resource.

本子功能用于其它模块对系统资源的读写访问。

4.2.1 SysGetPMState

4.2.1.1 Function Description 功能描述

Get PM's state.

获取 PM 的状态。

4.2.1.2 Argument Description 参数说明

- Definition 函数定义

uint16_t SysGetPMState(PMController_t emPMID);

- Input argument 输入参数

PMController_t emPMID.

PM's ID

PM 的 ID 号

- Output argument 输出函数

No

无

4.2.1.3 Processing flow 处理流程

This function is simple and the processing flow is omitted.

此函数逻辑简单，流程图省略。

4.2.2 SysSetPMState

4.2.2.1 Function Description 功能描述

Set PM's state.

设置 PM 的状态。

4.2.2.2 Argument Description 参数说明

- Definition 函数定义

bool_t SysSetPMState(PMController_t emPMID, uint16_t usPMState)

- Input argument 输入参数

PMController_t emPMID.

PM's ID

PM 的 ID 号

uint16_t usPMState

PM's state

PM 的状态

- Output argument 输出函数

No

无

4.2.2.3 Processing flow 处理流程

This function is simple and the processing flow is omitted.

此函数逻辑简单，流程图省略。

4.2.3 SysGetSysCMState

4.2.3.1 Function Description 功能描述

Get CM's state.

获取 CM 的状态。

4.2.3.2 Argument Description 参数说明

- Definition 函数定义

void SysGetSysCMState(SysCMState_t *pstSysCMState);

- Input argument 输入参数

No

无

- Output argument 输出函数

SysCMState_t *pstSysCMState

point to CM's state

指向 CM 的状态

4.2.3.3 Processing flow 处理流程

This function is simple and the processing flow is omitted.

此函数逻辑简单，流程图省略。

4.2.4 SysSetCMState

4.2.4.1 Function Description 功能描述

Set CM's state.

设置 CM 的状态。

4.2.4.2 Argument Description 参数说明

- Definition 函数定义

bool_t SysSetCMState(CMController_t emCMID, uint16_t usCMState)

- Input argument 输入参数

CMController_t emCMID.

CM's ID

CM 的 ID 号

uint16_t usCMState

CM's state

CM 的状态

- Output argument 输出函数

No

无

4.2.4.3 Processing flow 处理流程

This function is simple and the processing flow is omitted.

此函数逻辑简单，流程图省略。

4.3 Real time data management 实时数据管理

SWDD-PM-SS_SafR_SecR_A_003

This sub-function is used to other module to access the real time data.

本子功能用于其它模块对实时数据的读写访问。

4.3.1 SysRTDataInit

4.3.1.1 Function Description 功能描述

Real time data initialization.

实时数据初始化。

4.3.1.2 Argument Description 参数说明

- Definition 函数定义

void SysRTDataInit(void);

- Input argument 输入参数

No

无

- Output argument 输出函数

No

无

4.3.1.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.3.2 SysReadRTData

4.3.2.1 Function Description 功能描述

Read real time data.

读实时数据。

4.3.2.2 Argument Description 参数说明

- Definition 函数定义

bool_t SysReadRTData(SysRtDataAreaType_t emDataArea, puint8_t pucContent, uint16_t usLen, puint16_t pusRdLen);

- Input argument 输入参数

SysRtDataAreaType_t emDataArea

The type of data area

数据区类型

uint16_t usLen

pre-read data length

预读取数据长度

- Output argument 输出函数

puint8_t pucContent

Point to the readout data buffer

指向读出的数据缓冲区

puint16_t pusRdLen

Point to the length of readout data

指向读出的数据长度

4.3.2.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.4 File management 文件管理

SWDD-PM-SS_SafR_SecR_A_004

This sub-function is used for other module to access project files and configuration files.

本子功能用于其它模块对工程文件及配置文件的读写访问。

4.4.1 FileManagerInit

4.4.1.1 Function Description 功能描述

File management initialization.

文件管理初始化。

4.4.1.2 Argument Description 参数说明

- Definition 函数定义

void FileManagerInit(void);

- Input argument 输入参数

No

无

- Output argument 输出函数

No

无

4.4.1.3 Processing flow 处理流程

This function is simple and the processing flow is omitted.

此函数逻辑简单，流程图省略。

4.4.2 SysWriteFile

4.4.2.1 Function Description 功能描述

Write content to a file.

写文件。

4.4.2.2 Argument Description 参数说明

- Definition 函数定义

bool_t SysWriteFile(int32_t iFd, uint8_t ucContent[], uint16_t usLen);

- Input argument 输入参数

int32_t iFd

File descriptor

文件描述符

uint8_t ucContent[]

the content buffer

文件内容

uint16_t usLen

the length of the content

文件内容长度

- Output argument 输出函数

No.

无

4.4.2.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.4.3 SysReadFile

4.4.3.1 Function Description 功能描述

Read content from a file.

读文件。

4.4.3.2 Argument Description 参数说明

- Definition 函数定义

bool_t SysReadFile(int32_t iFd, uint16_t usLen);

- Input argument 输入参数

int32_t iFd

File descriptor

文件描述符

uint16_t usLen

the length of the content

文件内容长度

- Output argument 输出函数

No.

无

4.4.3.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.5 LED management 状态灯管理

SWDD-PM-SS_NSafR_SecR_A_005

This sub-function is used to manage LED.

本子功能用于对状态灯进行管理。

4.5.1 LEDManagerInit

4.5.1.1 Function Description 功能描述

LED management initialization.

状态灯管理初始化。

4.5.1.2 Argument Description 参数说明

- Definition 函数定义

void LEDManagerInit(void);

- Input argument 输入参数

No

无

- Output argument 输出函数

No

无

4.5.1.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.5.2 LEDManagerCycle

4.5.2.1 Function Description 功能描述

Management of LED states.

状态灯管理。

4.5.2.2 Argument Description 参数说明

- Definition 函数定义

void LEDManagerCycle (void);

- Input argument 输入参数

No

无

- Output argument 输出函数

No

无

4.5.2.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.6 Internal communication management 内部通讯资源管理

SWDD-PM-SS_SafR_SecR_A_006

This sub-function is used to internal communication module to access the system resource.

本子功能用于内部通讯模块对相关系统资源的读写访问。

4.6.1 SysCMBusInit

4.6.1.1 Function Description 功能描述

Internal communication system resource initialization.

内部通讯系统资源初始化。

4.6.1.2 Argument Description 参数说明

- Definition 函数定义

void SysCMBusInit(void);

- Input argument 输入参数

No

无

- Output argument 输出函数

No

无

4.6.1.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.6.2 SysGetMsg

4.6.2.1 Function Description 功能描述

Get message from message box.

从邮箱中获取信息。

4.6.2.2 Argument Description 参数说明

- Definition 函数定义

bool_t SysGetMsg (CMController_t emCMID, SysMsgType_t emMsgType, puint8_t pucContent, puint16_t pusLen);

- Input argument 输入参数

CMController_t emCMID.

CM's ID

CM 的 ID 号

SysMsgType_t emMsgType

Message type

消息类型

- Output argument 输出函数

puint8_t pucContent

point to the message content

指向消息内容

puint16_t pusLen

point to message length

指向消息长度

4.6.2.3 Processing flow 处理流程

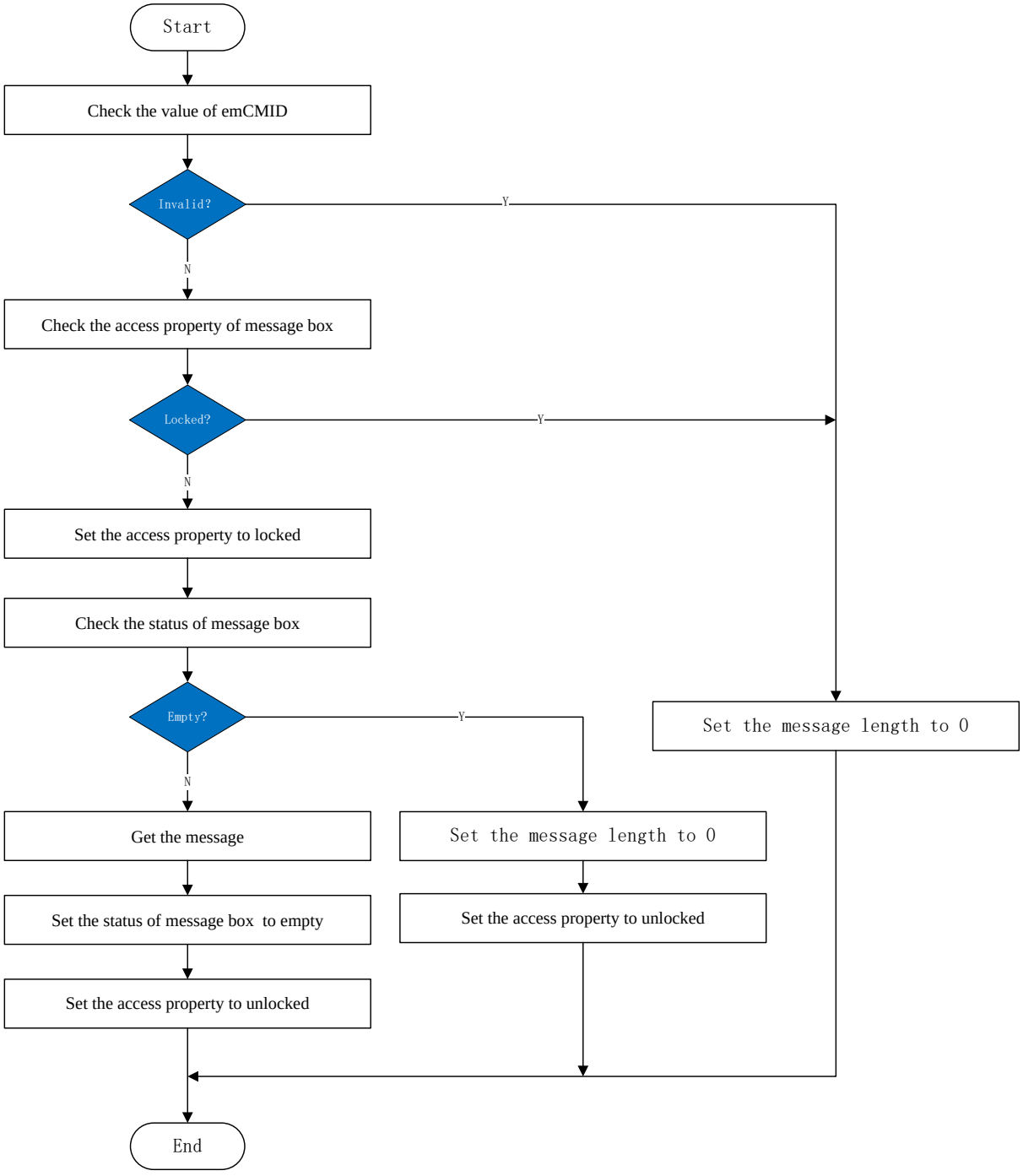

Figure4-2 Get message

图 4-2 获取消息

4.6.3 SysSetMsg

4.6.3.1 Function Description 功能描述

Set message to message box.

向邮箱中写入信息。

4.6.3.2 Argument Description 参数说明

- Definition 函数定义

bool_t SysSetMsg (CMController_t emCMID, SysMsgType_t emMsgType , puint8_t pucContent, uint16_t usLen);

- Input argument 输入参数

CMController_t emCMID.

CM's ID

CM 的 ID 号

SysP2PMsgType_t emMsgType

Message type

消息类型

puint8_t pucContent

point to the message content

指向消息内容

uint16_t usLen

message length

消息长度

- Output argument 输出函数

No.

无

4.6.3.3 Processing flow 处理流程

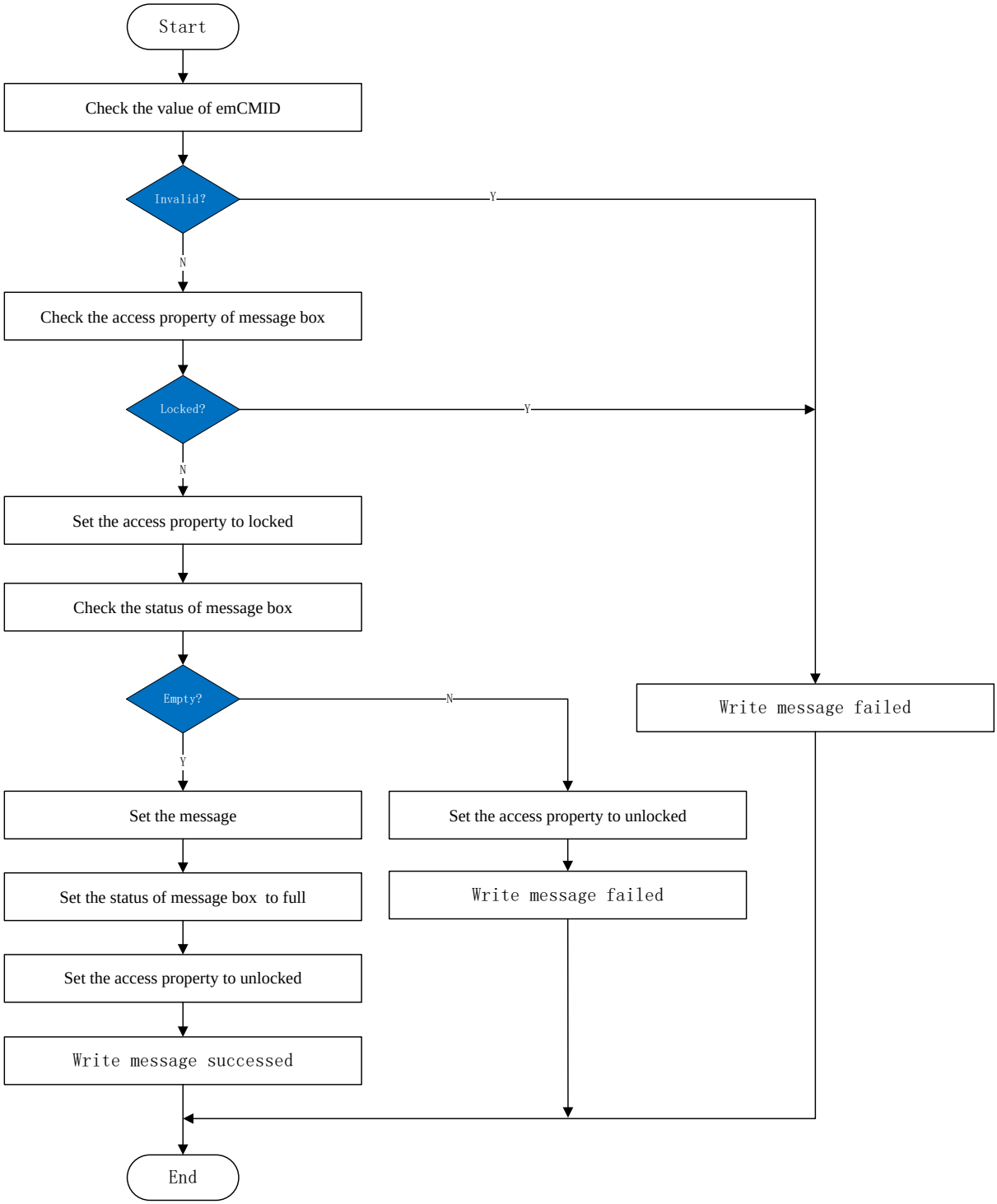

Figure4-3 Set message

图 4-3 写入消息

4.7 State management 状态管理

SWDD-PM-SS_SafR_SecR_A_007

This sub-function is used for state management module to access the system state.

本子功能用于状态管理模块对系统状态的读写访问。

4.7.1 SysStateMngtInit

4.7.1.1 Function Description 功能描述

State management management.

状态管理初始化。

4.7.1.2 Argument Description 参数说明

- Definition 函数定义

void SysStateMngtInit(void);

- Input argument 输入参数

No

无

- Output argument 输出函数

No

无

4.7.1.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.7.2 SysGetPMStateInfo

4.7.2.1 Function Description 功能描述

Get PM's state information.

获取 PM 状态信息。

4.7.2.2 Argument Description 参数说明

- Definition 函数定义

SysPMStateInfo_t* SysGetPMStateInfo(PMController_t emPMID);

- Input argument 输入参数

PMController_t emPMID.

PM's ID

PM 的 ID 号

- Output argument 输出函数

No.

无

4.7.2.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.8 Sync management 同步管理

SWDD-PM-SS_SafR_SecR_A_008

This sub-function is used for Sync module to access sync resource.

本子功能用于同步模块对同步资源的读写访问。

4.8.1 SysSyncInit

4.8.1.1 Function Description 功能描述

Sync resource initialization.

供同步模块使用的资源初始化。

4.8.1.2 Argument Description 参数说明

- Definition 函数定义

void SysSyncInit(void);

- Input argument 输入参数

No

无

- Output argument 输出函数

No

无

4.8.1.3 Processing flow 处理流程

This function is simple and the processing flow is omitted.

此函数逻辑简单，流程图省略。

4.8.2 SysEnSyncSrvQueue

4.8.2.1 Function Description 功能描述

Set sync-task to sync-queue.

将同步任务放入同步队列。

4.8.2.2 Argument Description 参数说明

- Definition 函数定义

void SysEnSyncSrvQueue(SysSyncSrvQueue_t* pstSyncSrvQueue, SysSyncTask_t* pstSyncTask);

- Input argument 输入参数

SysSyncSrvQueue_t* pstSyncSrvQueue

Point to sync-queue

指向同步队列

SysSyncTask_t* pstSyncTask

Point to sync-task

指向同步任务

- Output argument 输出函数

No.

无

4.8.2.3 Processing flow 处理流程

This function is simple and the processing flow is omitted.

此函数逻辑简单，流程图省略。

4.8.3 SysClearSyncSrvQueue

4.8.3.1 Function Description 功能描述

Clear sync-queue.

清空某同步队列。

4.8.3.2 Argument Description 参数说明

- Definition 函数定义

void SysClearSyncSrvQueue(SysSyncSrvQueue_t* pstSyncQueue, PMController_t emPMId);

- Input argument 输入参数

SysSyncSrvQueue_t* pstSyncSrvQueue

Point to sync-queue

指向同步队列

PMController_t emPMId

PM's ID

PM 的 ID

- Output argument 输出函数

No.

无

4.8.3.3 Processing flow 处理流程

This function is simple and the processing flow is omitted.

此函数逻辑简单，流程图省略。

4.9 Log 日志

SWDD-PM-SS_NSafR_SecR_A_009

This sub-function is used to read/write log.

本子功能用于读写日志。

4.9.1 LogWrite

4.9.1.1 Function Description 功能描述

Write log.

写日志。

4.9.1.2 Argument Description 参数说明

- Definition 函数定义

uint32_t LogWrite(uint32_t usLogID);

- Input argument 输入参数

uint32_t usLogID

Log ID

- Output argument 输出函数

Returns the write result. If an error is returned, the result shall be sent to the diagnostic software.

返回写结果。如果返回错误，将错误信息传给诊断软件。

4.9.1.3 Processing flow 处理流程

This function is simple and the processing flow is omitted.

此函数逻辑简单，流程图省略。

4.9.2 LogRead

4.9.2.1 Function Description 功能描述

Read log.

读日志。

4.9.2.2 Argument Description 参数说明

- Definition 函数定义

int32_t LogRead(LogInfo_t *pstLogInfo, uint16_t usLogicPos, uint32_t uiLogType, puint32_t puiNextIndex);

- Input argument 输入参数

uint16_t usLogicPos

Log's position

日志位置

uint32_t uiLogType

Log's type

日志类型

- Output argument 输出函数

LogInfo_t *pstLogInfo

Point to log information

指向日志信息

puint32_t puiNextIndex

Point to next log's index

指向下一条日志的索引号

4.9.2.3 Processing flow 处理流程

This function is simple and the processing flow is omitted.

此函数逻辑简单，流程图省略。

4.10 CRC algorithm CRC 算法

SWDD-PM-SS_SafR_SecR_A_010

This sub-function is used to calculate CRC.

本子功能用于计算 CRC。

4.10.1 SysCrc32Cal

4.10.1.1 Function Description 功能描述

Calculate 32 bit CRC.

计算 32 位 CRC。

4.10.1.2 Argument Description 参数说明

- Definition 函数定义

uint32_t SysCrc32Cal(uint32_t uiCrc, const puint8_t pucBuf, uint32_t uiCount);

- Input argument 输入参数

uint32_t uiCrc

The initial value of CRC

CRC 计算初始值

const puint8_t pucBuf

Point to the content which will be calcuted

指向待计算的内容

uint32_t uiCount

The length of the content

待计算内容的数据长度

- Output argument 输出函数

No

无

4.10.1.3 Processing flow 处理流程

This function is simple and the processing flow is omitted.

此函数逻辑简单，流程图省略。

4.10.2 SysCrc16Cal

4.10.2.1 Function Description 功能描述

Calculate 16 bit CRC.

计算 16 位 CRC。

4.10.2.2 Argument Description 参数说明

- Definition 函数定义

uint16_t SysCrc16Cal(const puint8_t pucBuf, uint16_t usCount);

- Input argument 输入参数

const puint8_t pucBuf

Point to the content which will be calcuted

指向待计算的内容

uint32_t uiCount

The length of the content

待计算内容的数据长度

- Output argument 输出函数

No

无

4.10.2.3 Processing flow 处理流程

This function is simple and the processing flow is omitted.

此函数逻辑简单，流程图省略。

4.11 Shared memory 共享内存

SWDD-PM-SS_SafR_SecR_A_011

This sub-function is used to read/write resource in shared memory.

本子功能用于读写共享内存上的资源。

4.11.1 GetLEDState

4.11.1.1 Function Description 功能描述

Get the state of LED.

获取状态灯状态。

4.11.1.2 Argument Description 参数说明

- Definition 函数定义

LEDState_t GetLEDState(LEDName_t emLED);

- Input argument 输入参数

LEDName_t emLED

The name of LED

LED 灯名称

- Output argument 输出函数

No.

无

4.11.1.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.11.2 SetLEDState

4.11.2.1 Function Description 功能描述

Set the state of LED.

设置状态灯状态。

4.11.2.2 Argument Description 参数说明

- Definition 函数定义

void SetLEDState(LEDName_t emLED, LEDState_t emLEDState);

- Input argument 输入参数

LEDName_t emLED

The name of LED

LED 灯名称

LEDState_t emLEDState

The state of LED

LED 状态

- Output argument 输出函数

No.

无

4.11.2.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.11.3 UpdateLEDRunningState

4.11.3.1 Function Description 功能描述

Update the state of LED.

更新状态灯状态。

4.11.3.2 Argument Description 参数说明

- Definition 函数定义

void UpdateLEDRunningState(void);

- Input argument 输入参数

No.

无

- Output argument 输出函数

No.

无

4.11.3.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.11.4 SysGetHandshakeSignal

4.11.4.1 Function Description 功能描述

Get handshake signal from shared memory.

从共享内存获取握手信号。

4.11.4.2 Argument Description 参数说明

- Definition 函数定义

bool_t SysGetHandshakeSignal(uint32_t uiSignalTrans, puint32_t puiSignal);

- Input argument 输入参数

uint32_t uiSignalTrans

the direction of the signal: core0 to core1 or core1 to core0

信号传输方向：core0 到 core1 或者 core1 到 core0

- Output argument 输出函数

puint32_t puiSignal

Point to the content of signal

指向信号内容

4.11.4.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.11.5 SysSetHandshakeSignal

4.11.5.1 Function Description 功能描述

Set handshake signal to shared memory.

向共享内存中设置握手信号。

4.11.5.2 Argument Description 参数说明

- Definition 函数定义

bool_t SysSetHandshakeSignal(uint32_t uiSignalTrans, uint32_t uiSignal);

- Input argument 输入参数

uint32_t uiSignalTrans

the direction of the signal: core0 to core1 or core1 to core0

信号传输方向：core0 到 core1 或者 core1 到 core0

uint32_t uiSignal

the content of signal

信号内容

- Output argument 输出函数

No.

无

4.11.5.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.11.6 SharedSetSOE

4.11.6.1 Function Description 功能描述

Set SOE to shared memory.

向共享内存中写入 SOE。

4.11.6.2 Argument Description 参数说明

- Definition 函数定义

bool_t SharedSetSOE(uint8_t ucQueueIndex, SoftSOE_t stSOE);

- Input argument 输入参数

uint8_t ucQueueIndex

SOE queue index

SOE 队列索引

SoftSOE_t stSOE

The content of SOE

SOE 内容

- Output argument 输出函数

No.

无

4.11.6.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.11.7 SharedGetSOE

4.11.7.1 Function Description 功能描述

Get SOE from shared memory.

从共享内存中读取 SOE。

4.11.7.2 Argument Description 参数说明

- Definition 函数定义

int32_t SharedGetSOE(uint8_t ucQueueIndex, SoftSOE_t* pstSOE, uint16_t usPos, puint32_t puiNextIndex);

- Input argument 输入参数

uint8_t ucQueueIndex

SOE queue index

SOE 队列索引

uint16_t usPos

the Position in the queue

在队列中的位置

- Output argument 输出函数

SoftSOE_t* pstSOE

Point to the SOE

指向 SOE

puint32_t puiNextIndex

point to the next position in the queue

指向下一个位置

4.11.7.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.11.8 SysGetSharedMemRtDataStartAddr

4.11.8.1 Function Description 功能描述

Get the start address of real time data in shared memory.

获取实时数据在共享内存中的起始地址。

4.11.8.2 Argument Description 参数说明

- Definition 函数定义

puint8_t SysGetSharedMemRtDataStartAddr(SysRtDataType_t emDataType);

- Input argument 输入参数

SysRtDataType_t emDataType

The type of real time data area

实时数据区的类型

- Output argument 输出函数

No

无

4.11.8.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.12 Version 版本

SWDD-PM-SS_NSafR_SecR_A_012

This sub-function is used to show the version.

本子功能用于显示版本号。

4.12.1 ShowVersion

4.12.1.1 Function Description 功能描述

Show version.

显示版本号。

4.12.1.2 Argument Description 参数说明

- Definition 函数定义

void ShowVersion(void);

- Input argument 输入参数

No.

无

- Output argument 输出函数

No.

无

4.12.1.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

——以下无正文