Document Title: PM_FW diagnostics module design description of Safety Control System

Document Number: 15-Q04-000065

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 自检模块设计说明书

文件编号：15-Q04-000065

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 diagnostics 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 自检模块的设计方案。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 diagnostics module (marked red) in the software hierarchy is shown below.

自检模块（标红）在软件层次中的位置如下图所示。

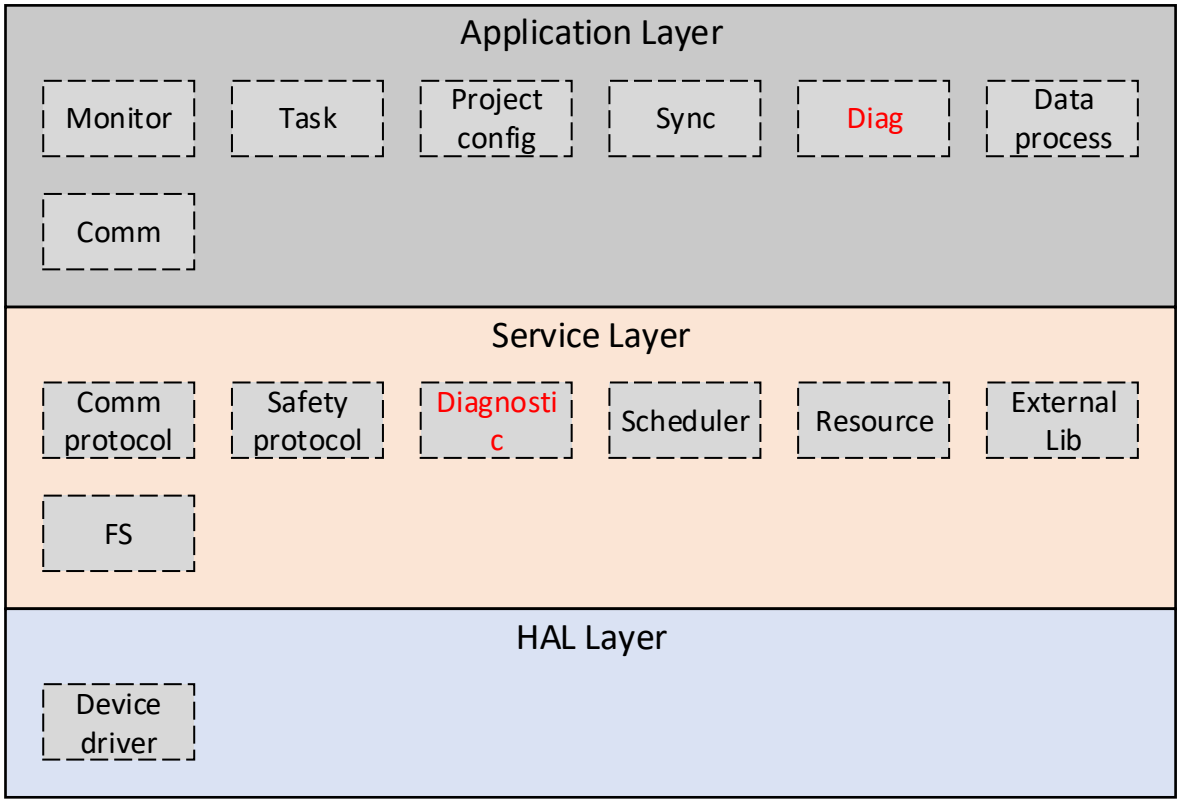

Figure 2-1 the location of the module

图 2-1 模块位置

3 Module design 模块设计

3.1 Function description 功能描述

Diagnose the system after power on and diagnose the system periodically during the system running. It includes CPU test, DDR test and exception test.

该模块提供了系统上电自检和周期自检功能。自检功能主要包括 CPU 自检、DDR 自检及异常自检。

3.2 Design concept 设计思路

3.2.1 Program relocation 程序重定位

The running space will be destroyed during DDR test, so the running space and test space must be separated. Please see the following test sequence:

内存自检时会破坏程序的运行空间，因此自检的空间和程序运行空间需要分开。整个自检流程如下所述：

The test code running in space1, while space2 and space3 are tested, when the test is finished, the test code will switch to space2;

运行在 space1 上的程序，对 space2 和 space3 进行自检，当自检完成后， space1 上的程序停止运行， space2 上的程序开始运行；

The test code running in space2, while space1 and space3 are tested, when the test is finished, the test code will switch to space3;

运行在 space2 上的程序，对 space1 和 space3 进行自检，当自检完成后， space2 上的程序停止运行， space3 上的程序开始运行；

The test code running in space3, while space1 and space2 are tested, when the test is finished, the test code will switch to space1;

运行在 space3 上的程序，对 space1 和 space2 进行自检，当诊断完成后， space3 上的程序停止运行， space1 上的程序开始运行。

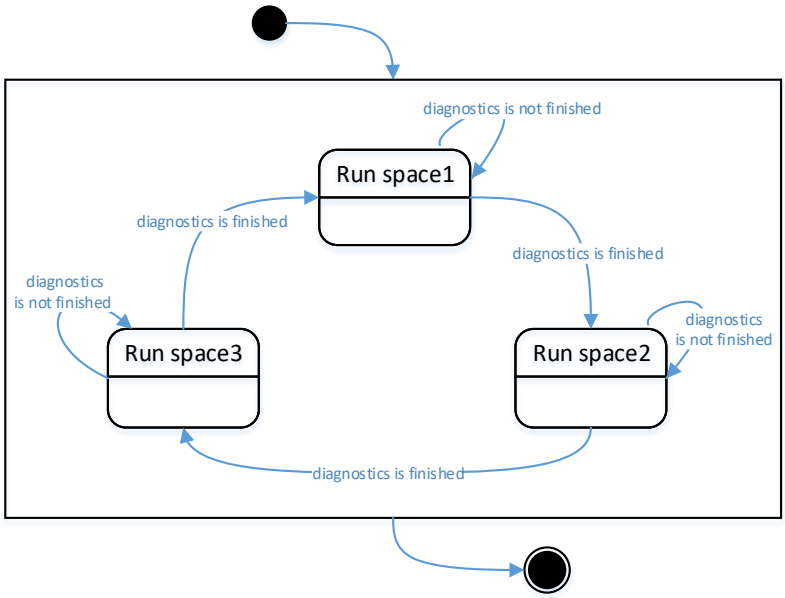

Figure 3-1 program relocation state transition diagram

图 3-1 程序重定位状态转换图

There are three identical programs, and only one program is running at any time. To ensure the program switching from one space to another space correctly, there are two measures:

运行在三个独立空间上的程序，逻辑上是一致的，为了确保三份程序运行时无扰切换，切换时需做到以下两点：

1) The data area snapshot (e.g. data1) must be copied to the next running space (e.g. data2) in one period;

1）将当前程序使用的数据区快照（例如 data1）拷贝到下一个即将运行程序的数据区（例如 data2）；

2) The stack pointer and PC registers etc. must be modified to the correct content.

2）将当前程序的栈指针及 PC 指针等更新为正确的内容。

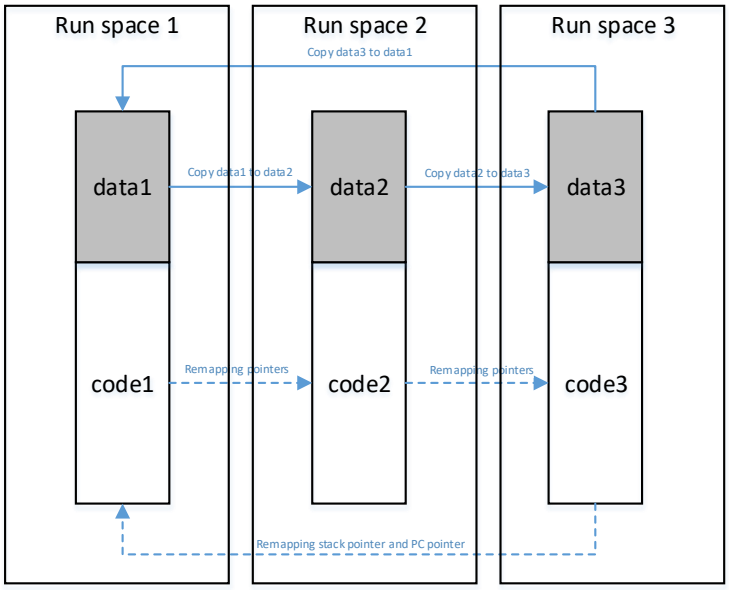

Figure 3-2 Program switching in different running spaces

图 3-2 运行空间切换示意图

3.2.2 CPU test 处理器自检

(1) Program running field save and resume

(1) 现场保护及恢复

The stack will be created before CPU test, and the context will be stored in this stack. When CPU test is finished, the context will be resumed from the stack.

自检开始前先创建一个栈，将当前的上下文内容保存在该栈内，当对应的自检项完毕后，再将上下文内容从栈中恢复出来。

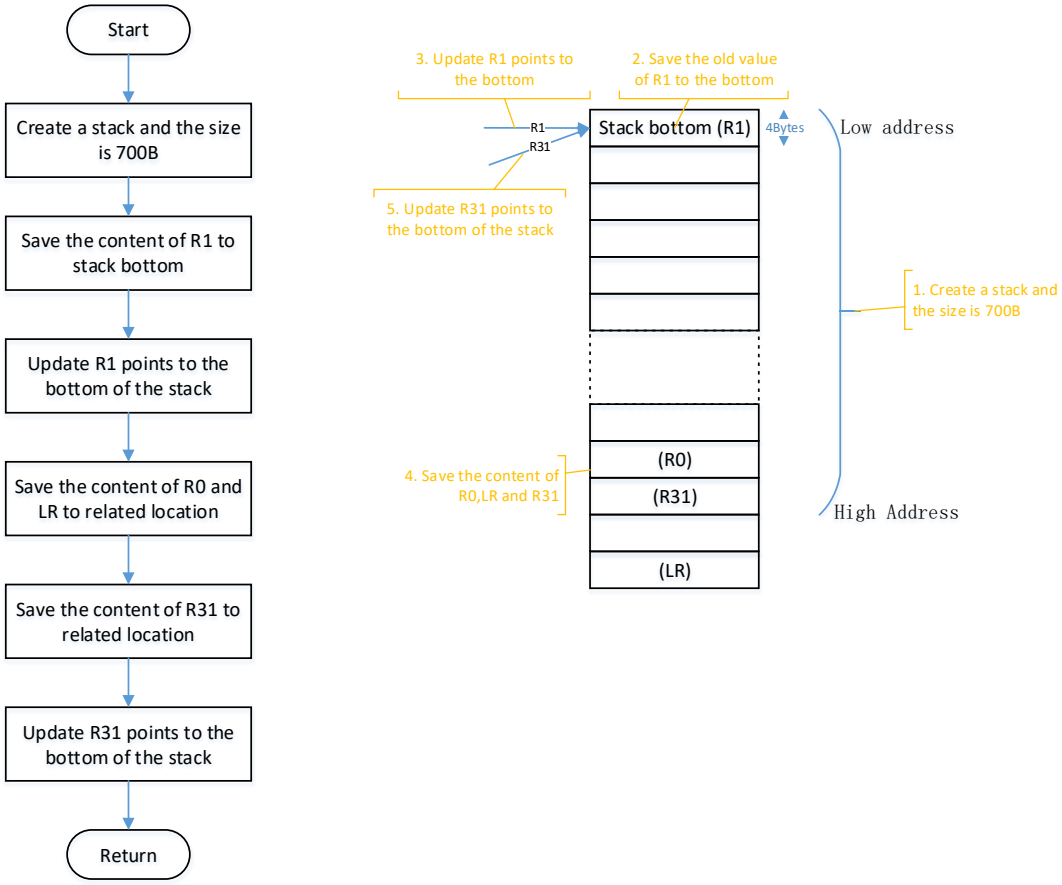

Figure3-3 create a stack processing flow

图 3-3 创建堆栈流程图

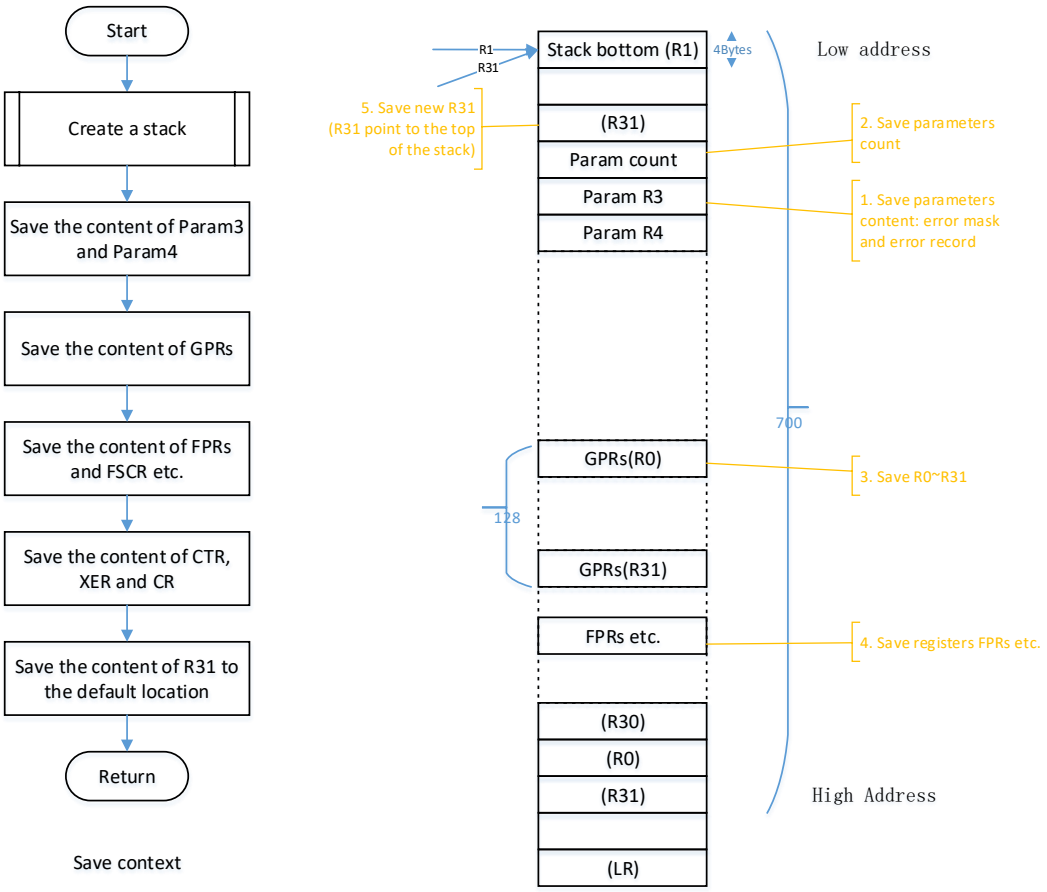

Figure3-4 save context processing flow

图 3-4 保存上下文流程图

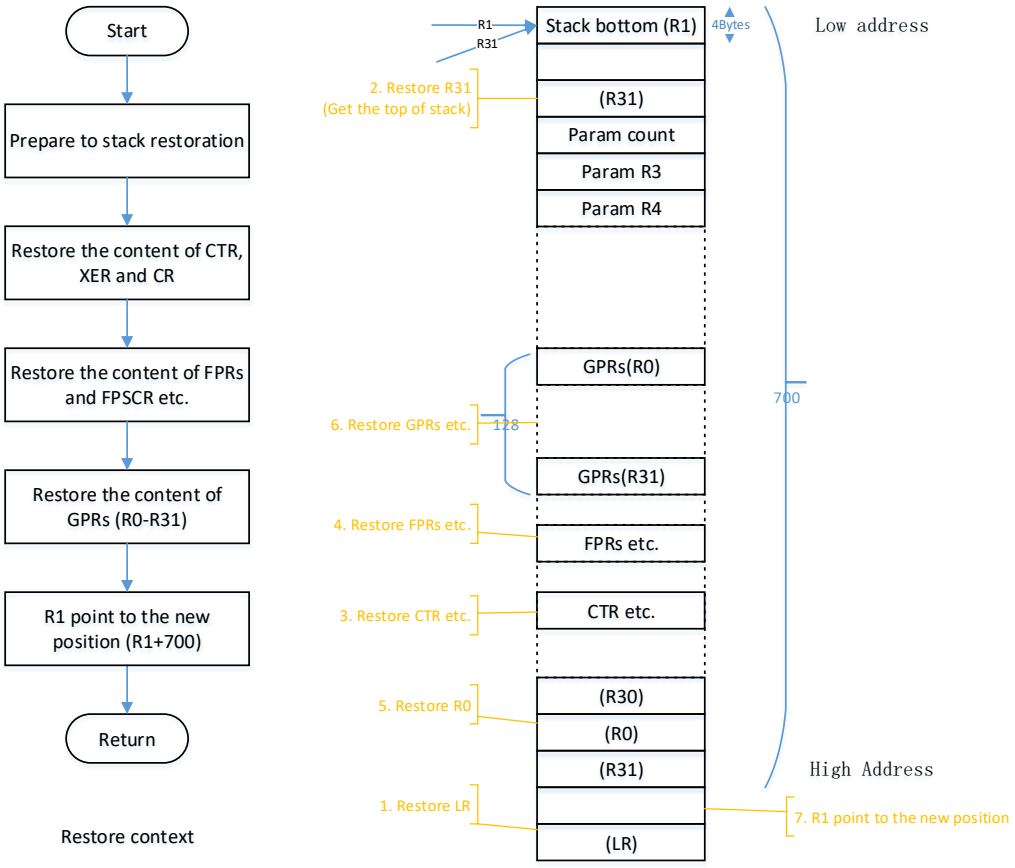

Figure3-5 Resume context processing flow

图 3-5 恢复上下文流程图

(2) Data bus test

(2) 数据总线自检

Data bus test will be executed on core1 during startup; and it will be executed on core0 periodically.

该项自检对 core1 只需启动时自检。对 core0 需要周期自检。

The test address is 0x0200 0000 (not used by PM_FW); write the related data to the address (0x0200 0000); read the data from the address (0x0200 0000). Compare the write-in data with the read-out data, if they are not the same, it means data bus test failed.

首先，设置一个确定的被测地址（0x0200 0000），该地址未被程序使用；其次，将相应数据写入该地址；最后，从被测地址读出数据，判断是否与写入的数据一致，若不一致则表明数据线自检失败，进入自检失败处理程序。

The test data are in the following table. Different data are used to test different data lines.

测试数据如下表所示，不同的数据，对应不同的数据线测试。

Data bus test
数据线自检

Test address 被测地址	bus 被测数据线	Write 1 写入 1	Readout (0) 读出 1	Write 0 写入 0	Readout (0) 读出 0
0x0200 0000	31st	0x8000 0000	0x8000 0000	0x0000 0000	0x0000 0000
	30th	0x4000 0000	0x4000 0000	0x0000 0000	0x0000 0000
	…				
	0th	0x0000 0001	0x0000 0001	0x0000 0000	0x0000 0000

(3) Address bus test

(3) 地址总线自检

Address bus test will be executed on core1 during startup; Data bus test and address bus test are executed indirectly via DDR test during operation. Address bus test will be executed periodically on core0.

该项自检对 core1 只需启动时自检，运行时通过内存自检间接实现数据线和地址线的功能自检。对 core0 需要周期自检。

The test data are in the following table. Different data are used to test different address lines.

不同的被测地址代表对不同的地址线进行测试，如下表所示。

Address bus test
地址线自检

Test address 被测地址线	Test address 被测试地址	Write data 写入数据	Read out data 读出数据	Test address 被测试地址	Write data 写入数据	Read out data 读出数据
27th	0x07FF FFF0	0x55	0x55	0x03FF FFF0	0xAA	0xAA
26th	0x0400 0000	0x55	0x55	0x0600 0000	0xAA	0xAA
	0x0400 0001	0xAA		0x0600 0001	0x55	
	0x0400 0002	0xAA		0x0600 0002	0x55	
	0x0400 0004	0xAA		0x0600 0004	0x55	
	…	0xAA		…	0x55	
	0x0500 0000	0xAA		0x0600 0000	0x55	
25th	0x0400 0000	0x55	0x55	0x0500 0000	0xAA	0xAA
	0x0400 0001	0xAA		0x0500 0001	0x55	
	0x0400 0002	0xAA		0x0500 0002	0x55	
	0x0400 0004	0xAA		0x0500 0004	0x55	
	…	0xAA		…	0x55	
	0x0480 0000	0xAA		0x0580 0000	0x55	
24th	0x0400 0000	0x55	0x55	0x0480 0000	0xAA	0xAA
	0x0400 0001	0xAA		0x0480 0001	0x55	
	0x0400 0002	0xAA		0x0480 0002	0x55	
	0x0400 0004	0xAA		0x0480 0004	0x55	
	…	0xAA		…	0x55	

	0x0440 0000	0xAA		0x04C0 0000	0x55	
23th	0x0400 0000	0x55	0x55	0x0440 0000	0xAA	0xAA
	0x0400 0001	0xAA		0x0440 0001	0x55	
	0x0400 0002	0xAA		0x0440 0002	0x55	
	0x0400 0004	0xAA		0x0440 0004	0x55	
	…	0xAA		…	0x55	
	0x0420 0000	0xAA		0x0460 0000	0x55	
…						
1st	0x0400 0000	0x55	0x55	0x0400 0001	0xAA	0xAA

(4) Registers test

(4) 寄存器自检

Coupling error, flip error, fixed point error and static error etc. will be detected by registers test. The algorithm is March SS.

寄存器自检目的是检查寄存器是否存在耦合错误，翻转错误，定点错误等各种静态错误，采用自检算法为面向字检测的 March SS 法。

(5) Instruction set test

(5)指令集自检

The instruction test include jump instruction test, integer/float instruction test and load/store instruction test, and the algorithm is March SS.

主要包含跳转指令、整数运算指令、浮点运算指令及数据加载存储指令自检，采用 March SS 算法。

(6) Instruction cache test

(6) 指令 cache 自检

The related instructions will be loaded to instruction cache, and run the instruction to verify the result.

指令 Cache 自检需要将指定指令加载到指令 Cache，然后运行，查看运行结果是否正确。

The following two instructions in the table below will be used:

指令缓存测试采用的两条指令如下表所示：

Binary **二进制**	**Assemble thumb** **汇编指令**
0x7D4A5A14	ADDI R10,R10,1
0x7D4B5050	SUBI R10,R10,1

(7) Data cache test

(7) 数据 cache 自检

。

32kB L1 data cache and 256kB L2 data cache are used in CPU of PM. L1 & L2 data cache can be checked periodically using March SS by the following steps:

CPU 使用 32kB L1 数据 cache 和 256kB L2 数据 cache，对数据 cache 采用周期 March SS 测试，步骤如下

1. "Flush" (backup) the content of data cache in DDR.

   刷新数据 cache 的内容

2. "Lock" part of unused block of memory to data cache so that the data cache can be accessed with the address of the locked block.

   锁定一片不使用的内存和数据 cache 的对应关系。

3. Perform March SS in the locked address. The locked block memory is not accessed but the data cache is actually read/write by March SS patterns.

   对这片内存进行 March SS 算法测试，即对这片内存对应的数据 cache 进行 March SS 算法测试。

4. "Unlock" after finishing March SS, and the data cache can be normally used as cache again.

   March SS 算法测试结束后，解除内存和数据 cache 的锁定。

$$
\begin{array}{l}
\{\ \Updownarrow(w0)\ ; \\
\quad M_0 \\
\Uparrow(r0,r0,w0,r0,w1)\ ; \quad \Uparrow(r1,r1,w1,r1,w0)\ ; \\
\qquad M_1 \qquad\qquad\qquad M_2 \\
\Downarrow(r0,r0,w0,r0,w1)\ ; \quad \Downarrow(r1,r1,w1,r1,w0)\ ; \\
\qquad M_3 \qquad\qquad\qquad M_4 \\
\Updownarrow(r0)\ \} \\
M_5
\end{array}
$$

Figure 3-6 March SS execute sequence

图 3-6 March SS 执行序列

(8) Stack test

(8) 栈空间自检

a) Setup the stack size and set the boundary of the stack

a) 设置栈空间大小及建立栈空间边界标记

The grey background means stack space. 16 bytes of 0x55 are written into the upper boundary, and 16 bytes of 0xAA are written into the lower boundary.

灰色背景部分表示栈空间。从上边界往上写入 16 字节特定值（0x55），从下边界往下写入 16 字节特定值（0xAA）。

Upper boundary 上边界（16B）	Flag: 0x55 标记值：0x55
	…
	Flag: 0x55 标记值：0x55
	Flag: 0x55 标记值：0x55
	Flag: 0x55 标记值：0x55
Stack space 栈空间	
Lower boundary 下边界（16B）	Flag:0xAA 标记值：0xAA
	Flag:0xAA 标记值：0xAA
	Flag:0xAA 标记值：0xAA

	…
	Flag:0xAA 标记值：0xAA

b) Boundary check

b) 检查边界

Read the related 16 bytes from upper boundary and compare them with 0x55, read the related 16 bytes from lower boundary and compare them with 0xAA, if they are not equal, that means the stack is destroyed.

将上边界往上的 16 字节读出与特定值 0x55 进行比较，将下边界往下的 16 字节读出与特定值 0xAA 进行比较，若出现不一致情况，则表明栈空间被破坏。

c) push stack test

c) 压栈自检

Execute push operation once, and check the stack pointer whether it is decreased 1 or not.

进行一次压栈操作，检查栈指针寄存器的值是否减 1。

d) pop stack test

d) 出栈自检

Execute pop operation once, and check the stack pointer whether it is increased 1 or not.

进行一次出栈操作，检查栈指针寄存器的值是否增 1。

(9) ECC test

(9) ECC 自检

Use the CPU's own ECC fault detection function.

使用 CPU 自带的 ECC 故障检测功能。

3.2.3 DDR test 内存自检

The algorithm is March SS.

自检算法 March SS。

$$\{\ \underset{M_0}{\Updownarrow(w0)}\ ;\ \underset{M_1}{\Uparrow(r0,r0,w0,r0,w1)}\ ;\ \underset{M_2}{\Uparrow(r1,r1,w1,r1,w0)}\ ;\ \underset{M_3}{\Downarrow(r0,r0,w0,r0,w1)}\ ;\ \underset{M_4}{\Downarrow(r1,r1,w1,r1,w0)}\ ;\ \underset{M_5}{\Updownarrow(r0)}\ \}$$

Figure 3-7 March SS execute sequence

图 3-7 March SS 执行序列

3.2.4 Code test 代码自检

(1) Firmware test

(1) 固件代码自检

The code will be loaded into DDR, and it will not be modified during the whole running time. The CRC32 will be used to check out whether the code is correct or not.

代码从 flash 加载到内存后，整个运行过程中，均不会被修改。按照安全标准要求，可以通过 32 位 CRC 校验码检测。

(2) User program test

(2)用户代码自检

The program will be tested periodically during the system running. The test algorithm is calculate the CRC of user program.

系统运行时通过周期性计算用户程序的 CRC 来实现对用户程序的自检。

3.2.5 Exception test 异常自检

First, set the condition and execute the related code to cause the exception.

首先，设置异常产生条件，执行产生异常语句，从而使异常产生。

Second, when the system caught the exception, the interrupt service routing will be executed, and the exception flag will be set and return from exception.

其次，系统捕获异常后，进入异常中断服务处理程序，在异常中断服务处理程序中设置异常标志，然后从中断服务程序中返回。

At last, the exception flag will be checked in the related function for whether the exception test is executed successfully via exception flag.

最后，在检查函数中检查是否有对应的异常标志被记录，根据该标志判断异常产生和捕获自检是否成功。

3.3 Interface function 接口函数

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

模块提供的接口函数如下：

1. void DiagInit(void)

Input argument 输入参数	Output argument 输出参数	Description 描述
No. 无。	No. 无。	Diagnostics module initialization. 自检模块初始化。

2. void DiagCycle(void)

Input argument 输入参数	Output argument 输出参数	Description 描述
No. 无。	No. 无。	Diagnostics module runs periodically. 自检模块周期运行。

3. void CPUTestInit(void)

Input argument 输入参数	Output argument 输出参数	Description 描述
No. 无。	No. 无。	CPU test initialization. 处理器自检初始化。

4. uint32_t CPUTestCycle(puint32_t puiVectorTest)

Input argument 输入参数	Output argument 输出参数	Description 描述
No. 无。	No. 无。	CPU test periodically 处理器周期自检。

5. void RamTestInit(void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No. 无。	No. 无。	RAM test initialization. 内存自检初始化。

6. uint32_t RamTestCycle(puint32_t puiVectorTest)

Input argument	Output argument	Description

输入参数	输出参数	描述
No. 无。	No. 无。	RAM test periodically 内存周期自检。

7. void ExceptionTestInit (void);

Input argument 输入参数	Output argument 输出参数	Description 描述
No. 无。	No. 无。	Exception test initialization. 异常自检初始化。

8. uint32_t ExceptionTestCycle (puint32_t puiVectorTest)

Input argument 输入参数	Output argument 输出参数	Description 描述
No. 无。	No. 无。	Exception test periodically 异常周期自检。

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.	DiagCtrlInfo_t	s_stDiagCtrlInfo	Control information 控制参数
2.	ExcptHandleInfo_t	s_stExcptInfo	Exception handling information 异常处理状态字

3.5 Data structure 数据结构

```
typedef enum
{
    DIAG_CPU = 0U,          /*CPU test sequence number 处理器测试序列号*/
    DIAG_CODE,              /*code test sequence number 代码测试序列号*/
    DIAG_RAM,               /*RAM test sequence number 内存测试序列号*/
    DIAG_EXCPT,             /*Exception test sequence number 异常测试序列号*/
    MAX_DIAG
}DiagSequence;
```

```
typedef struct PrjCodeTag
{
    uint32_t uiStartAddr; /* start address in RAM  在内存中的起始地址*/
    uint32_t uiCodeSize; /* the size of code  代码大小*/
    uint32_t uiCRC; /* CRC  校验码*/
}PrjCode_t;
```

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-DIAG_SafR_NSecR_A_001	CPU test 处理器自检
SWDD-PM-DIAG_SafR_NSecR_A_002	DDR test 内存自检
SWDD-PM-DIAG_SafR_NSecR_A_003	Code test 代码自检
SWDD-PM-DIAG_SafR_NSecR_A_004	Except test 异常自检

4 Design of sub-function 子功能设计

4.1 CPU test 处理器自检

SWDD-PM-DIAG_SafR_NSecR_A_001

4.1.1 CPUTestInit

4.1.1.1 Function Description 功能描述

CPU test initialization.

处理器自检初始化。

4.1.1.2 Argument Description 参数说明

- Definition 函数定义

```
void CPUTestInit(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。

4.1.2 CPUTestCycle

4.1.2.1 Function Description 功能描述

CPU test periodically.

处理器周期自检。

4.1.2.2 Argument Description 参数说明

- Definition 函数定义

void CPUTestCycle(puint32_t puiVectorTest);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

puint32_t puiVectorTest

Point to the error flag

自检出错指示标志

4.1.2.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.1.3 DataBusTest

4.1.3.1 Function Description 功能描述

Data bus test.

数据总线自检。

4.1.3.2 Argument Description 参数说明

- Definition 函数定义

void DataBusTest (void);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

No.

无。

4.1.3.3 Processing flow 处理流程

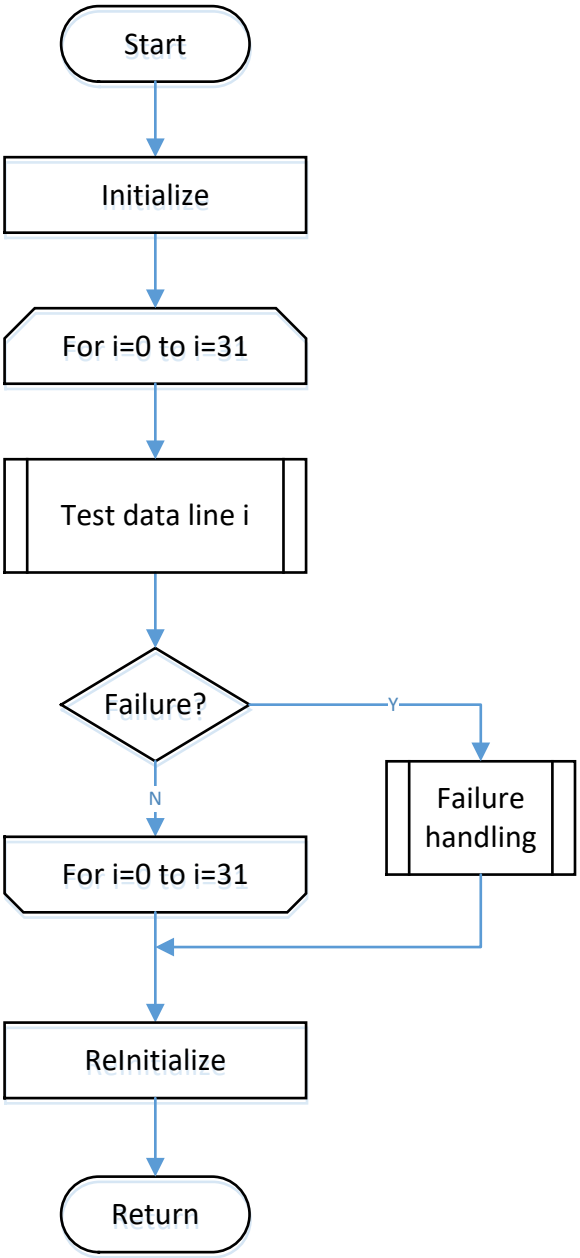

Figure4-1 Data bus test processing flow

图 4-1 数据总线自检流程图

4.1.4 AddressBusTest

4.1.4.1 Function Description 功能描述

Address bus test.

地址总线自检。

4.1.4.2 Argument Description 参数说明

- Definition 函数定义

void AddressBusTest (void);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

No.

无。

4.1.4.3 Processing flow 处理流程

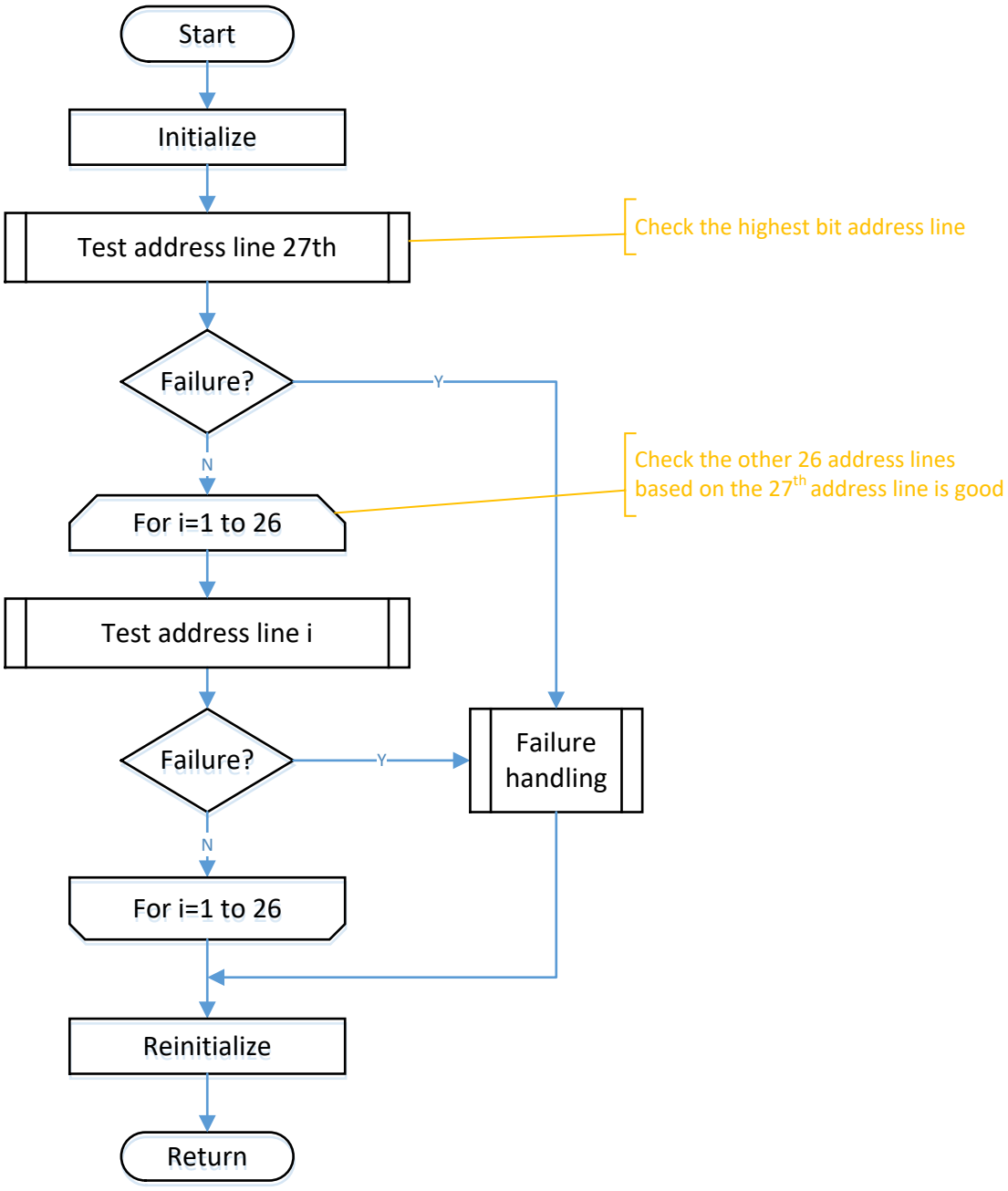

Figure4-2 Address bus test processing flow

图 4-2 地址总线自检流程图

4.1.5 RegisterTest

4.1.5.1 Function Description 功能描述

Registers test.

寄存器自检。

The following registers will be tested:

寄存器自检分类如下：

LR register test

LR 寄存器自检

CR register test

CR 寄存器自检

GPRs interword test

通用寄存器字间错误自检

GPRs intraword test

通用寄存器字内错误自检

FPRs interword test

浮点型寄存器字间错误自检

FPRs intraword test

浮点型寄存器字内错误自检

SPRs interword test

特殊功能寄存器字间错误自检

SPRs intraword test

特殊功能寄存器字内错误自检

SPRG error test

SPRG 寄存器错误自检

Critical SPRs test

关键功能寄存器自检

4.1.5.2 Argument Description 参数说明

- Definition 函数定义

void RegisterTest (void);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

No.

无。

4.1.5.3 Processing flow 处理流程

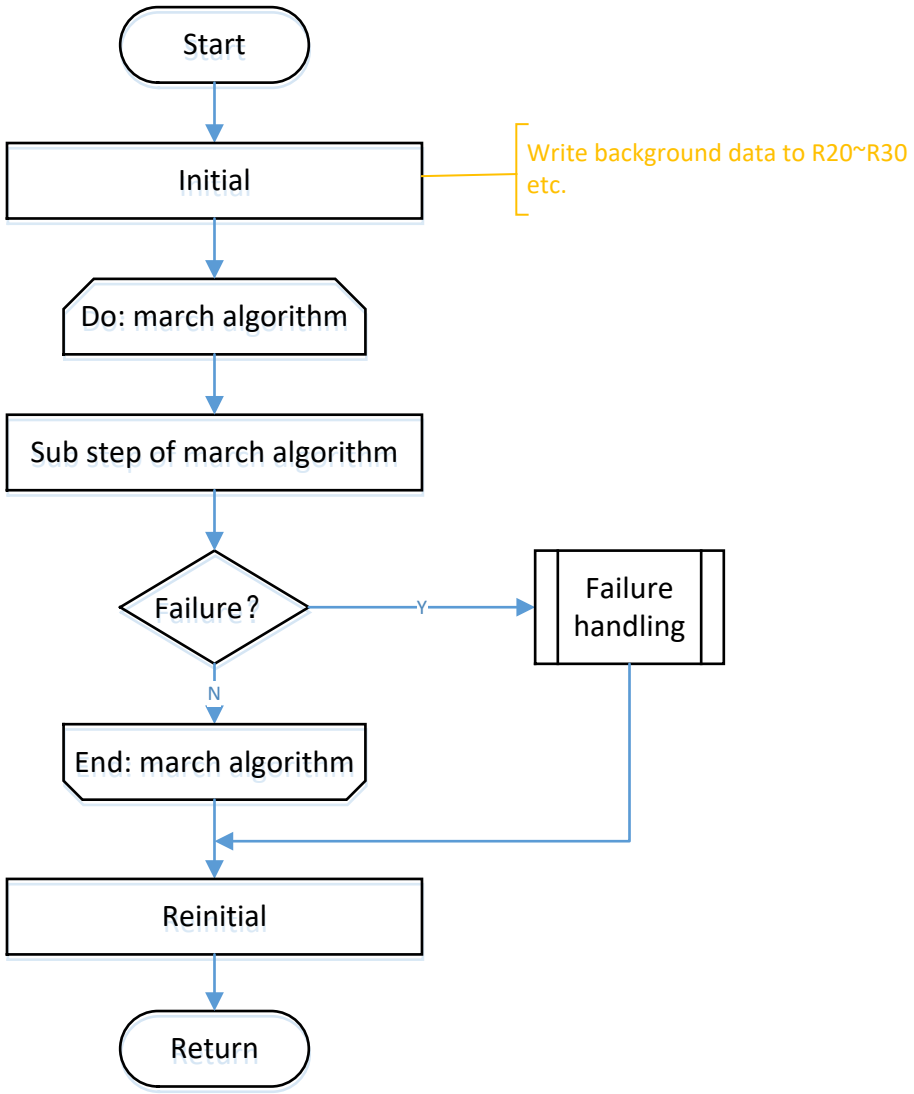

Figure4-3 Registers test processing flow

图 4-3 寄存器自检流程图

4.1.6 JumpInsTest

4.1.6.1 Function Description 功能描述

Jump instruction test.

跳转指令自检。

4.1.6.2 Argument Description 参数说明

- Definition 函数定义

void JumpInsTest (void);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

No.

无。

4.1.6.3 Processing flow 处理流程

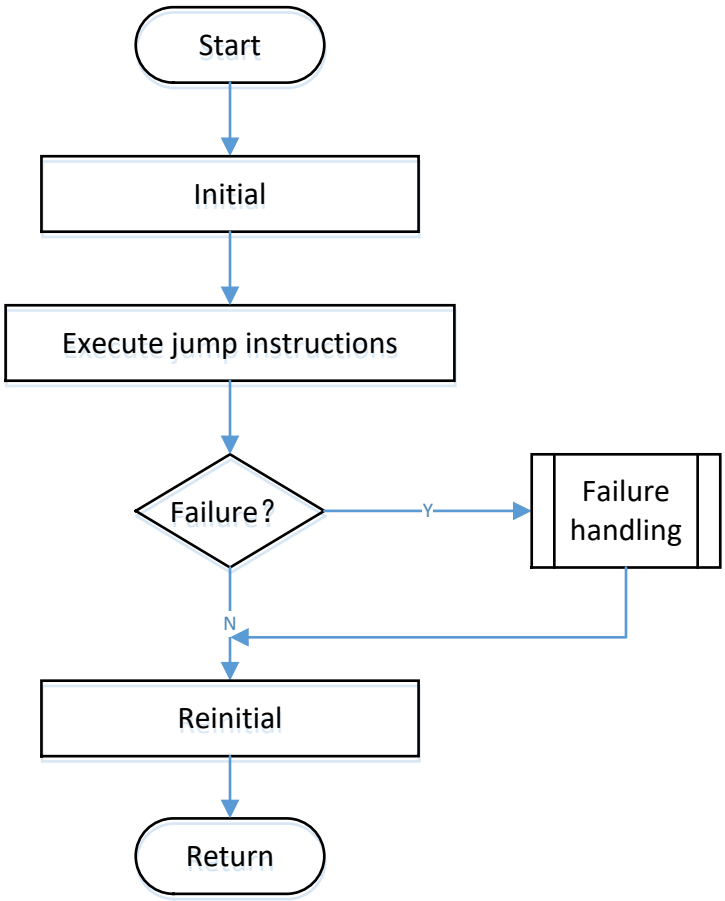

Figure4-4 Jump instruction test processing flow

图 4-4 跳转指令自检流程图

4.1.7 Load/store instruction test

4.1.7.1 Function Description 功能描述

Load/store instruction test.

数据加载存储指令自检。

Argument Description 参数说明

- Definition 函数定义

void CpuTestLsu (void);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

No.

无。

4.1.7.2 Processing flow 处理流程

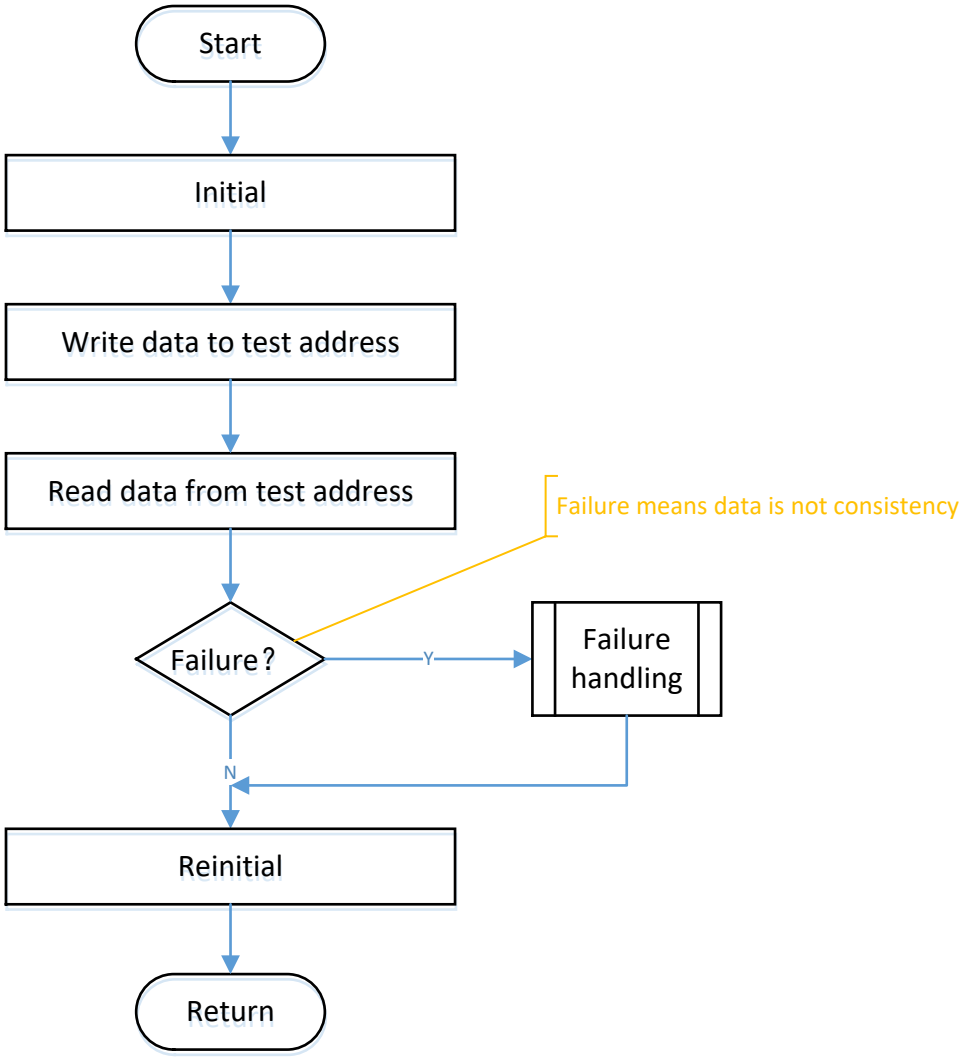

Figure4-5 Load/store instruction processing flow

图 4-5 数据加载存储指令自检流程图

4.1.8 Pipe line test 指令流水线自检

4.1.8.1 Function Description 功能描述

Instruction pipe line test.

指令流水线自检。

4.1.8.2 Argument Description 参数说明

- Definition 函数定义

void CpuTestPipeLine (void);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

No.

无。

4.1.8.3 Processing flow 处理流程

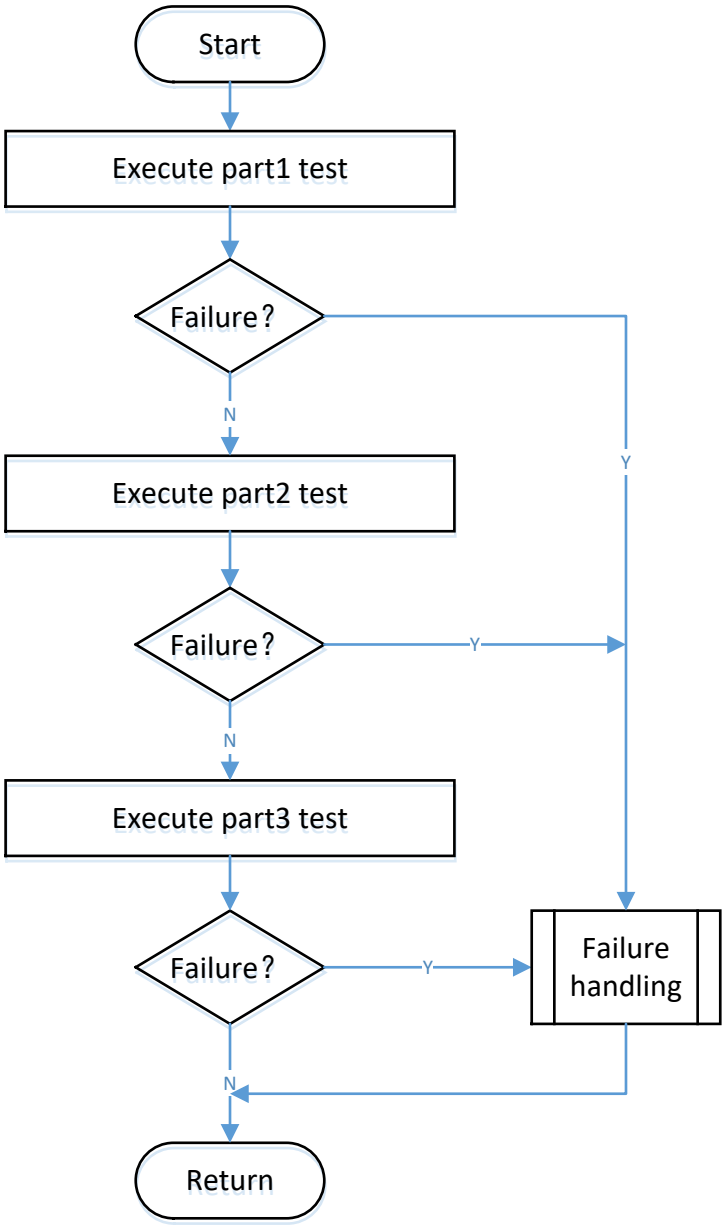

Figure4-6 Instruction pipe line test processing flow

图 4-6 指令流水线自检流程图

4.1.9 Instruction cache test 指令cache自检

4.1.9.1 Function Description 功能描述

Instruction cache test.

指令 cache 自检。

4.1.9.2 Argument Description 参数说明

- Definition 函数定义

void CpuTestInstructionCache (void);

➢ Input argument 输入参数

No.

无。

➢ Output argument 输出函数

No.

无。

4.1.9.3 Processing flow 处理流程

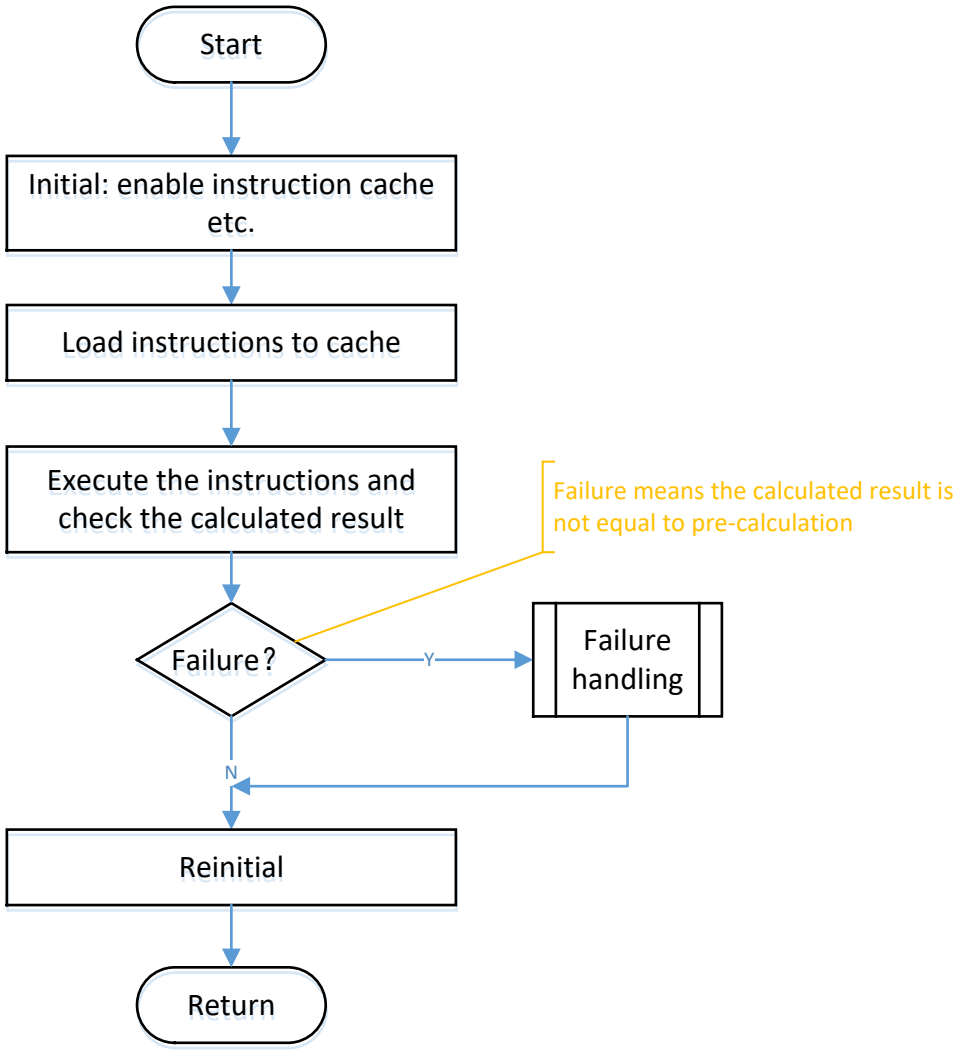

Figure4-7 Instruction cache test processing flow

图 4-7 指令 cache 自检流程图

4.1.10 Data cache test 数据cache自检

4.1.10.1 Function Description 功能描述

Data cache test.

数据 cache 自检。

4.1.10.2 Argument Description 参数说明

- Definition 函数定义

void CpuTestDataCache (void);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

No.

无。

4.1.10.3 Processing flow 处理流程

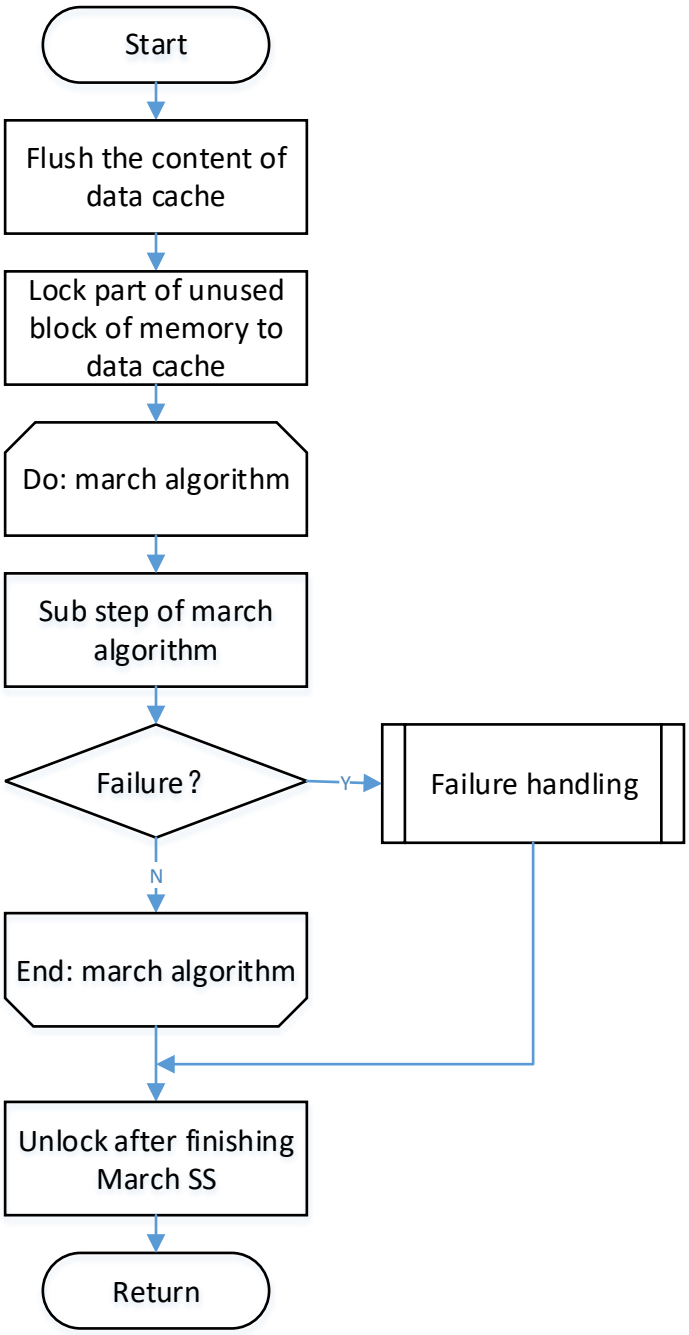

Figure4-8 Data cache test processing flow

图 4-8 数据 cache 自检流程图

4.1.11 Stack test 栈自检

4.1.11.1 Function Description 功能描述

Stack test.

栈空间自检。

4.1.11.2 Argument Description 参数说明

- Definition 函数定义

void StackTest (void);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

No.

无。

4.1.11.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.2 DDR test 内存自检

SWDD-PM-DIAG_SafR_NSecR_A_002

4.2.1 RamTestInit

4.2.1.1 Function Description 功能描述

RAM test initialization.

内存自检初始化。

4.2.1.2 Argument Description 参数说明

- Definition 函数定义

void RamTestInit(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

The processing flow is omitted.

流程图省略。

4.2.2 RamTestCycle

4.2.2.1 Function Description 功能描述

RAM test during running time in every 24 hours.

内存自检初始化。

4.2.2.2 Argument Description 参数说明

- Definition 函数定义

uint32_t RamTestCycle(puint32_t puiVectorTest);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

puint32_t puiVectorTest

Point to the error flag

自检出错指示标志

4.2.2.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.3 Code test 代码自检

SWDD-PM-DIAG_SafR_NSecR_A_003

4.3.1 Firmware test 固件自检

4.3.1.1 Function Description 功能描述

Firmware test will be done during system startup in u-boot.

在 u-boot 中完成代码自检。

4.3.1.2 Argument Description 参数说明

- Definition 函数定义

void FirmwareTest(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 User program test 用户程序自检

4.3.2.1 Function Description 功能描述

User program test will be done after the project download to the PM.

工程下装到PM后需执行进行用户程序自检。

4.3.2.2 Argument Description 参数说明

- Definition 函数定义

void UserProgramTest(puint32_t puiVectorTest);

- Input argument 输入参数

No.

无。

- Output argument 输出函数

puint32_t puiVectorTest

Point to the error flag

自检出错指示标志

4.3.2.3 Processing flow 处理流程

The processing flow is omitted.

流程图省略。

4.4 Exception test 异常自检

SWDD-PM-DIAG_SafR_NSecR_A_004

4.4.1 ExceptionTest

4.4.1.1 Function Description 功能描述

Exception test will be executed periodically.

对异常进行周期自检

4.4.1.2 Argument Description 参数说明

- Definition 函数定义

void ExceptionTestCycle(puint32_t puiVectorTest);

- Input argument 输入参数

No.

无

- Output argument 输出函数

puint32_t puiVectorTest

Point to the error flag

自检出错指示标志

4.4.1.3 Processing flow 处理流程

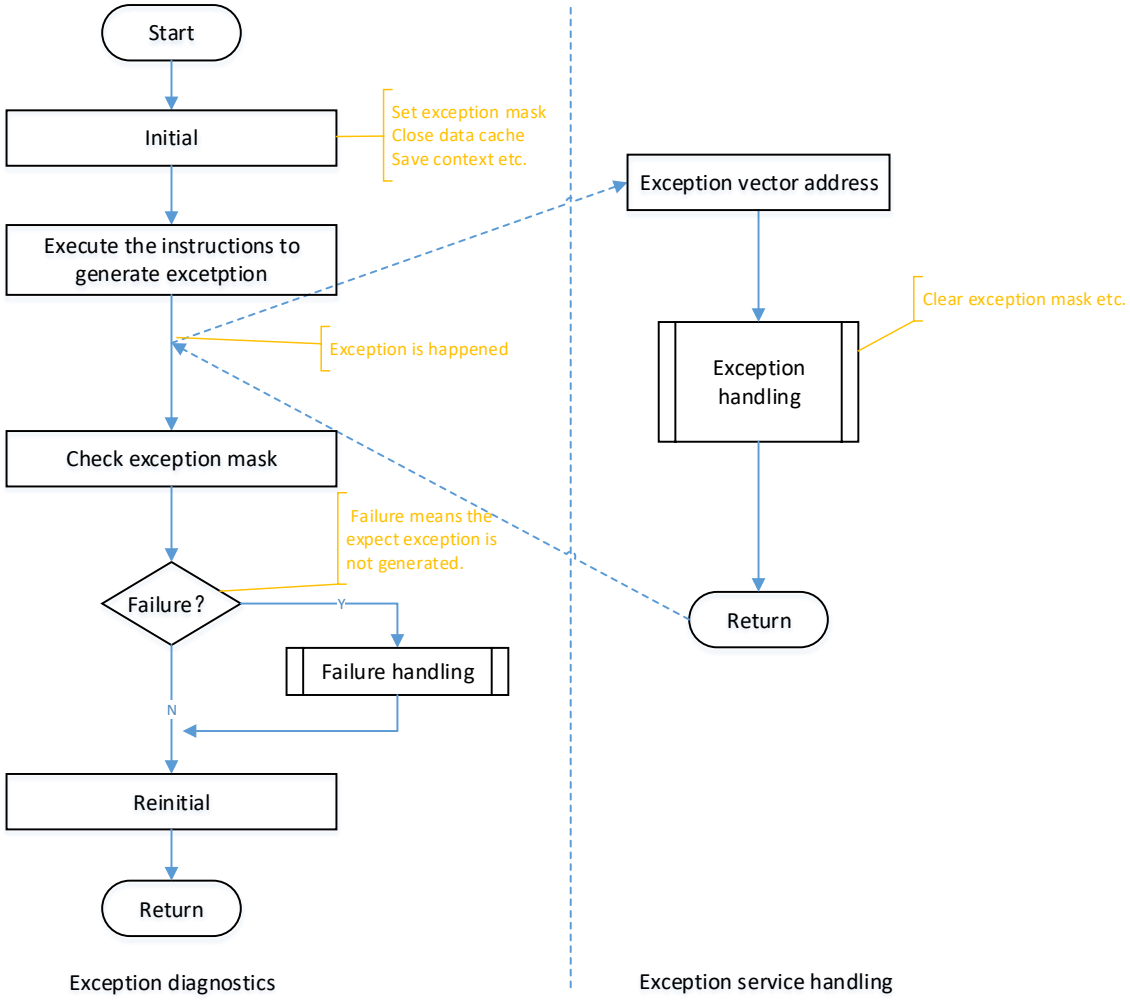

Figure4-9 Exception test processing flow

图 4-9 异常自检流程图

——以下无正文