Application Note

Safety Startup Application Note

Introduction

Safety plays an increasingly important role in the field of electronic applications. In electronic design, the level of component safety requirements is rising, and electronic equipment manufacturers are incorporating many new technology solutions into new component designs. Software technologies for improving safety are emerging. Standards of hardware and software safety requirements are also under continuous development.

This document describes how the project in N32G43x MCU perform the requirements of IEC60730 software safety related operations and related application code content.

This document applies to the N32G43x series products of Nsing Technologies.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Content

Content **2**

1. IEC60730 Class B Software Standard Introduction **3**

2. Test Point Process Description **4**

2.1 Check the Flow at Startup 5

2.1.1 CPU Self-check at Startup 6

2.1.2 Watchdog Self-check at Startup 7

2.1.3 FLASH Self-check at Startup 8

2.1.4 RAM Self-check at Startup 12

2.1.5 Clock Self-check at Startup 13

2.1.6 Control Flow Self-check at Startup 17

2.2 Self-check Process at Runtime 17

2.2.1 CPU Runtime Self-check 17

2.2.2 Stack Boundary Runtime Overflow Self-check 18

2.2.3 System Clock Runtime Self-check 19

2.2.4 FLASH Runtime Self-check 20

2.2.5 Watchdog Runtime Self-check 21

2.2.6 Partial RAM Runtime Self-check 21

3. Key Points of Software Library Migration **24**

4. Version History **25**

5. Disclaimer **26**

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road, Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

1. IEC60730 Class B Software Standard Introduction

To ensure the safety of electrical appliances, risk control measures during software operation need to be evaluated.

IEC60730, issued by the International Electrotechnical Commission, introduces the requirements for the evaluation of software for household appliances. In Appendix H(H.2.21), software is classified as follows:

Class A software: the software only implements the functions of the product and does not involve the safety control of the product. For example, software for household thermostats, lighting controls...

Class B software: software designed to prevent unsafe operation of electronic devices. For example, the software of washing machine with automatic door lock control, the software of induction cooker with overheating control...

Class C software: software designed to avoid certain specific danger. Such as automatic burner control and thermal cut-off for enclosed water heater (mainly for device that may cause explosions)

The specific evaluation requirements of class B software include components to be tested and related faults and test schemes, which are sorted out in the following table (refer to IEC60730 Table H.11.12.7):

Table 1-1 IEC60730 Class B Evaluation Requirement

Components to be detected		Fault/error	Fault classification	Nsing with library	Test Solution Overview
1.CPU	1.1 Register	Hysteresis (Stuck at)	MCU related	Y	Write relevant registers and check
	1.3 Program counter	Hysteresis (Stuck at)	MCU related	Y	When the PC crashes, the watchdog reset s triggered
2.Interruption		No interrupts or interrupts too frequently	Application related	N	Count the number of interrupts
3. The clock		Wrong frequency	MCU related	Y	Use HSI to measure HSE clock frequency
4. Memory	4.1 Non-volatile memory	All single bit errors	MCU related	Y	CRC integrity check of Flash
	4.2 Volatile memory	DC fault	MCU related	Y	1. SRAM March C test 2. Stack overflow detection
	4.3 Addressing (related to non-volatile and volatile memory)	Hysteresis (Stuck at)	MCU related	Y	Flash/SRAM tests are included
5. Internal data path	5.1 data	Hysteresis (Stuck at)	MCU related	N	Only for MCUs using external memory,
	5.2 addressing	Wrong address	MCU related	N	

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road, Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

					monolithic MCUs are not required
External communication	6.1 data	The Hamming distance is 3	Application related	N	Add verification in data transfer
	6.2 addressing	Wrong address	Application related	N	
	6.3 sequential	Wrong timing	Application related	N	Count the number of communication events
7. Input and output	7.1 digital I/O	Error defined in H27	Application related	N	None
	7.2 Analog input and output	Error defined in H27	Application related	N	None

2. Test Point Process Description

The check content of class B software package program is divided into two main parts: self-check at startup and periodic self-check at runtime. Self-test at startup includes:

- CPU self-check
- Watchdog self-check
- Flash integrity self-check
- RAM function self-check
- System clock self-check
- Control flow self-check

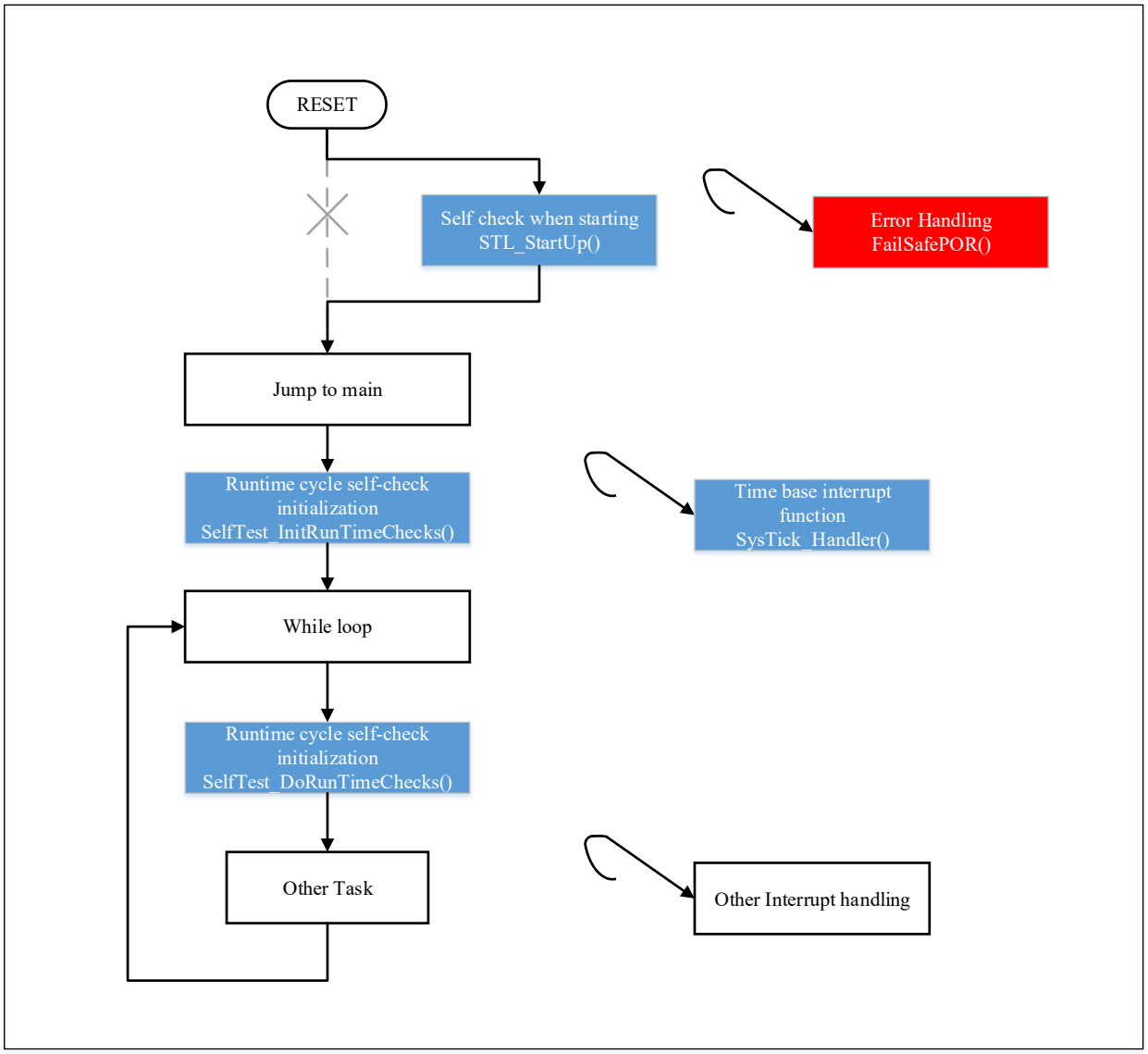
Periodic self-check at self-check:

- Partial CPU core register self-check
- Stack boundary overflow self-check
- System clock running self-check
- Flash CRC segmentation self-check
- Watchdog self-check
- Partial RAM self-check (in interrupt service routines)

The overall flow diagram is as follows:

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road, Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Figure 2-1 Self-check Flow

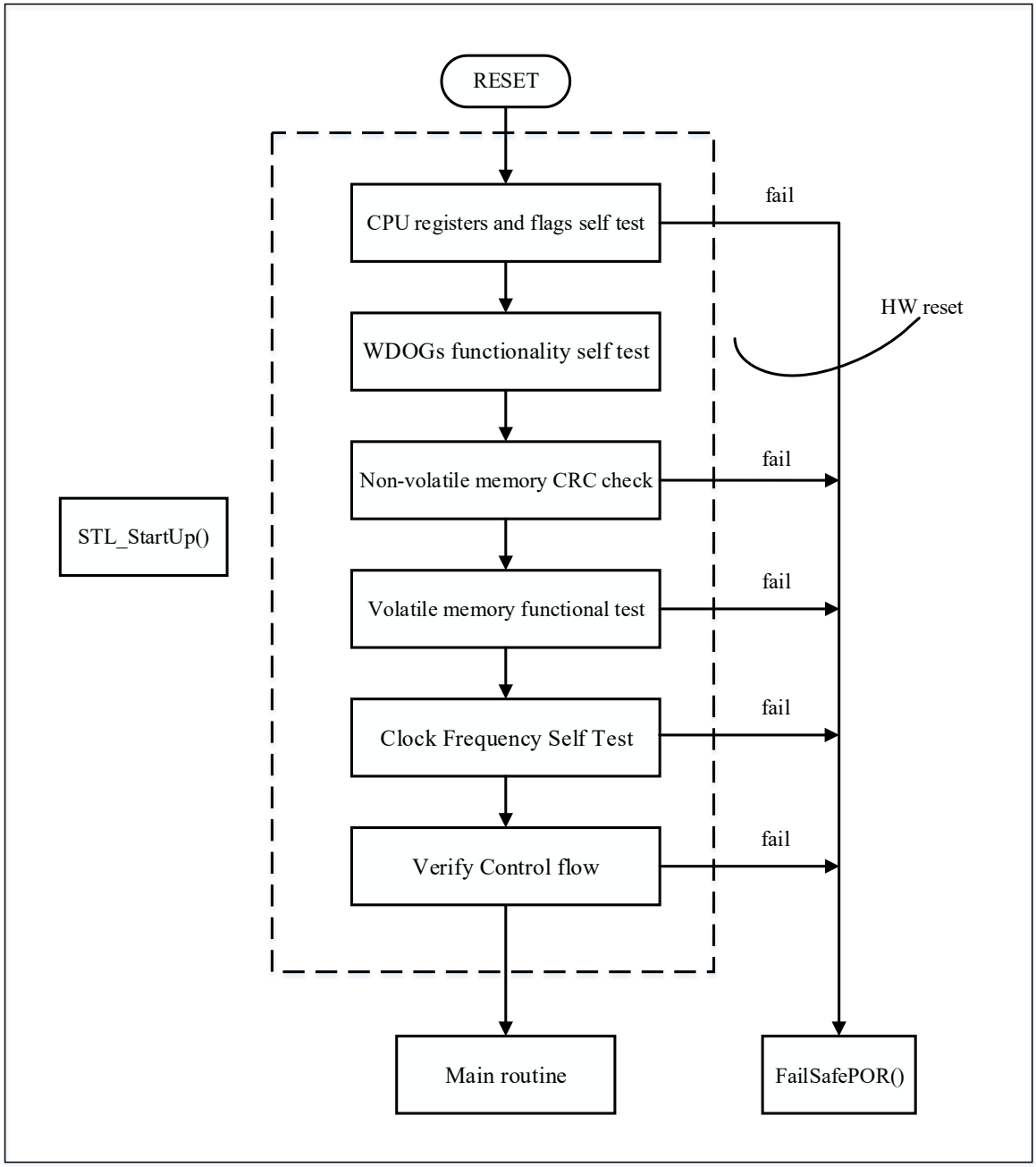


2.1 Check the Flow at Startup

Before the chip enters main function from startup, the startup self-check is carried out first. The startup file is modified to execute this part of the code. After the self-check process is over, the __iar_program_start function is called to jump back to main function.
The following is a flow diagram for performing a startup self-check:

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Figure 2-2 Self-check Flow at Startup



2.1.1 CPU Self-check at Startup

CPU self-check mainly checks whether the core flags, registers and so on are correct. If an error occurs, FailSafePOR () is called.

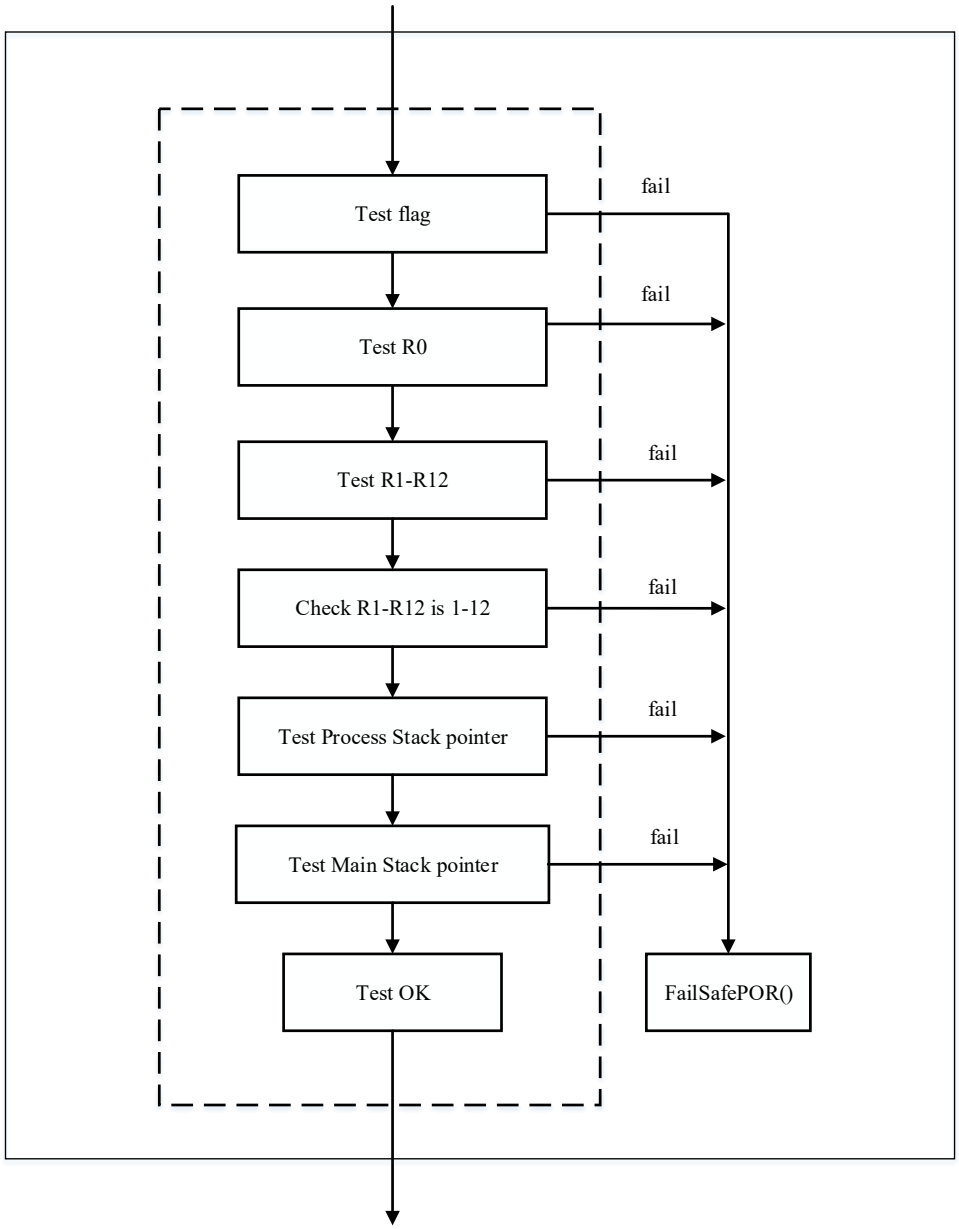
CPU self-check will be carried out at both startup and runtime. At a self-check will be conducted on the functionality of registers startup, R0~R12, PSP, MSP register and Z(zero), N(negative), C(carry), V(overflow) flag bit. During runtime, there will be periodic self-check, which is only applied to registers R1~R12.

Register self-check is implemented as follows: write 0xAAAAAAAA and 0x55555555 to the register respectively, and then compare whether the read value is the written value; write 1 after R1 is tested, write 2 after R2 is tested, and so on.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road, Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Flag bit self-check is implemented as follows: set the flag bits respectively; if an error is detected in the flag bits, enter the fault function. The diagram of self-check flow is as follows:

Figure 2-3 CPU Self-check Flow at Startup



2.1.2 Watchdog Self-check at Startup

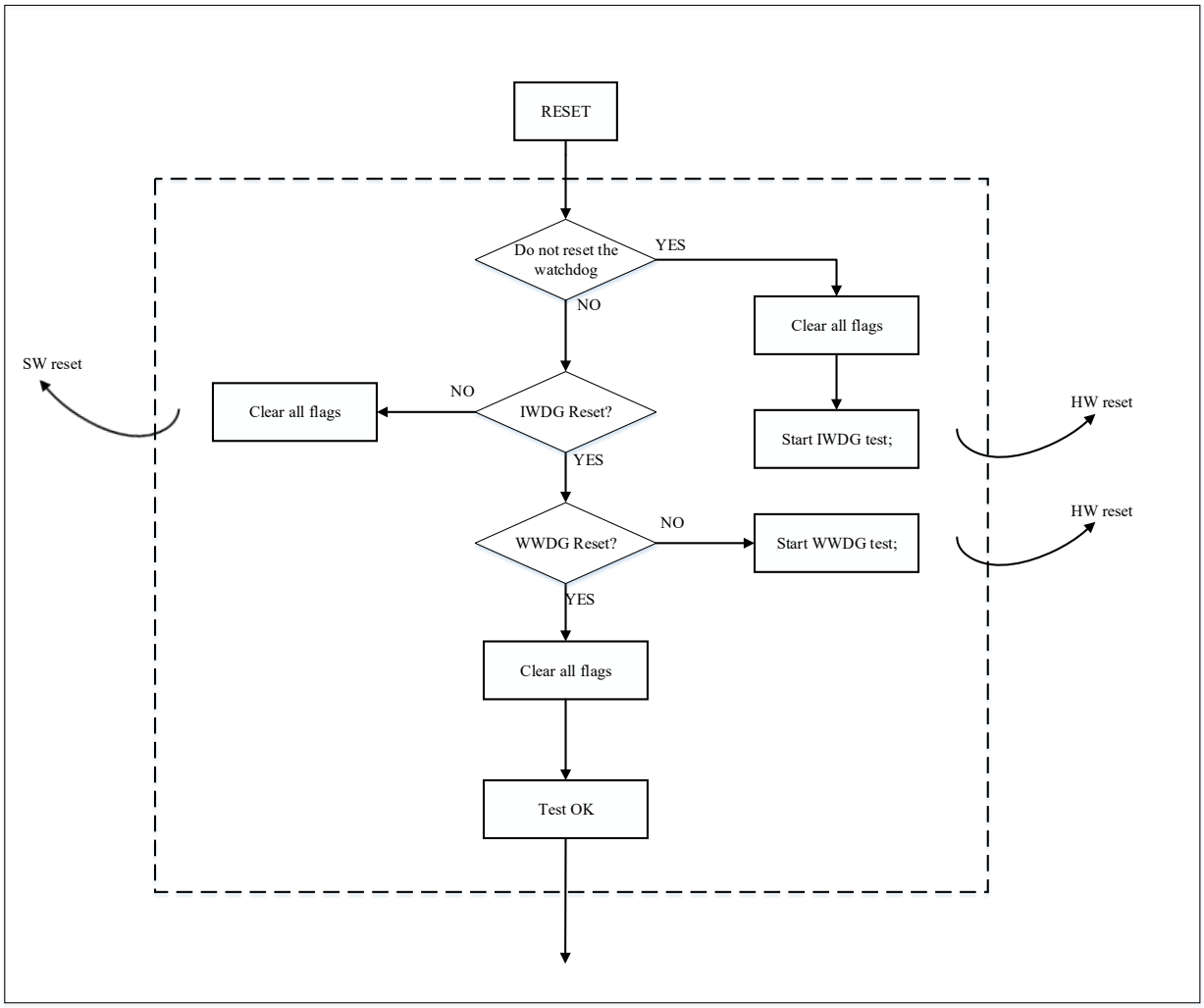
Watchdog self-check is to test the independent watchdog and window watchdog to confirm that they can be reset correctly to ensure that if the program crashes during runtime, the system can reset in time to prevent a deadlock.

The self-check process is: after the initial reset, clear all reset status register flag bits; start the IWDG test, reset the chip, and judge whether the IWDG reset flag bit is set; if it is set, start the WWDG test to reset the chip; if the WWDG reset flag bit is set, the watchdog test passes; clear all flags.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road, Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

The flow diagram is as follows:

Figure 2-4 Watchdog Self-check Flow at Startup

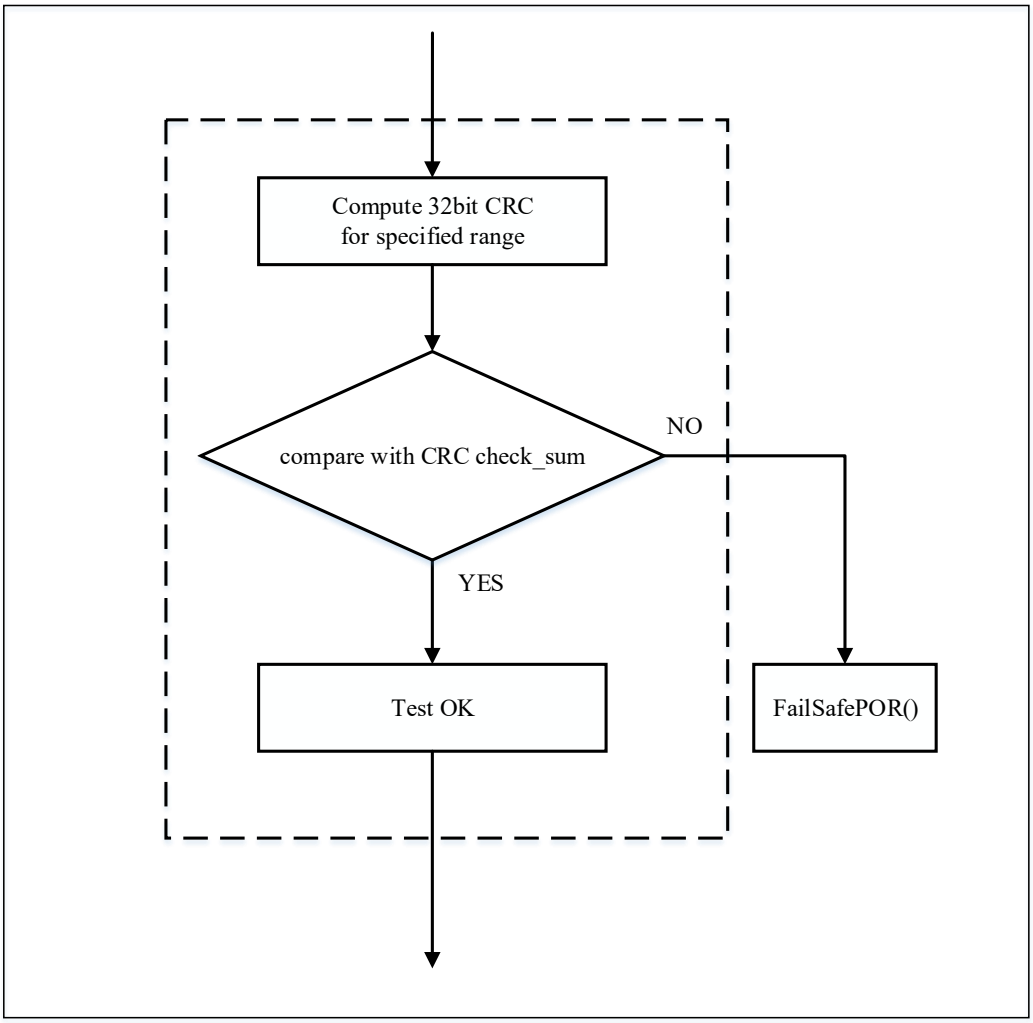


2.1.3 FLASH Self-check at Startup

FLASH self-check is a program that calculates Flash data with CRC algorithm. Then compares the result value with the CRC value stored in the specified location of Flash, which is calculated during compilation, to confirm the integrity of Flash.
The flow diagram is as follows:

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Figure 2-5 Flash Self-check Flow at Startup



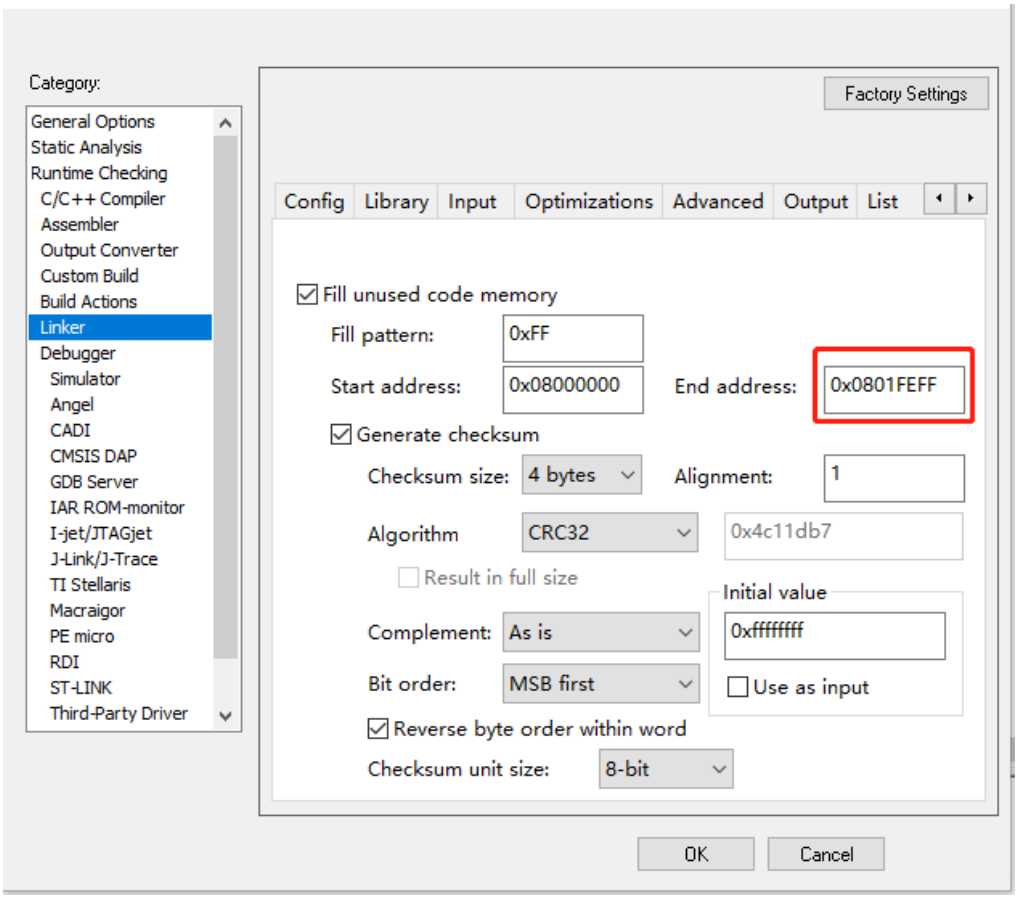
The address range of Flash for CRC calculation is configured according to the actual situation of the whole program. The configuration method is different in Keil and IAR.

IAR configuration:

The CRC calculation is supported in the IAR configuration options. Just by configuring the parameters properly, the CRC check_sum value will automatically be added to the selected Flash calculation range in the compiled file.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Figure 2-6 IRA Configuration Options



The range of calculating CRC in the program is configured in the .icf file, which can be modified according to the needs. The configured end address need to be incremented by 4 based on the value in above figure.

Figure 2-7 IRA .icf File Configuration

```
ecord_crc32.bat | crc_load.ini | N32G43x.icf
/*###ICF### Section handled by ICF editor, don't touch! ****/
/*-Editor annotation file-*/
/* IcfEditorFile="$TOOLKIT_DIR$\config\ide\IcfEditor\cortex_v1_0.xml" */
/*-Specials-*/
define symbol __ICFEDIT_intvec_start__ = 0x08000000;
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x08000000;
define symbol __ICFEDIT_region_ROM_end__   = 0x0801FF03;  /* Modify according to needs,Contains crc results */
define symbol __ICFEDIT_region_RAM_start__ = 0x20000100;
define symbol __ICFEDIT_region_RAM_end__   = 0x20007FFF;  /* Modify according to needs */

define symbol __ICFEDIT_region_CLASSB_start__ = 0x20000040;
define symbol __ICFEDIT_region_CLASSB_end__   = 0x20000100;
```

Keil configuration:

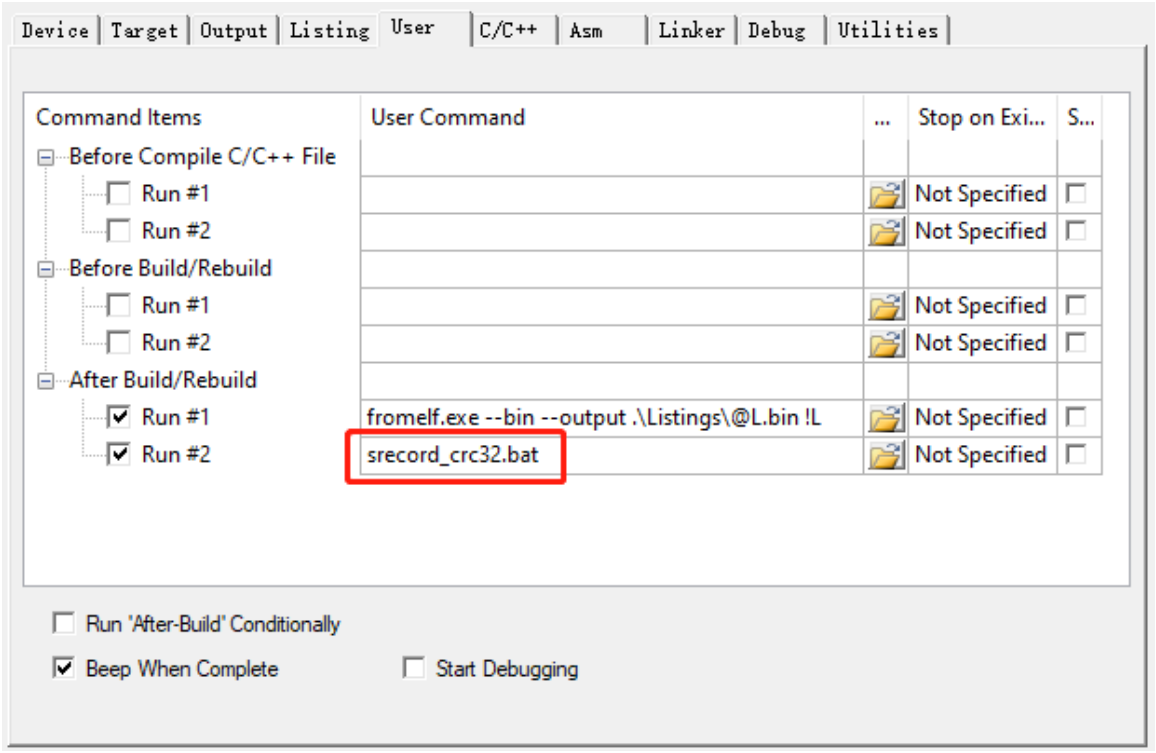
The configuration of Keil is more complicated. ARM officially recommends using the third-party software SRecord for ROM Self-Test in MDK-ARM.

According to the project configuration, after the compilation is completed, the script file srecord_crc32.bat will be called. Then by running the srec_cat.exe software, the data in the

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road, Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

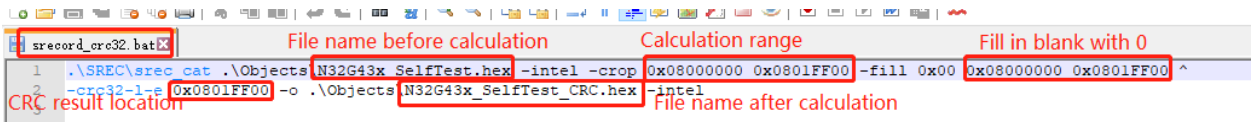
N32G43x_SelfTest.hex file generated by Keil compilation will be subjected to CRC calculation. The generated CRC check result will be added toa specified location to obtain the new N32G43x_SelfTest_CRC.hex file:

Figure 2-8 Keil Configuration Option



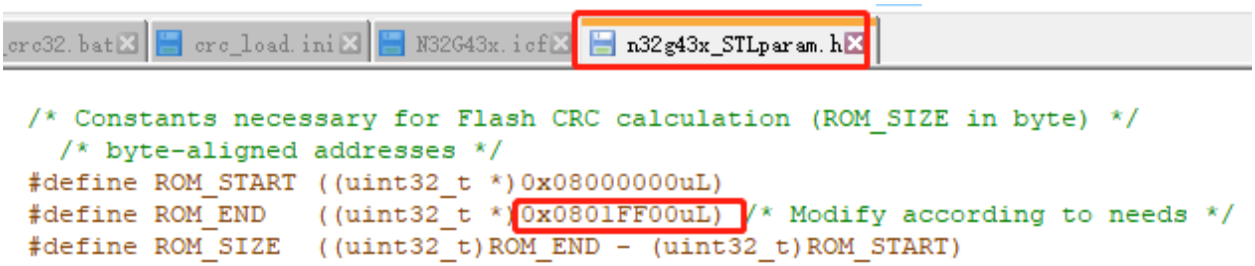
Open the .bat file with Notepad or other tools, and modify the following according to the actual application:

Figure 2-9 Keil .bat File Configuration



The range of calculating CRC in the program can be configured in the n32g43x_STLparam.h file according to requirements, which is consistent with the configuration in above figure:

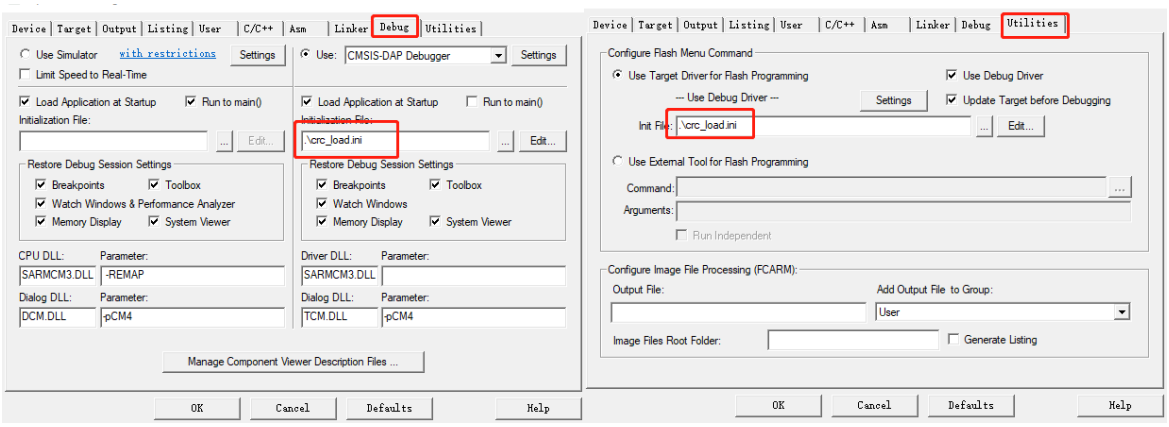
Figure 2-10 Keil .h File Configuration



Therefore, the final generated N32G43x_SelfTest_CRC.hex file needs to be used whether it is downloading or debugging. So, the .ini file needs to be added to the Keil configuration option to download the new .hex file. The configuration is as follows:

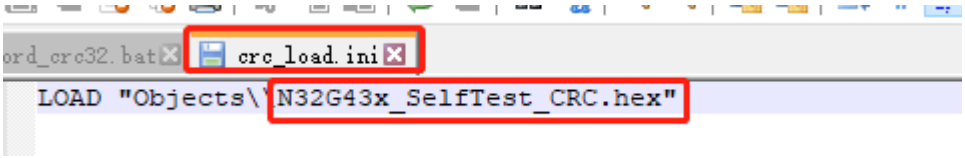
NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road, Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Figure 2-11 Keil Configuration to Add .ini File



It should be noted that the .ini file should also be configured with the file name of the actual application requirements that need to be modified.

Figure 2-12 Keil .ini File Configuration



2.1.4 RAM Self-check at Startup

SRAM self-check detects errors not only in the data region, but also in its internal address and data path.

SRAM self-check uses The March-C algorithm, which is used for SRAM testing of embedded chips as part of safety certification. All ranges of SRAM are checked at startup.

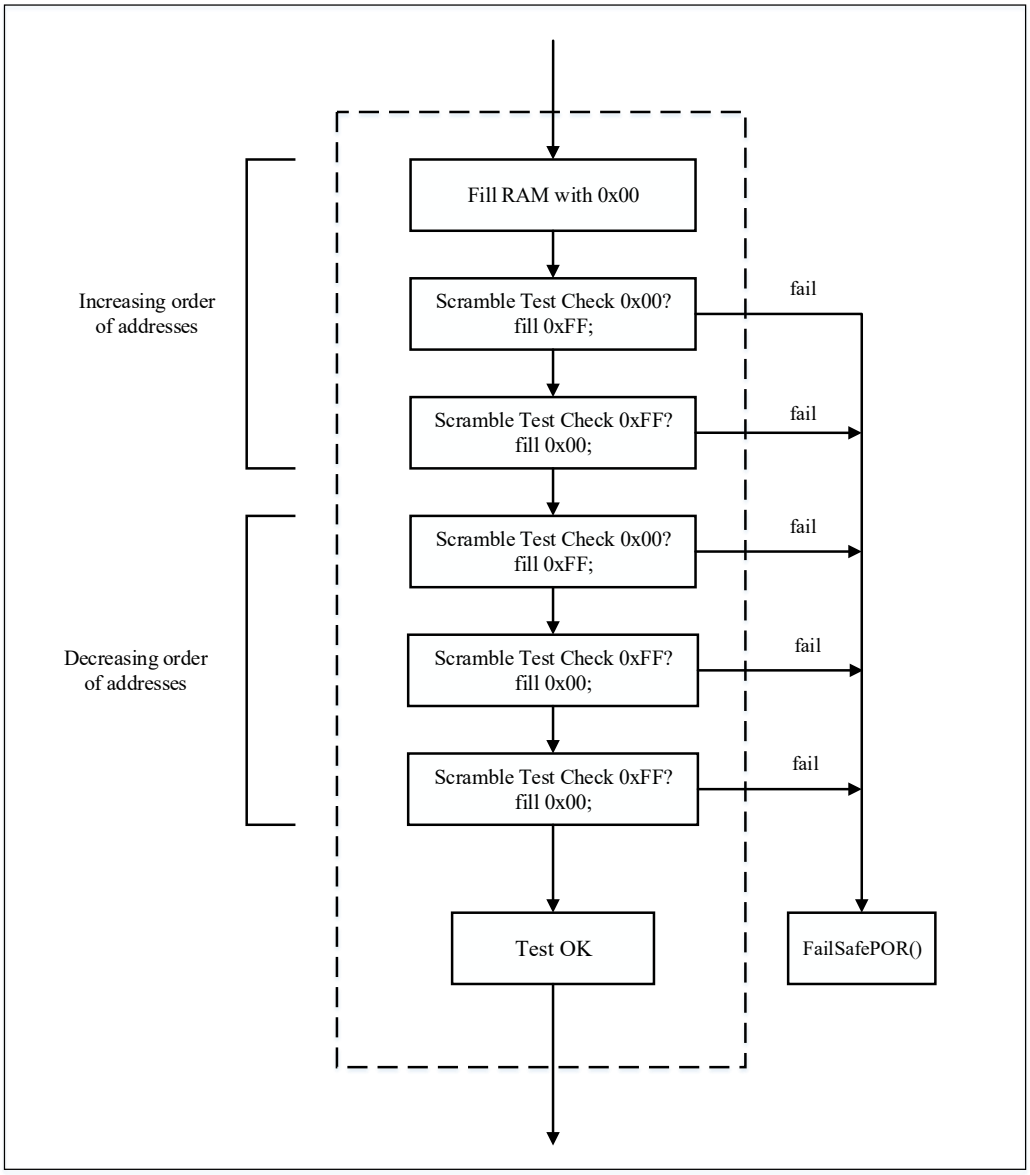
First, the whole SRAM is cleared. Then, set the SRAM data to 1 bit by bit, and check if this bit is set successfully. If this bit is set, continue to next bit till whole SRAM range is done. Otherwise, an error is reported; After all bits in SRAM are set, clear it to 0 bit by bit and check if this bit is cleared. If this bit is cleared, continue to next bit till entire SRAM range is done. Otherwise, an error is reported.

The test is divided into 6 loops, where the values 0x00 and 0xFF are alternately filled and checked word by word in the entire RAM. The first 3 loops are executed with increasing addresses, while the last 3 loops are executed with decreasing addresses.

The entire RAM self-check algorithm process is shown in the figure below:

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Figure 2-13 RAM Self-check Flow at Startup



2.1.5 Clock Self-check at Startup

The self-check process is as follows:

1. Start the high-speed external (HSE) clock source.
2. Before the test starts, the system clock source is set to high-speed internal (HSI) clock by default. Then, RTC (clock source is LSI) and SysTick (clock source is system clock) is initialized, and start to run at the same time. When the SysTick count is decremented from the reload value to 0, the current RTC count value is recorded to get the HSI Cnt.
3. Set the system clock source to HSE and initialize RTC (clock source is LSI) and SysTick (clock source is system clock) to obtain HSE Cnt in the same manner.
4. Take HSI clock frequency as the reference, calculate HSE frequency according to the formulas in the following flow chart. Then compare the frequency value with the expected range value: if it exceeds the expected value by +/-25%, the test fails. After the test, switch the system clock source to HSI. The expected range can be adjusted by users according to actual applications. Macros are

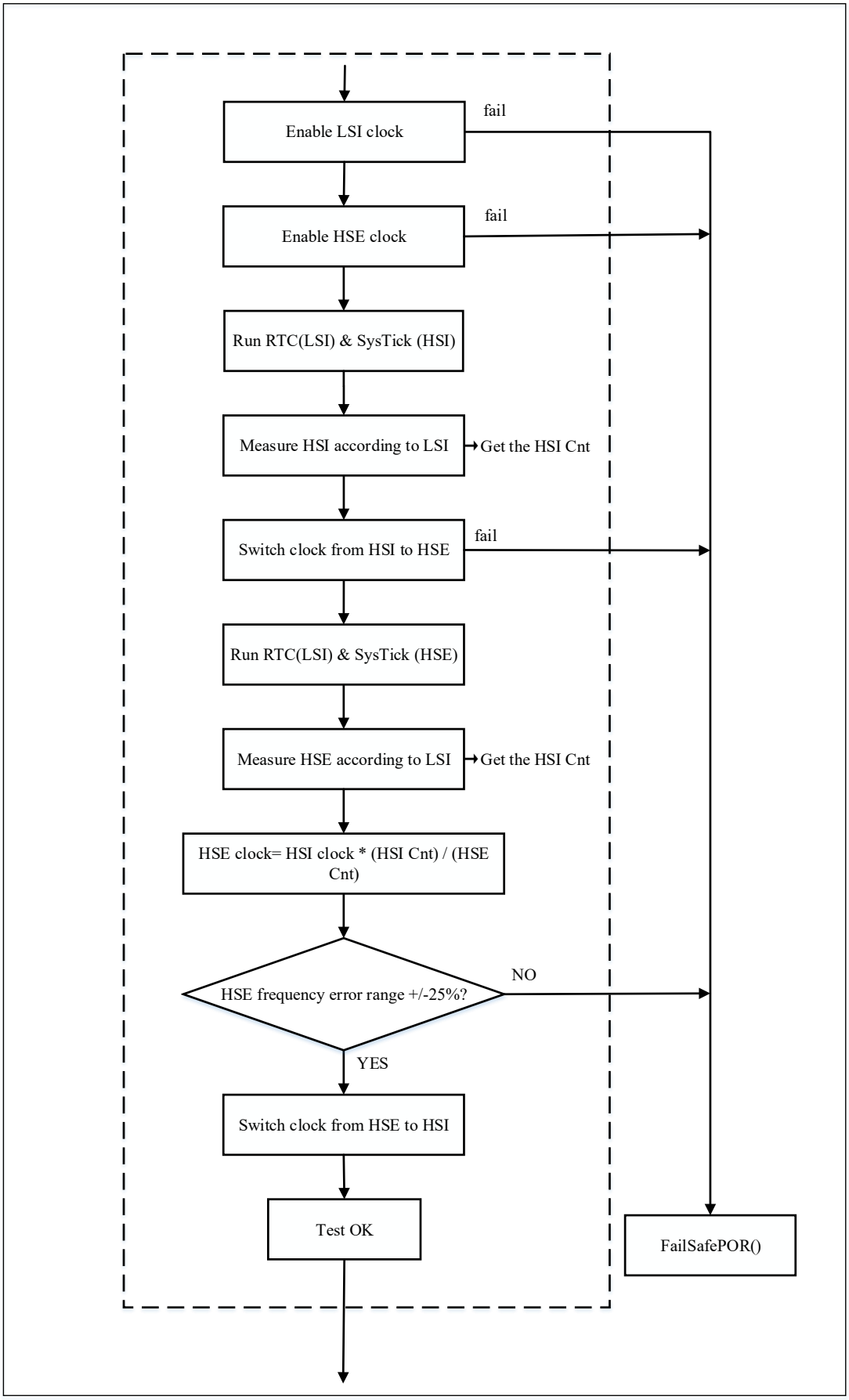
NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road, Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

defined as HSE_LimitHigh() and HSE_LimitLow().

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Figure 2-14 Clock Self-check Flow at Startup

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg



NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

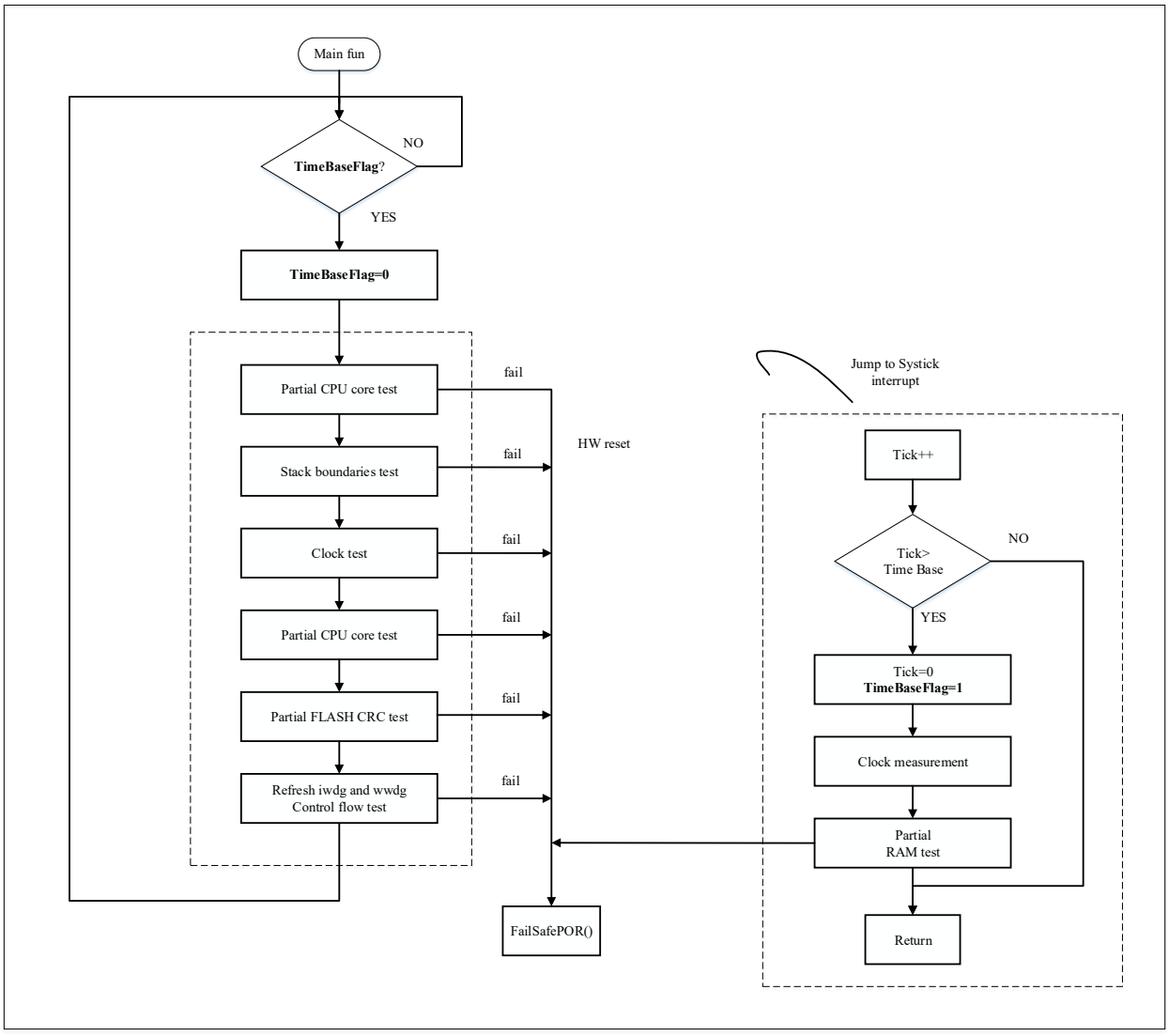
2.1.6 Control Flow Self-check at Startup

The self-check part of the startup ends with the control flow self-check pointer program. Initialize the variable CtrlFlowCnt to 0, and CtrlFlowCntInv to 0xFFFFFFFF. In each test step, CtrlFlowCnt adds a fixed value, and CtrlFlowCntInv subtracting the same fixed value. At the end of the startup self-check, judge whether the sum of the two values is still 0xFFFFFFFF.

2.2 Self-check Process at Runtime

If the startup self-check passes successfully, the run-time periodic self-check must be initialized before entering the main loop.
The runtime self-check is performed periodically based on SysTick. The run-time periodic self-check process is as follows:

Figure 2-15 Self-check Flow at Runtime



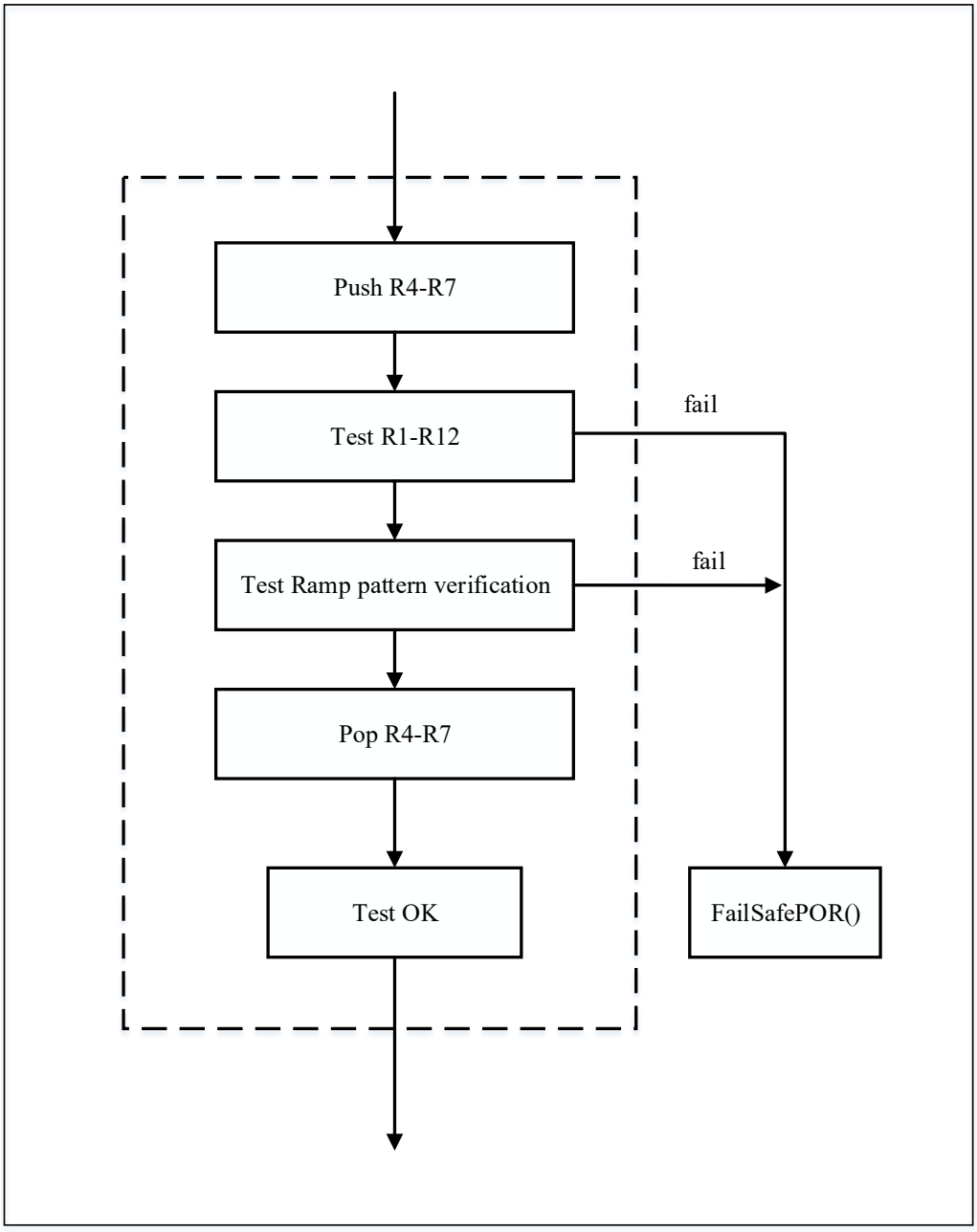
2.2.1 CPU Runtime Self-check

The CPU runtime periodic self-check is similar to the self-check at startup, except that the core

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

flags and stack pointers are not checked.

Figure 2-16 CPU Self-check Flow at Runtim



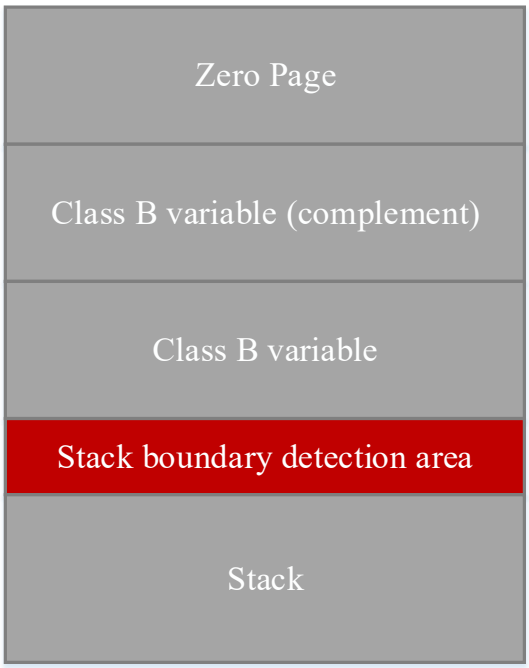
2.2.2 Stack Boundary Runtime Overflow Self-check

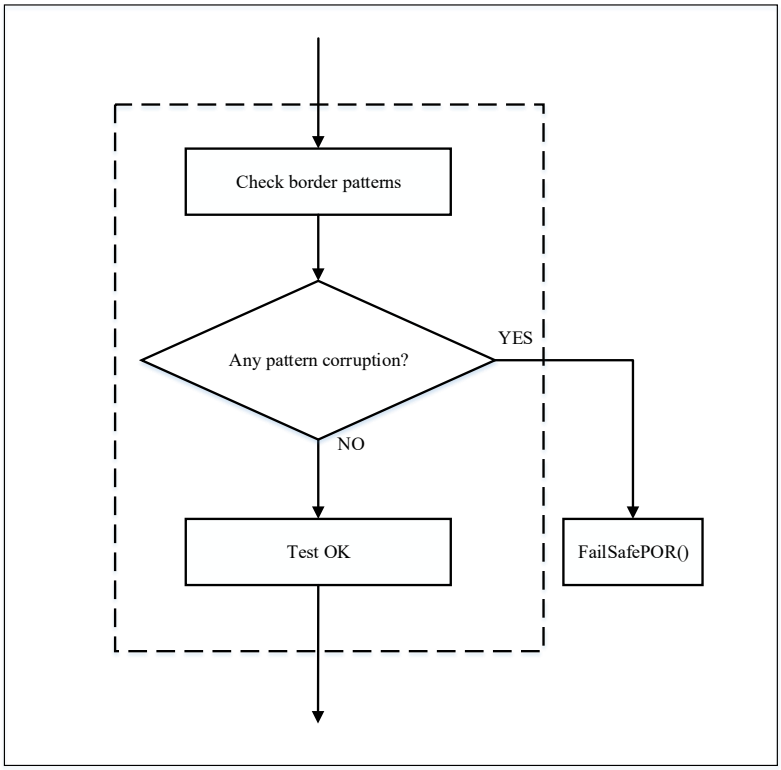
This test detects stack overflow by determining the data integrity of pattern array in the boundary detection area. If the original pattern data is corrupted, the test fails and a fault-tolerant program is invoked.

The lower address closely following the stack area is defined as the stack boundary detection area. This area can be configured differently depending on the device. The user must define enough areas for the stack and ensure that pattern is placed correctly.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Figure 2-17 Stack Boundary Self-check Flow at Runtime



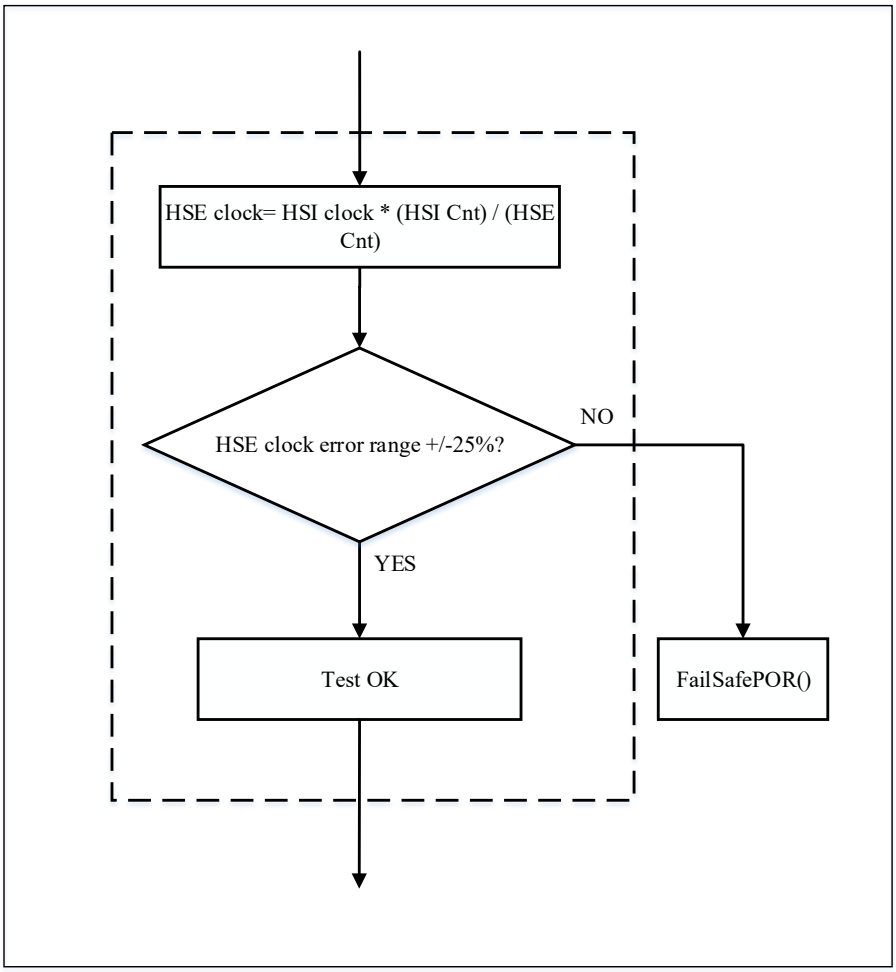


2.2.3 System Clock Runtime Self-check

The self-check of system clock during runtime is similar to that during startup. HSE frequency is calculated through HSI Cnt and HSE Cnt, and the process is as follows:

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Figure 2-18 System Clock Self-check Flow at Runtime

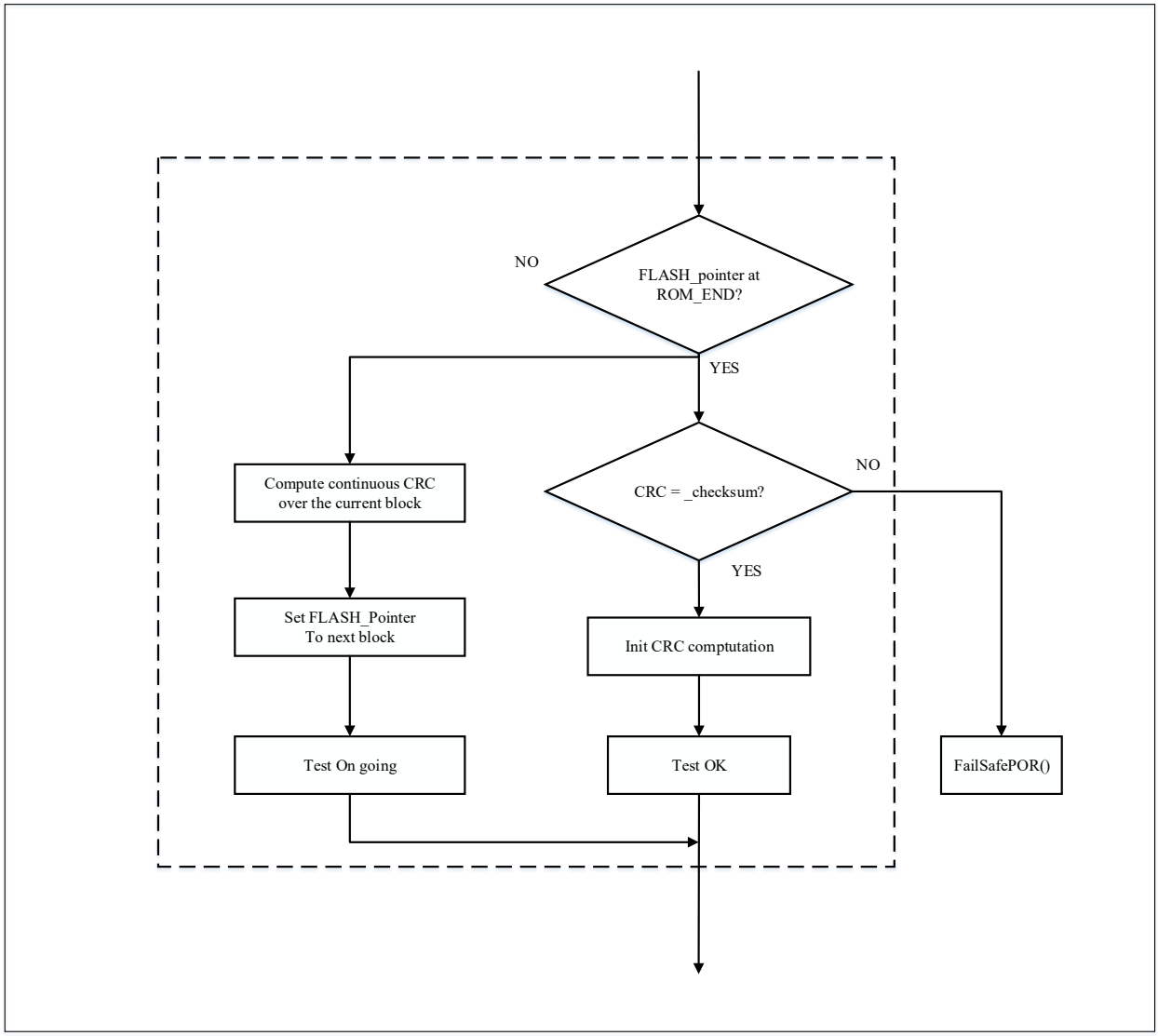


2.2.4 FLASH Runtime Self-check

The Flash CRC self-check is performed during the runtime. The self-check time length varies depending on the check range. The check range for segmented CRC calculation can be configured based on the size of the user application. When the CRC values are calculated to the last range, the CRC values are compared. If the values are not expected, the self-check fails.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Figure 2-19 Flash Self-check Flow at Runtime



2.2.5 Watchdog Runtime Self-check

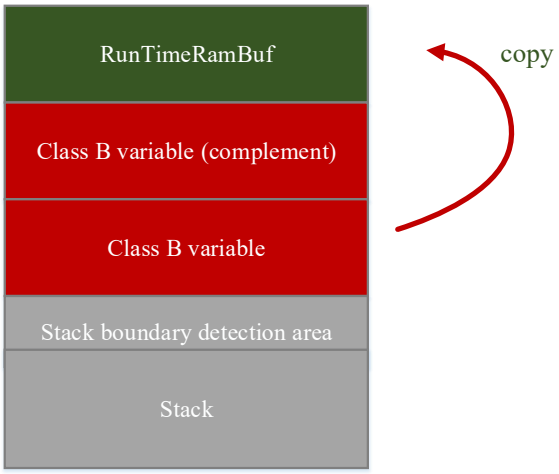
During runtime, watchdogs need to be fed regularly to ensure the normal operation of the system. The watchdog feeding part is placed at the end of STL_DoRunTimeChecks().

2.2.6 Partial RAM Runtime Self-check

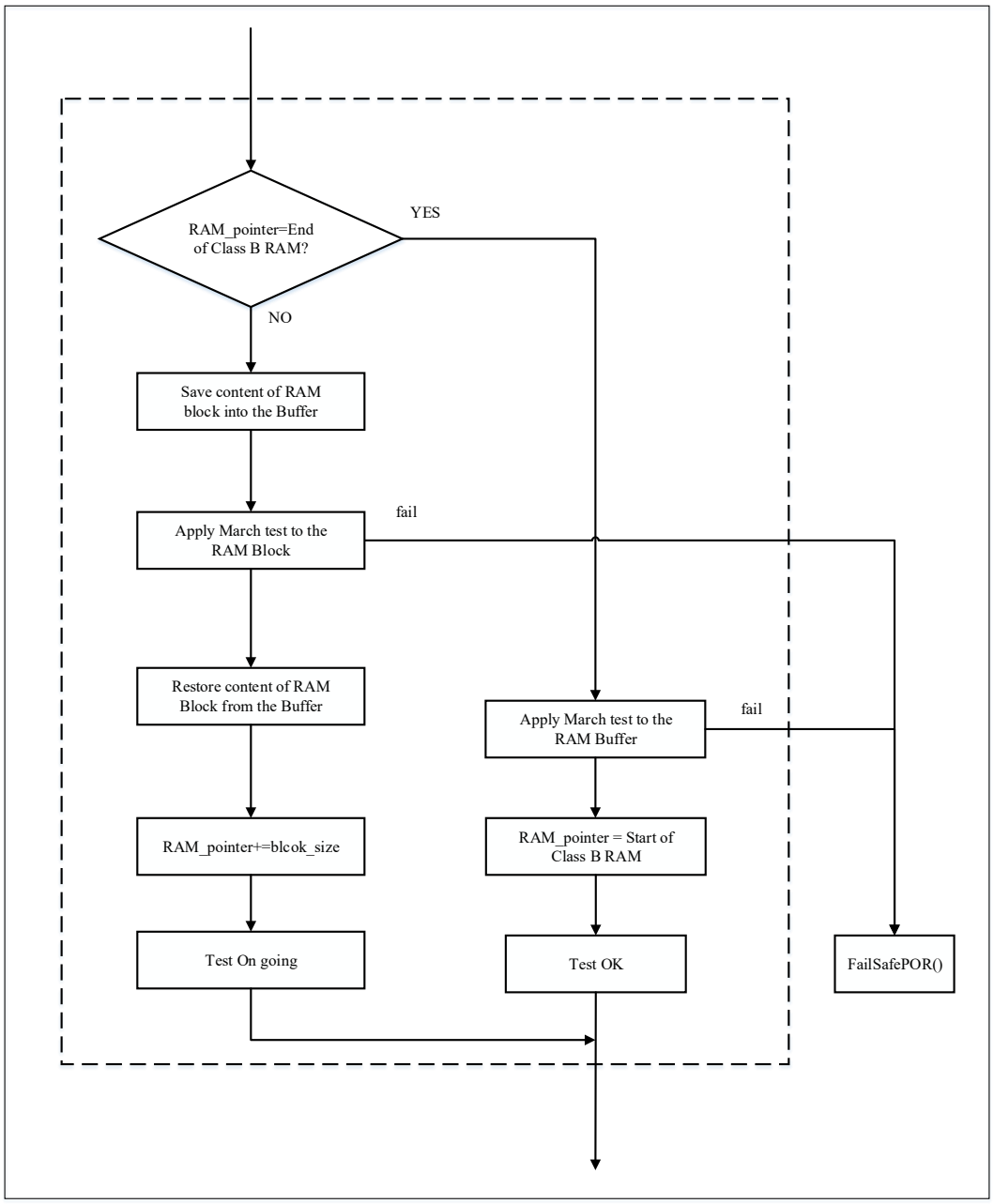
The RAM self-check at run time is done in the SysTick interrupt function. The test covers only thepartial memory allocated to the class B variable.
The area allocated to the class B variable, is divided into blocks, which each block consisting of 6 bytes. Before the March-C test, save the block data in the RunTimeRamBuf, and then put the data in RunTimeRamBuf back to the original position of the class B area after the test is completed. Repeat the operation until all tests in the class B area are completed.
After the class B area test is complete, the March-C test is performed in RunTimeRamBuf area. After the test is complete, the pointer is restored to the class B start address for the next test.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road, Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

Figure 2-20 RAM Self-check Flow at Runtime



NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg



NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

3. Key Points of Software Library Migration

- Before executing the user program, execute the STL_StartUp function (to start the self-check);
- Set WWDG and IWDG to prevent them from being reset when the program is running normally;
- Set up RAM and Flash self-check range for startup and runtime;
  - The range of CRC checksum, and the location where the checksum is stored in the Flash
  - The range of storage addresses for Class B variables
  - Location of stack boundary self-check area
- Troubleshoot detected faults.
- Add user-related fault detection content based on specific applications;
- Define the frequency of program runtime self-check according to the specific application;
- After the chip is reset, the STL_StartUp function must be called for startup self-check before initialization.
- Call STL_InitRunTimeChecks() before entering the main loop, and call STL_DoRunTimeChecks() in the main loop;
- Users can release Verbose comments to enter diagnostic mode and output text information through the Tx (PA9) pin of USART1.

Set the serial port to 115200Bits/s, no parity, 8-bit data, and 1 stop bit.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

4. Version History

Version	Date	Note
V1.0	2021.12.06	Initial release

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road,
Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg

5. Disclaimer

This document is the exclusive property of NSING TECHNOLOGIES PTE. LTD.(Hereinafter referred to as NSING).

This document, and the product of NSING described herein (Hereinafter referred to as the Product) are owned by NSING under the laws and treaties of Republic of Singapore and other applicable jurisdictions worldwide. The intellectual properties of the product belong to Nations Technologies Inc. and Nations Technologies Inc. does not grant any third party any license under its patents, copyrights, trademarks, or other intellectual property rights. Names and brands of third party may be mentioned or referred thereto (if any) for identification purposes only. NSING reserves the right to make changes, corrections. enhancements, modifications, and improvements to this document at any time without notice. Please contact NSING and obtain the latest version of this document before placing orders.

Although NATIONS has attempted to provide accurate and reliable information, NATIONS assumes no responsibility for the accuracy and reliability of this document. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. In no event shall NATIONS be liable for any direct, indirect, incidental, special, exemplary, or consequential damages arising in any way out of the use of this document or the Product.

NATIONS Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, Insecure Usage'. Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, all types of safety devices, and other applications intended to supporter sustain life. All Insecure Usage shall be made at user's risk. User shall indemnify NATIONS and hold NATIONS harmless from and against all claims, costs, damages, and other liabilities, arising from or related to any customer's Insecure Usage Any express or implied warranty with regard to this document or the Product, including, but not limited to. The warranties of merchantability, fitness for a particular purpose and non-infringement are disclaimed to the fullest extent permitted by law. Unless otherwise explicitly permitted by NATIONS, anyone may not use, duplicate, modify, transcribe or otherwise distribute this document for any purposes, in whole or in part.

NSING Technologies Pte. Ltd.
Add: NSING, Teletech Park #02-28, 20 Science Park Road, Singapore 117674
Tel: +65 69268090
Email: sales@nsing.com.sg