Memory
Self-diagnosis Program
Application Note

Outlines

This application note describes the self-diagnosis program for the memories.
The self-diagnosis program in this document is the library which is used to check the read from and write to the memory.
This library supports the sample program with the peripheral driver of TXZ series enclosed, and should be used after it is overwritten to the sample program.

© 2019
Toshiba Electronic Devices & Storage Corporation

Table of Contents

Outlines....1
Table of Contents....2
1. Preface....4
2. Outline of Self-diagnosis Test Library....5
2.1. Self-diagnosis Sample Project....5
3. Details of Self-diagnosis Test Library....6
3.1. RAM Test....6
3.2. Flash Memory Test....7
3.3. Setting for CRC Calculation Test....11
4. Modification of Linker Setting File for CRC32/CRC16 Calculation....12
5. List of Used Drivers....13
6. Reference Document....14
7. Revision History....15
RESTRICTIONS ON PRODUCT USE....16

Arm, Cortex and Keil are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

All other company names, product, and service names mentioned herein may be trademarks or registered trademarks of respective companies.

1. Preface

This application note describes the self-diagnosis program for the memories.
This library code should be used after it is added (overwritten) to the published sample program.

The explanation of this document uses the TMPM4K Group (1).
When the program is applied to a product, some appropriate modification may be necessary depending on the specifications of the product.

This sample program has been developed and evaluated under the conditions in the operation confirmation environment in “Self-diagnosis Program Application Note – Basic Setting”, and by using the TMPM4K4A 1.0.0 sample program and the reference manual released in February in 2019.

2. Outline of Self-diagnosis Test Library

This self-diagnosis test library has been checked on the evaluation board.

The following self-test function is supported.

Name	Description
RAM test	The checker patterns, 0x55555555 and 0xAAAAAAAA, are written to an address in the memory, respectively, and it is checked that the read data of the address is correct. This test is done to each address in a specified memory region (18 KB maximum).
FLASH test	The CRC32 value is calculated for the read data of a specified address region to check that the programmed values in the Flash memory are read correctly (128 KB maximum). This calculated CRC32 value is compared with the CRC32 value which was acquired at the program build. (This test is executed in IAR Embedded Workbench only.)
CRC calculation test	The CRC32 value of a specified memory region is calculated by the CRC calculation hardware (CRC-B).

2.1. Self-diagnosis Sample Project

In the “Project\Examples\Safety” folder, the following sample project for the self-diagnosis library is placed.

Project name	Operation	Utilized self-diagnosis library function
MEMORY_Sample	RAM test and Flash memory test are executed. The result is shown with the LED lighting.	safety_RAM() safety_FLASH()

3. Details of Self-diagnosis Test Library

This section describes the details of the self-diagnosis test library function.
This sample program is a self-diagnosis program for the memories.
The following setting is an example when a product in the TMPM4K group (1) is used.

The source code (.c file) of the self-diagnosis library is in the “Libraries\Safety\src” folder, and the header file (.h file) is in the “Libraries\Safety\inc” folder.

3.1. RAM Test

The checker patterns, 0x55555555 and 0xAAAAAAAA, are written to an address in the RAM, and it is checked that the read data of the address is correct.
This test is done to each address in a specified memory region (18 KB maximum).

When an error is detected, the address of the error is written to the result memory buffer.

After the test, the value of each memory address returns the value before the test, but the value when the test fails is undefined.

If the tested memory area includes a region modified by an interrupt process (a stack region or a region of the variables modified by the interrupt process), unexpected modification may occur.

In such a case, it is recommended that the test region be divided into a non-modified region and a modified one. The modified region should be tested with disabling the interrupt.

The “safety_RAM” function does not use any interrupts.

Source file: safety_ram.c
Header file: safety_ram.h
Used library: txz_gpio.c/.h

Function name	
bool safety_RAM(uint32_t start, uint32_t length, uint32_t *result_word)	
Input parameter	
uint32_t start	Test start RAM address. Rounded to 4-multiple number.
uint32_t length	Test region (Byte unit). Rounded to 4-multiple number.
Output parameter	
uint32_t *result_word	Memory buffer which stores the result. The address where an error is detected is written. (When no errors are detected, no writes are done.)
Return value	
bool	Test result (true: success, false: failure)

* If the return value cannot be confirmed for several seconds, it is supposed the test is not executed correctly. The process for the test failure should be done.
When, however, the terminal I/O output display is used, it takes several seconds to complete the display. The judgment of the test failure should be done by checking the display.

3.2. Flash Memory Test

The CRC32 value is calculated for the read data of a specified address region to check that the programmed values in the Flash memory are read correctly.

This function does not check the test region, but any region in the Flash memory can be specified.
The region should be specified correctly in the test program.

This library function does not use any interrupts.

Source file: safety_flash.c
Header file: safety_flash.h
Used library: txz_gpio.c/.h

Function name	
bool safety_FLASH(uint32_t start, uint32_t length, uint32_t crc32_value)	
Input parameter	
uint32_t start	Test start Flash memory address. The value can be set in units of 1 Byte.
uint32_t length	Test region (Byte) The value can be set in units of 1 Byte.
uint32_t crc32_value	The CRC32 value for the test region which has been calculated.
Output parameter	
None.	-
Return value	
bool	Test result (true: success, false: failure)

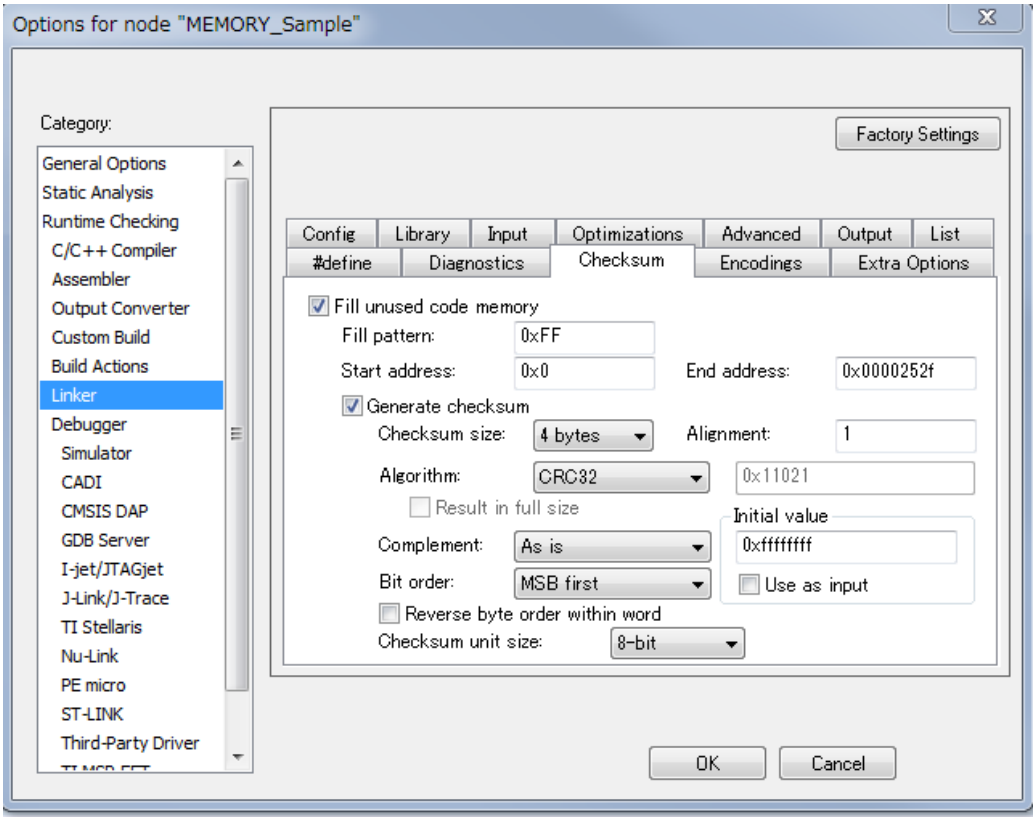
* If the return value cannot be confirmed for several seconds, it is supposed the test is not executed correctly. The process for the test failure should be done.
When, however, the terminal I/O output display is used, it takes several seconds to complete the display. The judgment of the test failure should be done by checking the display.

<In the case of IAR Embedded Workbench>

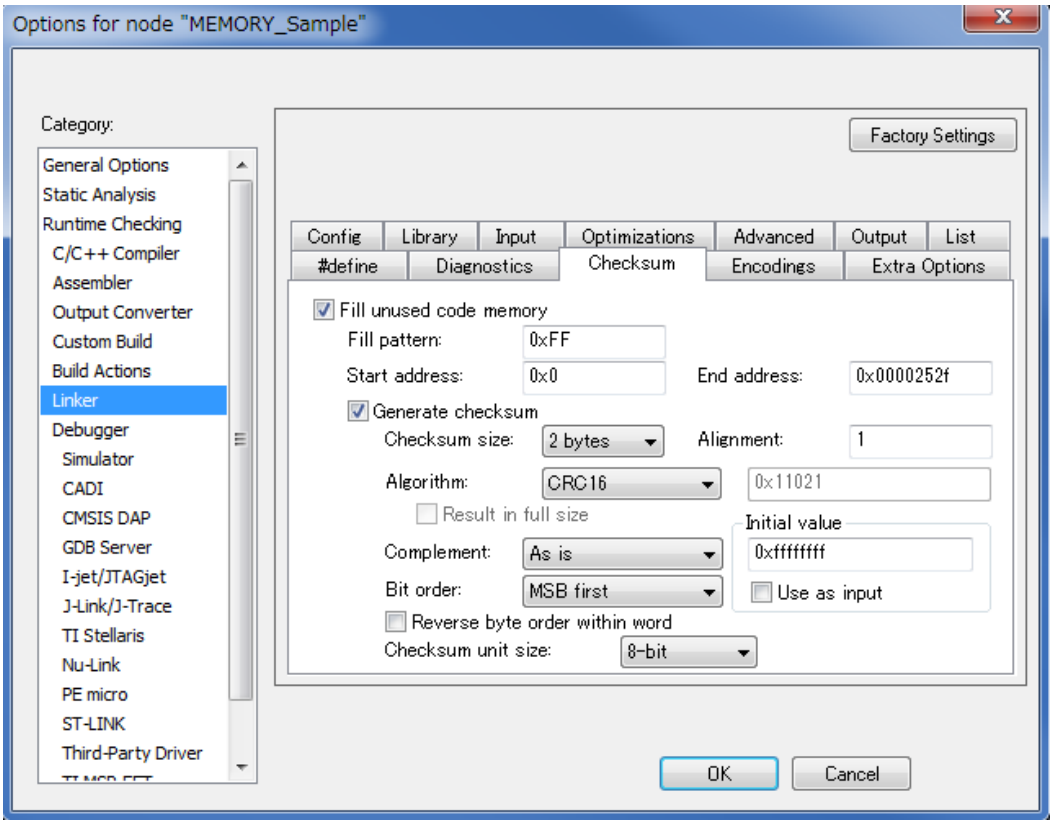
In order to use this function, the CRC32 value for a specified region of the Flash memory should be calculated in advance. The linker option of the IAR Embedded Workbench can be used for the calculation with the following setting.
(For the CRC16 value, the same setting is used.)

(As an example, the MEMORY_Sample project is used.)
1. Referring to “Modification of Linker Setting File for CRC32/CRC16 Calculation”, the linker setting file should be modified.
2. In the option setting of the target project, the setting in the following figure should be done in the “Checksum” tab in the “Linker” option.

In the case of CRC32:



In the case of CRC16:



Of these, "start address" and "end address" change for each project.
Generally, the start address is set to the first address of the Flash memory (0x0 for TMPM4K4A). And the end address is set to the last address of the Flash memory. It is, however, necessary to exclude the region which stores CRC32 and CRC16 values. So the addresses of the region should be known somehow.

3. The end address should be set to an arbitrary one (0x1000, and others), and the build should be done.
4. The address of the "__checksum" variable should be checked in the MAP file.

The section of "Modification of Linker Setting File for CRC32/CRC16 Calculation" describes how to check.
MEMORY_Sample.map (example of CRC32 (extract))

```
.rodata              const      0x000024a8     0x0  zero_init3.o [5]
.rodata              const      0x000024a8     0x0  packbits_init_single.o [5]
Initializer bytes    const      0x000024a8    0x85  <for P2-1>
CHECKSUM                        0x00002530     0x4  <Block>
  .checksum          const      0x00002530     0x4  Place holder __checksum
                                - 0x00002534  0x2390
```

The above example shows that "__checksum" variable which stores the checksum result of the Linker calculation is assigned to 0x2530. ".icf" setting of the linker is modified so that "__checksum" is placed to the last address of ReadOnly memory in the Flash memory in 1. step.
So the end address of the checked region should be the previous address of 0x2530, that is, 0x252F in the Flash memory.

5. The "checksum" tab in the Linker setting should be opened again. "End address" should be set to 0x252F, and the build should be done again. Then, the region of the checksum calculation and its value are shown as the message at the build.

6. Now the preparation has been done. Using the following variables in the program, the Flash memory region and the values of the CRC32 and the CRC16 can be utilized.

```
// use IAR Linker defined symbol "__checksum"
extern uint32_t __checksum;

// use IAR Linker defined symbol "__checksum_begin", "__checksum_end"
// following symbols are defined as assembler LABEL
extern uint8_t __checksum_begin;          // FLASH start
extern uint8_t __checksum_end;            // FLASH end except __checksum

// in IAR we can use Linker Symbol so use it
#define FLASH_TEST_START            (uint32_t)(&__checksum_begin)
#define FLASH_TEST_SIZE             (&__checksum_end - &__checksum_begin + 1)
#define FLASH_TEST_CRC32VALUE     (__checksum)

(snip)

      result = safety_FLASH(FLASH_TEST_START, FLASH_TEST_SIZE,
FLASH_TEST_CRC32VALUE);
```

In order to the DEBUGMSG is enabled in this MEMORY_Sample, the library implementation is set to “semi-hosting” for “general options/library configurations and library low level interface implementation”, set “stdout/stderr” to “via semi-hosting”, and the NDEBUG should be removed in “C/C++ compile/Preprocessor Defined symbols”. Then the build should be done. The terminal I/O displays as follows.

```
RAM (Non Stack): SUCCESS.
RAM (Stack): SUCCESS.
__checksum_begin=0x00000000, __checksum_end=0x0000252f, __checksum=0x3037947f
get_CRC32: start=0x00000000, length=0x00002530
safety_FLASH: value=0x3037947f, provided=0x3037947f
FLASH: SUCCESS. CRC=0x3037947f
Finish!.
```

3.3. Setting for CRC Calculation Test

The Flash memory test uses the CRC calculation function (CRC-B) to calculate the CRC32 and the CRC16 values of a specified memory region. Referring to this program code, the CRC calculation can be done using the CRC-B.
(Refer to “Libraries\Safety\src\safety_flash.c”.)

The following modification is necessary to operate the CRC32 calculation or the CRC16 calculation.

From Libraries\Safety\inc\ safety_flash.h;

<In the case of CRC32>

```
#define CRC32        0
#define CRC16        1
#define CRC          CRC32
```

<In the case of CRC16>

```
#define CRC32        0
#define CRC16        1
#define CRC          CRC16
```

4. Modification of Linker Setting File for CRC32/CRC16 Calculation

The MEMORY_Sample project uses the CRC32-B circuit to calculate the CRC32 and CRC16 values of a Flash memory address region. The calculated value is compared with the CRC32 and CRC 16 values in the binary file which is generated by the compiler to check if the values are identical or not, respectively. The CRC values of the Flash memory region should be calculated in advance using the linker function.
In order to do that, the common linker setting file TMPM4K4FYAUG.icf (Note1) for TMPM4K4A should be modified as follows.
TMPM4K4FYAUG.icf is the linker setting file which only the IAR Embedded Workbench can be used.
The linker setting file cannot be used in Arm® Keil®.

<In the case of IAR Embedded Workbench>

Description before modification

```
place in ROM_region        { readonly };
```

↓

Description after modification

```
define block CHECKSUM with alignment = 4 { ro section .checksum };
place in ROM_region { ro, last block CHECKSUM };
```

A modified linker setting file, TMPM4K4FYAUG_checksum.icf, is prepared in the Libraries\Safety folder.

This modification places the __checksum variable which stores the CRC32 value of the memory region set automatically by the linker to the end of the Readonly data. (When the linker option to generate the checksum is not set and __checksum variable is not used, this modification of the linker setting file affects nothing.)

The location of the __checksum variable can be checked using .map file which is generated at the build of the project.
In the case of MEMORY_Sample, .map file is generated in
Project\Examples\Safety\MEMORY_Sample\Debug\List.

Example of MEMORY_Sample.map (extract)

```
.text                ro code  0x00002450    0x38  packbits_init_single.o [5]
.iar.init_table      const    0x00002488    0x20  - Linker created -
.rodata              const    0x000024a8     0x0  zero_init3.o [5]
.rodata              const    0x000024a8     0x0  packbits_init_single.o [5]
Initializer bytes    const    0x000024a8    0x85  <for P2-1>
CHECKSUM                      0x00002530     0x4  <Block>
  .checksum          const    0x00002530     0x4  Place holder __checksum
                            - 0x00002534  0x2390
```

In the above example, the address 0x2530 is assigned to the __checksum variable.

Note1:Stored in C:\Program Files (x86)\IAR Systems\Embedded Workbench 8.0\arm\config\linker\Toshiba folder.

5. List of Used Drivers

This test library uses the driver and the code in the project of the TMPM4KxA_v1.0.0 version.

CMSIS library

Category	Source file name
Start-up	startup_TMPM4K4A.s
System (Clock setting and others)	system_TMPM4KxA.c

Periph_driver

Category	Source file name
GPIO	txz_gpio.c

In the Project examples

Category	Source file name
BSP (Evaluation board support)	bsp.c
LED output	bsp_led.c

6. Reference Document

For development, refer to the following documents.

- Datasheet of each product
- Reference Manual
- Self-diagnosis Program Application Note - Basic Setting
- ARM® Cortex®-M4 Processor technical Reference Manual
- ARMv7-M Architecture Reference Manual

7. Revision History

Revision	Date	Description
1.0	2019-08-27	First release

RESTRICTIONS ON PRODUCT USE

Toshiba Corporation and its subsidiaries and affiliates are collectively referred to as “TOSHIBA”.
Hardware, software and systems described in this document are collectively referred to as “Product”.

- TOSHIBA reserves the right to make changes to the information in this document and related Product without notice.
- This document and any information herein may not be reproduced without prior written permission from TOSHIBA. Even with TOSHIBA's written permission, reproduction is permissible only if reproduction is without alteration/omission.
- Though TOSHIBA works continually to improve Product's quality and reliability, Product can malfunction or fail. Customers are responsible for complying with safety standards and for providing adequate designs and safeguards for their hardware, software and systems which minimize risk and avoid situations in which a malfunction or failure of Product could cause loss of human life, bodily injury or damage to property, including data loss or corruption. Before customers use the Product, create designs including the Product, or incorporate the Product into their own applications, customers must also refer to and comply with (a) the latest versions of all relevant TOSHIBA information, including without limitation, this document, the specifications, the data sheets and application notes for Product and the precautions and conditions set forth in the "TOSHIBA Semiconductor Reliability Handbook" and (b) the instructions for the application with which the Product will be used with or for. Customers are solely responsible for all aspects of their own product design or applications, including but not limited to (a) determining the appropriateness of the use of this Product in such design or applications; (b) evaluating and determining the applicability of any information contained in this document, or in charts, diagrams, programs, algorithms, sample application circuits, or any other referenced documents; and (c) validating all operating parameters for such designs and applications. **TOSHIBA ASSUMES NO LIABILITY FOR CUSTOMERS' PRODUCT DESIGN OR APPLICATIONS.**
- **PRODUCT IS NEITHER INTENDED NOR WARRANTED FOR USE IN EQUIPMENTS OR SYSTEMS THAT REQUIRE EXTRAORDINARILY HIGH LEVELS OF QUALITY AND/OR RELIABILITY, AND/OR A MALFUNCTION OR FAILURE OF WHICH MAY CAUSE LOSS OF HUMAN LIFE, BODILY INJURY, SERIOUS PROPERTY DAMAGE AND/OR SERIOUS PUBLIC IMPACT ("UNINTENDED USE").** Except for specific applications as expressly stated in this document, Unintended Use includes, without limitation, equipment used in nuclear facilities, equipment used in the aerospace industry, lifesaving and/or life supporting medical equipment, equipment used for automobiles, trains, ships and other transportation, traffic signaling equipment, equipment used to control combustions or explosions, safety devices, elevators and escalators, and devices related to power plant. **IF YOU USE PRODUCT FOR UNINTENDED USE, TOSHIBA ASSUMES NO LIABILITY FOR PRODUCT.** For details, please contact your TOSHIBA sales representative or contact us via our website.
- Do not disassemble, analyze, reverse-engineer, alter, modify, translate or copy Product, whether in whole or in part.
- Product shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable laws or regulations.
- The information contained herein is presented only as guidance for Product use. No responsibility is assumed by TOSHIBA for any infringement of patents or any other intellectual property rights of third parties that may result from the use of Product. No license to any intellectual property right is granted by this document, whether express or implied, by estoppel or otherwise.
- **ABSENT A WRITTEN SIGNED AGREEMENT, EXCEPT AS PROVIDED IN THE RELEVANT TERMS AND CONDITIONS OF SALE FOR PRODUCT, AND TO THE MAXIMUM EXTENT ALLOWABLE BY LAW, TOSHIBA (1) ASSUMES NO LIABILITY WHATSOEVER, INCLUDING WITHOUT LIMITATION, INDIRECT, CONSEQUENTIAL, SPECIAL, OR INCIDENTAL DAMAGES OR LOSS, INCLUDING WITHOUT LIMITATION, LOSS OF PROFITS, LOSS OF OPPORTUNITIES, BUSINESS INTERRUPTION AND LOSS OF DATA, AND (2) DISCLAIMS ANY AND ALL EXPRESS OR IMPLIED WARRANTIES AND CONDITIONS RELATED TO SALE, USE OF PRODUCT, OR INFORMATION, INCLUDING WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY OF INFORMATION, OR NONINFRINGEMENT.**
- Do not use or otherwise make available Product or related software or technology for any military purposes, including without limitation, for the design, development, use, stockpiling or manufacturing of nuclear, chemical, or biological weapons or missile technology products (mass destruction weapons). Product and related software and technology may be controlled under the applicable export laws and regulations including, without limitation, the Japanese Foreign Exchange and Foreign Trade Law and the U.S. Export Administration Regulations. Export and re-export of Product or related software or technology are strictly prohibited except in compliance with all applicable export laws and regulations.
- Please contact your TOSHIBA sales representative for details as to environmental matters such as the RoHS compatibility of Product. Please use Product in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. **TOSHIBA ASSUMES NO LIABILITY FOR DAMAGES OR LOSSES OCCURRING AS A RESULT OF NONCOMPLIANCE WITH APPLICABLE LAWS AND REGULATIONS.**