SafeTI™ Hercules™ Diagnostic Library

Release Notes

Version: 2.3.1

Date: 26 October 2016

TABLE OF CONTENTS

1 New in this release

The following outlines the defects resolved and enhancements done in this release

- Maintenance update (See section 4 for bugs fixed)
- Added new diagnostic tests – see the user's guide for more details.
    - Diagnostic tests added for TMS570LS31x/21x, RM48x, TMS570LS12x/11x, RM46x, TMS570LS04x/03x, RM42x
        - ADC_SRAM_PARITY_TEST
        - ADC_SELFTEST_ALL - remapped existing ADC selftest to a test type.
        - L2INTERCONNECT_FAULT_INJECT changed to L2INTERCONNECT_RESERVED_ACCESS.
        - L3INTERCONNECT_FAULT_INJECT changed to L3INTERCONNECT_RESERVED_ACCESS.
        - L2INTERCONNECT_UNPRIVELEGED_ACCESS
        - L3INTERCONNECT_UNPRIVELEGED_ACCESS
        - VIM_SRAM_PARITY_TEST
        - VIM_SOFTWARE_TEST
        - DMA_SRAM_PARITY_TEST (not for RM42/ TMS570LS04x/03x)
        - DMA_SOFTWARE_TEST (not for RM42/ TMS570LS04x/03x)
        - HET_SRAM_PARITY_TEST
        - HET_ANALOG_LOOPBACK_TEST
        - HTU_SRAM_PARITY_TEST
        - MIBSPI_SRAM_PARITY_TEST
        - MIBSPI_ANALOG_LOOPBACK_TEST
        - SPI_ANALOG_LOOPBACK_TEST
        - CAN_SRAM_PARITY_TEST
        - CAN_ANALOG_LOOPBACK_TEST
        - GIO_ANALOG_LOOPBACK_TEST
        - SCI_ANALOG_LOOPBACK_TEST
        - LIN_ANALOG_LOOPBACK_TEST
    - Diagnostic tests added for TMS570LC43x, RM57Lx:
        - SRAM_RADECODE_DIAGNOSTICS
        - DMA_SOFTWARE_TEST
        - MIBSPI_ANALOG_LOOPBACK_TEST
        - PSCON_SELF_TEST
        - PSCON_ERROR_FORCING
        - PSCON_ERROR_FORCING_FAULT_INJECT
        - PSCON_SELF_TEST_ERROR_FORCING
        - PSCON_SELF_TEST_ERROR_FORCING_FAULT_INJECT
        - PSCON_PMA_TEST

- MEMINTRCNT_RESERVED_ACCESS
- MEMINTRCNT_SELFTEST
- MAINPERIPHINTRCNT_RESERVED_ACCESS
- PERIPHSEGINTRCNT_RESERVED_ACCESS
- PERIPHSEGINTRCNT_UNPRIVELEGED_ACCESS
- ADC_SELFTEST_ALL
- ADC_SRAM_PARITY_TEST
- VIM_SOFTWARE_TEST

- Updated API mapping table in the Safety Software Manual for SafeTI™ Diagnostic Library.
- Enabled handling of ESM event for PLL Slip detection.

2 Folder structure

The installer for this software installs by default to the C:\ti\Hercules folder. The folder structure is as shown below:

```
C:\ti\Hercules\SafeTI Diagnostic Library\  Installation Root folder
|
2.3.0                                     Product version number
    |
    +---build\                            Project files for demo
    |                                     application (Device specific)
    |
    +---build_safeTILib                   Project file for building library
    +---build_TPSDriverLib                Project file for building TPS Driver library
    +---demo_app\                         Demo application
    |   |
    |   +---common\                       Source code
    |   \---HALCoGen\                     HALCoGen configuration
    |                                     (Device specific)
    |
    +---docs                              Documentation.
    +---hal                               Device hardware abstraction
    +---libs                              Prebuilt libraries
    +---safety_library                    Source code for Diagnostic
    |                                     Library
    \---TPS_driver                        Source code for the TPS65381 Driver
```

3 Fixed in this release

SDOCM00120752	IAR: Add the variable ESM_AppCallback into the .noinit section.

Issue details:

The global variable "ESM_AppCallback" is initialized by the API SL_ESM_Init. If the user performs this initialization before calling main, the compiler generated code to initialize global variables resets this variable, unless it is in the .noinit section.

SDOCM00120751	IAR project .noinit section usage to be corrected.

Issue details:

```
#pragma location=".noinit"
volatile uint64 crcAtInit_VIMRAM = 0;
#pragma location=".noinit"
volatile uint64 crcAtInit_FLASH = 0;
#pragma location=".noinit"
volatile uint64 crcAtInit_FEE = 0;
#pragma location=".noinit"
volatile uint64 crcAtInit_StaticRAM = 0;
#pragma location=".noinit"
uint8 static_ram[8] = {4,2,3,4,5,5,7,9};
```

The initialization with 0 (= 0) will generate a warning and will cause that the variable did not can hold the CRC until the main().
Removing the "= 0" will fix this behavior.

SDOCM00120638	The imported "SL_RM57L843_NoOs" project does not build - multiple syntax erorrs and warnings

Issue details:

After importing the demo app project, a build fails with two errors:

1 File connections/SD560V2USB_Connection.xml does not exist. Was included by file C:/Users/a0321811/workspace_lwip_patch/SL_RM57L843_NoOS/targetConfigs/RM57L8xx.ccxml RM57L8xx.ccxml /SL_RM57L843_NoOS/targetConfigs

2 identifier "scilinREG" is undefined HL_sci.c /SL_RM57L843_NoOS/demo_app/HALCoGen/RM57L843_NoOS/source line 123 C/C++ Problem

SDOCM00120637	Safety Library Demo Build Instructions in .CHM file are incorrect

Issue details:

Multiple incorrect statements in the Demo application of the .chm file.

SDOCM00120246	Implementation of _SL_Kickoff_STC_execution in Safety Library Assumes WFI always enters standby.

Issue details:

The code for _SL_Kickoff_STC_execution assumes that the WFI always causes the standby state.
This isn't correct. The ARM Architecture manual explains that WFI is a hint - the instruction can be retired without
the CPU ever entering standby. It takes some cycles to enter standby due to the need to do things like flush the write
buffer, and if during this time an interrupt or debug request occurs - the CPU won't enter standby.

SDOCM00118840	The mapping between the API and the Safety Manual identifiers isn't up to date

Issue details:

The mapping between the API and the Safety Manual identifiers isn't up to date, some identifies changed the meaning with the update of the SM and some new were added in the SM.

SDOCM00100369	Flash address parity self test

Issue details:

1. Fix to implementation of SL_SelfTest_Flash for FLASH_ADDRESS_PARITY_SELF_TEST – Set a flag to indicate that an address parity self-test is being triggered *before* setting of the F021F_FPAROVR_ADD_INV_PAR bit in FPAROVR register.

2. Supporting fix to esmGroup2Handler – Clear the F021F_FPAROVR_ADD_INV_PAR bit if this flag is set.

SDOCM00097608	ADC self-test API returns UNDETERMINED.

Issue details:

ADC self-test API returns UNDETERMINED for the following cases:

1. WhenADCchannelisinuse/connected
2. WhenADCchannelisnotconnected

For other conditions (Short to ADRefHi & ADRefLo) the API works as expected.

SDOCM00105436	Flash test FLASH_ECC_ADDR_TAG_REG_MODE failing and is leading to nested aborts in RM42

Issue details:

Flash test FLASH_ECC_ADDR_TAG_REG_MODE failing and is leading to nested aborts in RM42.
In case of the RM48 and TMS570LS3x devices on testing the FLASH_ECC_ADDR_TAG_REG_MODE with a reduced frequency of 80MHz we observe the same behavior.

SDOCM00121322	SafeTI example

Issue details:

The examples for SafeTI library do not follow the required device initialization procedure which is that after reset, but before running any code, the application must initialize the core registers.

SDOCM00121324	**Confirm that SafeTI Library intialization code performs PBIST on PBIST ROM and STC ROM before other PBIST/LBIST tests.**

Issue details:

Two steps are missing from the device initialization procedure probably because they are not documented in the TRM but only in the data manual.

SDOCM00122551	**File sl_misc.c header include string.h isn't needed**

Issue details:

Please remove the #include <string.h> from sl_misc.c as it is not needed.

SDOCM00122369	**_SL_SelfTest_adcGetSingleData parameter check for data == NULL seems to be incorrect**

Issue details:

The following line in _SL_SelfTest_adcGetSingleData checks if data == NULL and accepts this:

if ((adc != sl_adcREG1) && (adc != sl_adcREG2) && (data != NULL)) {

I think it should be rewritten to:

if (((adc != sl_adcREG1) && (adc != sl_adcREG2)) || (data == NULL)) {

This will check if the parameter adc == ADC1 or ADC2 and that data != NULL.

SDOCM00122368	**sl_priv.c serverl parameters (adc) checked for NULL**

Issue details:

The following three functions in sl_priv.c are checking the parameter adc to be NULL

_SL_SelfTest_adcGetSingleData
_SL_SelfTest_adcStartConversion_selChn
_SL_SelfTest_adcIsConversionComplete

Code lines are:

if((adc != sl_adcREG1) && (adc != sl_adcREG2) && (adc != NULL) && (data != NULL)) {
if((adc != sl_adcREG1) && (adc != sl_adcREG2) && (adc != NULL)) {
if((adc != sl_adcREG1) && (adc != sl_adcREG2) && (adc != NULL)) {

adc == NULL doesn't seem to be a valid value.

Furthermore, all these checks are unconditional in terms of FUNCTION_PARAM_CHECK_ENABLED, not sure if this is intended.

SDOCM00122367	**SL_SelfTest_ADC FUNCTION_PARAM_CHECK_ENABLED for**

	(config->adcbase != NULL) seems to be incorrect

Issue details:

Correct line is:

```
   if( ((config->adcbase != sl_adcREG1) && (config->adcbase != sl_adcREG2)) || (config->adcbase == NULL)) {
```

or even simpler:

```
   if( (config->adcbase != sl_adcREG1) && (config->adcbase != sl_adcREG2) ) {
```

SDOCM00122365	**Bug in SL_adcCalibration?**

Issue details:

The function SL_adcCalibration contains the line:

```
   /** - Enable 12-BIT ADC  */
   sl_adcREG1->OPMODECR |= 0x80000000U;
```

This hardcoded register pointer seems to be wrong.
I guess this line should be changed to:

```
   /** - Enable 12-BIT ADC  */
   adc->OPMODECR |= 0x80000000U;
```

SDOCM00122364	**Second parameter (offset_error) of function SL_adcCalibration should be checked to be not NULL**

Issue details:

The following line:

```
if((adc != sl_adcREG1) && (adc != sl_adcREG2) && (adc != NULL)) {
```

Should be rewritten to:

```
if( ((adc != sl_adcREG1) && (adc != sl_adcREG2)) || (offset_error == NULL) ) {
```

In order to don't accept parameters adc == NULL (SDOCM00122362) and offset == NULL.

It would be even better to check if the parameter offset falls into the SRAM or not (CHECK_RANGE_RAM_PTR).

SDOCM00122363	**Doxygen comment for adcCalibration is incoorect**

Issue details:

Please compare comment with declaration, one parameter vs. two parameters

```
/** @fn void adcCalibration(sl_adcBASE_t *adc)
*   @brief Computes offset error using Calibration mode
*   @param[in] adc Pointer to ADC module:
*              - sl_adcREG1: ADC1 module pointer
*              - sl_adcREG2: ADC2 module pointer
*   This function computes offset error using Calibration mode
*
*/
/*SAFETYMCUSW 7 C MR: 14.7 <APPROVED> Comment_3*/
boolean SL_adcCalibration(sl_adcBASE_t * adc, uint32 * offset_error)
```

SDOCM00121814	**Wrong return value in SL_FLAG_SET**

Issue details:

The function SL_FLAG_SET() in the file sl_priv.c has the following return statement:

return sl_priv_flag_set[flag_id]; //code

which is wrong since the array index is erronous and can go out of bounds. The correct return statement should be:

return sl_priv_flag_set[flag_id-TESTTYPE_MIN]; //code

SDOCM00122281	**L2INTERCONNECT_FAULT_INJECT doesn't work when compiled with optimizations on**

Issue details:

The following line gets removed by the compiler if optimizations are turned on:

read_reserved_word = *((uint32*)SCR_RESERVED_LOCATION);

Please change to volatile to make this working:

read_reserved_word = *((volatile uint32*)SCR_RESERVED_LOCATION);

SDOCM00122253	**warning #64-D: shift count is too large**

Issue details:

Four times warning "#64-D: shift count is too large" is displayed during compilation of sl_selftest.c.

The reason is the following macro:

#define BIT(n) ((uint32)((uint32)1u <<(n)))

The variable n is 32 in all four cases, it is clear that a 32-bit variable can't hold 1 << 32 and therefore this warning is displayed.

Changing the macro as following should be safe to use and help to avoid these warnings:

#define BIT(n) ((uint32)((uint64)1u <<(n)))

SDOCM00122252	**Can't compile sl_esm.c with IAR EWARM 7.60.2**

Issue details:

The compilation of the file will fail with the following message:

Error[Pe513]: a value of type "void (__arm __irq *)(void)" cannot be assigned to an entity of type "sl_t_isrFuncPTR" C:\ti\Hercules\SafeTI Diagnostic Library\2.2.0\safety_library\source\sl_esm.c 66

The type sl_t_isrFuncPTR is defined as:

typedef void (__arm __fiq *sl_t_isrFuncPTR)(void);

So the definition for an IRQ and a FIQ aren't compatible and the compilation will fail.

Tested with two projects supplied with the library package:
RM48L950_NoOS_IAR
build_safeTILib

SDOCM00122282	**L3INTERCONNECT_FAULT_INJECT doesn't work when compiled with optimizations on**

Issue details:

The following line gets removed by the compiler if optimizations are turned on:

read_reserved_word = *((uint32*)PCR_RESERVED_LOCATION);

Please change to volatile to make this working:

read_reserved_word = *((volatile uint32*)PCR_RESERVED_LOCATION);

SDOCM00122354	**SL_SelfTest_SRAM(SRAM_PAR_ADDR_CTRL_SELF_TEST, ...)**

	calls ESM_ApplicationCallback

Issue details:

If the Self-Test function SRAM_PAR_ADDR_CTRL_SELF_TEST gets executed the application level callback ESM_ApplicationCallback gets executed to. This shouldn't be the case, as this is a self-test function ant not a fault injection.

The function esmGroup2Handler in sl_esm.c contains no code (flag) to detect a running SRAM_PAR_ADDR_CTRL_SELF_TEST:

```
    case ESM_G2ERR_B0TCM_ADDPAR:
      callbkParam1 = sl_tcram1REG->RAMPERADDR;
      /*SAFETYMCUSW 134 S MR: 12.2 <APPROVED> Comment_4*/
      callbkParam2 = (sl_tcram1REG->RAMERRSTATUS
                  & (TCRAM_RAMERRSTAT_WADDRPAR_FAIL |
TCRAM_RAMERRSTAT_RADDRPAR_FAIL));
      callbkParam3 = (uint32) 0u;
      /* Clear the WADDR/RADDR PAR Fail bits in TCM Regs */
      break;
```

There is also no flag set in SL_SelfTest_SRAM(SRAM_PAR_ADDR_CTRL_SELF_TEST).

How to distinguish between self-test and real fault?

SDOCM00122362	**SL_adcCalibration FUNCTION_PARAM_CHECK_ENABLED for adc == NULL seems to be broken**

Issue details:

The following code line for parameter checking accepts the parameter adc to be NULL, which seems to be incorrect

```
  if((adc != sl_adcREG1) && (adc != sl_adcREG2) && (adc != NULL)) {
```

I guess the API should only accept the parameter adc to be either sl_adcREG1 or sl_adcREG2, thus the line should be changed to:

```
if( (adc != sl_adcREG1) && (adc != sl_adcREG2) ) {
```

This will reject any incorrect pointer.

SDOCM00122731	**Unit tests involving switch to user mode fail when optimisation turned on**

Issue details:

Entry Condition unit test involving switch to user mode fail when run with optimisation (-O3). The return address (value in LR) on returning from data abort is incorrect when optimisation is turned on. The following assembly code in the function _excpt_vec_abort_data() needs to change in order to load correct value in LR:

```
__asm(" LDR R0, [SP, #100]");
__asm(" ADD R0, R0, #8");
__asm(" STR R0, [SP, #100]");
```

SDOCM00122730	**Memory Interconnect Tests fail when running with optimisation**

Issue details:

MEMINTRCNT_RESERVED_ACCESS test fails when running with optimisation (-O3). The return address (value in LR) on returning from data abort is incorrect when optimisation is turned on. The following assembly code in the function _excpt_vec_abort_data() needs to change in order to load correct value in LR:

```
__asm(" LDR R0, [SP, #100]");
__asm(" ADD R0, R0, #8");
__asm(" STR R0, [SP, #100]");
```

SDOCM00122733	**Running PBISTALGO_MARCH13N_2PORT algorithms on 2 port memories do not reach completion if optimisation enabled**

Issue details:

Running PBISTALGO_MARCH13N_2PORT algorithms on 2 port memories with optimisation enabled (-O3) ends up waiting infinitely in the loop:
while (TRUE != SL_SelfTest_Status_PBIST(&failInfoPBISTOthers));

This is because the required wait of 32 VBUS clock cycle when executing PBIST is optimised out:

```
/* Wait for 32 VBUS Clocks */
        for (tempVal=0u; tempVal<(VBUS_CLK_CYCLES + (VBUS_CLK_CYCLES * 1u));
tempVal++)
        {

        }
```

SDOCM00122548	**implementation for certain functionality is not setting/clearing flags**

Issue details:

Looking at code like the following from exception_handlers.it c seems like there are more checks needed to distinguish between a real fault and a self-test:

```
/*
 * DAbort due to access to illegal transcation to L2  Memory?
 * 0x00000008 indicates that it is an external abort caused by read and is AXI decode error
 * 0x88000000 is the reserved location accessed to create the L2 interconnect error trap AXI decode error
 */
        if ((0x00000008u == (0x00000008u & _SL_Get_DataFault_Status())) &&
                (0x88000000 == _SL_Get_DataFault_Address())) {
                maskDAbort = TRUE;
        }
```

1. there is no check for the self-test flag.

2. The function SL_SelfTestL2L3Interconnect is not setting the Self-Test Flag with SL_FLAG_SET.

How should the application know that a self-test is ongoing? I.e. in exception_handlers.c?

SDOCM00122547	**Exception_handlers.c has TODO which are not needed.**

Issue details:

There are some todo's in the file exception_handlers.c, these are not necessary and should be cleaned up.

SDOCM00122732	**DMA_ECC_TEST_MODE_1BIT fails with optimisation**

Issue details:

No error is reflected on reading the corrupt DMA ram location after deliberate injection of single bit ecc error as part of the diagnostic.

SDOCM00121725	**SRAM_PAR_ADDR_CTRL_SELF_TEST causes Application callback through ESM**

Issue details:

In app_main_NoOS.c the following test is executed:

retVal = SL_SelfTest_SRAM(SRAM_PAR_ADDR_CTRL_SELF_TEST, TRUE, &failInfoTCMRAM);

This test, even if it is a Self-Test, generates an interrupt served by the callback of the ESM module (in the file esm_application_callback.c) but actually it is not handled and considerated unknowncallback.
This shouldn't occur. Also need to identify any other self-tests which may have been missed similarly.

SDOCM00122735	**SRAM_ECC_ERROR_FORCING_2BIT test reults in nERROR on running with optimisation enabled**

Issue details:

SRAM_ECC_ERROR_FORCING_2BIT test reults in nERROR when run with -O3 optimisation. The program control within the SL_SelfTest_SRAM function never hits the instruction _SL_HoldNClear_nError() for clearing the nERROR, which is generated by read of the deliberately corrupted ram location. This is due to the variable "testType" getting overwritten and the program control never entering the following conditional code:

```
    if(SRAM_ECC_ERROR_FORCING_2BIT == testType){
        ram1uerraddr=sl_tcram1REG->RAMUERRADDR;
        ram2uerraddr=sl_tcram2REG->RAMUERRADDR;
        /*the esm interrupts for selftests which generate group 1 interrupts is blocked.Users will
have to rely on
        status functions to get the pass/failure information*/
        /* Check the error status on both banks */
        if ((((uint32)&sramEccTestBuff[2] & TCRAM_RAMUERRADDR_UNC_ERRADD) ==
ram1uerraddr)
            && (((uint32)&sramEccTestBuff[3] &
TCRAM_RAMUERRADDR_UNC_ERRADD) == (ram2uerraddr)) &&
            ((uint32)(1u << ESM_G3ERR_B1TCM_ECC_UNCORR) == (sl_esmREG->SR1[2]
& (uint32)(1u << ESM_G3ERR_B1TCM_ECC_UNCORR)))&&
            ((uint32)(1u << ESM_G3ERR_B0TCM_ECC_UNCORR) == (sl_esmREG->SR1[2]
& (uint32)(1u << ESM_G3ERR_B0TCM_ECC_UNCORR)))) {
            *sram_stResult = ST_PASS;
        } else {
            *sram_stResult = ST_FAIL;
        }
        /* Clear nError */
        _SL_HoldNClear_nError();

        /* Clear the ESM Status */
        sl_esmREG->SR1[2] = ((uint32)(1u << ESM_G3ERR_B0TCM_ECC_UNCORR) |
(uint32)(1u << ESM_G3ERR_B1TCM_ECC_UNCORR));
        /* Clear double bit error anyways */
        sl_tcram1REG->RAMERRSTATUS |= TCRAM_RAMERRSTATUS_DER;
        sl_tcram2REG->RAMERRSTATUS |= TCRAM_RAMERRSTATUS_DER;
        /* Compute uncorrupted ECC */
        sramEccTestBuff[2] = 0UL;
        sramEccTestBuff[3] = 0UL;
    }
```

Root cause: Reading the corrupt ram location results in data abort (which is the expected behaviour). But on returning from the _excpt_vec_abort_data() function, the variable "testType" gets overwritten. This is due to the following wrong assembly code used to return from _excpt_vec_abort_data() :

```
    __asm(" add SP, SP, #4 ");
    __asm(" ldmfd    r13!, {r0 - r4, r12, lr} ");
    __asm(" subs    pc, lr, #4 " );
```

4 Known Issues

SDOCM00102614	Version B implementation of the Stuck at zero test for efuse is not available in Diagnostic library
Issue details: The TRM specifies of two ways of doing the stuck at 0 test for efuse.(chapter 32.3.2.4). The version B is not implemented.	
Workaround None	

SDOCM00107044	TPS-Driver - Reset on enabling a ABIST and LBIST Run.
Issue details: Power reset is observed when enabling the ABIST/LBIST run by writing the LBIST_EN and ABIST_EN bits in the safety_bist_ctrl register of the TPS device.	
Workaround None	

SDOCM00112033	TPS_GetWatchdogFailureStatus giving failure status as TRUE when Watchdog fail count is 7 and Watchdog reset is not enabled
Issue details: The API TPS_GetWatchdogFailureStatus should return the failure status as true only when the watchdog failure count is 7 and the watchdog reset is enabled.But the failure status is returned as TRUE even when the watchdog failure count is 7 and watchdog reset is not enabled.	
Workaround None	

SDOCM00112116	AMUX diagnostics failing with Hitex kit
Issue details: The AMUX diagnostics tests are failing when tested on the HiTex Kit.The AMUX diagnostics pass in the RM46 and the TMS570LS1227 control card.	
Workaround None	

SDOCM00114805	MISRA-C violation detected incorrectly in TPS_Interface.c
Release Note Incorrect violations in TPS driver detected in LDRA	
Workaround None	

SDOCM00116435	**IAR doesn't support EXTERNAL_SP_INIT**

Release Note

The current version of the Safety Diagnostic Library doesnt support use of an external SP init function (The EXTERNAL_SP_INIT macro that disables the available API for SP initialization is absent, which is present for CCS)

Workaround

Application can use the SafeTI Diagnostic Library function for SP Init or define the globals required by the SafeTI Diagnostic Library but still use external API for Stack pointer init.

SDOCM00122662	**Safety Library doesn't document which APIs result in an interrupt**

Release Note

Some Safety Library API calls by design cause an ESM interrupt.

For example SL_SelfTest(SRAM) with some particular test types may generate an interrupt.

The customer needs to know when they must have installed the ESM interrupt handler and which functions may trigger it. But this isn't listed in the Safety Manual documentation.

See http://e2e.ti.com/support/microcontrollers/hercules/f/312/p/540238/1972316#1972316

Also BTW the documentation is missing the description for the GROUP2 ESM handler. It only lists group 1.
Could the documentation files be out of date w. respect to the source code from which they are generated?

Workaround

Application should setup the ESM handlers used by SafeTI diagnostic library using the SL_ESM_Init API.

SDOCM00122760	**FLASH_ADDRESS_PARITY_SELF_TEST fails on unit testing for RM42Lx platform**

Release **Note**

FLASH_ADDRESS_PARITY_SELF_TEST fails on unit testing for diagnosric functionality on RM42Lx platform

Workaround

None

SDOCM00122761	**DMA_SOFTWARE_TEST fails**

Release	Note
DMA_SOFTWARE_TEST conducts test of MPU on DMA access to memory locations with defined read/write access. The diagnostic fails, the likely cause being that the memory locations under test are not 64-bit aligned. A #pragma directive is required to ensure 64-bit alignement of the allocated memories.	

Workaround

None

SDOCM00122762	**SCI and LIN modules untested by unit testing**

Release	Note
SafeTI_Library product version 02.03.02 does not have unit testing implemented for SCI and LIN module. This is a pending work item to be implemented in subsequent product version.	

Workaround

None

SDOCM00122763	**Static Analysis gaps exist for RM57Lx platform**

Release	Note
The following mandatory standards are not met as per the LDRA static analysis report for RM57Lx platform: 47S - array bounds exceeded 50S - use of shift operators on signed types 61S - switch statement contains default only 96S - use of mixed mode arithmetic 120S - use of bit operators on signed type 434S - signed/unsigned conversion without cast 488S - value outside range of underlying types 24D - procedure definition has no associated prototype Resolving these is a pending work item to be resolved in subsequent product versions.	

Workaround

None

SDOCM00122766	**IAR projects for RM57x and TMS570LC43x throw many compilation errors**

Release Note
SafeTI Library projects for RM57x and TMS570LC43x throw many compilation errors. Likely to be an issue with linker command file settings and IAR project settings. Version :IAR Workbench 7.60.2
Workaround None

SDOCM00122767	**MEMORY_INTERCONNECT_SELFTEST fails for TMS570LC43 on unit testing**
Release Note MEMORY_INTERCONNECT_SELFTEST fails for TMS570LC43 on unit testing. Likely root cause unknown. Test settings: compiler - arm 5.2.6 optimisation - O3	
Workaround None	