SafeTI™ Hercules™ Diagnostic Library Release Notes

Contents

List of Tables

Trademarks

SafeTI, Hercules are trademarks of Texas Instruments.
Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
All other trademarks are the property of their respective owners.

1 New in This Release

- Added Support for TMS570LS07x/09x and RM44x Family of Devices
- Bug Fixes and Documentation Updates

2 Folder Structure

The installer, by default, installs to the C:\ti\Hercules folder. The folder structure is as follows:

```
C:\ti\Hercules\SafeTI Diagnostic Library\  Installation Root folder
|
2.4.0                                     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
    +---Test                              Test automation unit for testing the API.
    \---TPS_driver                        Source code for the TPS65381 Driver
```

3 Fixed in This Release

Table 1. Fixed in This Release

References	Headline	Description
SDOCM00122888	ADC Memory Init support is missing in SL_Init_Memory	ADC Memory Init support is missing in SL_Init_Memory. It returns FALSE when used "SL_Init_Memory(RAMTYPE_MIBADC1_RAM | RAMTYPE_MIBADC2_RAM))"
SDOCM00122935	Clarification of the SL_SelfTest_SRAM Description in SafeTIDiagnosticLibrary-User'sGuide-v2.3.1	The description for 'SL_SelfTest_SRAM ' in 'SafeTIDiagnosticLibrary-User'sGuide-v2.3.1.chm' contains: [in] param1 - Pointer to structure that is used to return the test status and results. **NOTE:** Contents of structure param1 is valid only for self-test. Not valid for fault injection modes. Issues are: • The parameter is called 'sram_stResult', not 'param1'. • It's an 'enum', not a structure. • What does 'only valid for self-test' mean? Can the note be enhanced to further clarify what this means and why it is not valid for fault injection? This would add more clarity.
SDOCM00122931	Using BIT_SET Macro for Status Flag Clear is not Adviced	Using BIT_SET Macro for Status Flag Clear is not Adviced. For example, BIT_SET(s|_tcram1REG → RAMERRSTATUS, TCRAM_RAMERRSTATUS_ADDR_SERR); BIT_SET is iplemented as "|=", which can clear the bits if they are already set.

Table 1. Fixed in This Release (continued)

References	Headline	Description
SDOCM00122928	SafeTI: Bug in SL_SelfTest_DMA() with DMA_SOFTWARE_TEST test	Test fails to first expected ok-return, based on register content, the Region 0 error is found: if (sl_dmaSoftwrTestConfig(&dma_test_varA,&dma_test_varB, DMA_PERMISSION_READ_ACCESS,&dmaCTRLPKT)) After some code inspection and careful reading of the device-specific TRM, it was found that the end-region is set like this sl_dmaREG->DMAMPR0E = (uint32)(srcAddr) + sizeof(uint32); This resulted the DMAPR0S to be ....240 and DMAPR0E be +4 so .....244. The variables A and B used in testing are stack variables and for me those addresses are after each other ..240 and ...244, respectively. After the end region setting was changed like this (-1), the test started to work sl_dmaREG->DMAMPR0E = (uint32)(srcAddr) + sizeof(uint32)-1U; Based on the device-specific TRM, the PR0E is end address of the region so the initial code is wrong and sets region 1 byte too far. Given the severity of this bug, it is imperative that you also check the test suite implementation to insure it can catch this error and that the expected results for passing is correct.

SDOCM00122867	SL_SelfTest_Status_CCMR4F Self Test Error Type (STET) is interchanged	The two error types are interchanged in the following code:
		`if (CCMR4F_CCMSR_STET == (uint32) (ccmr4fREG1->_CCMSR & CCMR4F_CCMSR_STET)) {`
		`failInfoccmr4f->failInfo = CCMR4F_ST_ERR_COMPARE_MATCH;`
		`} else {`
		`failInfoccmr4f->failInfo = CCMR4F_ST_ERR_COMPARE_MISMATCH;`
		`}`
		According to the device-specific TRM and design spec, a 1 means MISMATCH and a 0 means MATCH:
		0 = self test failed during Compare Match Test
		1 = self test failed during Compare Mismatch Test

SDOCM00122933	Handling of nERROR pin clearing in CCMR4F_SELF_TEST_ERROR_FORCING	In the SafeTI Diagnostic library function boolean, SL_SelfTest_CCMR4F (SL_SelfTestType testType, boolean bMode, SL_CCMR4F_FailInfo * config) test condition CCMR4F_SELF_TEST_ERROR_FORCING, the ESM error condition that is triggered is a group 1 error and, as such, has no dedicated nERROR or interrupt response unless configured as such. The test forces the error then sets the results appropriately, but after the test is completed, the software calls _SL_HoldNClear_nError(), which attempts to clear the nERROR signal that has not been set. Given, the CCMR4F is still functionally active during this test, there could be a real CCMR4F error condition or other error that would then be masked if it were to occur during the execution of the test or if an error should happen after the assertion of the nERROR reset bit. For more information, see the *ERROR Pin Timing - Example 5* figure in the *TMS570LS12x/11x 16/32-Bit RISC Flash Microcontroller Technical Reference Manual*. For explanation and advisory against such assertion of the nERROR reset prior to a real errror occurrence, see the *ERROR Pin Timing - Example 5* figure. This same issue is present in the test of CCMR5F. It is also worth noting that. This same issue is present in the test of CCMR5F. It is also worth noting that these are the only instances where a group 1 error is handled in this way.
SDOCM00122915	Please enhance the documentation for the following (destructive) peripheral tests	The following peripheral tests are destructive, (if they are disturbing the normal operation of the module they apply to in some way): • SL_SelfTest_DMA(DMA_SOFTWARE_TEST) • SL_SelfTest_DMA(DMA_SRAM_PARITY_TEST) • SL_SelfTest_VIM(VIM_SRAM_PARITY_TEST) • SL_SelfTest_VIM(VIM_SOFTWARE_TEST) • SL_SelfTest_HET(HET_SRAM_PARITY_TEST) • SL_SelfTest_HTU(HTU_SRAM_PARITY_TEST) All of the tests listed are disturbing the normal operation of the respective module. DMA tests will reconfigure a DMA channel to perform the test or the HET_SRAM_PARITY_TEST will introduce a fault in the RAM that will cause the HET to stop operation. Thus, normal operation will somehow be disturbed by these tests, even if the test performs a cleanup afterwards and the module can continue its normal operation.

Table 1. Fixed in This Release (continued)

References	Headline	Description

SDOCM00122912	SL_Init_ResetReason returns RESET_TYPE_DEBUG incase reset reason is unknown.	The final else branch in the function SL_Init_ResetReason causes it to return RESET_TYPE_DEBUG if the reset reason is unknown, not signaled by SysEsr.
		`else`
		`{`
		`/*default reset type*/`
		`retVal = RESET_TYPE_DEBUG;`
		`}`
		Some IDE's including CCS do set the PC to 0 to simulate a CPU reset, in this case the else branch shown above is executed.
		However, this simulated reset is quite different from a rel CPU reset and from a System Reset which equals the Debug (ICEPICK) reset. Therefore, the customer can't rely on the value returned by SL_Init_ResetReason.

SDOCM00122904	Error Pin check Entry condition check is not valid for the function SL_SelfTest_PBIST().	PBIST test / self Test does not affect Error pin in case of Failure so, Error Pin check Entry condition check is not valid for the function SL_SelfTest_PBIST().
		The following check must be removed..
		`/* If nERROR is active then do not proceed with tests that trigger nERROR */`
		`if((boolean)(TRUE) == SL_ESM_nERROR_Active()){`
		`SL_Log_Error(FUNC_ID_ST_PBIST, ERR_TYPE_ENTRY_CON, 2u);`
		`return(retVal);`

SDOCM00122845	Incorrect use of size of operator in Safety library demo code	The statements: • for(i = 0;i <(sizeof(all2portmemories)/sizeof(uint32));i++) • for(j =0;j<(size of(all2portalgos)/siz eof(uint32));j++) are incorrect. There is no need to divide by the size of (uint32).
SDOCM00122769	Wrong shift in Safety Library CRC example file	Line in the sys_startup.c files `crcAtInit_FLASH = SL_CRC_Calculate((uint64 *) (uint32)&ulFlashStartAddr), ((((uint32)&ulFlashEndAddr)- ((uint32)&ulFlashStartAddr)) >> 6));` should be “>>3” (divide by 8, not divide by 64)
SDOCM00122826	Documentation: For GIO, not all ports are available on all devices. Parameter range checks only validate for superset.	Document that the customer should ensure that the GIO port/instance is valid for the given device.
SDOCM00122825	Documentation: For DCAN, not all ports are available on all devices. Parameter range checks only validate for superset.	Document that the customer should ensure that the DCAN port/instance is valid for the given device.
SDOCM00122824	Documentation: For MibSPI, not all ports are available on all devices. Parameter range checks only validate for superset.	Document that the customer should ensure that the MibSPI port/instance is valid for the given device.
SDOCM00122802	Document the meanings of the fields of the structure _SL_STC_FailInfo	The meaning of the three detailed fault fields in this struct is counterintuitive; improve the documentation to make this more clear. Here is a short description of the meaning for all four fields: • In an STC pass case (stResult==ST_PASS), CPU1Failure, CPU2Failure, TimeOutFailure should all be ST_FAIL. • In an STC failure case (stResult==ST_FAIL), one or more of CPU1Failure, CPU2Failure, TimeOutFailure would indicate ST_PASS to indicate that as the reason of failure.
SDOCM00122842	Documentation for SL_SelfTest_PBIST, SL_SelfTest_PBIST_StopExec and SL_SelfTest_Status_PSCON is wrong	Documentation for SL_SelfTest_PBIST, SL_SelfTest_PBIST_StopExec and SL_SelfTest_Status_PSCON is wrong there is no function SL_SelfTest_PBIST_ExecStatus.

Table 1. Fixed in This Release (continued)

References	Headline	Description
SDOCM00122840	Description of the parameter flash_stResult in SL_SelfTest_Flash() is wrong	The description of the parameter flash_stResult says that this is a pointer to a structure and in the additional note it is stated, that member param1 is not valid for all tests. However, flash_stResult is a enum and not a structure and also has no member param1.
SDOCM00122819	Document that _SL_HoldNClear_nError unconditionally resets the nError pin	The function _SL_HoldNClear_nError unconditionally resets the nError pin in many self test functions. However, the application might rely on the nError pin staying activated once a real fault was detected. Even if it is unlikely, the current implementation might mask the nError pin and could cause issues on application level.
SDOCM00122812	Redefine SL_FLAG_SET/CLEAR/GET with argument type of SL_SelfTestType to avoid implicit type cast	Redefine SL_FLAG_SET/CLEAR/GET with argument type of SL_SelfTestType to avoid implicit type cast.
SDOCM00122808	Example ESM_ApplicationCallback many ESM SRx writes are faulty and clear all set bits instead of only one	The Status Registers (SRx) are clear on write (WPC), which means that the status bit gets cleared if a one in written to it. Many if not all of the status clear operations in ESM_ApplicationCallback are written with in a read modify write way with a logic or which causes all pending bits to be cleared instead of only one. This could lead to masking real faults and should be corrected in this example provided with the library.
SDOCM00122838	SL_SelfTestL2L3Interconnect documentation is incorrect	The description for SL_SelfTestL2L3Interconnect lists two test types: • L2L3INTERCONNECT_RESERVED_ACCESS • L2L3INTERCONNECT_UNPRIVELEGED_ACCESS However, these do not exist. Instead, there are four: • L3INTERCONNECT_RESERVED_ACCESS • L2INTERCONNECT_RESERVED_ACCESS • L3INTERCONNECT_UNPRIVELEGED_ACCESS • L2INTERCONNECT_UNPRIVELEGED_ACCESS
SDOCM00122829	DMA ECC Single bit test mode: EDACMODE assumed to be enabled	In the case of DMA ECC test for single bit failures, the SBERR bit indicates if a single bit error has occurred and was corrected by the ECC logic. The auto-correction is controlled by EDACMODE. In the implementation, the EDACMODE is not forced to be enabled, and if in the default application configuration is not enabled, the SBERR bit will not be set. Assumption should be documented in the device-specific user's guide.
SDOCM00122811	_SL_HoldNClear_nError unconditionally resets the nError pin	The function _SL_HoldNClear_nError unconditionally resets the nError pin in many self test functions. However, the application might rely on the nError pin staying activated once a real fault was detected. Even if it is unlikely, the current implementation might mask the nError pin and could cause issues on application level. Clear indication in the API documentation is required.
SDOCM00122791	(Child) Implementation of _SL_Kickoff_STC_execution in Safety Library Assumes WFI always enters standby.	Really all platforms. The code for _SL_Kickoff_STC_execution assumes that the WFI always causes the standby state. This is not 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 will not enter standby. If the CPU does not enter standby, then even though WFI is executed the code will resume execution after the WFI. LBIST will not run and the CPU will not be reset.
SDOCM00122789	SL_SelfTest_STC nERROR entry condition check doesn't set retval to FALSE	The nERROR entry condition check in SL_SelfTest_STC does not set retval to FALSE. This is no functional issue as the variable is initialized to FALSE at the beginning but it is inconsistent to the other condition checks which explicitly set the retval variable.
SDOCM00121630	Need #defines for PBIST RAM groups to make code readable. Otherwise it's a support problem.	For more information, see the forum post. Having a hard coded hex # instead of something humanly readable makes it difficult to understand what needs fixing when changing/adapting to a lower end MCU with subset of ESRAM.

4 Known Issues

Table 2. Summary of Known Issues

HERCULES_SW-5107 *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. For more information, see the *Stuck at Zero Test* chapter in the *TMS570LS12x/11x 16/32-Bit RISC Flash Microcontroller Technical Reference Manual*. The version B is not implemented.

Devices All device platforms.

Workaround This is a feature gap and not a bug. Customers can implement this using the description in the TRM.

HERCULES_SW-5113 *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.

Devices All device platforms

Workaround ABIST and LBIST must be run independently. Trying to run both ABIST and LBIST at the same time is not allowed.

HERCULES_SW-5126 *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.

Devices All device platforms

Workaround The TPS65381A-Q1 datasheet has been updated to provide clarity on the working of the watchdog. When the fail counter reaches 7, device will not cause a RESET. The counter on reaching 7+1(or the next bad event after it has reached 7) will put the device to RESET state.

HERCULES_SW-5195 *DMA_SOFTWARE_TEST fails*

Issue details 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 alignment of the allocated memories.

Devices All device platforms

Workaround Added a dummy variable in-between to get 64 bit alignment.

HERCULES_SW-6014 *demo_app in the SDL sw incorrectly uses sizeof unary operator in calling PBIST routine.*

Issue details The issue is in the demo_app included with the SDL SW.at line 1936 as shown below where all2portmemories is an array of 64bit values. The obvious intention is for the sizeof to return the number of elements in the array since each element is associated with a 2 port RAM, but size of returns the number of bytes in the array. This would then return a much larger index value than the array has elements.

```
/running PBISTALGO_MARCH13N_1PORT algorithms on 2 port memories/
for(i = 0;i < (sizeof(all2portmemories));i++)
{
/* only run March13N algorithm. */
{
retVal = SL_SelfTest_PBIST( PBIST_EXECUTE,
all2portmemories[i],
PBISTALGO_MARCH13N_2PORT);
```

Devices TMS570LC43x,TMS570LS04x,TMS570LS12x,TMS570LS31x,RM42x,RM46x,RM48x,RM57LX

Workaround Please change the for loop implementation as given below:
for(i = 0;i < (sizeof(all2portmemories) / 8;i++)

HERCULES_SW-6087 *The user's guide does not contain the macros for PBIST RAM groups for TMS570LC4357 and RM57.*

Issue details In the SafeTI Diagnostic Library - User's Guide, the macros for PBIST RAM Groups under Modules are for Cortex-R4 devices. The user guide doesn't list the macros for TMS570LC4357 and RM57x's PBIST RAM Groups. The macros for R5 devices need to be updated here.

Devices TMS570LC43x, RM57Lx

Workaround The macros for Cortex-R5 devices exist in the source code. This is a documentation issue.

HERCULES_SW-6124 *Flash self test using SDL on TMS570LC43x and RM57Lx devices causes abort.*

Issue details

In SL_SelfTest_Flash API, the loop variable "content" is used outside the for() loop as shown below. If the if() condition inside the for(..) loop fails, the "break" is not executed, and the "content" will be 32.When "content" is 32, the index becomes 8. Since the size of CAM_INDEX is 8 (from 0 to 7, register), it run into data abort when index is 8.

```
for(content = 0U; content<32U; content++)
{
    if((volatile uint32 *)(sl_epcREG1-
>CAM_CONTENT[content] & EPC_CAM_CONTENT_ADDR) == (volatile uint32 *)(flashBadECC1
& EPC_CAM_CONTENT_ADDR))
   {
       break;
   }
}
/* Calculate the index location */
index = content/4;
index = content - index*4;
content = content/4;
BF_SET(sl_epcREG1-
>CAM_INDEX[content], EPC_CAM_INDEX_CLEAR, EPC_CAM_INDEX_START(index),
EPC_CAM_INDEX_LENGTH);
```

Devices

TMS570LC43x, RM57Lx

Workaround

The workaround is to not use the loop variable outside the for(…) loop. Adding a counter before if(..) in for(..) loop, and using this counter outside the loop will solve the problem.

```
unsigned int counter = 0;
for(content = 0U; content<32U; content++) {
counter ++;
if((volatile uint32 )(sl_epcREG1-
>CAM_CONTENT[content] & EPC_CAM_CONTENT_ADDR) == (*volatile uint32
*)(flashBadECC1 & EPC_CAM_CONTENT_ADDR)) {
break;
}
/* Calculate the index location */
index = counter/4;
index = counter - index*4;
counter = counter/4;
BF_SET(sl_epcREG1-
>CAM_INDEX[*counter*], EPC_CAM_INDEX_CLEAR, EPC_CAM_INDEX_START(index),
EPC_CAM_INDEX_LENGTH);
```

HERCULES_SW-6130 *Fix for wrong shift in CRC calculation not applied to code for IAR toolchain.*

Issue details

Bug fix for issue SDOCM00122769 where in the sys_startup.c files crcAtInit_FLASH = SL_CRC_Calculate((uint64 *)((uint32)&ulFlashStartAddr), ((((uint32)&ulFlashEndAddr)-((uint32)&ulFlashStartAddr)) >> 6)); was changed to >>3. This was applied only to TI toolchain and not in IAR code in the demo application.

Devices

TMS570LC43x,TMS570LS04x,TMS570LS12x,TMS570LS31x,RM42x,RM46x,RM48x,RM57LX

Workaround

Modify crcAtInit_FLASH = SL_CRC_Calculate((uint64 *)((uint32)&ulFlashStartAddr), ((((uint32)&ulFlashEndAddr)-((uint32)&ulFlashStartAddr)) >> 6)); to crcAtInit_FLASH = SL_CRC_Calculate((uint64 *)((uint32)&ulFlashStartAddr), ((((uint32)&ulFlashEndAddr)-((uint32)&ulFlashStartAddr)) >> 3));

in IAR demo application.

The same fix has been applied to TI toolchain code in the SDOCM00122769 fix.

5 References

- *TMS570LS12x/11x 16/32-Bit RISC Flash Microcontroller Technical Reference Manual*

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (April 2015) to A Revision	Page
• Updates were made in Section 1.	2
• Update was made in Section 2.	2
• Updates were made in Section 3.	2
• Updates were made in Section 4.	6

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265