SafeTI™ Hercules™ Diagnostic Library

Release Notes

Version: 2.4.0

Date: 28 November 2017

TABLE OF CONTENTS

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 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
|
2.4.0                                     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
    +---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

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 & results Note: contents of structure param1 is valid only for self-test. Not valid for Fault Injection modes. Issues are: 1. The parameter is called 'sram_stResult', not 'param1'. 2. It's an 'enum', not a structure. 3. What does 'only valid for self-test' mean? Can the note be enhanced to further clarify what this means and why it isn't 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 Eg.. BIT_SET(sl_tcram1REG->RAMERRSTATUS, TCRAM_RAMERRSTATUS_ADDR_SERR); BIT_SET is iplemented as "|=" which can clear the bits if they are already set.

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 TRM reading I found that 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 & B used in testing are stack variables and for me those addresses are after each other ..240 and ...244 respectively. After I changed end region setting like this (-1) the test started to work
		sl_dmaREG->DMAMPR0E = (uint32)(srcAddr) + sizeof(uint32)-1U;
		
		Based on 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 we 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 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 SW calls _SL_HoldNClear_nError() which will attempt to clear the nERROR signal that hasn't 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. See TRM fig. 12-8 nERROR example 5 diagram for explanation and advisory against such assertion of the nERROR reset prior to a real errror occurrence. 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, i.e. 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 tests listed are disturbing the normal operation of the respective module. E.g. DMA tests will reconfigure a DMA channel to

		perform the test or HET_SRAM_PARITY_TEST will introduce a fault in the RAM which will cause the HET to stop operation. Thus normal operation will somehow be disturbed by these tests, even if the test will perform a cleanup afterwards and the module can continue its normal operation.

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().
		
		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 sizeof operator in Safety library demo code	The statements:
		for(i = 0;i < (sizeof(all2portmemories)/sizeof(uint32));i++)
		and
		for(j =0;j<(sizeof(all2portalgos)/sizeof(uint32));j++)
		are incorrect. There is no need to divide by the sizeof (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	Please 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 to me, please 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.
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 therefore also has no member param1.
SDOCM00122819	Please 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	Please redefine SL_FLAG_SET/CLEAR/GET with argument type of SL_SelfTestType to avoid implicit type cast	Please 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 will cause all pending bits to be cleared instead of only one. This could lead to masking real faults and thus 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 don't 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 isn't forced to be enabled, and if in the default application configuration isn't enabled, the SBERR bit will not be set. Assumption Should be documented in the user 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 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. If the CPU doesn't enter standby - then even though WFI is executed the code will resume execution after the WFI. LBIST won't run and the CPU won't 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 doesn't 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.	See 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

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	

SDOCM00122761	**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 alignement of the allocated memories.
Workaround
Added a dummy variable in-between to get 64 bit alignment.