Application Note

AWR294x Safety Diagnostic Library (SDL) User Reference

Allen Yin *China Central FAE*

ABSTRACT

This document was translated from a simplified Chinese source. (ZHCAG86)

AWR2944 is a high-performance mmWave radar chip introduced by Texas Instruments, integrating an RF front-end, a signal processing unit, and a microcontroller. Featuring a dual-core architecture, the chip includes an ARM Cortex-R5F Main Subsystem (MSS) and a C66x DSP Signal Processing Subsystem (DSS), supports the 76-81GHz band, and is widely applied in critical fields such as automotive Advanced Driver Assistance Systems (ADAS).

This paper systematically introduces the architecture and application of the AWR2944 Safety Diagnostic Library (SDL). The SDL implements chip-level fault detection and diagnosis, including: the Data Porting Layer (DPL) providing hardware abstraction; the Core Compare Module (CCM) implementing CPU lockstep monitoring; the Dual Clock Comparator (DCC) detecting clock drift; the ECC module providing memory error correction; the Bus Safety module protecting interconnect buses; the Memory Cyclic Redundancy Check (MCRC) implementing data integrity validation; parity checking detecting single-bit errors; the Programmable Built-In Self-Test (PBIST) executing memory self-tests; reset management tracking system status; the ROM checksum verifying firmware integrity; the windowed watchdog (RTI) monitoring execution flow; and the Self-Test Controller (STC) conducting logic gate circuit testing. All modules report safety events uniformly through the ESM (Error Signal Module). This paper provides SDL module and API descriptions, test cases, and configuration workflows, serving as a complete safety diagnostic technical reference for developing radar applications that meet functional safety requirements.

Table of Contents

List of Figures

List of Tables

Trademarks

All trademarks are the property of their respective owners.

1 Introduction to AWR294x and Safety Modules

Automotive Advanced Driver Assistance Systems utilize various sensors (mmWave radar, LiDAR, cameras, and satellite navigation) to perceive the surrounding environment at any time during vehicle operation, performing obstacle identification and tracking, and providing driving decision-assist information to the driver through system computational analysis. Among these, mmWave radar features excellent all-weather ranging and velocity-measuring performance alongside strong anti-interference capability, making it widely applied within ADAS systems.

Based on the RFCMOS process, TI integrates the RF/intermediate-frequency circuits, voltage-controlled oscillator (VCO), analog-to-digital converter (ADC), microcontroller (MCU), and digital signal processor (DSP) of the mmWave radar data link into a single chip, significantly reducing the cost and development difficulty of automotive radar.

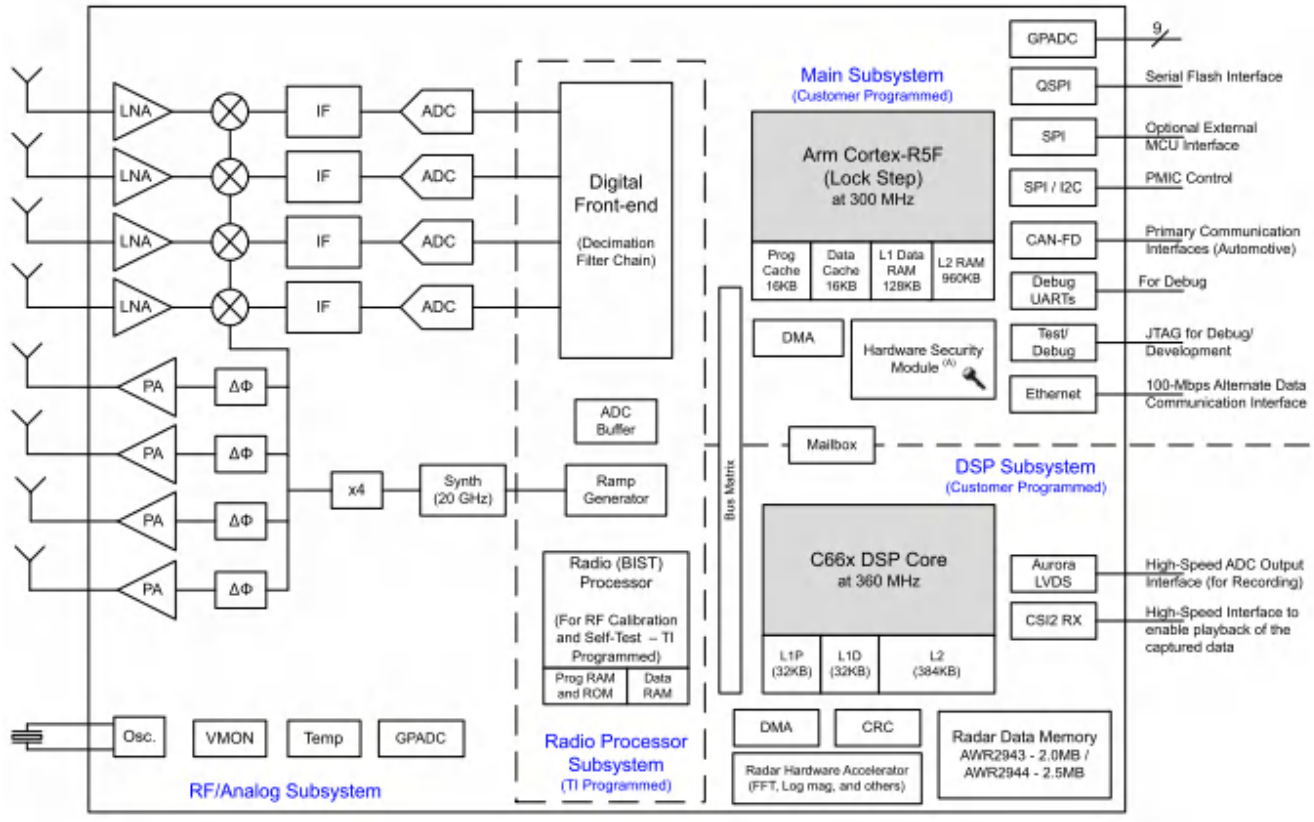

Figure 1-1. AWR2943/4 System Block Diagram

The functional safety of automotive mmWave radar chips is the core cornerstone of building automotive autonomous driving and active safety systems. As the control hub of core automotive sensors, the output of the mmWave radar chip directly drives decision-control systems critical to human safety, such as Autonomous Emergency Braking (AEB) and Adaptive Cruise Control (ACC). Once a chip fails to report or falsely reports due to random hardware failures, environmental electromagnetic interference, or systemic defects, it will directly trigger catastrophic traffic collisions or unintended braking. Therefore, strictly adhering to automotive functional safety standards (such as ISO26262) by deploying redundant architectures, real-time self-tests, and fault-tolerant mechanisms inside the chip is key to ensuring the radar system remains safe and controlled under extreme or failure conditions, thereby safeguarding human life on the road.

The safety modules of AWR2944 form the core of the chip's functional safety architecture, providing standardized interfaces through the Software Diagnostic Library (SDL) to help developers implement applications that comply with functional safety standards. In safety systems, detecting random hardware faults and guiding the system to a safe state is of paramount importance. The safety modules integrate various hardware safety mechanisms, including memory ECC error detection and correction, ESM (Error Signal Module)

uniform monitoring of safety events, CCM core lockstep comparison, DCC clock monitoring, PBIST memory self-test, and RTI watchdog diagnostics. The SDL encapsulates these hardware mechanisms through API interfaces, providing error capture and safety diagnostic functions, while the error response strategies are implemented by the application layer in accordance with the device safety manual requirements. This architecture supports bare-metal or RTOS environments, allowing system integrators to use the APIs provided by the SDL to configure ECC, set up interrupt handling, and execute diagnostic tests, thereby building radar application systems that meet functional safety standards like ISO26262 with high diagnostic coverage and fail-safe response capabilities.

2 In-Depth Guide to AWR294x Safety Diagnostic Library

The AWR294x series SoC provides comprehensive hardware safety mechanisms and Software Diagnostic Library (SDL) support for functional safety applications. Serving as a standardized interface layer, the SDL encapsulates various built-in safety functions of the chip, helping developers build application systems compliant with functional safety standards.

In safety-critical applications, timely detection of random hardware faults and guiding the system to a safe state is a core requirement. The SDL achieves this goal through error capture modules and safety mechanism modules, providing typical hardware safety mechanisms such as memory ECC error detection and correction, ESM (Error Signal Module) uniform monitoring of safety events, core lockstep comparison, clock monitoring, and memory self-test. The SDL provides configuration interfaces in the form of software APIs, such as ECC configuration APIs and interrupt setup reference examples, which system integrators can utilize to implement software diagnostics and meet safety system goals.

The SDL architecture is divided into two major stages based on the device operating mode. During the startup phase, self-test diagnostics are executed to verify normal device operation, and continuous diagnostic functions are initialized via SDL APIs; during the execution phase, the ESM processor receives real-time error notifications detected by continuous diagnostics, while SDL APIs are invoked to execute periodic diagnostic tasks. Error response strategies are flexibly managed by the application layer based on device safety manual requirements.

The SDL adopts a modular design, supporting bare-metal or RTOS environment deployment, covering the diagnostic needs of multiple IP cores. Applications interact with the SDL through standard APIs and can extend functionality by combining with external modules. This layered architecture provides a complete and flexible software diagnostic solution for developing mmWave radar applications compliant with functional safety standards such as ISO26262. (Note: The current mmWave Radar SDK version 4.7 does not contain the SDL and its associated code; version 4.6 must be downloaded).

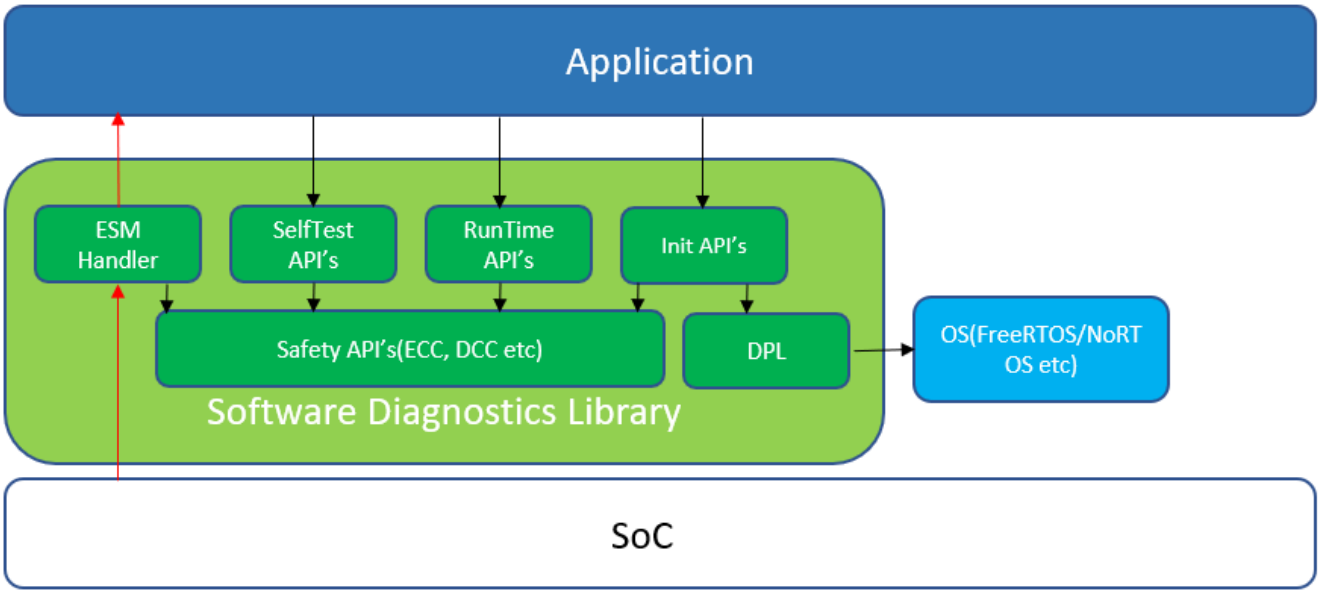

Figure 2-1. AWR294x SDL Architecture

2.1 DPL Data Porting Layer

The DPL (Data Porting Layer) is the low-level abstraction layer of the SDL, shielding differences in interrupt management, delay control, and address mapping across different chip platforms and processor cores. All upper-layer SDL modules access low-level hardware resources through DPL interfaces.

The design goal is to decouple SDL modules from specific hardware implementations, enabling the same set of SDL code to be ported to different chips such as AM263x, AM273x, and AWR2944. The DPL provides two main sets of capabilities: interrupt management (registration/unregistration/enabling/disabling) and basic utilities

(delay/address translation). On the AWR2944, it simultaneously supports interrupt management for both R5F and C66x DSP cores.

The DPL interface is registered to the SDL library in the form of a function pointer structure SDL_DPL_Interface, containing the following interfaces:

Table 2-1. DPL Interface Functions

Interface	Corresponding Function	Description
enableInterrupt	SDL_TEST_enableInterrupt	Enables the specified interrupt number
disableInterrupt	SDL_TEST_disableInterrupt	Disables the specified interrupt number
registerInterrupt	SDL_TEST_registerInterrupt	Registers an interrupt callback
deregisterInterrupt	SDL_TEST_deregisterInterrupt	Unregisters an interrupt callback
delay	ClockP_sleep	Delay (µs level)
addrTranslate	SDL_TEST_addrTranslate	Translates physical address to local address

The DPL itself does not provide independent tests; it acts as a prerequisite for all SDL modules. The usage method is to call SDL_TEST_dplInit() at the application entry to initialize the DPL before initializing specific SDL modules. Note that the DPL must be initialized before all other SDL modules.

2.2 CCM Core Compare Module

The CCM (Core Compare Module) implements the CPU lockstep mechanism. In the AWR2944, two R5F cores can be configured in lockstep mode, where the CCM compares the outputs (instructions, addresses, data) of both cores at every clock cycle, determining a CPU hardware fault when a discrepancy is detected. The CCM supports two categories of monitoring:

- Output Compare Block monitoring (SDL_CCM_MONITOR_TYPE_OUTPUT_COMPARE_BLOCK), comparing the instruction-level execution outputs of the master and slave cores;
- VIM Compare monitoring (SDL_CCM_MONITOR_TYPE_VIM). The lockstep mode makes CPU1 run as the diagnostic mirror of CPU0, with both executing the same interrupt handling. Interrupt signals are simultaneously sent to CPU0 and to CPU1 through a delay circuit, and their respective outputs are sent to the CCMR5 module via a dedicated bus for comparison. The output of CPU0 is also delayed to align timing. The CCMR5 compares the two signal paths, and if a mismatch occurs, it generates an MSS_CCMR5_ERR error interrupt, thereby detecting hardware faults in the CPU cores, with the error reported through the ESM module.

The low-level mechanism of this module is described in reference [1]: the lockstep comparator continuously monitors the CPU instruction and data buses, triggering an ESM safety response immediately upon detecting any deviation.

The CCM functional test (sdl_ccm_example) contains three sub-items:

1. Normal self-test: SDL_CCM_selfTest() verifies the basic functionality of the output compare block.
2. Error-forcing self-test: Calling SDL_CCM_selfTest() with the SDL_CCM_SELFTEST_TYPE_ERROR_FORCING mode to force error injection and verify the detection path.
3. Error injection: SDL_CCM_injectError() injects comparison errors to verify the ESM interrupt response and clearing workflow.

The VIM test (sdl_ccm_vim) targets comparison errors in the VIM path with a similar workflow.

Table 2-2. CCM Module Function Descriptions

Key Function	Functions
SDL_CCM_init	Initializes the CCM module
SDL_CCM_verifyConfig	Verifies the CCM configuration
SDL_CCM_selfTest	Executes self-test (normal/error-forcing modes)
SDL_CCM_injectError	Injects a CCM error
SDL_CCM_getErrorType	Gets the error type

Table 2-2. CCM Module Function Descriptions (continued)

Key Function	Functions
SDL_CCM_clearError	Clears the CCM error

2.3 DCC Dual Clock Comparator

The DCC (Dual Clock Comparator) uses a known reference clock path to measure the frequency of another clock signal, detecting whether the clock drift exceeds the allowed range. Reference [1] provides the detailed working principle:

The DCC contains two internal down-counters, counter0 and counter1, driven by clock0 and clock1 respectively. The application sets seed values for both counters and configures a tolerance window counter valid0 for clock0. Upon startup, both counters decrement simultaneously. When counter0 decrements to 0, it triggers valid0 to begin decrementing. If counter1 also decrements to 0 while valid0 is decrementing, the frequency ratio is considered normal, and all counters reload seed values to continue monitoring (continuous mode).

Two possible error conditions:

1. counter1 decrements to 0 before counter0 → clock1 is too fast or clock0 is too slow (including clock0 stuck).
2. counter1 is still not 0 after both counter0 and valid0 have decremented to 0 → clock1 is too slow or clock1 is stuck.

Either error freezes the counters, allowing the CPU to read counter values and analyze the cause of the fault.

In continuous mode, the DCC runs continuously without software intervention; in single-shot mode, it stops automatically after completing one measurement and generates a completion interrupt.

In the AWR2944, the MSS contains 4 DCC instances, and the DSS contains 2:

Table 2-3. DCC Instances in MSS and DSS

Instance	Subsystem	Reference Clock Source	Test Clock Source Example
MSS_DCCA	MSS	XTALCLK/RCCLK10	PLL_CORE / CR5F_CLK / SYS_CLK / QSPI / RTIA / SPIA / MCANA / SCIA
MSS_DCCB	MSS	XTALCLK/RCCLK10/SYS_CLK	PLL_CORE / CR5F_CLK / XTALCLK / CPTS / RTIB / SPIB
MSS_DCCC	MSS	XTALCLK/RCCLK10/SYS_CLK	PLL_DSS / GPIO[0-2] / WDT_CLK / XTALCLK
MSS_DCCD	MSS	XTALCLK/RCCLK10/SYS_CLK	PLL_PER / WDT_CLK / MCANA / GPIO[8-10,0-1]
DSS_DCCA	DSS	XTALCLK/RCCLK/SYS_CLK	SCIA / SYS_CLK / HWA_CLK / RTIA / WDT / DSP_CLK / PLL_DSP
DSS_DCCB	DSS	XTALCLK/RCCLK/RSS_CLK	HWA_CLK / CSI2_RX / WDT / RSS_CLK / PLL_PER

The basic workflow of DCC test code is as follows:

Table 2-4. DCC Test Workflow

Step	Action	Description
1	SDL_TEST_dplInit	Initializes the DPL
2	SDL_DCCA_clockInit	Configures the target clock source
3	SDL_ESM_init	Registers DCC error events
4	SDL_DCCAppGetClkRatio SDL_DCCAppSetSeedVals	Calculates the clock ratio and seed values, determining the reference/test clock frequency ratio and tolerance window
5	SDL_DCC_configure	Configures DCC parameters (seed values/tolerance/ clock selection)
6	SDL_DCC_verifyConfig	Verifies configuration correctness

Table 2-4. DCC Test Workflow (continued)

Step	Action	Description
7	SDL_DCC_enableIntr	Enables DCC completion/error interrupts
8	SDL_DCC_enable	Starts DCC monitoring
9	SDL_DCCAppWaitForCompletion	Waits for the completion interrupt or error interrupt

Typical clock combinations in test cases:

Table 2-5. Typical Clock Combinations in DCC Test Cases

Use Case	Reference Clock	Test Clock	Mode	Expectation
UC-1	XTAL_CLK(40MHz)	MSS_MCANA_CLK(30MHz)	Continuous	Frequency ratio deviation, generates error
UC-2	XTAL_CLK(40MHz)	MSS_MCANA_CLK(30MHz)	Single-shot	Normal completion
UC-3	XTAL_CLK(40MHz)	SYS_CLK(150MHz)	Continuous	Normal operation

List of Key Functions Provided by DCC Module

Table 2-6. DCC Module Key Function Descriptions

Key Function	Functions
SDL_DCC_configure	Configures clock sources, seed values, and tolerance windows
SDL_DCC_verifyConfig	Verifies configuration correctness
SDL_DCC_enable	Starts DCC monitoring
SDL_DCC_disable	Stops DCC
SDL_DCC_enableIntr	Enables completion/error interrupts
SDL_DCC_clearIntr	Clears interrupt flags
SDL_DCC_getStatus	Gets completion/error status

2.4 ECC Error Correction Code

The ECC module provides Single Error Correction (SEC) and Double Error Detection (DED) for on-chip memory through the ECC Aggregator and ECC Wrapper. Reference [1] states that the ECC Aggregator communicates with the ECC Wrapper of each memory via a serial interface, with the Wrapper responsible for actual ECC detection and correction. The AWR2944 chip contains 5 ECC Aggregators:

- MSS_ECC_AGG_R5A (Cache/TCM/VIM of R5F Core A)
- MSS_ECC_AGG_R5B (Cache/TCM/VIM of R5F Core B)
- MSS_ECC_AGG_MSS (L2/MBOX/RETRAM/GPADC/TPTC)
- DSS_ECC_AGG (L3/MBOX/TPTC)
- HSM_ECC_AGGR (HWA)

Each Aggregator manages multiple RAM IDs, where each ID corresponds to an ECC-protected memory module.

Core features of the ECC Aggregator:

- Reduces memory soft errors through ECC (SEC is correctable, DED is detectable but uncorrectable)
- Supports parity checking on the serial FIFO interface
- Control bits have redundancy protection, and a single-bit flip can trigger an interrupt
- Supports Inject Only mode for diagnostic testing
- Provides software-readable SEC/DED error status, containing row address and flipped bit position details

ECC testing covers 11 memory regions (r5_atcm0, r5_btcm, mss_l2, mss_tptc, mcan, dss_ecc_hwa, dss_l3_ecc, dss_mailbox, dss_tptc, dsp_edc, dss_l2_edc).

General test workflow:

1. The R5F core executes SDL_EXCEPTION_init to register exception handlers.
2. SDL_ECC_initMemory initializes ECC protection for the target RAM.
3. Initializes the ESM and configures high- and low-priority interrupts.

4. SDL_ECC_init initializes the ECC aggregator (specifying the RAM ID list).
5. SDL_ECC_injectError injects single-/double-bit errors, waiting for the ESM interrupt.
6. In the ESM callback, SDL_ECC_getESMErrorInfo and SDL_ECC_getErrorInfo are called to obtain detailed error information, followed by clearing the interrupt.

Table 2-7. ECC Module Function Descriptions

Key Function	Functions
SDL_ECC_init	Initializes the ECC aggregator, configuring the RAM subtype list
SDL_ECC_initMemory	Initializes ECC protection for the specified RAM
SDL_ECC_injectError	Injects SEC/DED errors into the specified memory
SDL_ECC_getESMErrorInfo	Parses ECC error types from the ESM interrupt source
SDL_ECC_getErrorInfo	Gets detailed ECC information (address, flipped bits, etc.)
SDL_ECC_clearNIntrPending	Clears pending interrupts
SDL_ECC_ackIntr	Acknowledges ECC interrupts

CAUTION

On the R5F, the exception handler (SDL_EXCEPTION_init) must be initialized first, otherwise DED may not be reported correctly.

2.5 ECC Bus Safety

ECC Bus Safety extends ECC protection from memory to bus interconnect modules. Reference [1] points out that the command/address buses on the system interconnect bus have parity checking and redundancy protection, while the data bus has SEC and DED protection. This module covers approximately 60 bus nodes in both MSS and DSS subsystems.

Testing is divided into three categories for each bus node: Single-Bit Error Correction (SEC), Double-Bit Error Detection (DED), and Redundancy Error Detection (RED). Coverage includes: TPTC read/write channels, DMA controllers, Mailbox, FIFO, CPSW, QSPI, GPADC, MCRC, and other modules.

Testing method: Triggers data read/write on the target bus node through EDMA transfer, injecting ECC errors and verifying the ESM response.

Table 2-8. ECC Bus Safety Module Function Descriptions

Key Function	Functions
SDL_ECC_BUS_SAFETY_*_SEC_Test	Single-bit error correction test for each node
SDL_ECC_BUS_SAFETY_*_DED_Test	Double-bit error detection test for each node
SDL_ECC_BUS_SAFETY_*_RED_Test	Redundancy error detection test for each node

2.6 HWA Hardware Accelerator Safety

HWA (Hardware Accelerator) is the radar signal processing hardware accelerator in the AWR2944. The HWA internally contains multiple DMEMs (Data Memory) and a Window RAM, which are protected by parity checking. The HWA safety module detects parity errors in the DMEMs and the lockstep execution state of the Finite State Machine (FSM).

The HWA belongs to the DSS subsystem, with the ESM instance being SDL_ESM_INST_DSS_ESM. RAM ID 22 in DSS ECC Aggregator corresponds to the HWA PARAM RAM.

Testing covers 8 DMEMs each for DMA0 and DMA1, and the Window RAM, totaling 18 parity test items plus 1 FSM lockstep test. Each test invokes SDL_HWA_memParityExecute or SDL_HWA_fsmLockStepExecute.

Table 2-9. HWA Safety Module Function Descriptions

Key Function	Functions
SDL_HWA_memParityExecute	Executes parity checking test for the specified memory
SDL_HWA_fsmLockStepExecute	Executes FSM lockstep test

Table 2-9. HWA Safety Module Function Descriptions (continued)

Key Function	Functions
SDL_HWA_ESM_CallbackFunction	HWA ESM error callback

2.7 MCRC Cyclic Redundancy Check

MCRC (Memory Cyclic Redundancy Check) verifies memory data integrity through hardware-accelerated CRC calculation. This module uses a maximum 64-bit polynomial $f(x) = x^{64} + x^4 + x^3 + x + 1$ to construct PSA (Parallel Signature Analysis) registers, performing calculations on 8/16/32/64-bit data widths.

MCRC supports two independent channels, which can perform CRC calculations on multiple memory regions in parallel and supports three modes:

Auto Mode: DMA automatically transfers data for CRC calculation without CPU involvement. When a mismatch occurs, an interrupt is generated, and the hardware provides a 16-bit sector ID to locate the faulty sector.

Semi-CPU Mode: DMA remains responsible for data transfer, but an interrupt is triggered after each sector compression is completed, prompting the CPU to read the signature register and manually compare it with the preset value.

Full-CPU Mode: The CPU is fully responsible—both transferring data to the signature register and reading the result for comparison. No interrupts, no DMA, and all counters are disabled.

Full-CPU Mode Testing (mcrc_full_cpu):

Table 2-10. MCRC Full-CPU Mode Workflow

Step	Action	Description
1	SDL_MCRC_init	Initializes the MCRC channel
2	SDL_MCRC_channelReset	Resets the CRC channel
3	SDL_MCRC_config	Configures mode, pattern size, and sector count
4	Input raw data	Writes known data patterns
5	SDL_MCRC_computeSignCPUmode	Calculates the CRC signature
6	Compare signature	Compares with the reference signature

Semi-CPU Mode Testing (mcrc_semi_cpu):

Table 2-11. MCRC Semi-CPU Mode Workflow

Step	Action	Description
1	SDL_MCRC_init	Initializes the MCRC channel
2	SDL_MCRC_channelReset	Resets the CRC channel
3	SDL_MCRC_config	Configures the channel to Semi-CPU mode
4	EDMA operation	Transfers data into PSA registers
5	Waiting for interrupt	MCRC completes one sector compression
6	Read PSA Sector Signature	Compares with the reference signature

Auto Mode Testing (mcrc_auto_cpu):

Table 2-12. MCRC Auto Mode Workflow

Step	Action	Description
1	SDL_MCRC_init SDL_MCRC_channelReset	Initializes and resets the MCRC channel
2	SDL_MCRC_config	Sets SDL_MCRC_OPERATION_MODE_AUTO
3	EDMA transfer	Transfers both data and pre-calculated signatures to the CRC Value Register
4	MCRC auto comparison	Hardware completes signature comparison
5	Interrupt gets result	CRC Fail interrupt or completion notification

Key Function Description:

Table 2-13. MCRC Module Function Descriptions

Key Function	Functions
SDL_MCRC_init	Initializes MCRC channels (watchdog/block preload values)
SDL_MCRC_channelReset	Resets the CRC channel
SDL_MCRC_config	Configures mode, pattern size, and sector count
SDL_MCRC_configCRCType	Configures the CRC polynomial
SDL_MCRC_computeSignCPUmode	Calculates CRC signature in Full-CPU mode
SDL_MCRC_getPSASigRegAddr	Gets PSA signature register address
SDL_MCRC_getPSASectorSig	Gets sector signature value
SDL_MCRC_getPSASig	Reads PSA signature value
SDL_MCRC_enableIntr	Enables MCRC interrupts

2.8 Parity

Parity checking detects single-bit flip errors in memory, only capable of detecting odd numbers of bit errors and unable to correct them. In the AWR2944, it is primarily used for the TPCC memory of EDMA (MSS and DSS), R5F TCM memory, and DSS L1 and L2 memories. Errors are reported through the ESM, and the software must clear the error status by writing to the control register corresponding to the RAM's parity.

DMA TPCC Parity Testing (sdl_dma_parity):

Table 2-14. DMA TPCC Parity Checking Workflow

Step	Action	Description
1	SDL_TEST_dplInit	Initializes the DPL
2	SDL_ESM_init	Registers DMA parity events
3	Parity_funcTest	Executes functional parity testing
4	ESM callback handling	Disables parity for the corresponding RAM and clears status after detecting an error

TCM Parity testing and DSS L1/L2 Parity testing workflows are similar, targeting R5F TCM and DSS L1/L2 memories respectively.

Table 2-15. Parity Module Function Descriptions

Key Function	Functions
Parity_funcTest	Executes functional parity testing
SDL_ESM_applicationCallbackFunction	ESM callback function handling parity errors

2.9 PBIST Programmable Built-In Self-Test

PBIST (Programmable Built-In Self-Test) is a dedicated coprocessor featuring a specialized instruction set for memory testing. The PBIST reads test algorithms and memory configuration information from the on-chip ROM, executing tests directly on target memories via a dedicated data path without CPU intervention.

Compared to the CPU running software tests, the advantages of PBIST include: a dedicated path bypassing the CPU access path, a more efficient dedicated instruction set, smaller code size, and lower hardware area overhead. The AWR2944 contains two PBIST instances:

- TOP_PBIST (covering all memories in the MSS subsystem)
- DSS_DSP_PBIST (covering all memories in the DSS subsystem)

The recommended test algorithm is March13N, which can detect address decoder faults, Stuck-At faults, coupling faults, state coupling faults, parametric faults, write recovery faults, and read/write logic faults.

PBIST Test Workflow (taking TOP PBIST as an example):

Table 2-16. TOP PBIST Test Workflow

Step	Action	Description
1	SDL_TEST_dplInit	Initializes the DPL
2	SDL_PBIST_configNeg	Executes negative test, configuring incorrect algorithm parameters to verify PBIST detection capability
3	Disable Cache	Disables L1P/L1D Cache to prevent interference with test results
4	SDL_PBIST_selfTest	Executes normal PBIST, running March algorithm to detect memory faults
5	Waiting for completion/timeout	Waits for PBIST completion interrupt or timeout
6	Get results	Reads test status (pass/fail)

Note: Cache must be disabled during PBIST, and the test execution time is relatively long.

Table 2-17. PBIST Module Function Descriptions

Key Function	Functions
SDL_PBIST_selfTest	Executes PBIST self-test
SDL_PBIST_configNeg	PBIST negative test configuration (fault injection)

2.10 Reset Management

The Reset module provides reset cause query and software reset generation. By reading the reset status register, the software can determine the cause of the last reset (power-on reset/watchdog reset/software reset/ external pin reset/HSM watchdog reset), thereby selecting different initialization strategies during startup.

Reset Test Workflow:

Table 2-18. RESET Module Test Workflow

Step	Action	Description
1	SDL_getWarmResetCause	Reads the warm reset cause
2	Categorized handling	Selects initialization strategy based on the reset cause
3	SDL_r5fGetResetCause	Read R5F core reset cause
4	SDL_generateSwWarmReset	If power-on reset, triggers software warm reset to verify path
5	SDL_rcmDspLocalReset	Executes DSP local reset

Table 2-19. RESET Module Function Descriptions

Key Function	Functions
SDL_getWarmResetCause	Gets warm reset cause
SDL_r5fGetResetCause	Gets R5F core reset cause
SDL_generateSwWarmReset	Triggers software warm reset
SDL_rcmDspLocalReset	Triggers DSP local reset

Note

The reset status register is automatically cleared after being read; critical data must be saved before a software warm reset.

2.11 ROM Checksum

ROM Checksum calculates a checksum of the MCU ROM content and compares it with a preset reference value to detect accidental modifications or corruption of the ROM content. This is one of the means to establish trust during system startup.

ROM Checksum Test Workflow:

Table 2-20. ROM Content Validation Workflow

Step	Action	Description
1	SDL_TEST_dpllnit	Initializes the DPL
2	SDL_ROM_Checksum_compute	Calculates the ROM checksum
3	Compare results	Compares with the preset reference value

The return value has three cases: SDL_PASS (pass), SDL_EBADARGS (argument error), or others (integrity check failure).

Table 2-21. ROM Module Function Descriptions

Key Function	Functions
SDL_ROM_Checksum_compute	Calculates ROM checksum and verifies integrity

2.12 RTI Real-Time Windowed Watchdog

The RTI DWWD (Digital Windowed Watchdog) requires watchdog servicing (kicking) to be completed within a specific time window; kicking too early or too late will trigger a safety response. It effectively detects CPU execution flow anomalies (runaway, deadlock, execution delay).

Window size ratio configuration:

Table 2-22. RTI Window Ratio Configuration

Window Size	Open Window Ratio	Closed Window Ratio
100%	100%	0% (Traditional watchdog behavior)
50%	50%	50%
25%	25%	75%
12.5%	12.5%	87.5%
6.25%	6.25%	93.75%
3.125%	3.125%	96.875%

The test workflow covers four use cases (UC1-UC4):

Table 2-23. RTI Module Test Workflow

Step	Action	Description
1	Configure RTI parameters	Sets window size and preload values
2	Start watchdog	Calls SDL_RTI_start
3	UC1: Correct Service	Servicing within the open window, no error interrupt
4	UC2: Permanent Fault	No servicing triggers timeout, ESM interrupt triggered
5	UC3: Early Service	Servicing within the closed window, ESM interrupt triggered
6	UC4: Window Switching	Configures different window ratios to verify validation timing

Table 2-24. RTI Module Function Descriptions

Key Function	Functions
SDL_RTI_config	Configures windowed watchdog parameters (window size/preload values)
SDL_RTI_verifyConfig	Verifies the RTI configuration
SDL_RTI_start	Starts windowed watchdog
SDL_RTI_service	Services (kicks) the watchdog
SDL_RTI_writeWinSz	Dynamically modifies the window size
SDL_RTI_getStatus	Gets interrupt status
SDL_RTI_clearStatus	Clears interrupt status

2.13 STC Self-Test Controller

The STC (Self-Test Controller) performs Logic Built-In Self-Test (LBIST) on the CPU core logic gate circuits based on OPMISR (On-Product Multiple Input Signature Register). The STC mainly consists of a ROM interface, an FSM sequence controller, register files, a Bypass/ATE interface, and a VBUSP interface.

The STC reads test vectors (including Segment ID, Pattern Count, and Golden MISR values) from the on-chip ROM, shifts the scan data into the logic under test, and finally compares the 896-bit MISR value generated by the OPMISR controller with the Golden MISR value in the ROM, writing the result into the status register. It supports the Stuck-at fault model and the Launch-on-System-Clock transition delay method.

In the AWR2944, the STC instances for MSS and DSS run independently and can be triggered separately.

STC Module Test Workflow:

Table 2-25. STC Module Test Workflow

Step	Action	Description
1	SDL_TEST_dplInit	Initializes the DPL
2	SDL_ESM_init	Registers STC events
3	SDL_STC_config	Configures STC parameters (test duration/interval/ algorithm)
4	SDL_STC_selfTest	Executes STC self-test
5	Waiting for completion/timeout	Waits for test completion or timeout
6	SDL_STC_getResult	Reads test results
7	Clear interrupt	Clears STC interrupt flags

Table 2-26. STC Module Function Descriptions

Key Function	Functions
SDL_STC_selfTest	Executes STC LBIST self-test
SDL_STC_config	Configures STC parameters
SDL_STC_getResult	Gets STC test results

Note

During the STC, the CPU suspends execution of user code, requiring a balance between diagnostic coverage and system availability.

3 Summary

The SDL provides a plug-and-play, functionally safety-compliant diagnostic solution for the AWR294x series, reducing the complexity of safety mechanism integration through standardized APIs. Its modular design enables system integrators to quickly build functional safety applications, significantly shortening the development cycle. The layered architecture of the SDL achieves decoupling of hardware diagnostics from application logic, improving code maintainability and portability. As the official diagnostic library provided by TI, the SDL has been widely validated in safety-critical domains such as automotive radar, providing developers with a reliable path to implement functional safety diagnostics.

4 Glossary

RBL - ROM Bootloader -	ROM Bootloader
SBL - Secondary Bootloader -	Secondary Bootloader
ROM - Read Only Memory -	Read Only Memory
HWA - Hardware Accelerator -	Hardware Accelerator
MSS - Main SubSystem (R5F) -	Main SubSystem (R5F)
DSS - DSP SubSystem -	DSP SubSystem
RSS - Radar SubSystem -	Radar SubSystem
BSS - BIST SubSystem -	BIST SubSystem
DSP - Digital Signal Processors -	Digital Signal Processors
DFE - Digital Front End -	Digital Front End
MPU - Memory Protection Unit -	Memory Protection Unit
EDMA - Enhanced Direct Memory Access -	Enhanced Direct Memory Access
RTI - Real Time Interrupt -	Real Time Interrupt
PBIST - Programmable Built-In Self-Test -	Programmable Built-In Self-Test
ESM - Error State Machine -	Error State Machine
MCRC - Memory Cyclic Redundancy Check -	Memory Cyclic Redundancy Check
DCC - Dual Clock Comparator -	Dual Clock Comparator
GPADC - General-Purpose Analog Digital Converter -	General-Purpose Analog Digital Converter
TCMA/TCMB - Tightly Coupled Memory -	Tightly Coupled Memory
R5FA/R5FB - ARM Real Time Processor with Floating Point Unit -	ARM Real Time Processor with Floating Point Unit
TPTC - Third Party Transfer Controller -	Third Party Transfer Controller
TPCC - Third Party Channel Controller -	Third Party Channel Controller
PRCM - Power, Reset and Clock Management -	Power, Reset and Clock Management
STC - Self-Test Controller -	Self-Test Controller
WDT - Watchdog Timer -	Watchdog Timer
WWDT - Windowed Watchdog Timer -	Windowed Watchdog Timer
VIM - Vectored Interrupt Manager -	Vectored Interrupt Manager
IRQ - Interrupt Request -	Interrupt Request
FIQ - Fast Interrupt Request -	Fast Interrupt Request
SECDEC - Single Error Correction Double Error Code -	Single Error Correction Double Error Code
ECC - Error Correction Code -	Error Correction Code

5 References

- *AWR294x Technical Reference Manual (Rev. D)*
- *MMWAVE-MCUPLUS-SDK Software development kit (SDK)*
- *AWR2943/AWR2944 Single-Chip 76 to 81GHz FMCW Radar Sensor datasheet (Rev. E)*

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, regulatory 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 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, TI’s General Quality Guidelines, 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. Unless TI explicitly designates a product as custom or customer-specified, TI products are standard, catalog, general purpose devices.

TI objects to and rejects any additional or different terms you may propose.

Last updated 10/2025