

CoverageMaster winAMS

Automated embedded software unit test tool
 Performs unit testing on unmodified MPU target code
 Automatically create input test data using the static analysis tool 'CasePlayer2'
 Complies with ISO26262 Automotive Functional Safety Standard

Automated Embedded Software Unit Test Tool Target Code Executed using GAIO's ISS

CoverageMaster winAMS is an automated embedded software unit test tool that executes the target code without modifications in order to achieve reliable testing results. CoverageMaster is configured with an MPU Instruction Set Simulator (ISS) in order to test the C source code logic, as well as check for possible implementation issues.

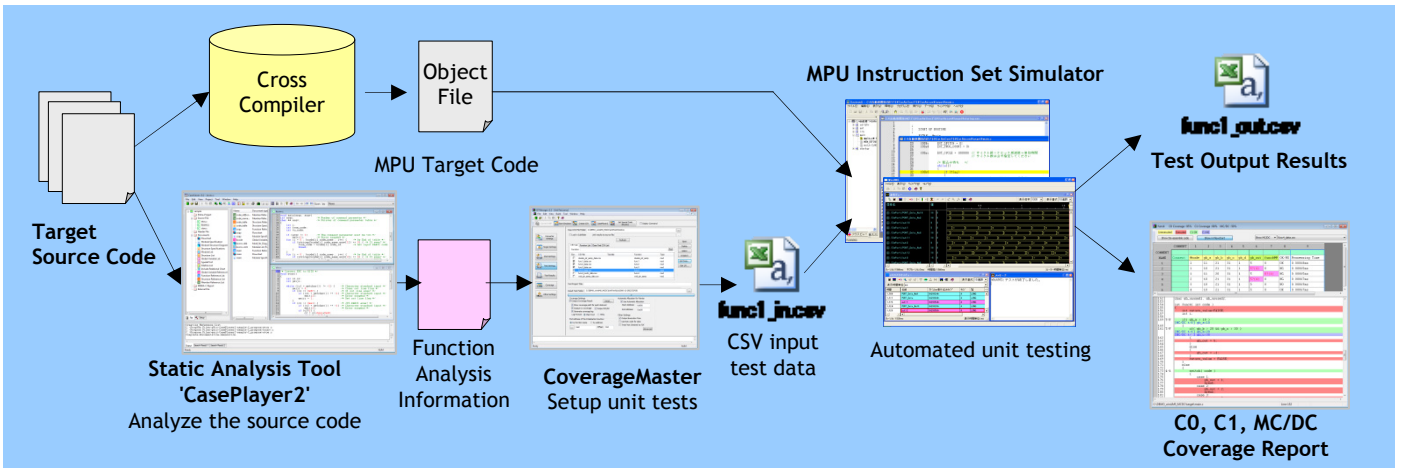
Compliance with ISO26262 Automotive Functional Safety Standard

The ISO26262 automotive functional safety standard is an adaptation of the IEC 61508 functional safety meta-standard. The "structural coverage metrics at the software unit level" are defined in "Part 6-9 : Software unit testing".
 The standard requires coverage measurement, including Statement Coverage (C0), Branch Coverage (C1), and MC/DC (Modified Condition/Decision Coverage) to be carried out in accordance with the Automotive Safety Integrity Levels (ASIL). CoverageMaster winAMS meets these coverage measurement requirements.

No Hook Code or Test Driver Required

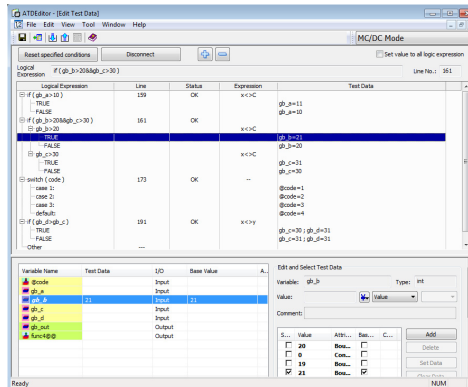
Specialized hook code or test drivers are not required for unit testing with CoverageMaster WinAMS. The target MPU code is executed as is, for reliable as close to the actual device as possible test results. As an additional advantage, this means that setting up a separate test environment is not required.

Also, the safety standard states that software unit testing should be performed in an environment as close to the target environment as possible. Unit testing is performed using the target MPU object file without inserted hook code or test drivers, making CoverageMaster WinAMS a prime choice for performing software unit tests in compliance with the ISO26262 safety standard.



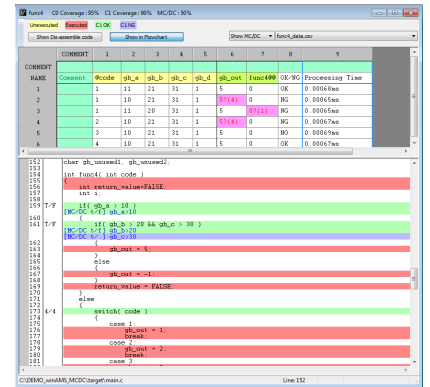
Detect Input/Output Variables Automatically

Save time and reduce selection errors by having the global input/output variables used by the target function automatically listed for you. Made possible through the use of static analysis information provided by CasePlayer2.



Create Optimized Input Test Data Combinations

Automatically create optimized C1 & MC/DC test data combinations. Made possible through the use of static analysis information provided by CasePlayer2.

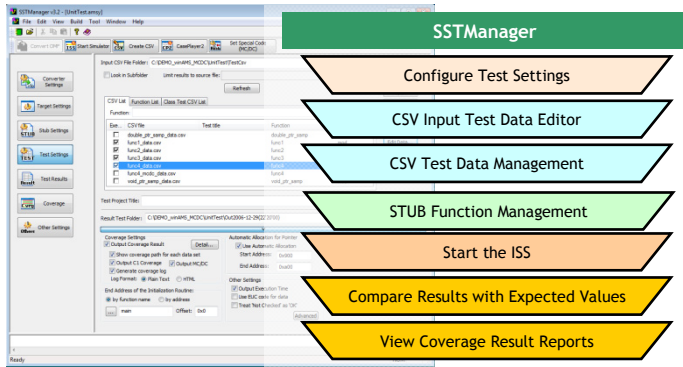


Easy to Understand Coverage Results

Coverage results for C0, C1, and MC/DC are output in a colorized easy to understand format with numerous methods for reference and analysis.

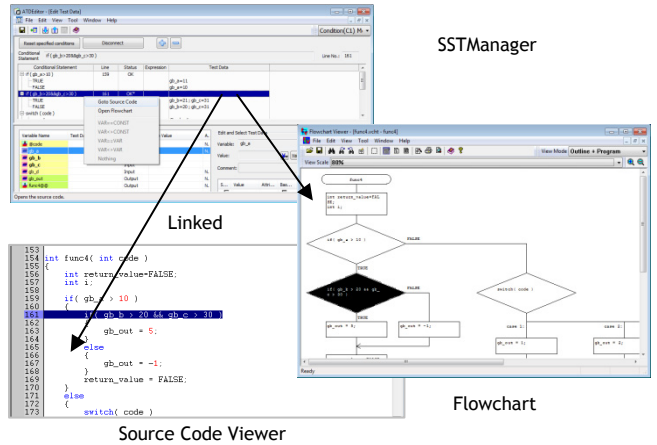
Unit Test Integrated Management Tool 'SSTManager'

SSTManager is an integrated management tool used to access the unit testing processes. These include configuring test settings, creating test data, managing test data and STUB functions, executing test operations, starting the ISS and viewing test reports.



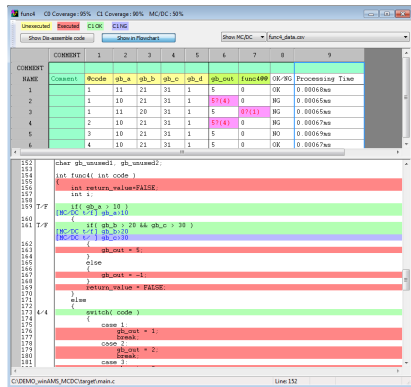
Access to Source Code and Program Documents

The source code and CasePlayer2 created program documents may be easily accessed for reference during unit testing. This allows the user to obtain a better understand of the program's components and structure simply and quickly.



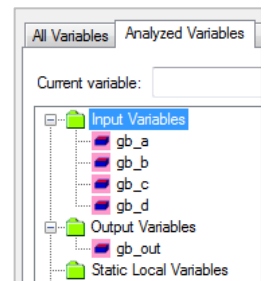
Create Unit Test Coverage Reports Automatically

Unit test coverage reports are created automatically after testing has been completed. The coverage results for C0, C1 and MC/DC may be saved for later reference, or viewed and analyzed using the dedicated coverage view tool. Coverage results are colorized for easy recognition and understanding.



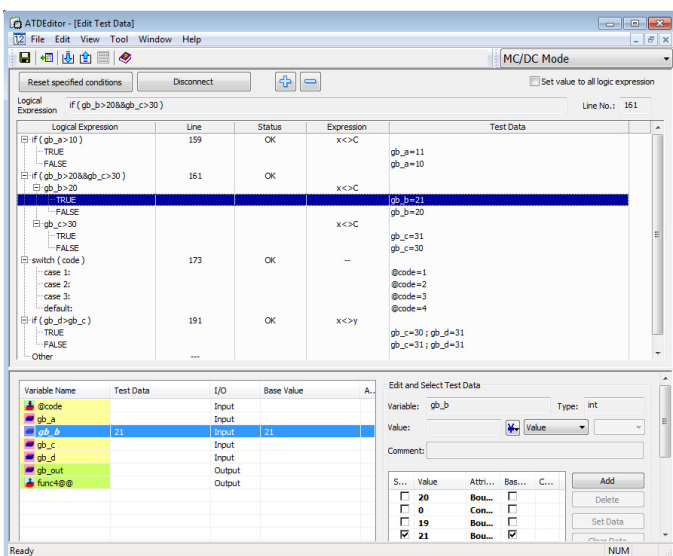
Detect Input/Output Variables Automatically

Save time and reduce selection errors by having the global input/output variables used by the target function automatically listed for you. From the list the user may then easily choose which items to add to the unit test, including additional variables from the full list as desired. Made possible through the use of static analysis information provided by CasePlayer2.



Create Optimized Input Test Data Combinations

Automatically create optimized C1 & MC/DC test data combinations. The user may choose to only create the minimal number of tests needed to achieve coverage results in order to keep the test as efficient and simple as possible. Or the user may choose to include additional test data combinations to test using specific test values. Made possible through the use of static analysis information provided by CasePlayer2.



MPU / Dev. Environment Support List

	Status	Supported Dev Environment		Status	Supported Dev Environment
Fujitsu			Infineon Technologies		
FM3 (Coretex-M3)	OK	Keil, IAR, GNU (GCC)	TriCore	OK	TASKING
FM38L	OK	Fujitsu	Renesas Electronics (HITACH, Mitsubishi)		
FM38FX	OK	Fujitsu	M16C	OK	GAIO, Renesas
FM38LX	OK	Fujitsu	M32R	OK	GAIO, Renesas
FM316LX	OK	Fujitsu	M32C	OK	GAIO, Renesas
FR20/30	OK	Fujitsu	SH2/2E	OK	GAIO, Renesas, GNU (GCC)
FR60Lite	OK	Fujitsu	SH2A	OK	GAIO, Renesas
FR80	OK	Fujitsu	SH2A-FPU	OK	GAIO, Renesas
Renesas Electronics (NEC)			SH3/3E	OK	GAIO, Renesas, GNU (GCC)
78K0	OK	GAIO, NEC	SH4	OK	GAIO, Renesas, GNU (GCC)
78K0R	OK	GAIO, NEC	SH4A	Call	GAIO, Renesas
V850	OK	GAIO, NEC, GHS	H8/S	OK	GAIO, Renesas
Freescale Semiconductor			H8/SX	OK	Renesas
PPC500	OK	Diab	H8/300H	OK	GAIO, Renesas
PPC600	OK	Diab, GNU (GCC)	R32C/100	Call	GAIO, Renesas
PPC800	OK	Diab	R8C/Tiny	OK	Renesas
MPC5500	OK	Diab	RX600	OK	GAIO, Renesas
MPC83xx	OK	Diab, GNU (GCC)	Sony		
MIPS			SPC900/970	Call	GAIO
MIPS	Call	GNU (GCC)	Toshiba Semiconductor		
ARM			TLCS870C	OK	Toshiba
ARM7	OK	GAIO, ARM, GNU (GCC), GHS	TLCS900(L/H)	OK	Toshiba
ARM9	OK	GAIO, ARM, GNU (GCC), GHS	TX19	OK	Toshiba
Coretex-M3	OK	Keil, IAR, GNU (GCC)	TX03 (Coretex-M3)	OK	Keil, IAR, GNU (GCC)

System Requirements

OS: Windows XP SP2 or later, Windows Vista (32-bit) / Windows 7 (32/64-bit)
Recommended System Requirements : Pentium 2GHz or higher, 2GB RAM or more