

CoverageMaster Setup Manual

Version 1.0
2014/07/11

Table of Contents

Table of Contents	1
CoverageMaster Setup Manual	3
Introduction	3
Tool Version	3
Tool Relation Diagram.....	3
Setup Flow.....	4
CasePlayer2 Setup	6
1. Project Settings	6
2. Add Source Files	8
3. Analysis Settings	10
4. Preprocessor Settings.....	12
5. C Option Parameter Settings	14
(Reference) ASM Instruction Definition Settings	15
6. Create Documents	16
7. Create Undefined Function Skeleton	17
8. Build Object File.....	17
(Reference) Other Relevant FAQs.....	18
CoverageMaster winAMS Setup	19
9. Project Settings	19
10. Converter Settings	21
11. Target Settings	25
12. Test Settings.....	27
13. Stub Settings	29
14. Build Object File.....	32
(Reference) Other Relevant FAQs.....	33

CoverageMaster Setup Manual

Introduction

The purpose of this document is to explain how to setup GAIO TECHNOLOGY's test tools, CoverageMaster winAMS and CasePlayer2. Please refer to the tool's help manual and tutorial documentation for details regarding information not covered in this document.

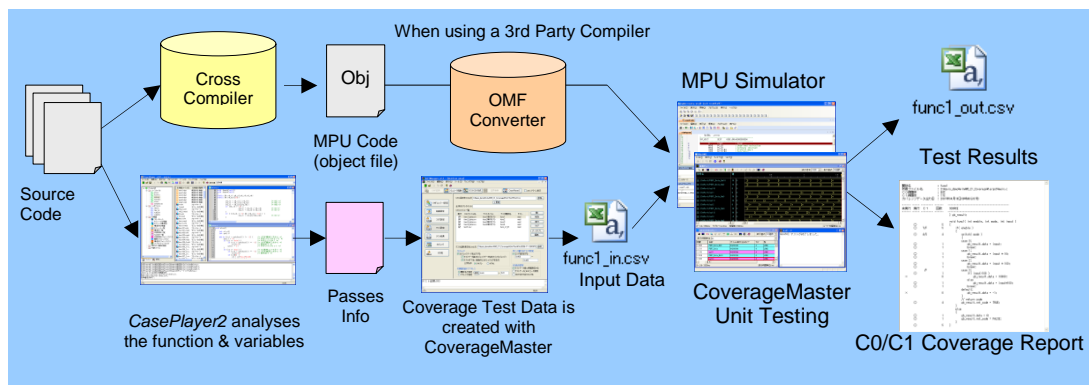
Additional technical support related information and FAQs can also be found at the following link: <http://www.gaio.com/support/user/techpaper.html>.

Tool Version

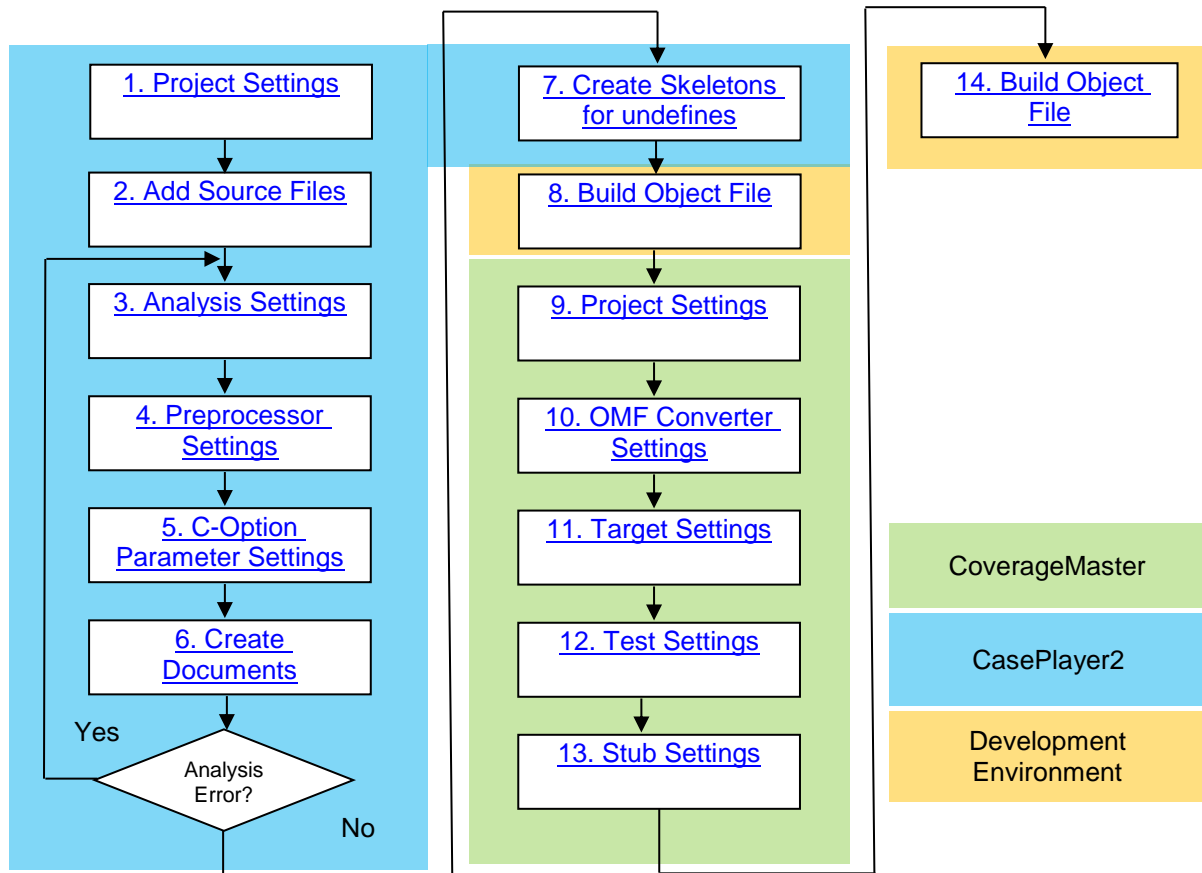
The following tool versions were used as a base for this document.

CoverageMaster winAMS	V3.6.2
CasePlayer2	V5.6.2

Tool Relation Diagram



Setup Flow



Setup CoverageMaster and CasePlayer2 settings in the following order:

1. CasePlayer2 Setup
 - [1.1. Project Settings](#)
 - [1.2. Add Source Files](#)
 - [1.3. Analysis Settings](#)
 - [1.4. Preprocessor Settings](#)
 - [1.5. C-Option Parameter Settings](#)
 - [1.6. Create Documents](#)

CasePlayer2 setup is complete when the Create Documents step completes successfully with the “Complete-Documentation-Generation” message output in the message view.

If an analysis error occurs, verify and make necessary changes to the 1.3 Analysis Settings, 1.4 Preprocessor Settings, and/or 1.5 C-Option Parameter Settings, then Create Documents again until the error has been resolved.

2. CoverageMaster winAMS Setup

[2.1. Project Settings](#)

[2.2. OMF Converter Settings](#)

[2.3. Target Settings](#)

[2.4. Test Settings](#)

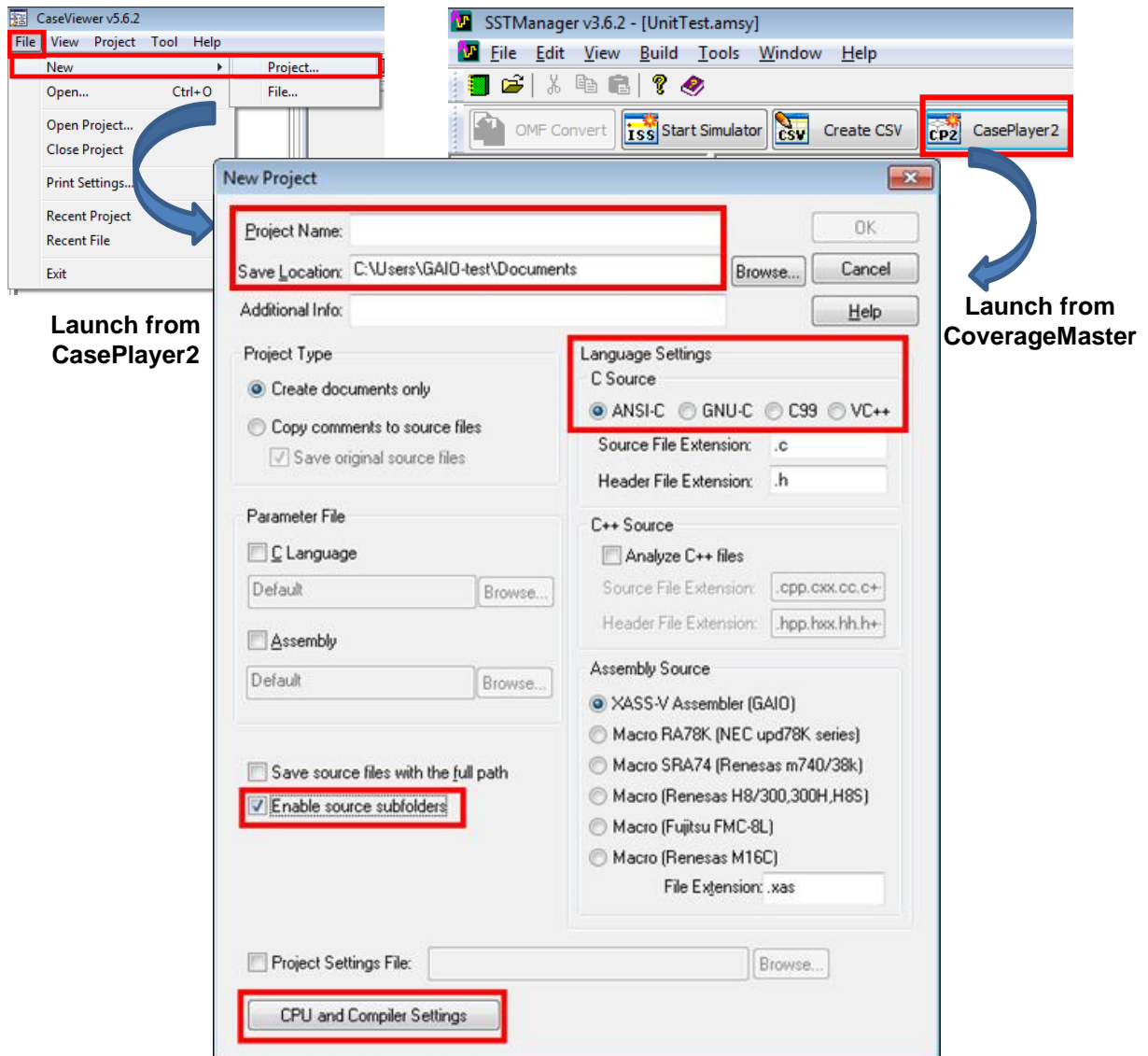
[2.5. Stub Settings](#)

CoverageMaster setup is complete when unit test results and coverage results can be obtained for a sample test function.

In an error occurs, verify and make necessary changes to the 2.2 OMF Converter Settings, 2.3 Target Settings, 2.4 Test Settings and/or 2.5 Stub Settings, then run the test again until the error has been resolved.

CasePlayer2 Setup

1. Project Settings



Launch from
CasePlayer2

Launch from
CoverageMaster

1.1. Project Name, Location

Enter a name for the project here. A new folder with the project name will be created at the selected location to store the project file.

1.2. Language Settings – C Source

Select the setting that most closely matches that of the source code. CasePlayer2 will perform analysis of the source code according to this setting.

- An extended language option is required for settings other than “ANSI-C”.
- The MISRA-C checker can only be used when this setting is set to “ANSI-C”.
- If included header files are “GNU-C” or “C99” it is recommended to use that setting even if the source code is “ANSI-C”.

1.3. Enable source subfolders

This setting is recommended for users who wish to use subfolders to organize source files added to the CasePlayer2 project. When enabled, the user can easily add source files to CasePlayer2 using the same subfolder name and structure of the existing source file subfolder structure. Note that this setting cannot be changed after project creation.

1.4. CPU and Compiler Settings

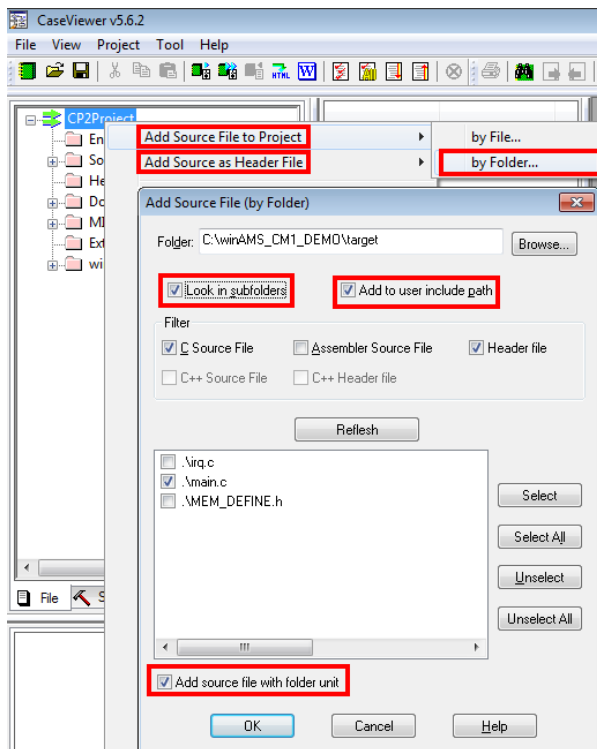
An analysis error can occur if compiler specific descriptions are included in the source code. These compiler specific descriptions must be added to the “C Option Parameter” – “User Keyword” settings in order to be properly recognized by CasePlayer2. By selecting your CPU and compiler combination using this setting, it is possible to automatically add known descriptions for the selected combination to the CasePlayer2 “User Keyword” section.

Note: this setting need not be changed when CasePlayer2 is linked with CoverageMaster because the setting will be automatically set to match the CPU and Compiler set in the CoverageMaster project.

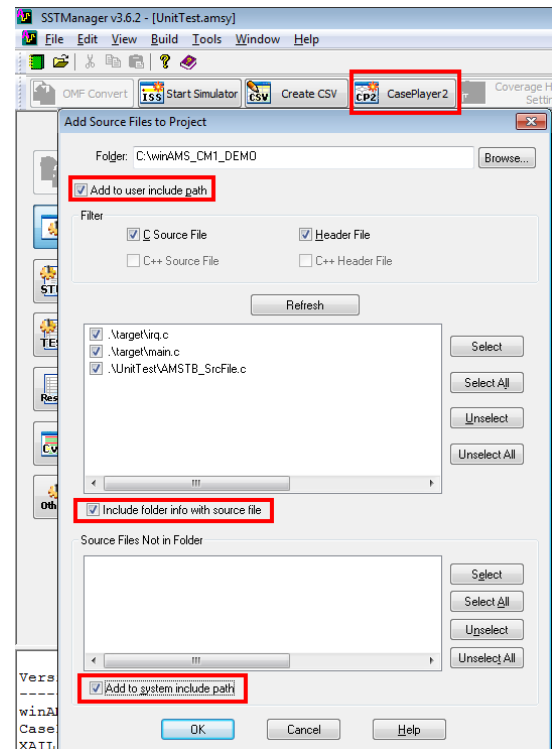
Related FAQ: 019_07: I tried to analyze the registered source files. However, the ‘CasePlayer2-E-SYN : ; expected.’ error message is displayed.

http://www.gaiotech.com/support/user/faq/winams/faq_019_07.html

2. Add Source Files



**Launch from
CasePlayer2**



**Launch from
CoverageMaster**

2.1. Add Source File to Project

Source files can be added to CasePlayer2 individually by file or by folder. It is sufficient for unit testing to add only the source files that contain the functions to be tested. Furthermore, it is not necessary to add the header files included by the source files, but the location of the included header files must be registered (explained later in this document).

Tips: The difference in analysis results between adding and not adding header files

The Global Variable List, typedef List, #define List and Include Relation Diagram will be output even without adding header files to the CasePlayer2 project.

For functions/variables declared in header files, the header files must be added to the CasePlayer2 project in order to output the Flowchart, Module Specification, Module Structure Diagram, Structure Specification, Module List, Structure List, Global Variable Reference List, Function Reference List, Structure Reference List and Member Reference List for these items.

*In summary, for CoverageMaster winAMS unit test purposes, not adding the header files can reduce CasePlayer2 analysis time. But for CasePlayer2 document creation purposes adding the header files is recommended.

2.2. Look in subfolders

Enable this option in order to also look for source file types located in subfolders of the selected folder.

2.3. Add to user include path / Add to system include path

In order to analyze header files included in the source files, the location (folder path) of the header file must be registered. Instead of manually adding the locations later, enable these setting to automatically register header file locations when adding header files to the project.

Note: adding a large number of include paths can increase analysis times so it is recommended to remove unneeded path registrations.

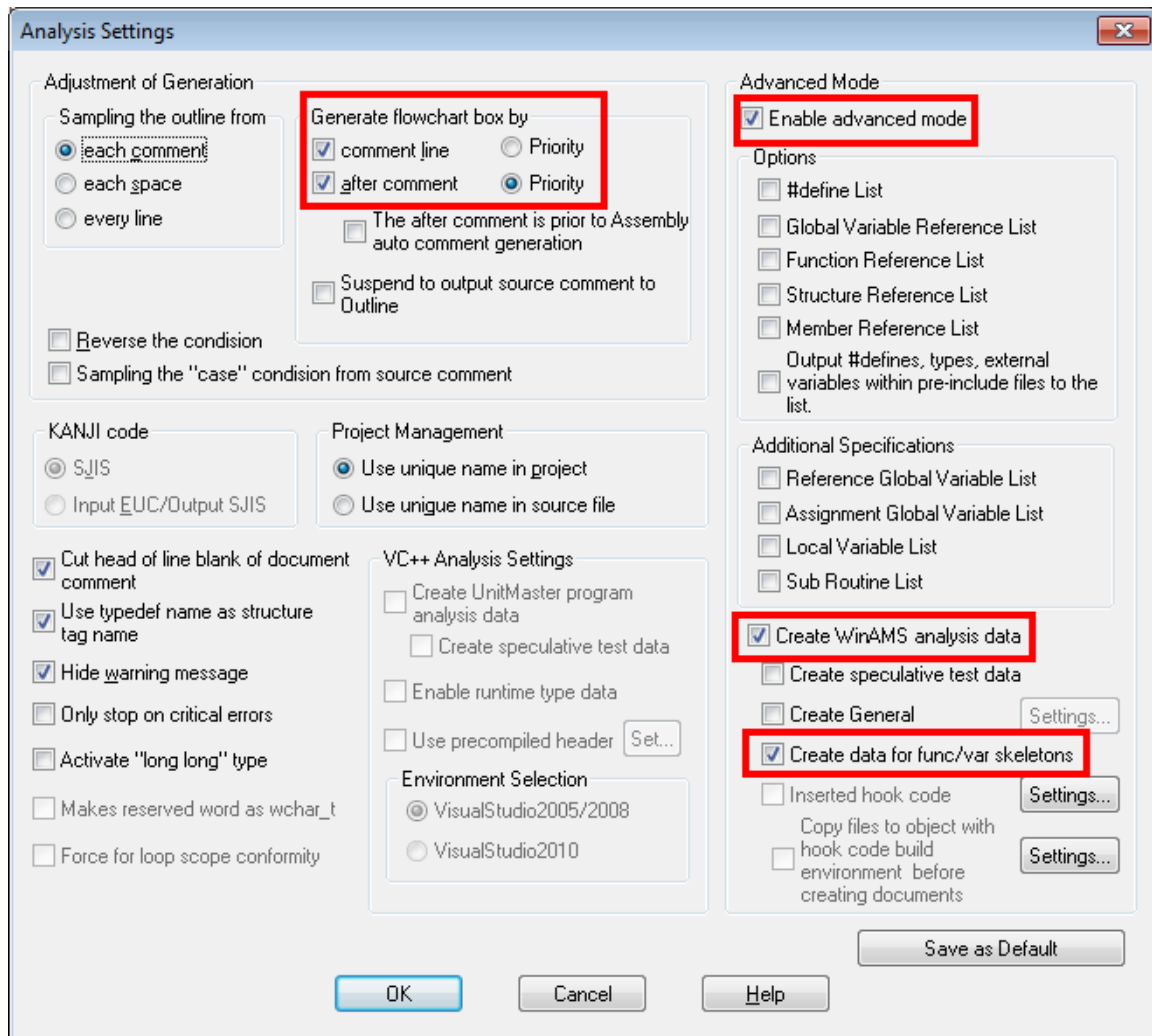
2.4. Add source file with folder unit

Enable this option in order to maintain the same folder structure when adding source files to CasePlayer2. In this way, subfolders of the same name as your source file subfolders will be created underneath the CasePlayer2 projects "Source File" folder.

2.5. Add Source as Header File

Depending on the application, some source files may include other C source files like header files. In such cases use this option to add these included C source files as header files.

3. Analysis Settings



3.1. Generate flowchart box by

Enable these options in order to extract comments from the source files and output to the generated program documents. Special CasePlayer2 format comments are not needed, standard comments in `/* comment */` and `// comment` format will be extracted.

3.2. Enable advanced mode, Create WinAMS analysis data

These settings are necessary for generating analysis data to be used with CoverageMaster winAMS. Enable both settings.

Related FAQ D01_01: How should I set up CasePlayer2 to use with CoverageMaster winAMS?

http://www.gaio.com/support/user/faq/winams/faq_D01_01.html

3.3. Create data for func/var skeletons (optional)

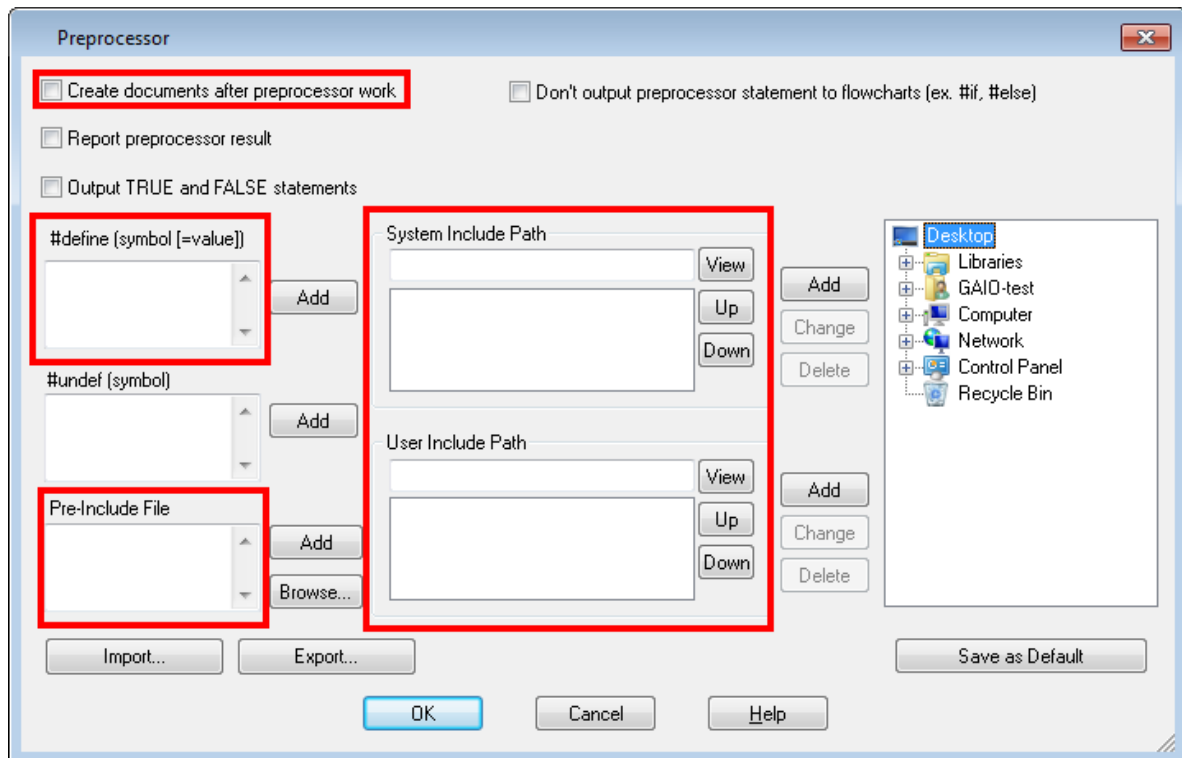
When enabled, this setting will detect undefined functions and variables during analysis. Using this analysis data the [“Create Undefined Function Skeleton...”](#) feature can be used to easily create skeletons for undefined functions or variables.

Tips: CoverageMaster linked Stub Creation

With CoverageMaster it is possible to auto generate stub functions. However, this is only normally possible for functions defined in the source code. For system calls and libraries, stub functions cannot be auto generated unless the “Create undefined function skeleton” feature is used.

Note: the source file containing the function/variable skeletons must be added to the development environment and build the object with the target application. (Depending on the development environment, the library link may need to be removed to avoid a double definition error.) The same source file can be used for undefined function skeletons and CoverageMaster stub functions.

4. Preprocessor Settings



4.1. Create documents after preprocessor work

Enabling this setting will create program documents after preprocessor expansion. For example, with a function definition like below based on the AUTOSAR standard, this setting must be enabled in order to properly output the function name to program documents.

```
#define FUNC(rettype, memclass) rettype
FUNC(void, RTE_CODE) Rte_XXXX(void)
{
}
```

With this setting disabled, the function name will be output as “FUNC”. With this setting enabled, the function name will be correctly output as “Rte_XXXX”.

4.2. #define (symbol [= value])

Enter the preprocessor controls used during the build of the source files to be analyzed here.

Note: preprocessor controls defined within header files included by the source files are valid so they do not need to be set here.

Related FAQ D01_01: How should I set up CasePlayer2 to use with CoverageMaster winAMS?

http://www.gaio.com/support/user/faq/winams/faq_D01_01.html

4.3. Pre-Include File

Specify the pre-include file here if one exists for your development environment.

Related FAQ 019_06: Using the original Renesas Electronics cross-compiler, I get the following error when I run the CasePlayer2 Analysis: "CasePlayer2-E-SYN : name (xxx) is not declared."
http://www.gaiocom.com/support/user/faq/winams/faq_019_06.html

Related FAQ 005_01: Using the original Renesas Electronics (formerly NEC Electronics) cross-compiler, how can I use an I/O port as an Input/Output variable for my test?
http://www.gaiocom.com/support/user/faq/winams/faq_005_01.html

4.4. System Include Path, User Include Path

Enter the path to header files included by the source files here.

Add [#include <header.h>] format header files to the System Include Path section.

Add [#include "header.h"] format header files to the User Include Path section.

*Note: each subfolder where the included files reside must be set, not only the parent folder.

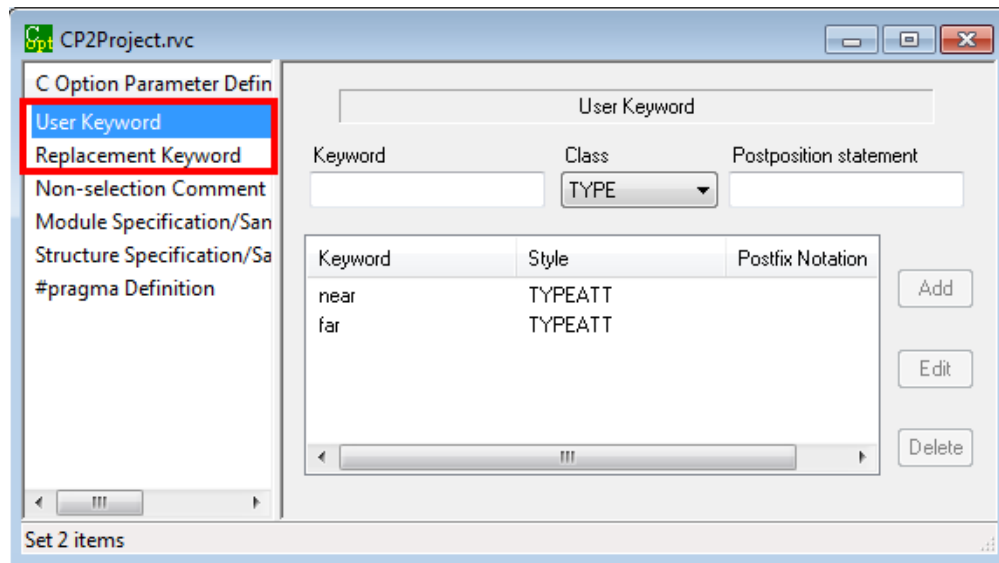
Related FAQ 019_04: The 'file i/o error' messages are output when executing the CasePlayer2's static analysis.
http://www.gaiocom.com/support/user/faq/winams/faq_019_04.html

Related FAQ 019_02: How is the retrieval sequence of CasePlayer2's include path settings?
http://www.gaiocom.com/support/user/faq/winams/faq_019_02.html

Related FAQ 019_03: Can I use a relative path to the 'user include path'?
http://www.gaiocom.com/support/user/faq/winams/faq_019_03.html

Related FAQ 019_05: The header files in the sub folder cannot be retrieved in spite of the include path setting.
http://www.gaiocom.com/support/user/faq/winams/faq_019_05.html

5. C Option Parameter Settings



5.1. User Keyword

CasePlayer2 will perform analysis according to the [1.1. Language Settings – C Source](#) setting. Compiler specific keywords not part of the selected ANSI-C, GNU-C, or C99 standard must be entered here.

***Note: an analysis error will occur for non-standard keywords not entered here.**

Related FAQ 019_05: I tried to analyze the registered source files. However the 'CasePlayer2-E-SYN : ; expected.' error message is displayed.

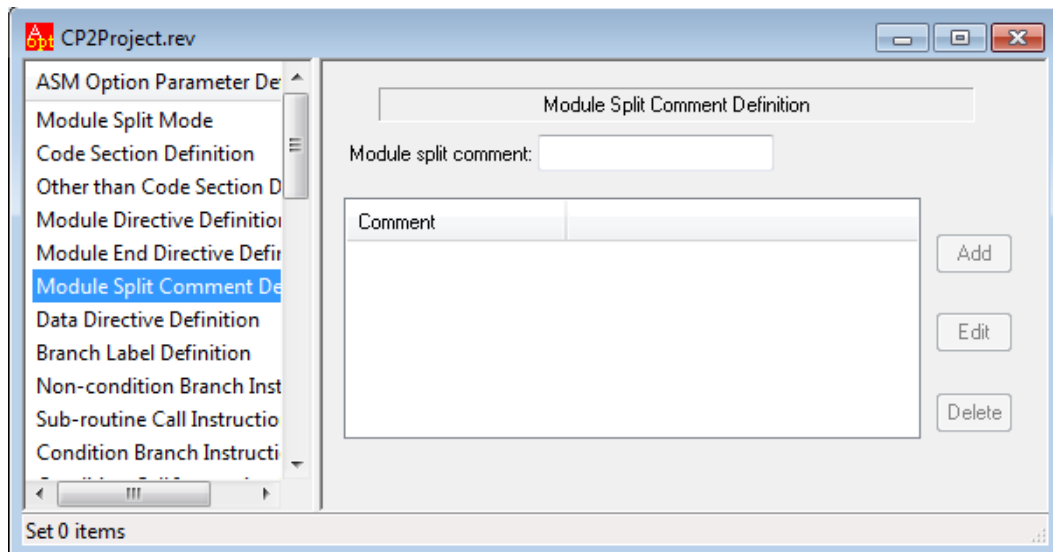
http://www.gaio.com/support/user/faq/winams/faq_019_07.html

5.2. Replacement Keyword

Use this setting to enter a "New Keyword" to be replaced by an "Existing Keyword" during analysis (the source file will not be changed).

For example, the keyword "int8" is not a valid type and will result in an analysis error. In this example int8 could be replaced with "char" or other valid type. The [User Keyword](#) setting could also be used to avoid this analysis error.

(Reference) ASM Instruction Definition Settings

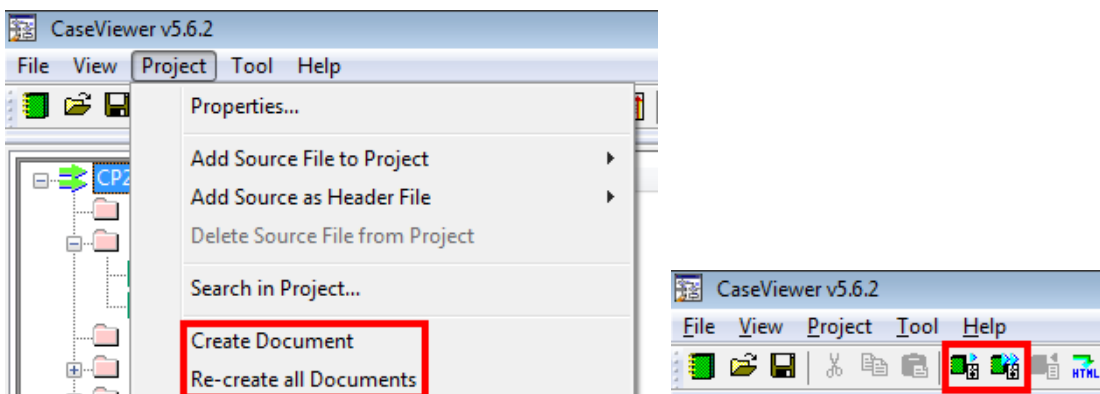


CasePlayer2 also features the ability to create program documents (flowcharts, module specifications, module structure diagrams, module lists) from assembly code. Since the assembly language commands (mnemonics) differs per MPU, use these settings to enter the assembly language commands (mnemonics).

Related FAQ 462_02: CasePlayer2 can create flowcharts and program documents from Assembly Code, but what are the supported Assembly languages?

http://www.gaio.com/support/user/faq/winams/faq_462_02.html

6. Create Documents



6.1. Create Documents

After completing all the necessary CasePlayer2 settings (from 1.Project Settings to 5.C Option Parameter Settings), click “Create Documents” to analyze source files and create program documents.

Related FAQ 019_05: I tried to analyze the registered source files. However the ‘CasePlayer2-E-SYN : ; expected.’ error message is displayed.

http://www.gaio.com/support/user/faq/winams/faq_019_07.html

Tips: Create Documents vs Re-create all Documents

Create Documents will analyze source files updated since the last analysis. Re-create all Documents will analyze all source files registered in the project.

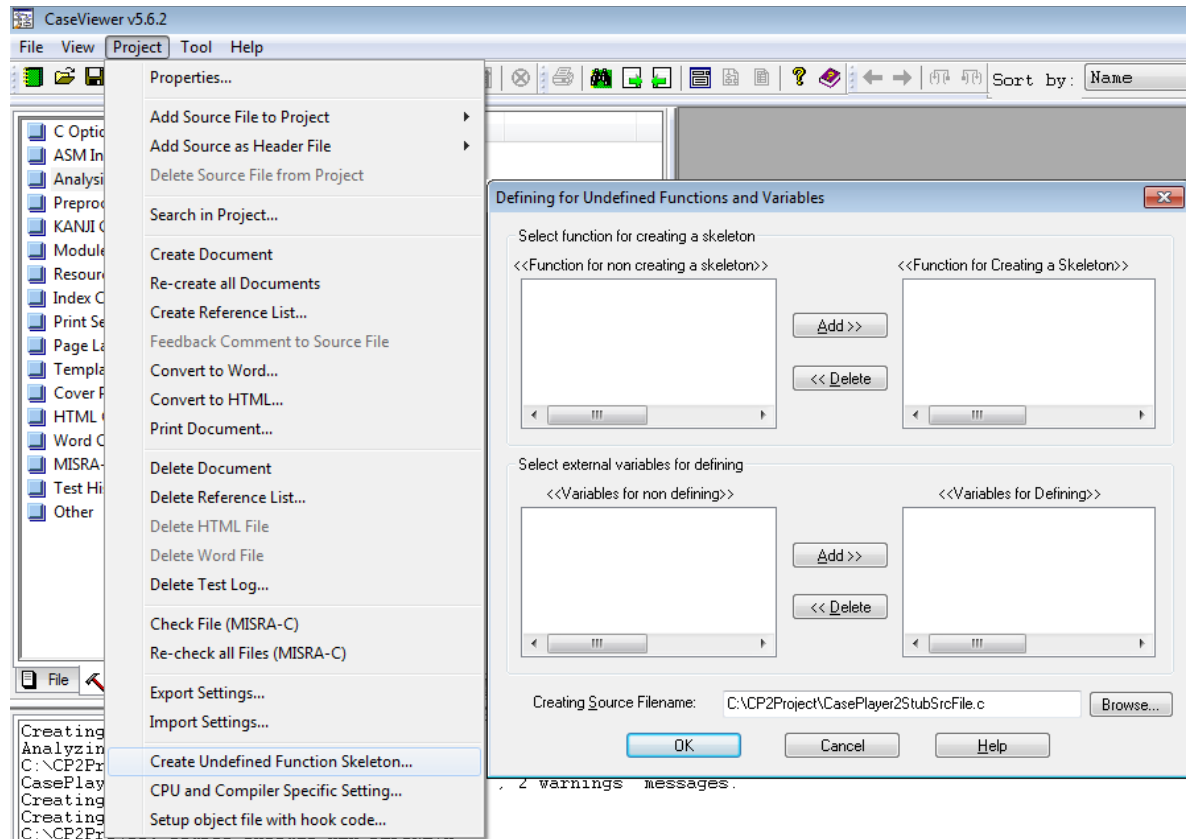
*Note: If analysis settings are changed, all source files will be re-analyzed even if the Create Documents option is selected.

The following [7. Create Undefined Function Skeleton](#) and [8. Build Object File](#) sections are not directly related to setting up CasePlayer2. However, these sections may be useful for users also setting up CoverageMaster winAMS.

7. Create Undefined Function Skeleton

7.1. Create Undefined Function Skeleton

After enabling the [Create data for func/var skeletons](#) setting, skeletons for undefined functions or variables can be automatically generated using this feature.



8. Build Object File

8.1. Build Object File

After using the [Create Undefined Function Skeleton](#) feature, the next step is to add the source file created with the generated function / variable skeletons to your build environment and build the object file. This will enable CoverageMaster to access these functions / variables during testing.

*Note: the object file must be compiled and linked with debug information.

Tips:

In a later step stub functions will also be generated using CoverageMaster and require an object file build. As such, building the object file may be performed at that time instead in order to avoid additional builds.

(Reference) Other Relevant FAQs

462_01: Do I have to match the version of both CoverageMaster (SSTManager) and CasePlayer2?

http://www.gaiotech.com/support/user/faq/winams/faq_462_01.html

C01_01: The execution error message 'error code 5941' appears when generating Word files.

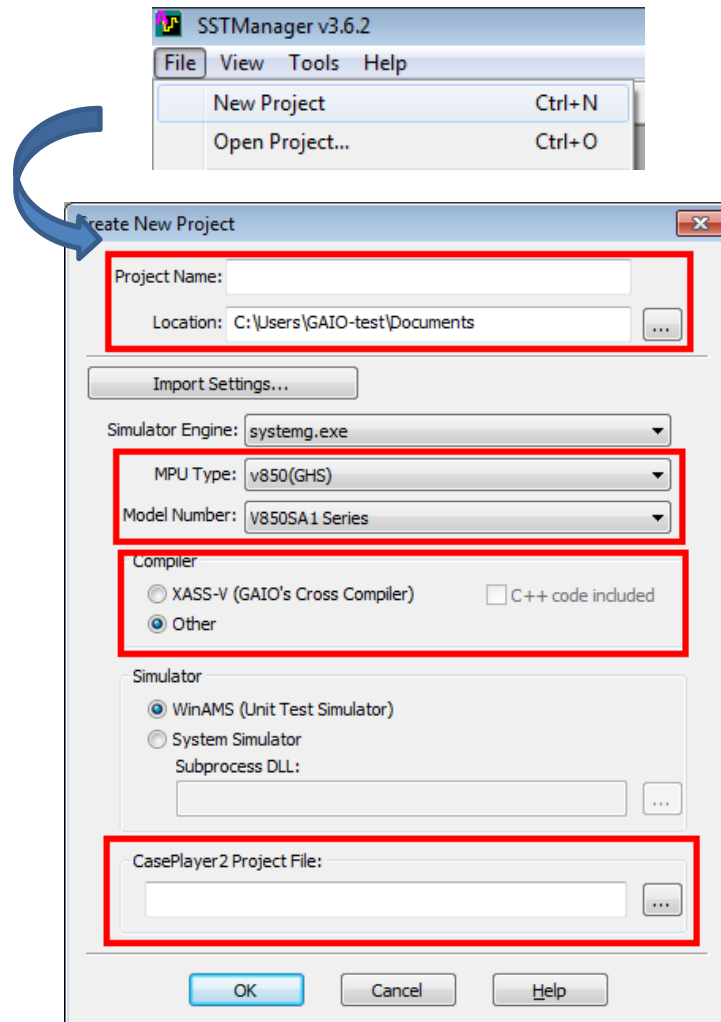
http://www.gaiotech.com/support/user/faq/winams/faq_C01_01.html

C01_02: The execution error message "<Template List> *****.dot" appears when generating Word files.

http://www.gaiotech.com/support/user/faq/winams/faq_C01_02.html

CoverageMaster winAMS Setup

9. Project Settings



9.1. Project Name, Location

Enter a name for the project here. A new folder with the project name will be created at the selected location to store the project file.

9.2. MPU Type

Select the MPU type (MPU core) here. When using a cross compiler other than GAIO's cross compiler, the compiler name will be indicated in parenthesis ().

Example:

- Select "v850" when using the v850 MPU core with GAIO's cross compiler.
- Select "v850(GHS)" when using the v850 MPU core with the GreenHills cross compiler.

9.3. Model Number

Select the MPU model number (series) when applicable. In some cases, compile mode and/or endian selections may be available.

*Note: if your MPU model number/series is not available, select a model/series that uses the same MPU core.

Related FAQ 017_05: I cannot find my MPU type in the list of 'Test Project Settings' dialog.
http://www.gaio.com/support/user/faq/winams/faq_017_05.html

9.4. Compiler

If you are using GAIO's cross compiler select "XASS-V". For other cross compilers select "Other".

Related FAQ 017_01: The 'OMF Convert' button is disabled. How can I use it?
http://www.gaio.com/support/user/faq/winams/faq_017_01.html

Tips: OMF Converter

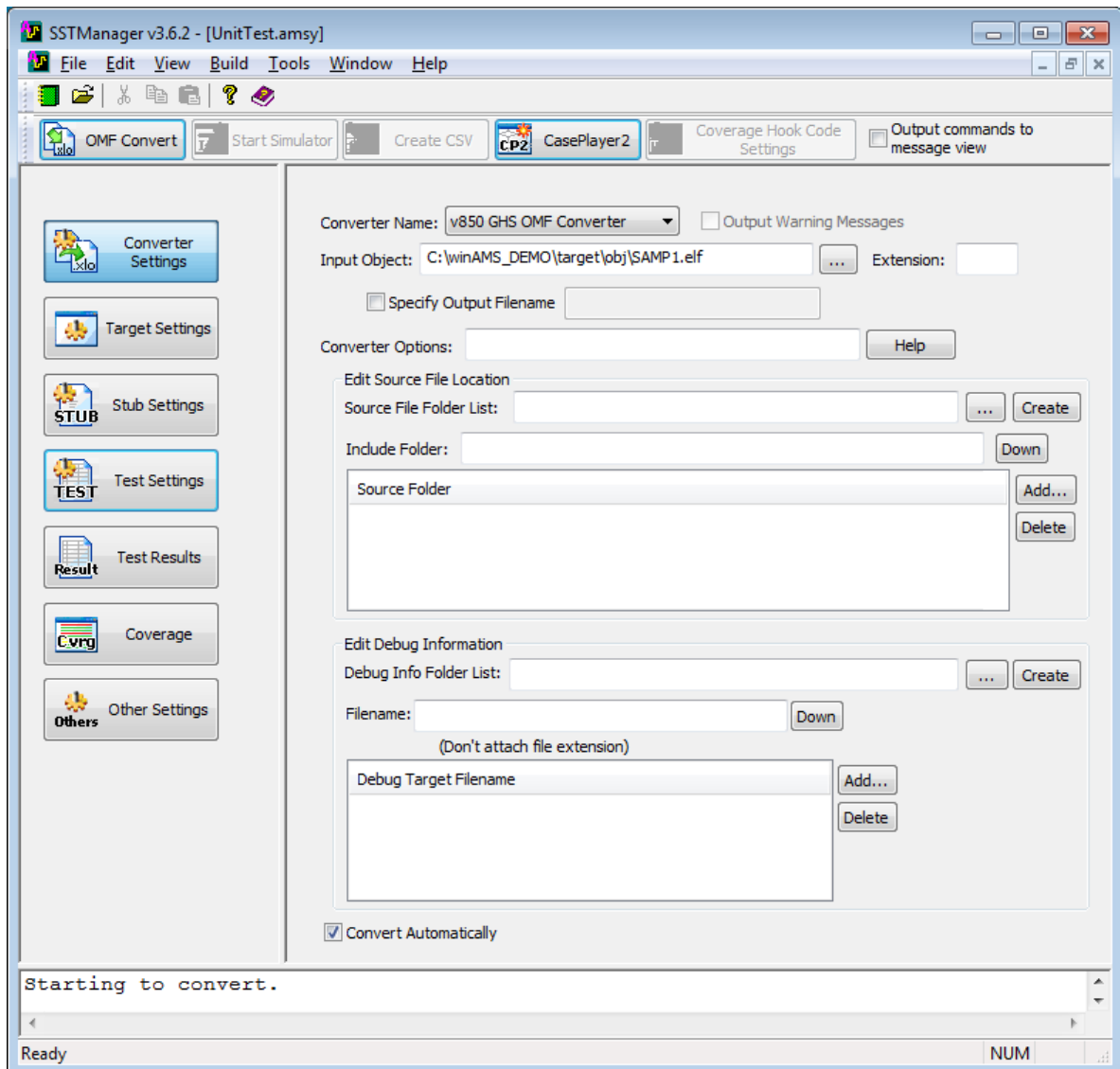
The OMF Converter only need be used when the object file was compiled with a compiler other than GAIO's cross compiler.

9.5. CasePlayer2 Project File

Set the CasePlayer2 project file here to link it with the CoverageMaster project.

Related FAQ 462_01: Do I have to match the version of both CoverageMaster (SSTManager) and CasePlayer2?
http://www.gaio.com/support/user/faq/winams/faq_462_01.html

10. Converter Settings



10.1. What is the OMF Converter?

The object file format must be in GAIO's native *.xlo format in order to run the object file with GAIO's MPU simulator environment. Therefore, when using an object file compiled by a compiler other than GAIO's cross compiler, the OMF converter is used to convert the file format to an *.xlo format file.

*Note: when using GAIO's cross compiler the OMF converter is unnecessary and will be greyed out.

Related FAQ 011_03: What initial settings will be required regarding the object code in order to start unit test using CoverageMaster?

http://www.gaio.com/support/user/faq/winams/faq_011_03.html

Related FAQ 017_01: The 'OMF Convert' button is disabled. How can I use it?

http://www.gaio.com/support/user/faq/winams/faq_017_01.html

10.2. Converter Name

The OMF Converter is set according to the MPU(compiler) set in the Project Settings. In case multiple selections are available use this setting to select the OMF Converter.

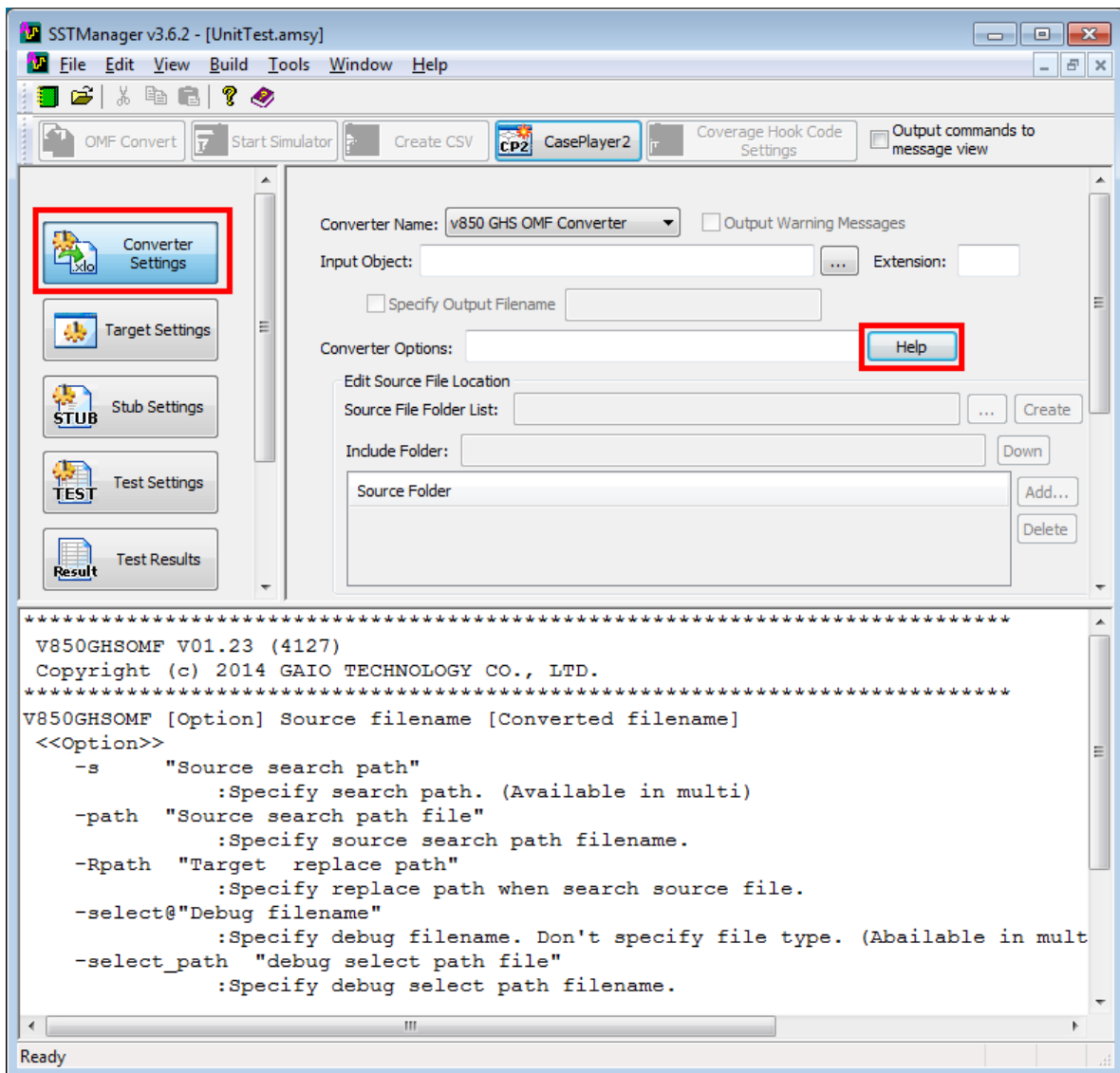
10.3. Input Object

Set the object file built using your cross compiler here.

*The object file must be compiled and linked and the object code executable. In addition, debug information must be included in the object file. Therefore, it is necessary to enable the option in your development environment to output debug information to the object file.

10.4. Converter Options

Set OMF Converter options here. The OMF converter options available for the selected OMF converter can be output to the message view by clicking the "Help" button.



Tips: “-funcinfo” option

Debug information is not correctly output for function arguments with the M16C, R8C/Tiny, R32C/100, MPC83xx(e300) MPU cores. In order to properly perform the OMF conversion in such cases the “-funcinfo” option should be used. The CasePlayer2 generated analysis file “CP2Project.arg” can be used to provide the required function information.

Example:

```
-funcinfo C:\winAMS_DEMO\CP2Project\CP2Project.arg
```

10.5. Edit Source File Location

Source file path information is recorded in the object file at the time of build. If the object file or source files are moved after building, the path information may no longer match. In such cases these settings can be used to designate the location of the source files. This can be done using either of the following two converter options.

```
-s “source search path”  
-path “source search path file”
```

10.6. Edit Debug Information

It may take longer to perform the OMF conversion and test the converted object file if the input object is of a large size. In such cases these settings can be used to reduce the debug information to only what is necessary for the functions to be tested in order to improve performance.

Tips: reducing debug information

Testing may not perform correctly if there is insufficient debug information.

For example, consider the following situation:

Source File A: the source file where the test function exists.

Source File B: the source file where external variable definitions exist that are accessed by the test function.

In this case if only source file A is selected, then test data cannot be assigned to external variables defined in source file B. Therefore, debug information need be present for both source files.

Debug information needed for testing:

1. Test function
Debug information of the source file where the test function is defined.
2. Initialization End Address
If for example, the initialization end address is set to function main, then debug information of the source file where main is defined.

*Note: this is not needed if the initialization end address is set by address in hexadecimal
3. Variables accessed by the test function (when used for test data)
Debug information of the source files for external variables accessed by the test function.
4. Functions called by the test function
If the test function includes calls to other functions, then the debug information of the source files where the functions called are defined.
5. Stub functions
If stub functions are used, then the debug information of the source files where the stub functions are defined.

10.7. Convert Automatically

Enable this option to automatically convert the input object file. The conversion will be automatically performed when the timestamp of the input object file is newer than the converted *.xlo object file.

10.8. OMF Convert

Click this button to start the OMF conversion. The conversion results will be displayed in the message view at the lower part of the window.

Tips: OMF converter error and warning messages

If an error occurs during the conversion, changes to the Converter Settings, or in some cases changes to the compiler options during build need to be performed.

In many cases the warning messages shown do not affect testing and can be ignored.

Related FAQ 017_02: How to hide the warning messages when converting OMF?

http://www.gaio.com/support/user/faq/winams/faq_017_02.html

Related FAQ 017_03: The OMF conversion is terminated incompletely and no error message appears.

http://www.gaio.com/support/user/faq/winams/faq_017_03.html

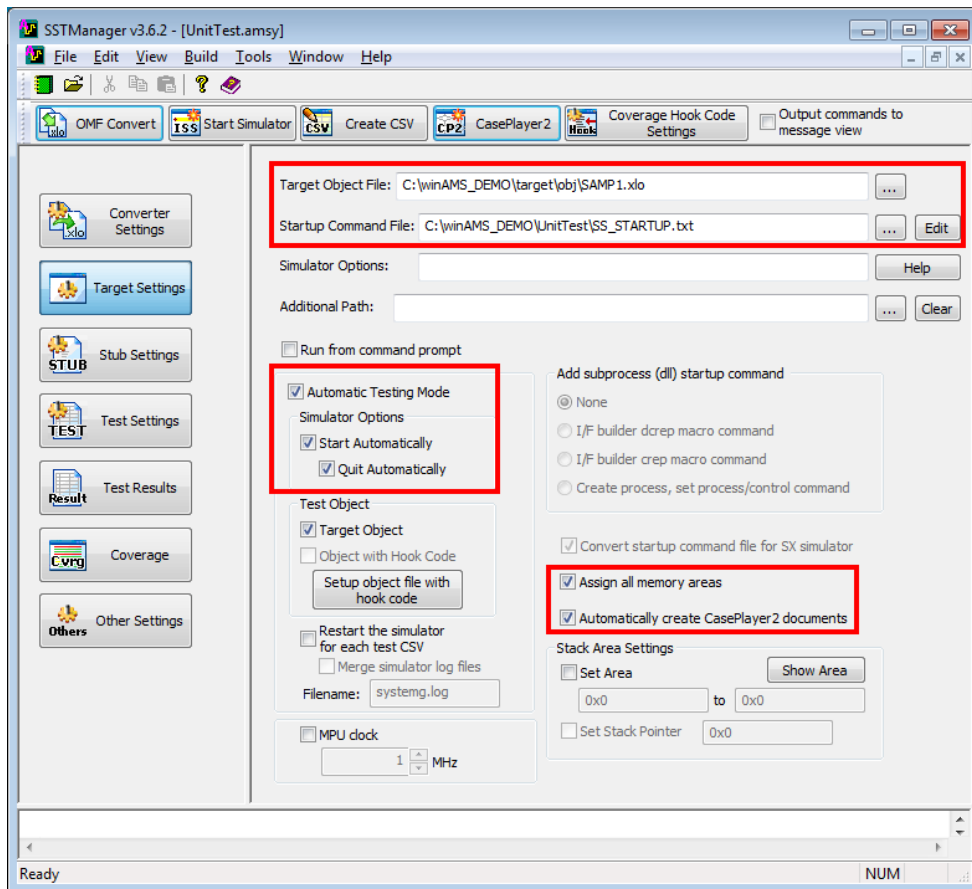
Related FAQ 017_04: I met the error message 'Failed to read the object file' while converting OMF.

http://www.gaio.com/support/user/faq/winams/faq_017_04.html

Related FAQ 017_06: Using the Renesas Compiler CubeSuite+ for RL78, I ran the OMF Converter, but when running Simulation I get the "Source file doesn't exist" error.

http://www.gaio.com/support/user/faq/winams/faq_017_06.html

11. Target Settings



11.1. Target Object File

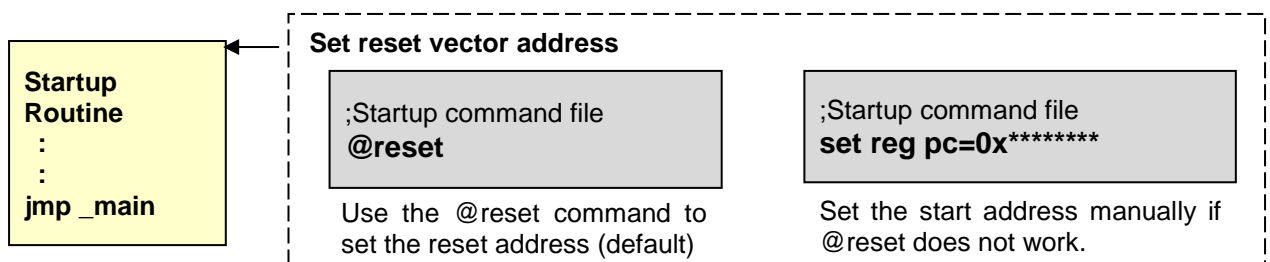
Set the target object (*.xlo) file here. After the OMF conversion the converted object file will be automatically set.

11.2. Startup Command File

The startup command file is a script file for the MPU simulator used to make settings to the simulator environment. The same commands usable in the simulator (debugger) can be set in the startup command file. A startup command file with the default commands will be created for new projects.

An important setting set in the startup command file is the MPU start address. This is explained further in the diagram below.

Reset vector (program start location) setting



Other features include:

Assign memory areas for access

Example: assign/read/write/execute 0x*****

Modify variables to break from loops

The simulator will run starting from the reset vector set in the startup command file until reaching the function or address set as the initialization end address. Since hardware modules are not simulated, it is possible an infinite loop may be encountered during this time as the program waits for the hardware status. In such cases commands can be entered into the startup command file to break from the loop.

Related FAQ 011_03: What initial settings will be required regarding the object code in order to start unit test using CoverageMaster?

http://www.gaio.com/support/user/faq/winams/faq_011_03.html

Related FAQ 000_01: The MPU simulator stops while performing unit test. How can I examine its cause?

http://www.gaio.com/support/user/faq/winams/faq_000_01.html

Related FAQ 326_01: How to handle optional address data with Simulator macro command?

http://www.gaio.com/support/user/faq/winams/faq_326_01.html

11.3. Automatic Testing Mode, Start Automatically, Quit Automatically

The “Automatic Testing Mode” must be enabled for unit testing.

Enable the “Start Automatically” setting to automatically start simulation after the MPU simulator has been launched.

Enable the “Quit Automatically” setting to automatically quit the simulator after the simulation has completed.

Tips: Simulator modes

If both the “Start Automatically” and “Quit Automatically” options are set, the simulator will run automatically in the background using the lite (LiX) simulator. If either option is disabled, the simulator (debugger) UI will be displayed during execution. In general the lite simulator is run in the background for quicker test performance. However, the simulator UI is useful for initial testing during setup and for debugging.

11.4. Assign all memory areas

Enable this setting to assign memory to all available areas for read, write and execution during simulation. This can be useful for resolving memory assignment or memory protection related simulation errors.

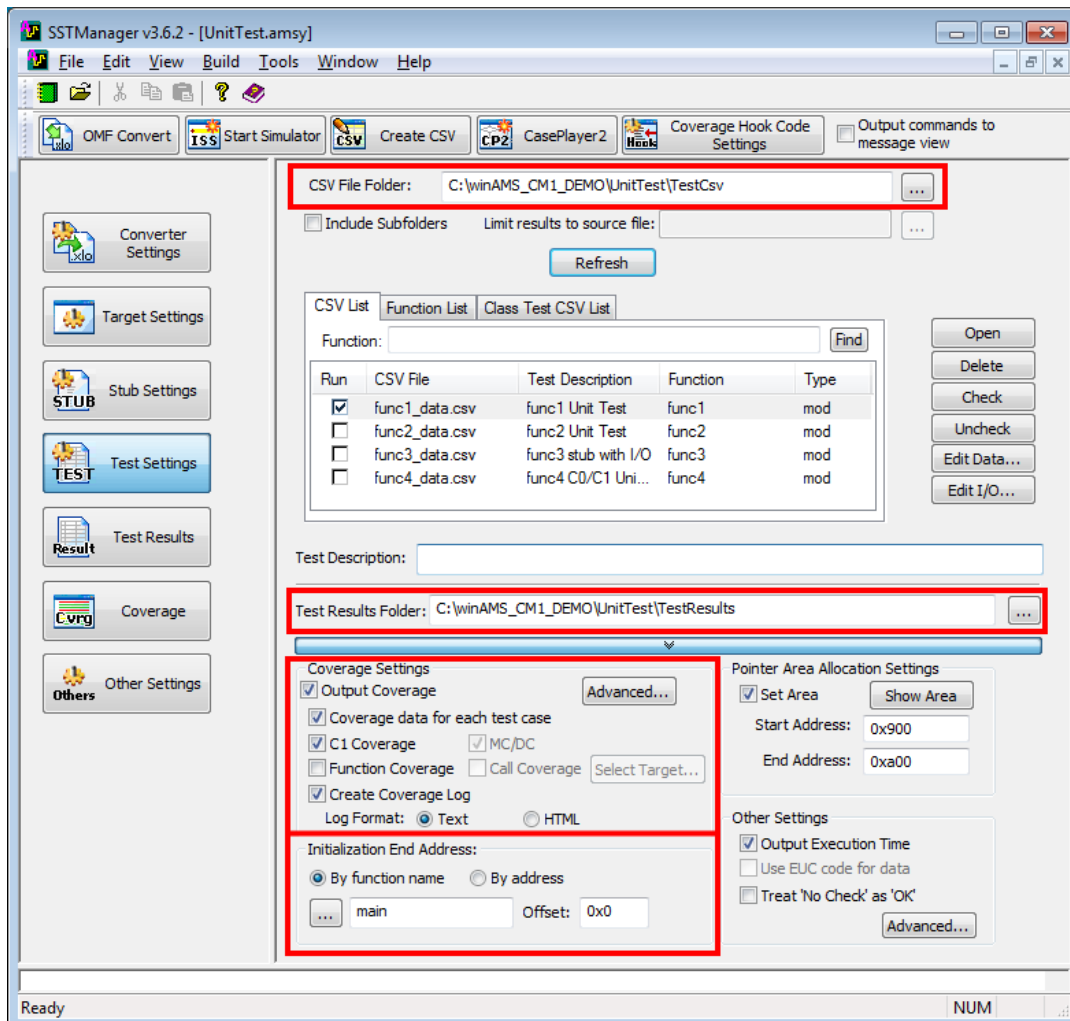
Tips: assigning memory areas the same as the MPU

When the “Assign all memory areas” setting is disabled, only the memory areas statically allocated within the target object file are assigned. Reserved MPU memory areas such as SFR however, will not be assigned. In this case these memory areas must be assigned using the “Assign” command in the startup command file.

11.5. Automatically create CasePlayer2 documents

Enable this setting to automatically perform the CasePlayer2 analysis when creating test CSV files or editing test data values if the source files have been modified since the last analysis was performed.

12. Test Settings



12.1. CSV File Folder

Select the folder to save test CSV files to. Newly created test CSV files will be saved to this folder.

Tips: Modifying CSV files

Test CSV files used by CoverageMaster are open format files. In this way test CSV files can be modified / created manually and placed in the CSV file folder for use. Click the "Refresh" button in order to update the CSV file list.

12.2. Test Results Folder

Select the folder to save test results to. Test result CSV files will be saved to this folder with the same name as the test CSV file that was run.

Tips: Test results folder

In addition to test result CSV files, test reports and coverage log files are output to the test results folder. The most recent test will overwrite previous tests. If you wish to retain previous test results, backup or rename the test results folder, or change the "Test Results Folder" to output results to a different folder.

12.3. Coverage Settings

Enable coverage measurement related settings here. The “Output Coverage”, “Coverage data for each test case”, “C1 Coverage” and “Create Coverage Log” are the standard recommended settings to be enabled.

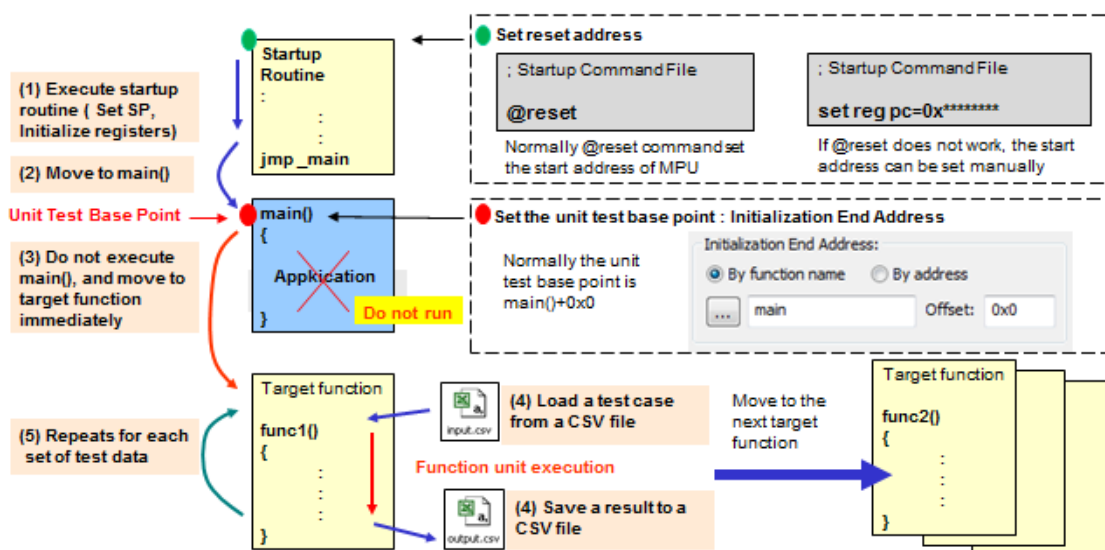
12.4. Initialization End Address (Important!)

During simulation, the program will run starting with the reset vector set in the startup command file up until reaching the initialization end address (or function) set here. Then the unit test function will be called for testing.

In order to perform unit tests the initialization of the target (such as setting the stack pointer, initializing register values) must first be performed. For this purpose CoverageMaster is setup to run your targets initialization procedure prior to running unit tests.

The initialization end address can be set by function name, or by address (or address label).

The diagram below further outlines the simulator execution procedure.



Tips: where should I set the initialization end address?

The reset address set in the startup command file and the initialization end address setting depend on the structure of your application. In some cases such as with hardware dependencies or OS calls it may not be possible to run the entire initialization procedure of your application. In these cases the initialization end address may need to be set to an earlier point. The system simulator (debugger) can be used here to perform step executions to locate an appropriate point to set the initialization end address to.

If an appropriate initialization end address cannot be found, it can be set to a point immediately after the reset address set in the startup command file, and the necessary stack and register initial values can be manually entered into the startup command file. Consult your targets architecture manual for stack and register initial values.

Related FAQ 011_01: What is the 'Offset' parameter at the 'Initialization End Address' setting?

http://www.gaio.com/support/user/faq/winams/faq_011_01.html

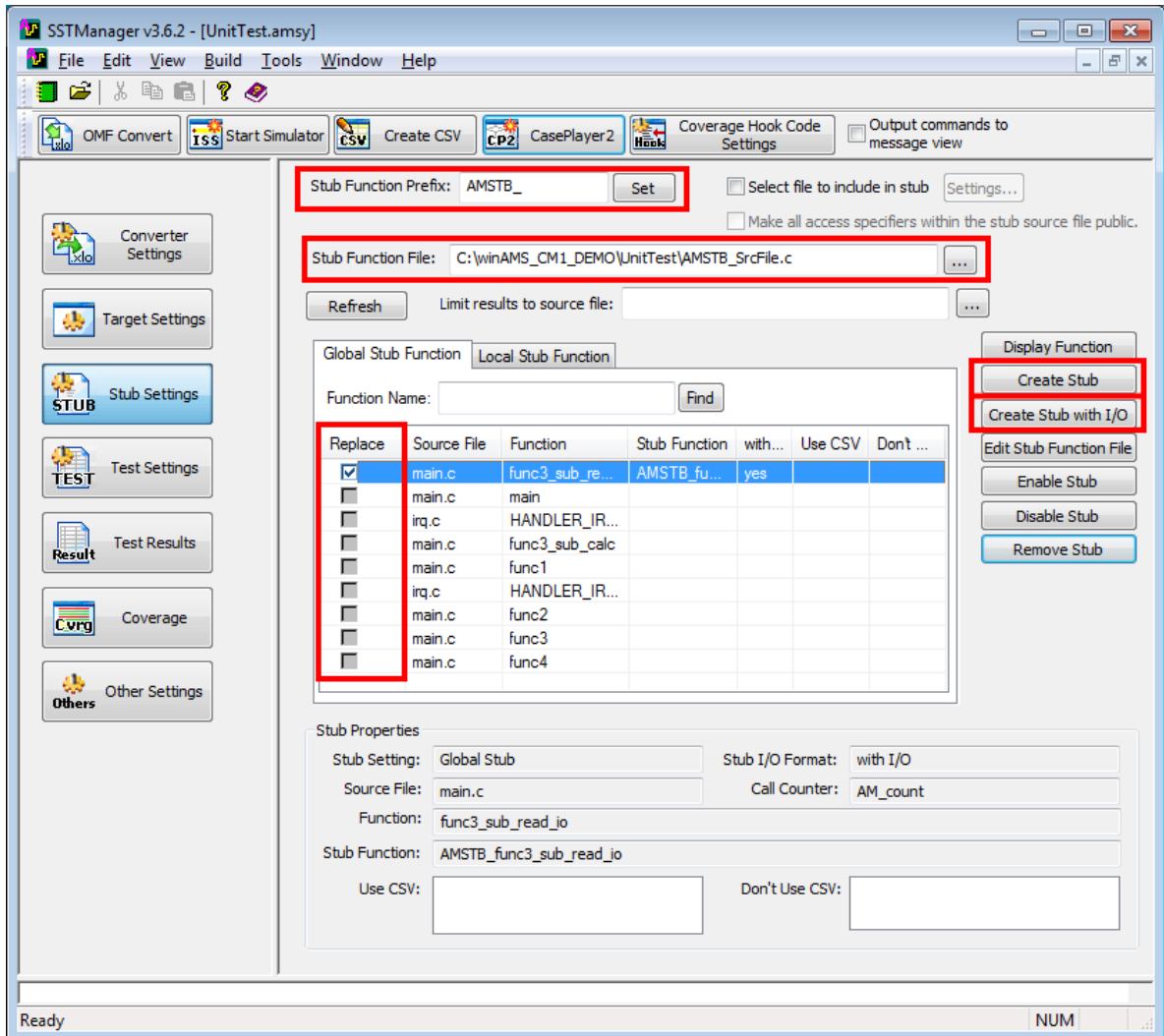
Related FAQ 011_02: What is 'Initialization End Address' in the Test Setting view?

http://www.gaio.com/support/user/faq/winams/faq_011_02.html

Related FAQ 011_03: What initial settings will be required regarding the object code in order to start unit test using CoverageMaster?

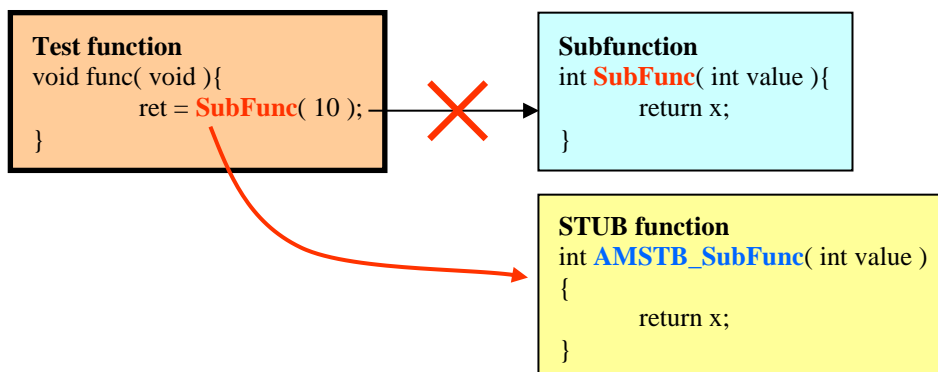
http://www.gaio.com/support/user/faq/winams/faq_011_03.html

13. Stub Settings



13.1. What are Stub Functions?

Stub functions are functions used to replace function calls (subfunctions) within the test function. Stub functions take on the same interface as the function it will replace and allow the user to control the value returned by the function called.



Stub functions can be created as **Global Stub Functions** or **Local Stub Functions**. In most cases it is recommended to create global stub functions.

Stub functions once created can then be applied globally to replace all function calls with the stub function for all tests. Or test CSV files can be set individually to use or not use stub functions.

13.2. Stub Function Prefix

In order for stub functions to be usable during testing the stub source file must be added to the development environment and the object file built. To avoid multiple definition errors, the prefix set here will be applied to the stub function name. The default prefix is "AMSTB_".

13.3. Stub Function File

The stub function source file where the stub functions will be created. This can be modified to meet the needs of your environment if needed.

13.4. Create Stub and Create Stub with I/O

"Create Stub" will create an empty stub function template. The user will then need to enter the stub function procedure by hand.

"Create Stub with I/O" will create the stub function and automatically create Input/Output variables and procedure that imitates the original function. The stub function can then be used as is, or modified as desired by the user.

Create Stub with I/O Example

```
int AMSTB_subfunc( int index )
{
    static int volatile AMIN_return;
    static int volatile AMOUT_index;

    AMOUT_index = index;
    return AMIN_return;
}
```

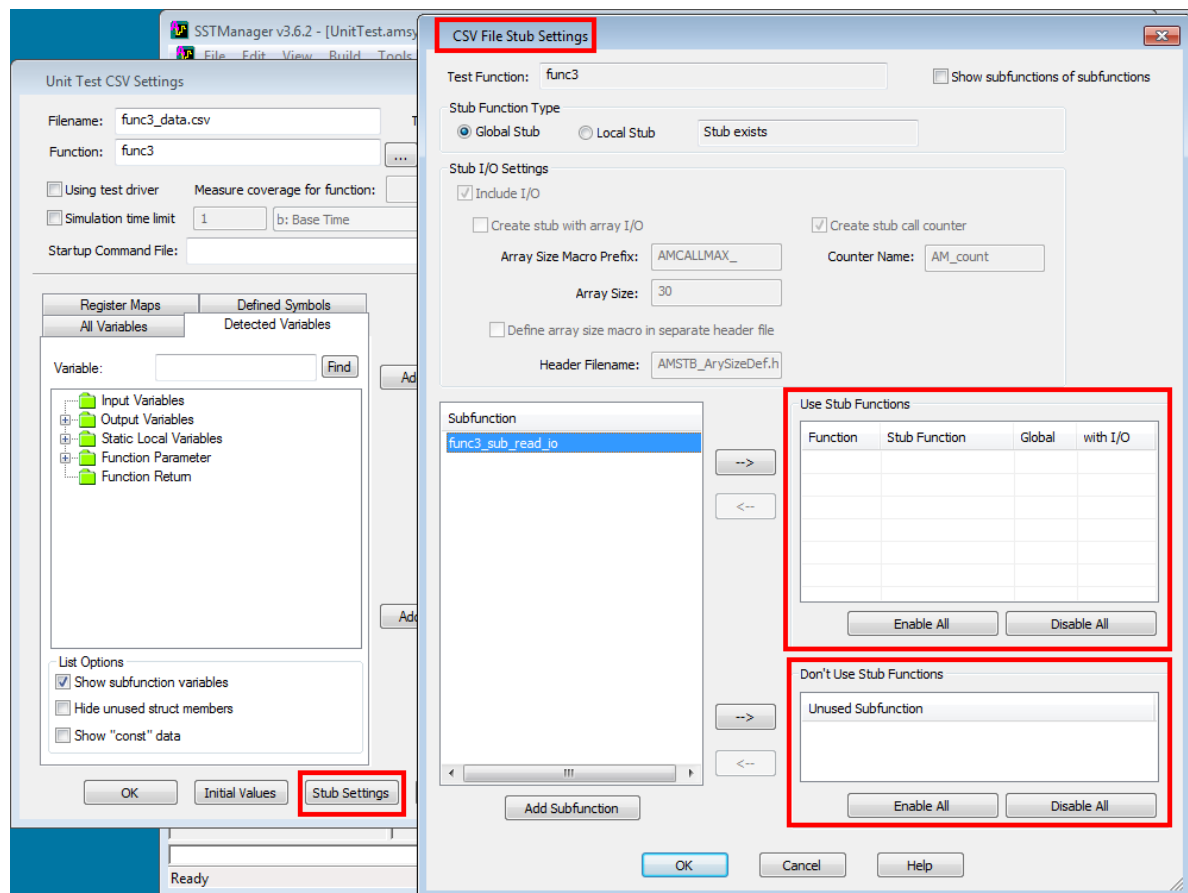
Create Stub Example

```
int AMSTB_subfunc( int index )
{

}
```

13.5. Replace checkbox

Check the checkbox in the "Replace" column of the "Global Stub Function" tab in order to replace all function calls for the selected function with the stub function.



13.6. CSV File Stub Settings

From the Test CSV creation dialog click the “Stub Settings” button to open the “CSV File Stub Settings” dialog. From this dialog stub functions can be assigned for use or not use when testing the CSV file. In addition, if the stub function has not already been created, it will be created after applying settings in this window by clicking the “OK” button.

13.7. Use Stub Functions

Select a subfunction in the list, then add it to the “Use Stub Functions” list in order to replace the selected subfunction with a stub function when testing this CSV file.

13.8. Don't Use Stub Functions

Select a subfunction in the list, then add it to the “Don't Use Stub Functions” list in order to cancel a globally set stub function and not use the stub function when testing this CSV file.

Related FAQ 102_09: How to create a stub for an inline function or macro function?
http://www.gaio.com/support/user/faq/winams/faq_102_09.html

Related FAQ 002_04: How can I create and manage stub functions?
http://www.gaio.com/support/user/faq/winams/faq_002_04.html

Related FAQ 002_05: Is it possible to create a stub function from a function written in assembly language?
http://www.gaio.com/support/user/faq/winams/faq_002_05.html

Related FAQ 002_01: How to enable/disable a stub function for each test CSV file?
http://www.gaio.com/support/user/faq/winams/faq_002_01.html

Related FAQ 002_02: The created stub function cannot be linked and executed.
http://www.gaio.com/support/user/faq/winams/faq_002_02.html

Related FAQ 002_03: How to create a stub of a sub-routine called by a function pointer?
http://www.gaio.com/support/user/faq/winams/faq_002_03.html

Related FAQ A03_01: How to create a stub function in case that the same stub function is called more than once?
http://www.gaio.com/support/user/faq/winams/faq_A03_01.html

14. Build Object File

14.1. Build Object File

After creating a [Stub Function](#) the stub function source file must be added to the development environment and built (compiler & link). This will enable CoverageMaster to use the stub functions during testing.

*Note: the object file must be compiled and linked with debug information.

Important Note:

The stub function source file is enclosed with “#ifdef WINAMS_STUB” at top and “#endif /* WINAMS_STUB */” at the bottom so it can be easily enabled / disabled during compiling. The preprocessor compiler switch “WINAMS_STUB” must be added when building the object file for testing with CoverageMaster or else the stub functions will not be built.

(Reference) Other Relevant FAQs

Related FAQ 021_01: How to test a function in a 'Loadable Module' of VxWorks?

http://www.gaio.com/support/user/faq/winams/faq_021_01.html

Related FAQ 115_01: How to use an assembly symbol as a test condition variable?

http://www.gaio.com/support/user/faq/winams/faq_115_01.html

Related FAQ 115_02: How to test the function using '#pragma inline asm' feature of Renesas compiler?

http://www.gaio.com/support/user/faq/winams/faq_115_02.html

Related FAQ 115_03: I cannot generate a Reference List for variables defined in Assembly language.

http://www.gaio.com/support/user/faq/winams/faq_115_03.html

Related FAQ 316_01: How to test an Interrupt handler?

http://www.gaio.com/support/user/faq/winams/faq_316_01.html

Related FAQ 453_01: I cannot find a static function in the 'Function List'.

http://www.gaio.com/support/user/faq/winams/faq_453_01.html

Related FAQ 102_08: How to test an inline function?

http://www.gaio.com/support/user/faq/winams/faq_102_08.html

Related FAQ 301_02: How to test a function created using an assembly language?

http://www.gaio.com/support/user/faq/winams/faq_301_02.html

CoverageMaster Setup Manual

*All company names and product names are trademarks or registered trademarks of their respective owners.

*Unauthorized reproduction of the information contained in this document is prohibited.

GAIO TECHNOLOGY CO., LTD. (JAPAN)

■User Support / Technical Information

<http://www.gaio.com/support/support.html>

■For technical support that cannot be resolved using the FAQs, send your inquiry to: user@gaio.co.jp

Please provide your user ID when requesting support.

*Note: a maintenance agreement is required for technical support.