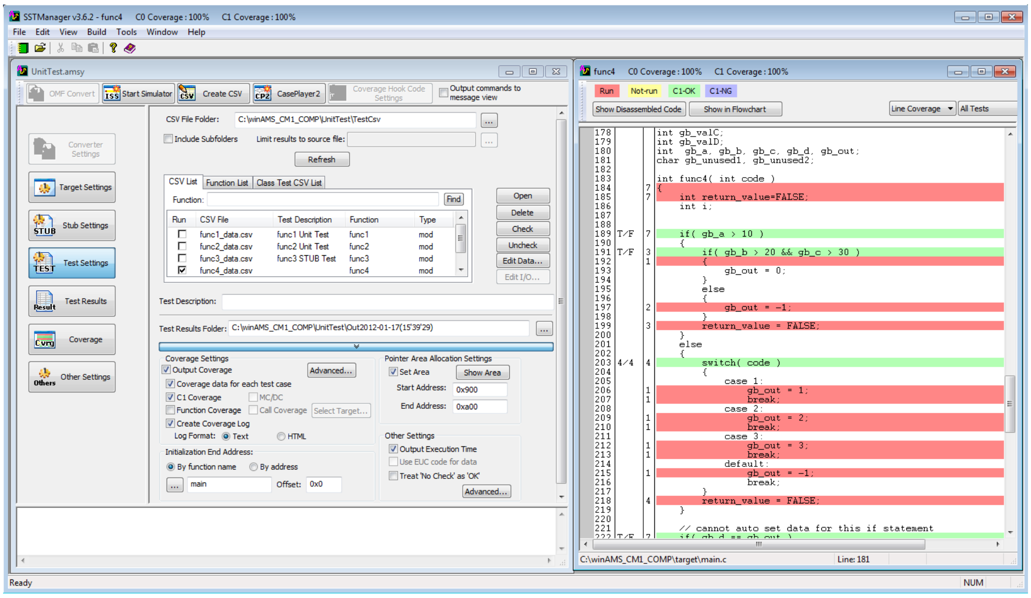
Solution House
GAIO
TECHNOLOGY



□ Output commands to message view

CSV File Folder: C:\winAMS_CM1_COMP\UnitTest\TestCsv

□ Include Subfolders Limit results to source file:

Function:

Run	CSV File	Test Description	Function	Type
□	func1_data.csv	func1 Unit Test	func1	mod
□	func2_data.csv	func2 Unit Test	func2	mod
□	func3_data.csv	func3 STUB Test	func3	mod
☑	func4_data.csv		func4	mod

Test Description:

Test Results Folder: C:\winAMS_CM1_COMP\UnitTest\Out2012-01-17(15'39'29)

Coverage Settings
- ☑ Output Coverage
- ☑ Coverage data for each test case
- ☑ C1 Coverage □ MC/DC
- □ Function Coverage □ Call Coverage
- ☑ Create Coverage Log
- Log Format: ● Text ○ HTML

Initialization End Address:
- ● By function name ○ By address
- main Offset: 0x0

Pointer Area Allocation Settings
- ☑ Set Area
- Start Address: 0x900
- End Address: 0xa00

Other Settings
- ☑ Output Execution Time
- □ Use EUC code for data
- □ Treat 'No Check' as 'OK'

```
178          int gb_valC;
179          int gb_valD;
180          int  gb_a, gb_b, gb_c, gb_d, gb_out;
181          char gb_unused1, gb_unused2;
182
183          int func4( int code )
184       7  {
185       7      int return_value=FALSE;
186              int i;
187
188
189 T/F   7      if( gb_a > 10 )
190              {
191 T/F   3          if( gb_b > 20 && gb_c > 30 )
192       1          {
193                      gb_out = 0;
194                  }
195                  else
196                  {
197       2              gb_out = -1;
198                  }
199       3          return_value = FALSE;
200              }
201              else
202              {
203 4/4   4          switch( code )
204                  {
205                      case 1:
206       1                  gb_out = 1;
207       1                  break;
208                      case 2:
209       1                  gb_out = 2;
210       1                  break;
211                      case 3:
212       1                  gb_out = 3;
213       1                  break;
214                      default:
215       1                  gb_out = -1;
216                          break;
217                  }
218       4          return_value = FALSE;
219              }
220
221              // cannot auto set data for this if statement
222 T/F   7      if( gb_d == gb_out )
```

C:\winAMS_CM1_COMP\target\main.c Line: 181

CoverageMaster Setup Manual

Version 1.0.3
2020/11/11

Table of Contents

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.

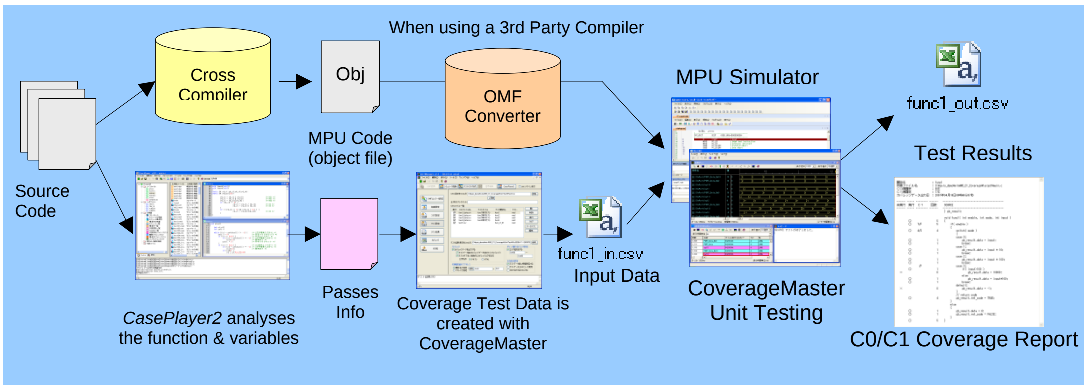
Additional technical support related information and FAQs can also be found at the following link: https://www.en.gaio.co.jp/eng/support/user/techpaper.html

Tool Version

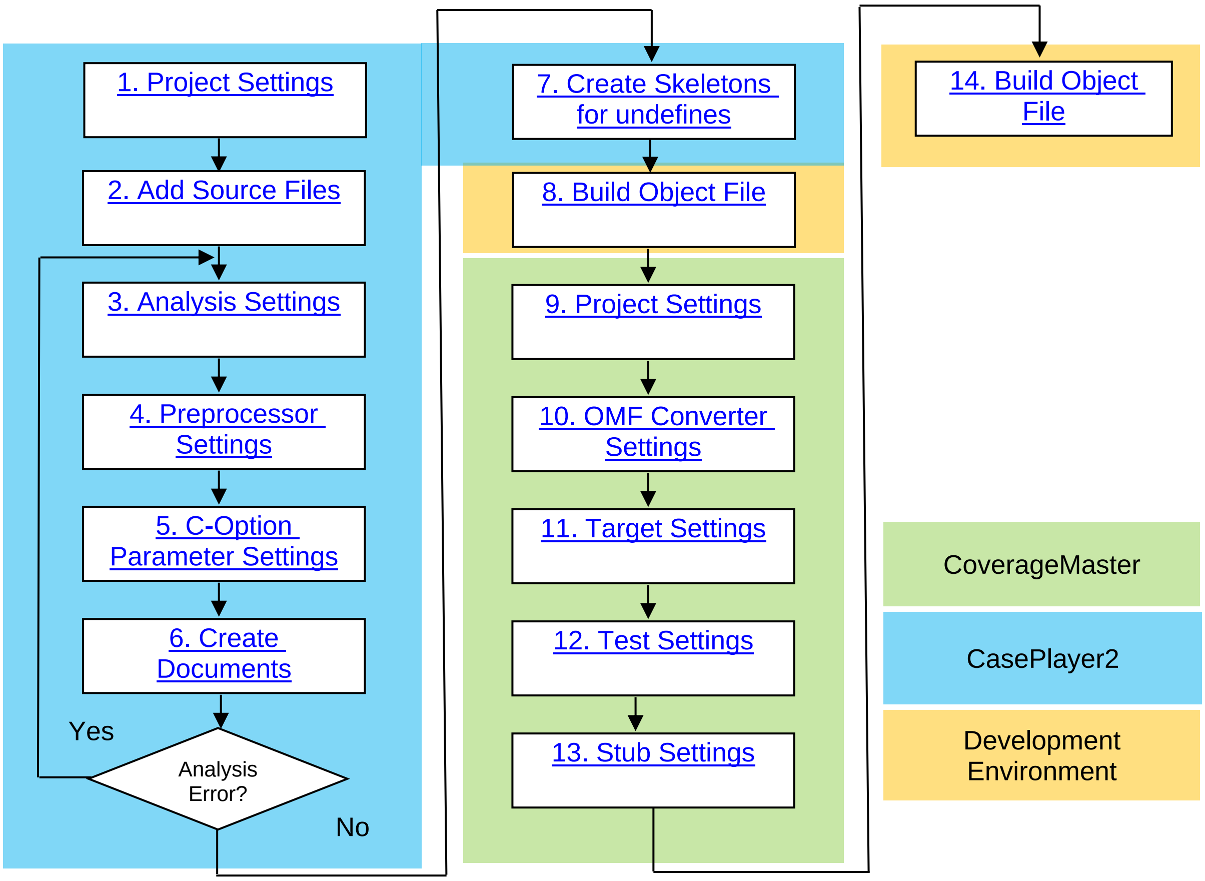
The following tool versions were used as a base for this document.

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.

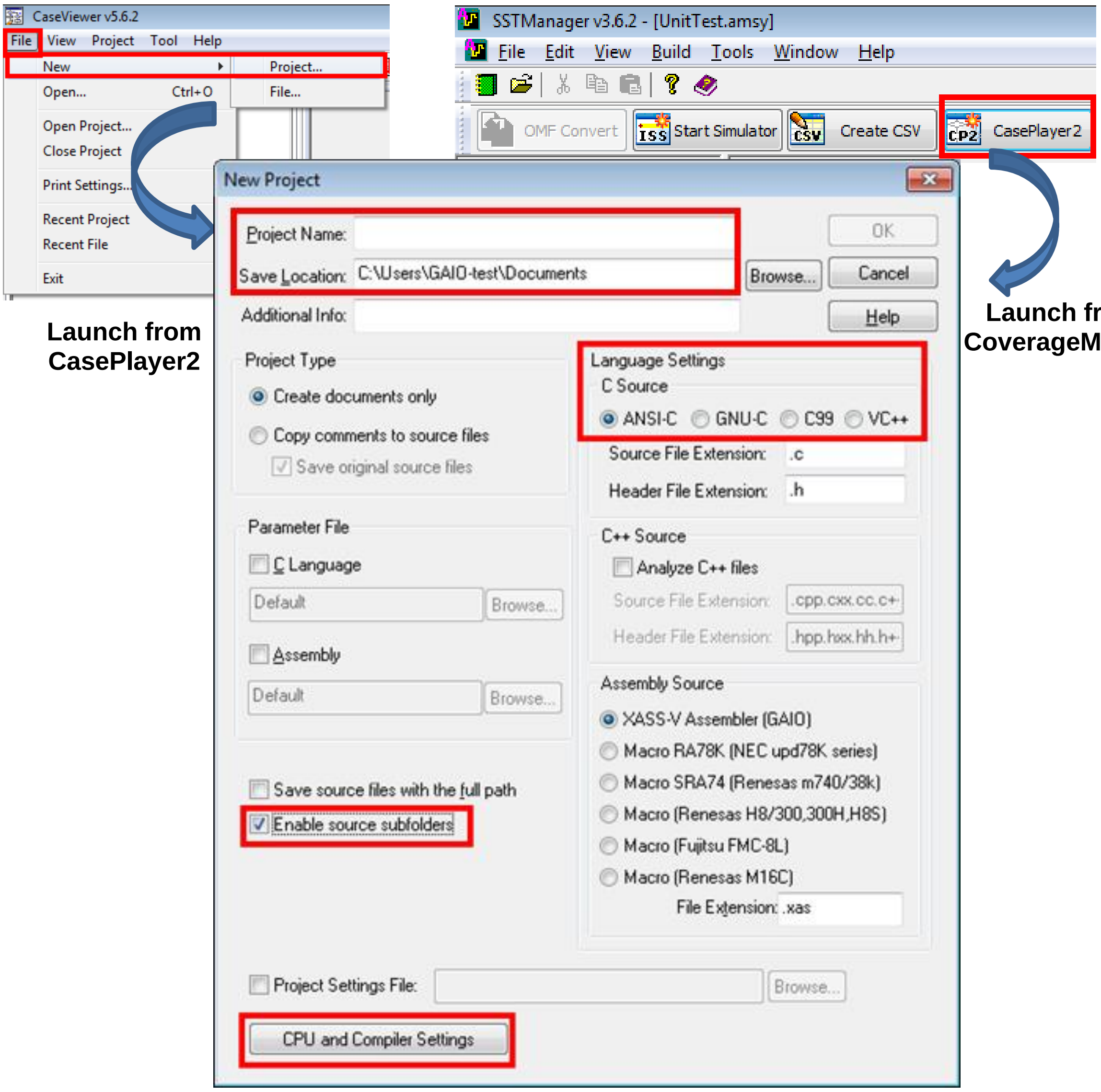
2. CoverageMaster winAMS Setup

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



New Project

Project Name:

Save Location: C:\Users\GAIO-test\Documents

Additional Info:

Project Type
- ● Create documents only
- ○ Copy comments to source files
- ☑ Save original source files

Parameter File
- □ C Language: Default
- □ Assembly: Default

- □ Save source files with the full path
- ☑ Enable source subfolders

Language Settings

C Source
- ● ANSI-C ○ GNU-C ○ C99 ○ VC++
- Source File Extension: .c
- Header File Extension: .h

C++ Source
- □ Analyze C++ files
- Source File Extension: .cpp.cxx.cc.c+
- Header File Extension: .hpp.hxx.hh.h+

Assembly Source
- ● XASS-V Assembler (GAIO)
- ○ Macro RA78K (NEC upd78K series)
- ○ Macro SRA74 (Renesas m740/38k)
- ○ Macro (Renesas H8/300,300H,H8S)
- ○ Macro (Fujitsu FMC-8L)
- ○ Macro (Renesas M16C)
- File Extension: .xas

□ Project Settings File:

CPU and Compiler 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

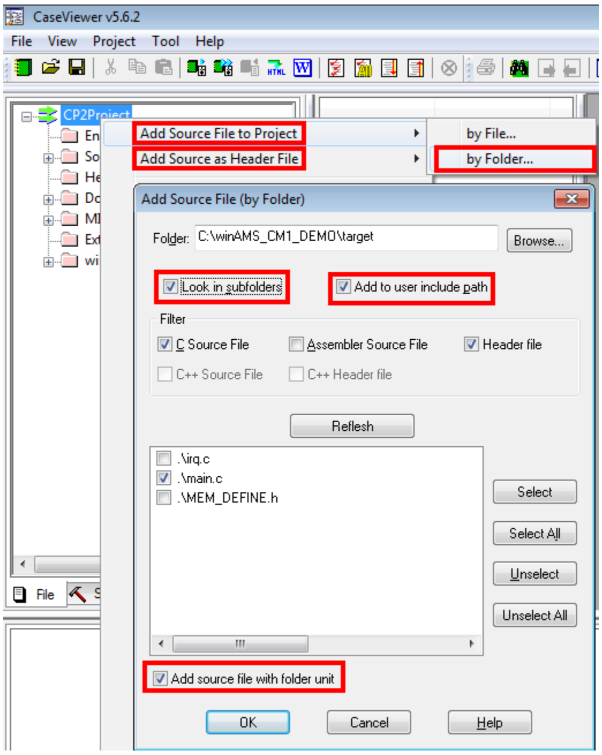
An analysis error can occur if compiler specific descriptions are included in the source code. These compiler specific descriptions must be added to the “C Option Parameter” – “User Keyword” settings in order to be properly recognized by CasePlayer2. By selecting your CPU and compiler combination using 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.

https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_019_07.html

2. Add Source Files



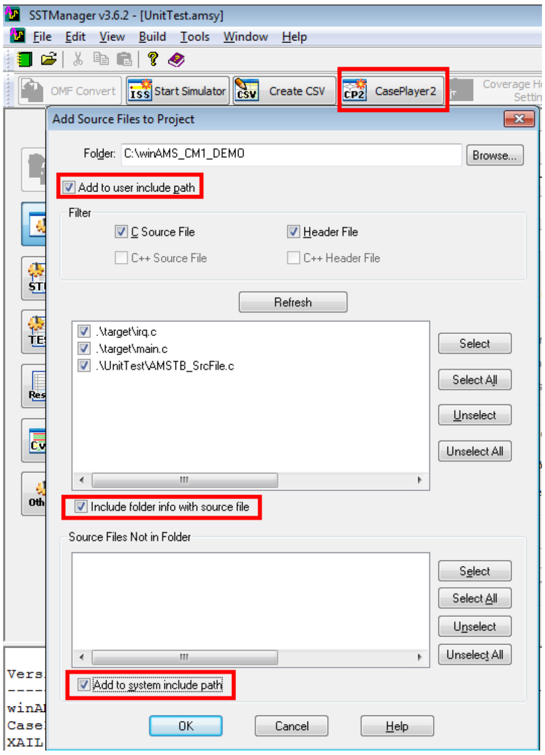
Folder: C:\winAMS_CM1_DEMO\target

☑ Look in subfolders
☑ Add to user include path

☑ C Source File
☐ Assembler Source File
☑ Header file
☐ C++ Source File
☐ C++ Header file

☐ .\irq.c
☑ .\main.c
☐ .\MEM_DEFINE.h

☑ Add source file with folder unit



Folder: C:\winAMS_CM1_DEMO

☑ Add to user include path

☑ C Source File
☑ Header File
☐ C++ Source File
☐ C++ Header File

☑ .\target\irq.c
☑ .\target\main.c
☑ .\UnitTest\AMSTB_SrcFile.c

☑ Include folder info with source file

☑ Add to system include path

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.

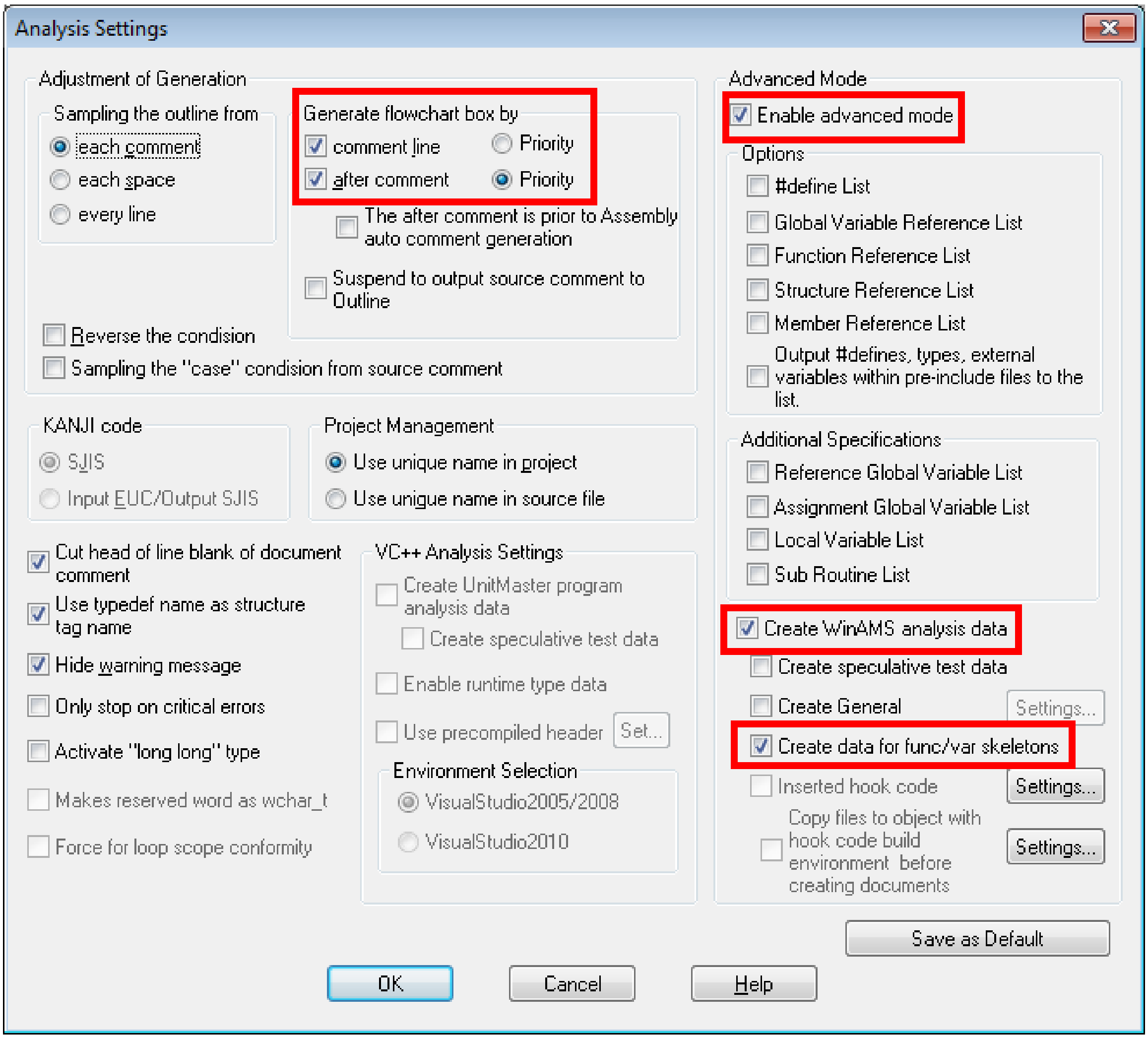
2.4. Add source file with folder unit

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



Adjustment of Generation

Sampling the outline from
- ● each comment
- ○ each space
- ○ every line

Generate flowchart box by
- ☑ comment line ○ Priority
- ☑ after comment ● Priority
- □ The after comment is prior to Assembly auto comment generation
- □ Suspend to output source comment to Outline

- □ Reverse the condision
- □ Sampling the "case" condision from source comment

KANJI code
- ● SJIS
- ○ Input EUC/Output SJIS

Project Management
- ● Use unique name in project
- ○ Use unique name in source file

- ☑ Cut head of line blank of document comment
- ☑ Use typedef name as structure tag name
- ☑ Hide warning message
- □ Only stop on critical errors
- □ Activate "long long" type
- □ Makes reserved word as wchar_t
- □ Force for loop scope conformity

VC++ Analysis Settings
- □ Create UnitMaster program analysis data
- □ Create speculative test data
- □ Enable runtime type data
- □ Use precompiled header

Environment Selection
- ● VisualStudio2005/2008
- ○ VisualStudio2010

Advanced Mode
- ☑ Enable advanced mode

Options
- □ #define List
- □ Global Variable Reference List
- □ Function Reference List
- □ Structure Reference List
- □ Member Reference List
- □ Output #defines, types, external variables within pre-include files to the list.

Additional Specifications
- □ Reference Global Variable List
- □ Assignment Global Variable List
- □ Local Variable List
- □ Sub Routine List

- ☑ Create WinAMS analysis data
- □ Create speculative test data
- □ Create General
- ☑ Create data for func/var skeletons
- □ Inserted hook code
- □ Copy files to object with hook code build environment before creating documents

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?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_D01_01.html

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

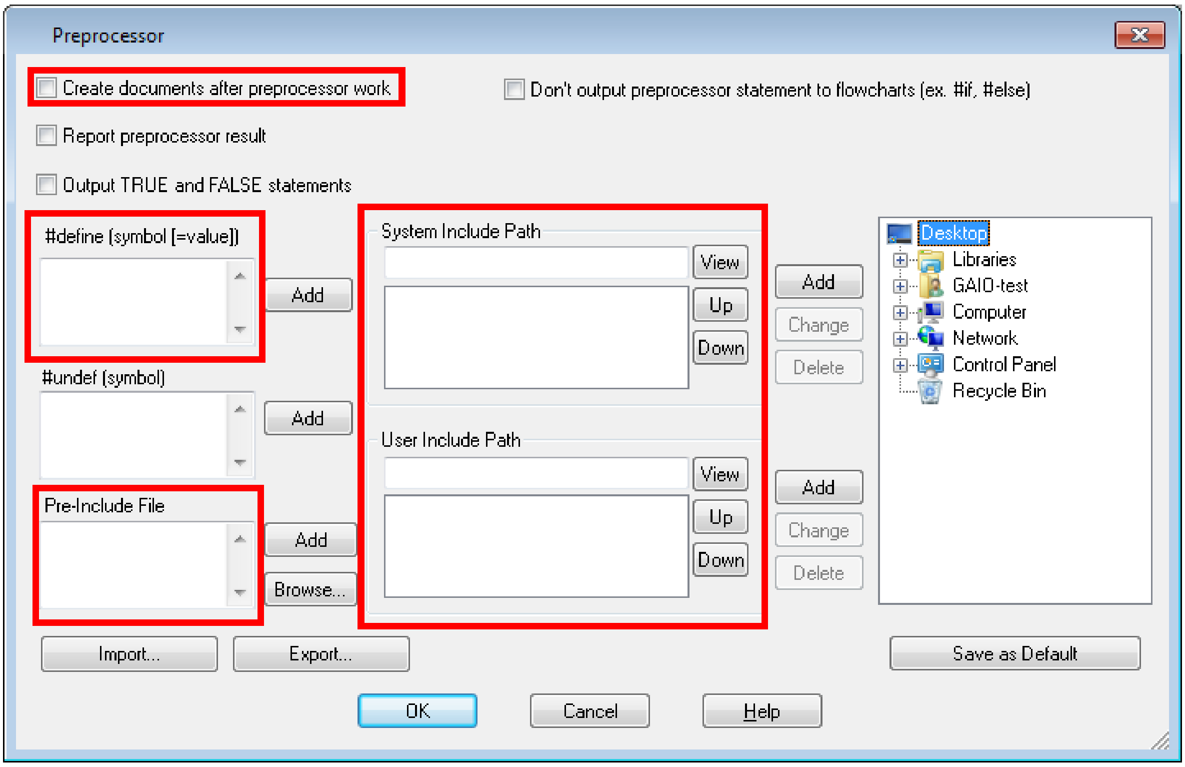
When enabled, this setting will detect undefined functions and variables during analysis. Using this analysis data the “Create Undefined Function Skeleton…” feature can be used to easily create skeletons for undefined functions or variables.

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?
https://www.en.gaio.co.jp/eng/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."

https://www.en.gaio.co.jp/eng/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?
https://www.en.gaio.co.jp/eng/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.

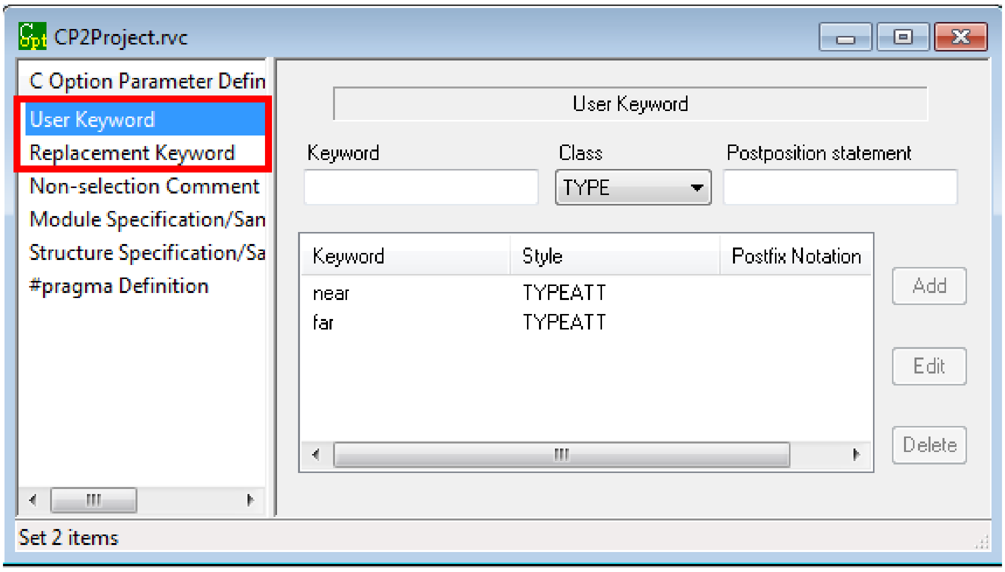
Related FAQ 019_04: The 'file i/o error' messages are output when executing the CasePlayer2's static analysis.
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_019_04.html

Related FAQ 019_02: How is the retrieval sequence of CasePlayer2's include path settings?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_019_02.html

Related FAQ 019_03: Can I use a relative path to the 'user include path'?
https://www.en.gaio.co.jp/eng/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.
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_019_05.html

5. C Option Parameter Settings



Keyword

Class: TYPE

Postposition statement

Keyword	Style	Postfix Notation
near	TYPEATT	
far	TYPEATT	

Set 2 items

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.

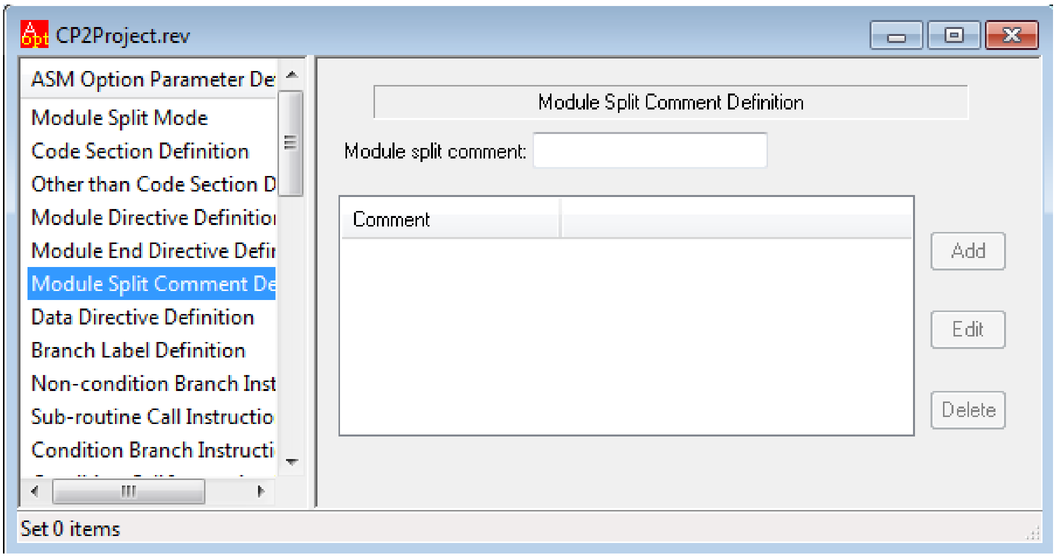
Related FAQ 019_05: I tried to analyze the registered source files. However, the ‘CasePlayer2-E-SYN : ; expected.’ error message is displayed.
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_019_07.html

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.

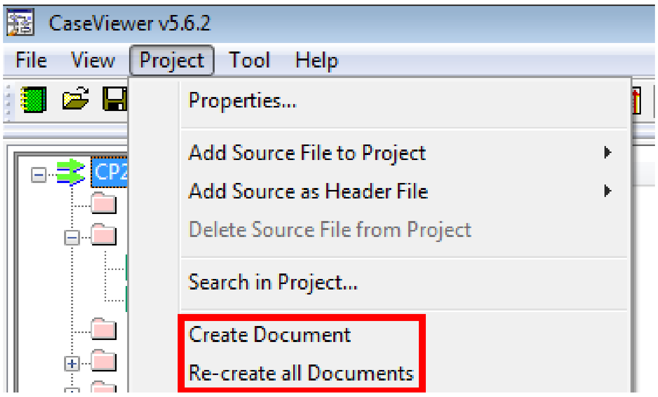
(Reference) ASM Instruction Definition Settings

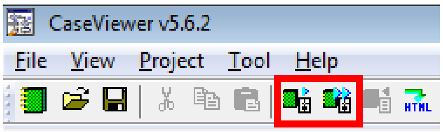

CP2Project.rev
ASM Option Parameter De
Module Split Mode
Code Section Definition
Other than Code Section D
Module Directive Definitio
Module End Directive Defir
Module Split Comment De
Data Directive Definition
Branch Label Definition
Non-condition Branch Inst
Sub-routine Call Instructio
Condition Branch Instructi
Module Split Comment Definition
Module split comment:
Comment
Add
Edit
Delete
Set 0 items

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?
https://www.en.gaio.co.jp/eng/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_07: I tried to analyze the registered source files. However, the ‘CasePlayer2-E-SYN : ; expected.’ error message is displayed.
https://www.en.gaio.co.jp/eng/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.

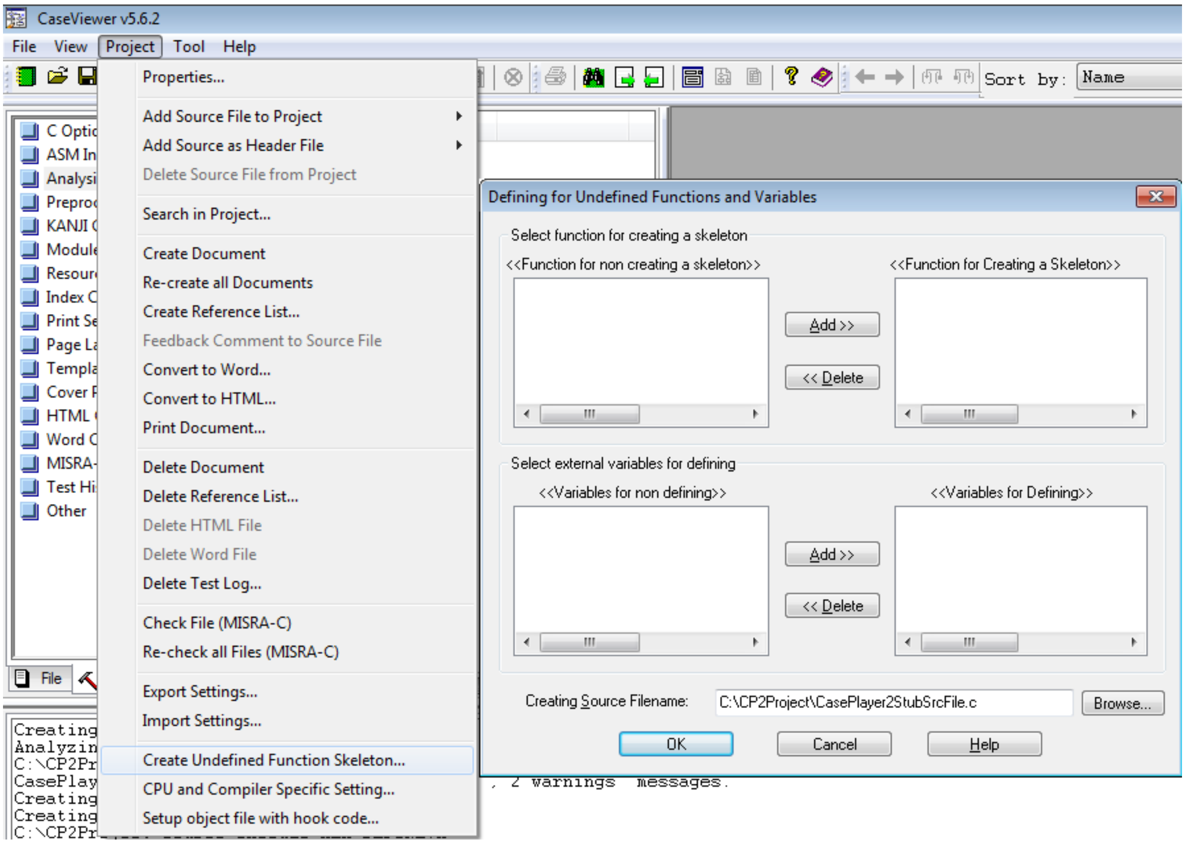
*Note: If analysis settings are changed, all source files will be re-analyzed even if the Create Documents option is selected.

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.



Creating Source Filename: C:\CP2Project\CasePlayer2StubSrcFile.c

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

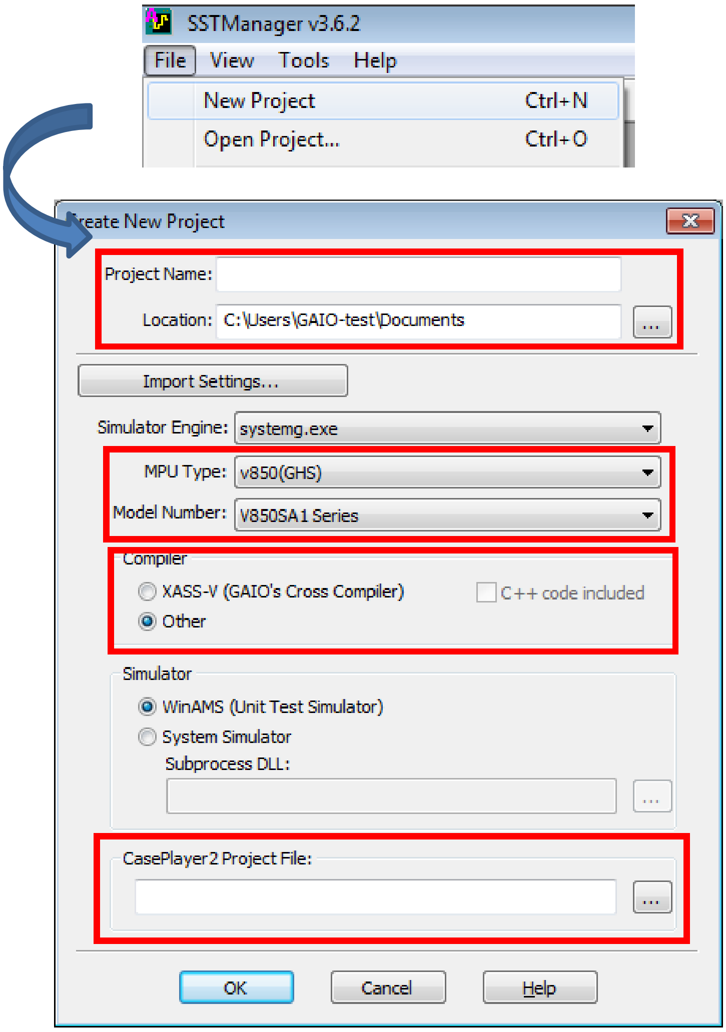
462_01: Does the version of CasePlayer2 and CoverageMaster have to match?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_462_01.html

C01_01: How do I fix a [Run-time Error '5941'] that occurs during Word conversion?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_C01_01.html

C01_02: How do I fix a [<Template List> ******.dot] error that occurs during Word conversion?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_C01_02.html

CoverageMaster winAMS Setup

9. Project Settings



Project Name:

Location: C:\Users\GAIO-test\Documents

Simulator Engine: systemg.exe

MPU Type: v850(GHS)

Model Number: V850SA1 Series

Compiler

○ XASS-V (GAIO's Cross Compiler) □ C++ code included

● Other

Simulator

● WinAMS (Unit Test Simulator)

○ System Simulator

Subprocess DLL:

CasePlayer2 Project File:

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_06: I cannot find my MPU type in the test project settings dialog.
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_017_06.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?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_017_01.html

Tips：OMF Converter

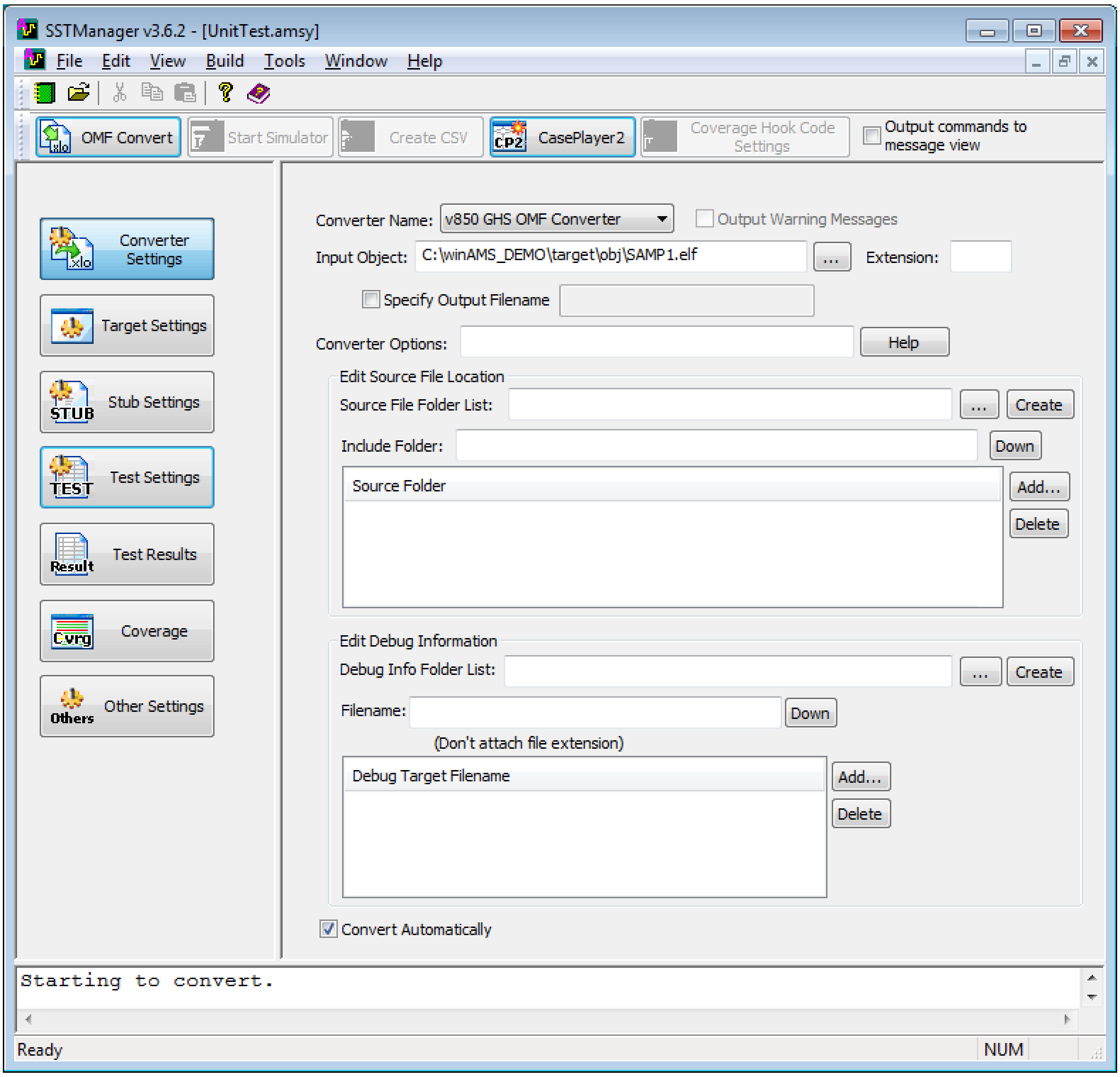
The OMF Converter only need be used when the object file was compiled with a compiler other than GAIO's cross compiler.

9.5. CasePlayer2 Project File

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

Related FAQ 462_01: Does the version of CasePlayer2 and CoverageMaster have to match?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_462_01.html

10. Converter Settings



SSTManager v3.6.2 - [UnitTest.amsy]

□ Output commands to message view

Converter Name: v850 GHS OMF Converter

□ Output Warning Messages

Input Object: C:\winAMS_DEMO\target\obj\SAMP1.elf

Extension:

□ Specify Output Filename

Converter Options:

Edit Source File Location

Source File Folder List:

Include Folder:

Edit Debug Information

Debug Info Folder List:

Filename:

(Don't attach file extension)

☑ Convert Automatically

```
Starting to convert.
```

Ready NUM

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 are required regarding the object code in order to setup CoverageMaster?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_011_03.html

Related FAQ 017_01: The "OMF Convert" button is disabled. How can I use it?
https://www.en.gaio.co.jp/eng/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.

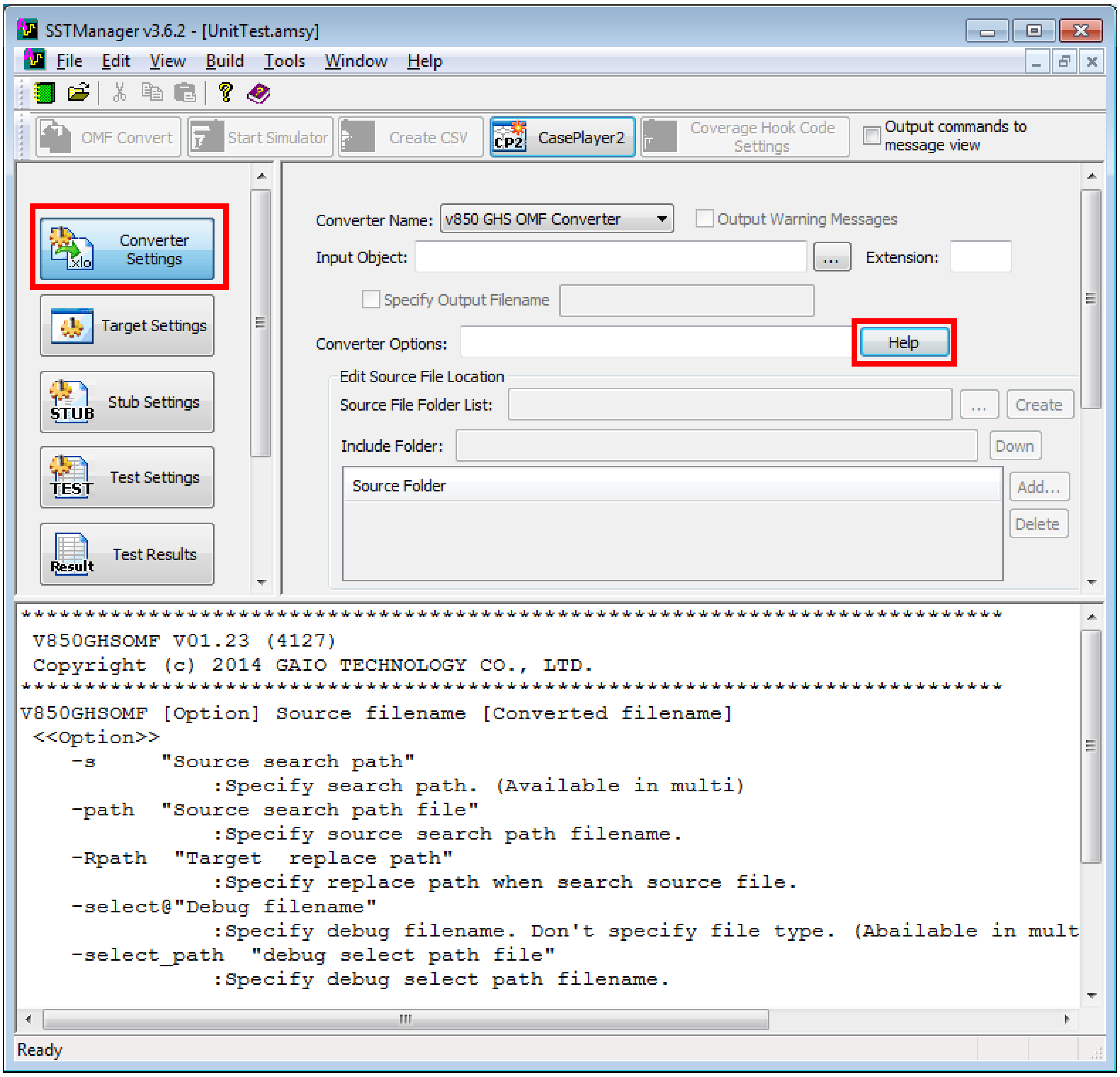
10.3. Input Object

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.



Converter Name: v850 GHS OMF Converter

```
*****************************************************************************
 V850GHSOMF V01.23 (4127)
 Copyright (c) 2014 GAIO TECHNOLOGY CO., LTD.
*****************************************************************************
V850GHSOMF [Option] Source filename [Converted filename]
 <<Option>>
    -s     "Source search path"
               :Specify search path. (Available in multi)
    -path  "Source search path file"
               :Specify source search path filename.
    -Rpath  "Target  replace path"
               :Specify replace path when search source file.
    -select@"Debug filename"
               :Specify debug filename. Don't specify file type. (Abailable in mult
    -select_path  "debug select path file"
               :Specify debug select path filename.
```

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.

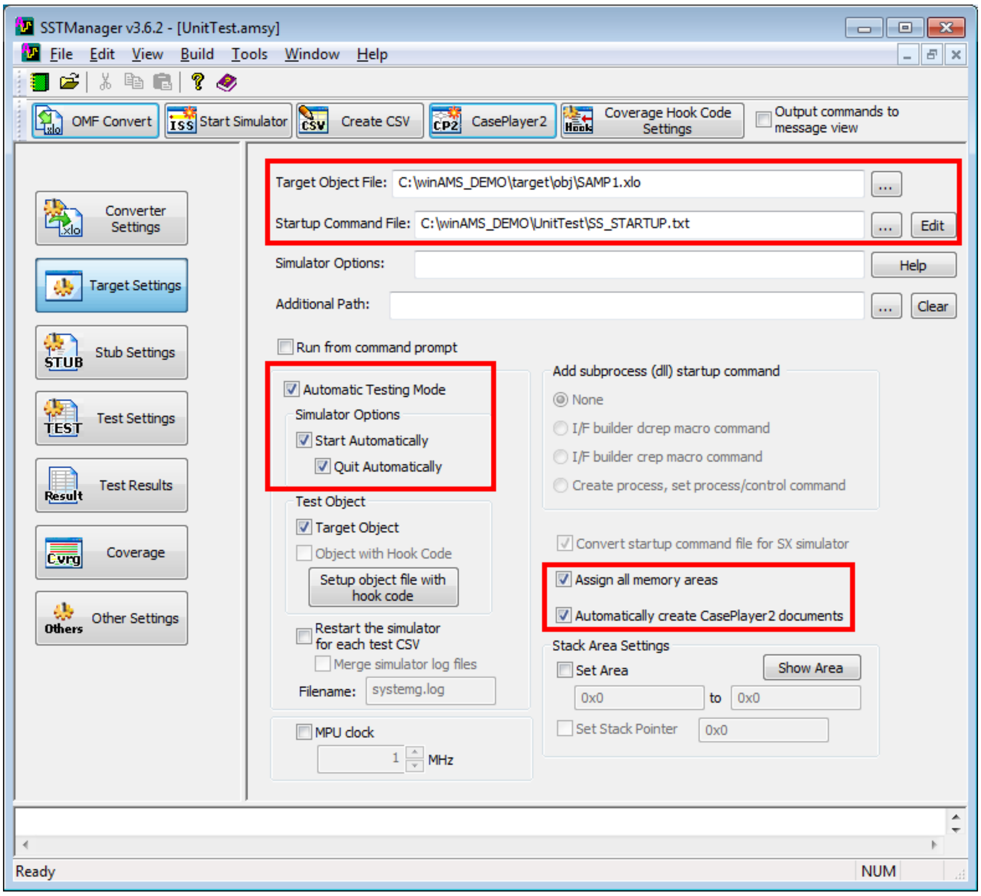
Related FAQ 017_02: How to hide the OMF conversion warning messages?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_017_02.html

Related FAQ 017_03: The tool terminates during OMF conversion.
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_017_03.html

Related FAQ 017_04: "Failed to read the object file" error during OMF conversion.
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_017_04.html

Related FAQ 017_05: When I try to OMF convert an object file built with the IAR compiler, the conversion fails with the error "Could not open the input file."
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_017_05.html

11. Target Settings



□ Output commands to message view

Target Object File: C:\winAMS_DEMO\target\obj\SAMP1.xlo

Startup Command File: C:\winAMS_DEMO\UnitTest\SS_STARTUP.txt

Simulator Options:

Additional Path:

□ Run from command prompt

☑ Automatic Testing Mode

Simulator Options

☑ Start Automatically

☑ Quit Automatically

Test Object

☑ Target Object

□ Object with Hook Code

Setup object file with hook code

□ Restart the simulator for each test CSV

□ Merge simulator log files

Filename: systemg.log

□ MPU clock

1 MHz

Add subprocess (dll) startup command

● None

○ I/F builder dcrep macro command

○ I/F builder crep macro command

○ Create process, set process/control command

☑ Convert startup command file for SX simulator

☑ Assign all memory areas

☑ Automatically create CasePlayer2 documents

Stack Area Settings

□ Set Area

0x0 to 0x0

□ Set Stack Pointer 0x0

11.1. Target Object File

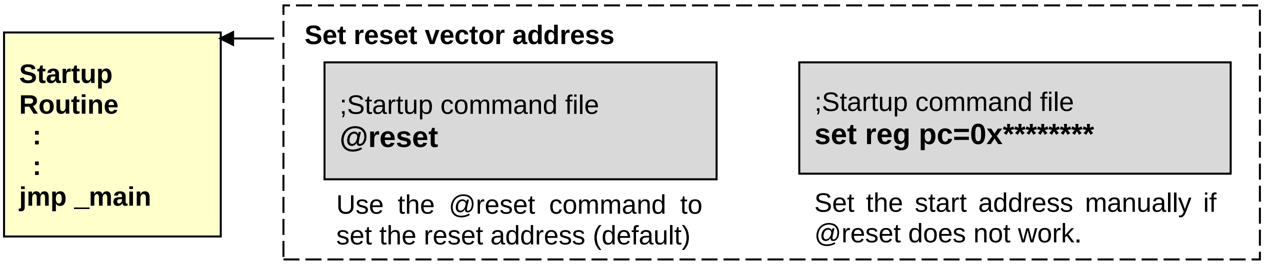
Set the target object (*.xlo) file here. After the OMF conversion the converted object file will be automatically set.

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



Startup Routine
:
:
jmp _main

Set reset vector address

```
;Startup command file
@reset
```

Use the @reset command to set the reset address (default)

```
;Startup command file
set reg pc=0x********
```

Set the start address manually if @reset does not work.

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 are required regarding the object code in order to setup CoverageMaster?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_011_03.html

Related FAQ 000_01: The simulator stops with an error. How can I output a simulation log to investigate the cause?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_000_01.html

Related FAQ 326_01: How can I read an address and change its procedure using a macro entered in the startup command file?
https://www.en.gaio.co.jp/eng/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. 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.

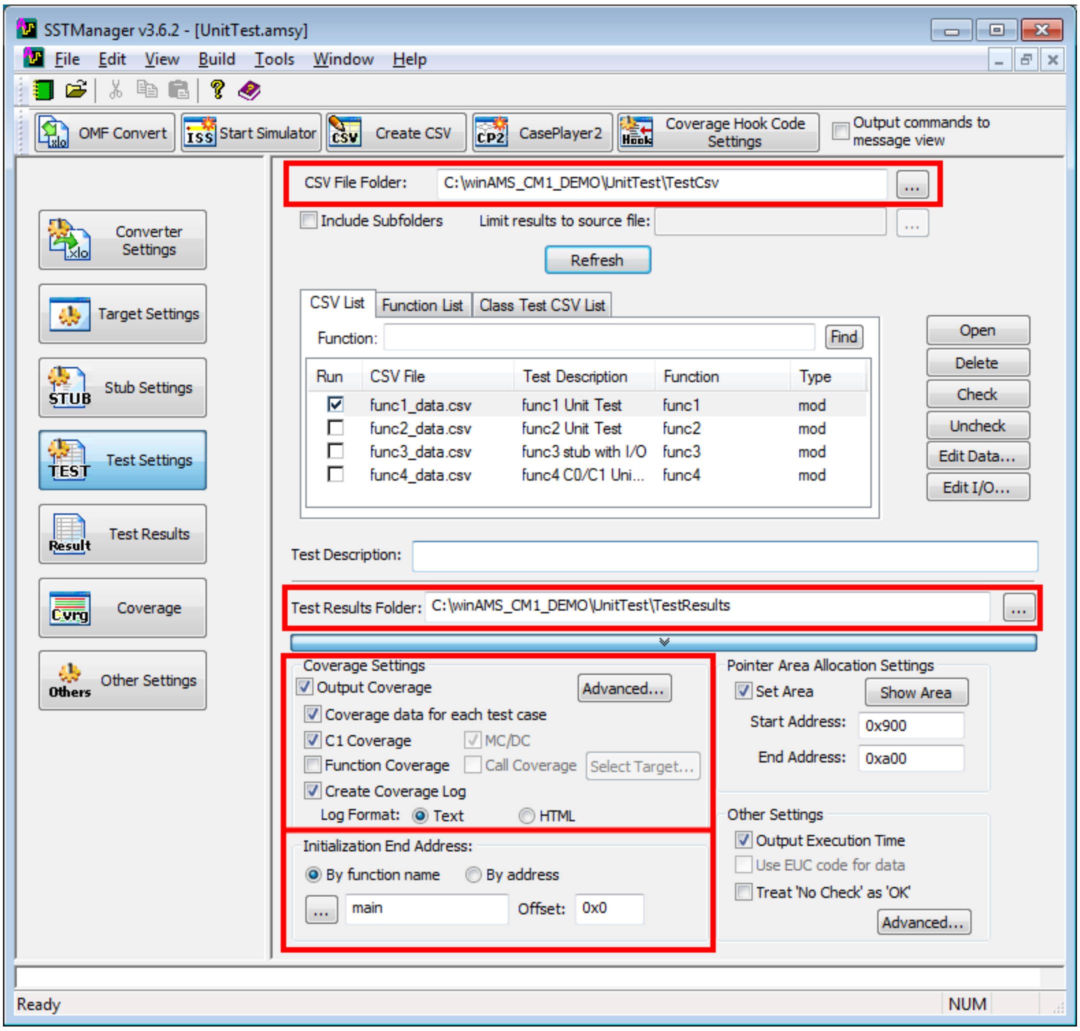
Tips: assigning memory areas the same as the MPU

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



SSTManager v3.6.2 - [UnitTest.amsy]

□ Output commands to message view

CSV File Folder: C:\winAMS_CM1_DEMO\UnitTest\TestCsv

□ Include Subfolders   Limit results to source file:

Function:

Run	CSV File	Test Description	Function	Type
☑	func1_data.csv	func1 Unit Test	func1	mod
□	func2_data.csv	func2 Unit Test	func2	mod
□	func3_data.csv	func3 stub with I/O	func3	mod
□	func4_data.csv	func4 C0/C1 Uni...	func4	mod

Test Description:

Test Results Folder: C:\winAMS_CM1_DEMO\UnitTest\TestResults

Coverage Settings
☑ Output Coverage
☑ Coverage data for each test case
☑ C1 Coverage   ☑ MC/DC
□ Function Coverage   □ Call Coverage
☑ Create Coverage Log
Log Format: ● Text   ○ HTML

Initialization End Address:
● By function name   ○ By address
main   Offset: 0x0

Pointer Area Allocation Settings
☑ Set Area
Start Address: 0x900
End Address: 0xa00

Other Settings
☑ Output Execution Time
□ Use EUC code for data
□ Treat 'No Check' as 'OK'

Ready   NUM

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.

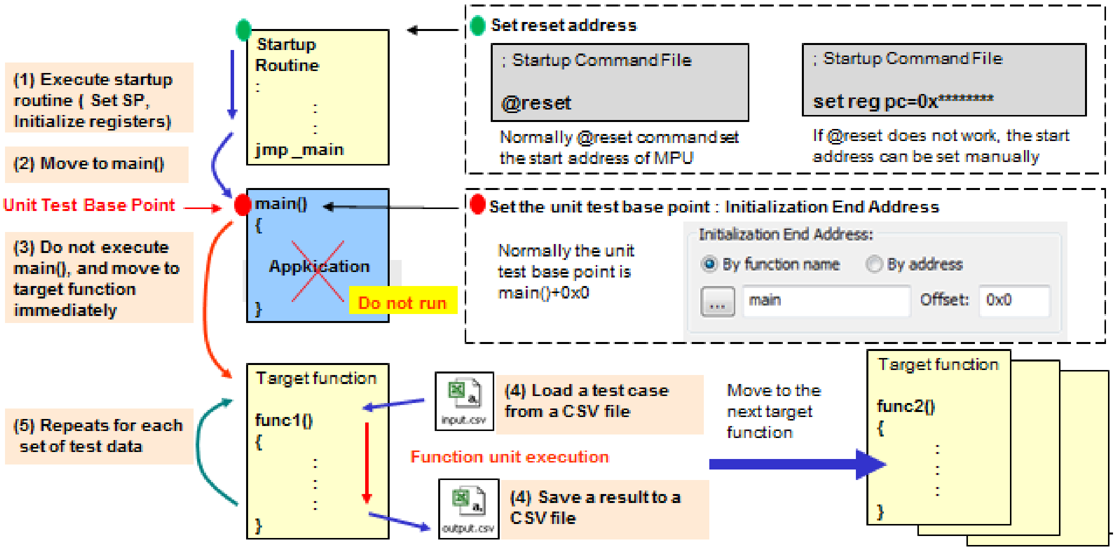
12.4. Initialization End Address (Important!)

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.


(1) Execute startup routine (Set SP, Initialize registers)
(2) Move to main()
Unit Test Base Point
(3) Do not execute main(), and move to target function immediately
(5) Repeats for each set of test data
Startup Routine
:
:
:
jmp _main
main()
{
Application
}
Do not run
Target function
func1()
{
:
:
}
Set reset address
Normally @reset command set the start address of MPU
If @reset does not work, the start address can be set manually
Set the unit test base point : Initialization End Address
Normally the unit test base point is main()+0x0
input.csv
(4) Load a test case from a CSV file
Function unit execution
output.csv
(4) Save a result to a CSV file
Move to the next target function
Target function
func2()
{
:
:
:
}

```
; Startup Command File
@reset
```

```
; Startup Command File
set reg pc=0x********
```

Initialization End Address:
● By function name ○ By address
... main Offset: 0x0

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.

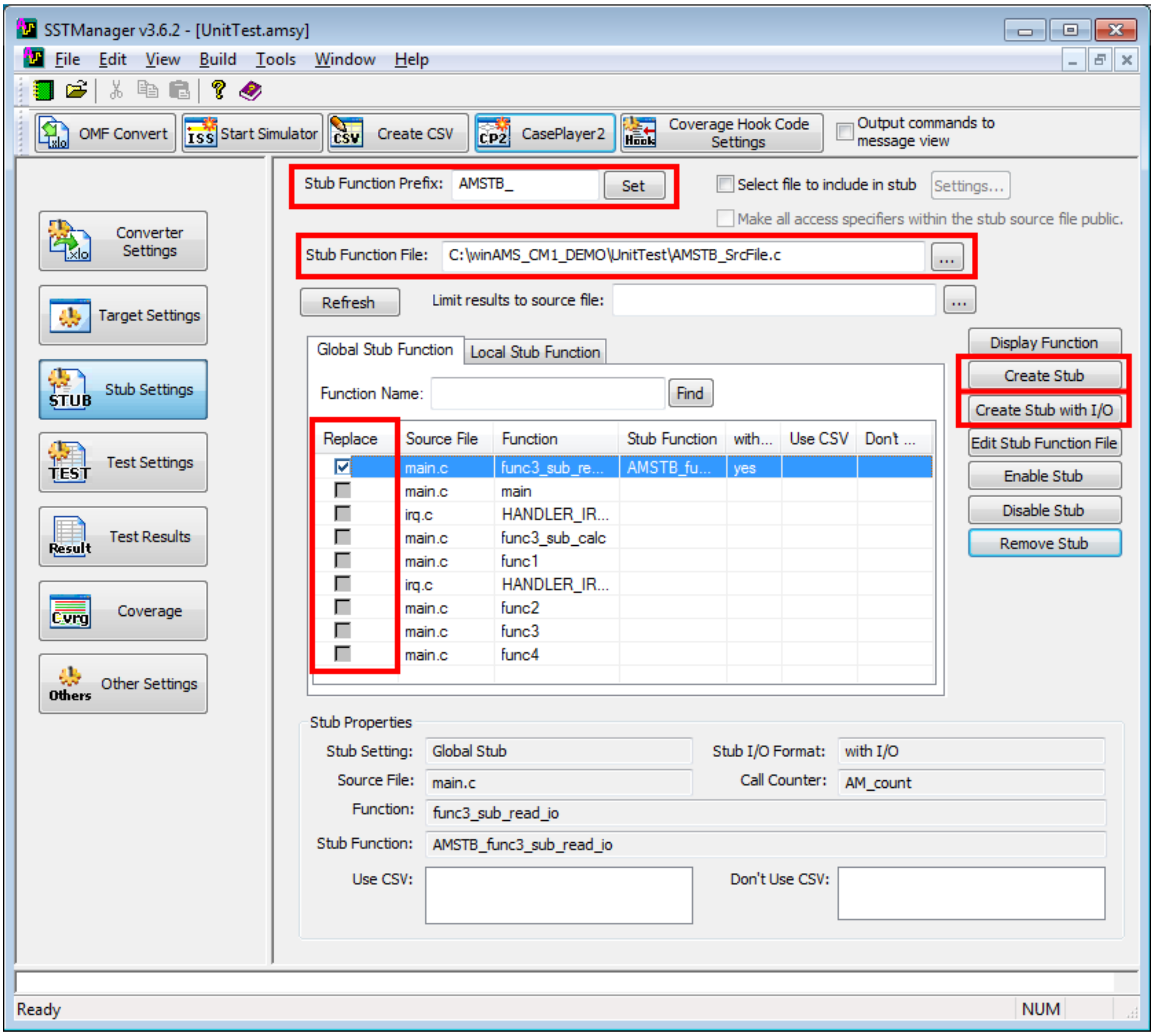
If an appropriate initialization end address cannot be found, it can be set to a point immediately after the reset address set in the startup command file, and the necessary stack and register initial values can be manually entered into the startup command file. Consult your targets architecture manual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

Related FAQ 011_01: What is the "Offset" setting of the "Initialization End Address"?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_011_01.html

Related FAQ 011_02: What is the "Initialization End Address" setting in the Test Setting screen?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_011_02.html

Related FAQ 011_03: What initial settings are required regarding the object code in order to setup CoverageMaster?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_011_03.html

13. Stub Settings



□ Output commands to message view

Stub Function Prefix: AMSTB_

□ Select file to include in stub

□ Make all access specifiers within the stub source file public.

Stub Function File: C:\winAMS_CM1_DEMO\UnitTest\AMSTB_SrcFile.c

Limit results to source file:

Global Stub Function | Local Stub Function

Function Name:

Replace	Source File	Function	Stub Function	with...	Use CSV	Don't ...
☑	main.c	func3_sub_re...	AMSTB_fu...	yes		
□	main.c	main				
□	irq.c	HANDLER_IR...				
□	main.c	func3_sub_calc				
□	main.c	func1				
□	irq.c	HANDLER_IR...				
□	main.c	func2				
□	main.c	func3				
□	main.c	func4				

Stub Properties

Stub Setting: Global Stub

Stub I/O Format: with I/O

Source File: main.c

Call Counter: AM_count

Function: func3_sub_read_io

Stub Function: AMSTB_func3_sub_read_io

Use CSV:

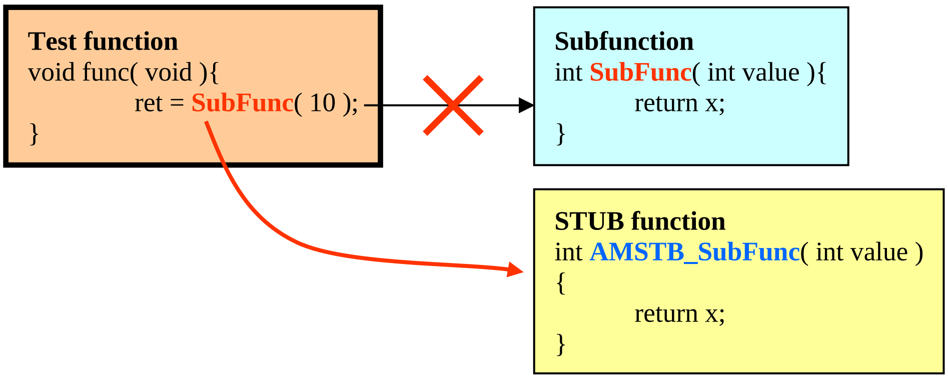
Don't Use CSV:

Ready

NUM

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.



Test function

```
void func( void ){
        ret = SubFunc( 10 );
}
```

Subfunction

```
int SubFunc( int value ){
        return x;
}
```

STUB function

```
int AMSTB_SubFunc( int value )
{
        return x;
}
```

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

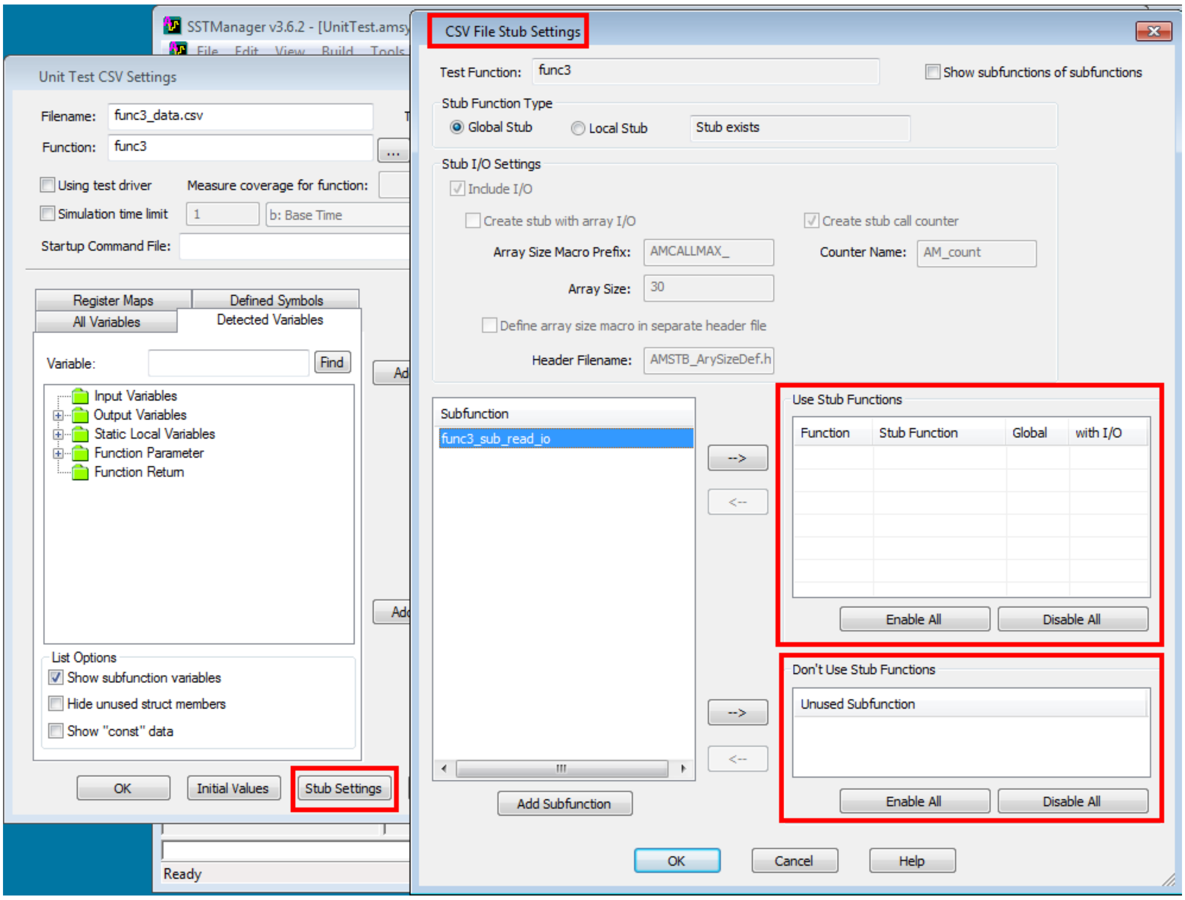
```
int AMSTB_ subfunc( int index )
{
   static int volatile AMIN_return;
   static int volatile AMOUT_index;

   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.



Unit Test CSV Settings

Filename: func3_data.csv

Function: func3

□ Using test driver Measure coverage for function:

□ Simulation time limit 1 b: Base Time

Startup Command File:

Variable:

- Input Variables
- Output Variables
- Static Local Variables
- Function Parameter
- Function Return

List Options

☑ Show subfunction variables

□ Hide unused struct members

□ Show "const" data

CSV File Stub Settings

Test Function: func3 □ Show subfunctions of subfunctions

Stub Function Type: ● Global Stub ○ Local Stub Stub exists

Stub I/O Settings

☑ Include I/O

□ Create stub with array I/O ☑ Create stub call counter

Array Size Macro Prefix: AMCALLMAX_ Counter Name: AM_count

Array Size: 30

□ Define array size macro in separate header file

Header Filename: AMSTB_ArySizeDef.h

Subfunction

func3_sub_read_io

Use Stub Functions

Function	Stub Function	Global	with I/O

Don't Use Stub Functions

Unused Subfunction

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 a macro defined function?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_102_09.html

Related FAQ 002_04: How can I create and manage stub functions?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_002_04.html

Related FAQ 002_05: Is it possible to create a stub function for a function written in assembly code?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_002_05.html

Related FAQ 002_01: How to enable/disable global stub functions for an individual test (test CSV file)?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_002_01.html

Related FAQ 002_02: Function symbol error when trying to use a stub function.
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_002_02.html

Related FAQ 002_03: How to create a stub function for a function called by a function pointer?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_002_03.html

Related FAQ A03_01: How to create the stub function when the same function is called more than once?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_A03_01.html

14. Build Object File

14.1. Build Object File

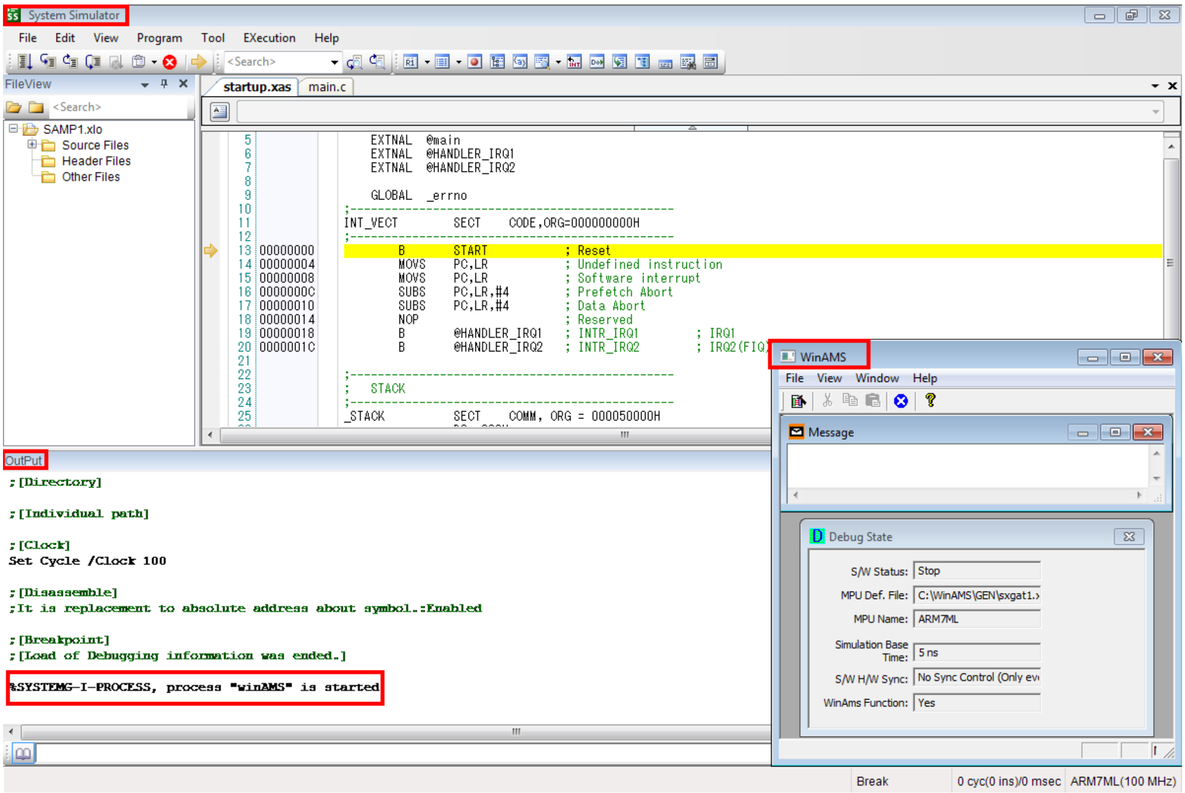
After creating a Stub Function the stub function source file must be added to the development environment and built (compiler & link). This will enable CoverageMaster to use the stub functions during testing.

*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) Using the System Simulator and Debugging



```
            EXTNAL  @main
            EXTNAL  @HANDLER_IRQ1
            EXTNAL  @HANDLER_IRQ2

            GLOBAL  _errno
          ;----------------------------------------------------
          INT_VECT        SECT    CODE,ORG=00000000H
          ;----------------------------------------------------
00000000        B       START           ; Reset
00000004        MOVS    PC,LR           ; Undefined instruction
00000008        MOVS    PC,LR           ; Software interrupt
0000000C        SUBS    PC,LR,#4        ; Prefetch Abort
00000010        SUBS    PC,LR,#4        ; Data Abort
00000014        NOP                     ; Reserved
00000018        B       @HANDLER_IRQ1   ; INTR_IRQ1         ; IRQ1
0000001C        B       @HANDLER_IRQ2   ; INTR_IRQ2         ; IRQ2(FIQ)

          ;----------------------------------------------------
          ;   STACK
          ;----------------------------------------------------
          _STACK          SECT    COMM, ORG = 00050000H
```

```
;[Directory]

;[Individual path]

;[Clock]
Set Cycle /Clock 100

;[Disassemble]
;It is replacement to absolute address about symbol.:Enabled

;[Breakpoint]
;[Load of Debugging information was ended.]

%SYSTEMG-I-PROCESS, process "winAMS" is started
```

S/W Status: Stop
MPU Def. File: C:\WinAMS\GEN\sxgat1.x
MPU Name: ARM7ML
Simulation Base Time: 5 ns
S/W H/W Sync: No Sync Control (Only ev
WinAms Function: Yes

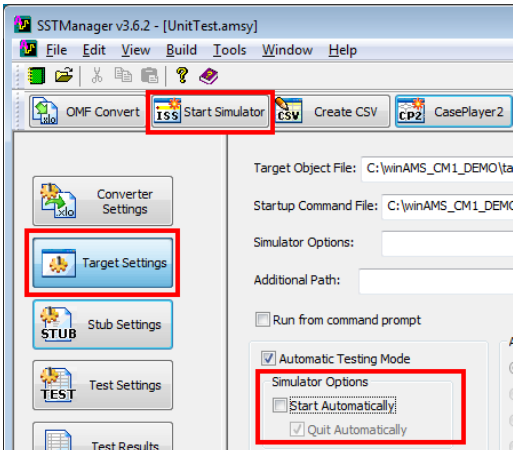
Break | 0 cyc(0 ins)/0 msec | ARM7ML(100 MHz)

Debugger UI Mode and Start / Quit Automatically Mode

If both the “Start Automatically” and “Quit Automatically” options are set in the Target Settings screen, 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. 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.

Using the Debugger UI Mode

Disable the “Start Automatically” option in the Target Settings screen, then click the “Start Simulator” button to start the System Simulator (debugger).



Target Object File: C:\winAMS_CM1_DEMO\ta
Startup Command File: C:\winAMS_CM1_DEMO
Simulator Options:
Additional Path:
□ Run from command prompt
☑ Automatic Testing Mode
Simulator Options
□ Start Automatically
☑ Quit Automatically

(Reference) Other Relevant FAQs

Related FAQ 021_01: How do I test using a VxWorks "loadable module"?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_021_01.html

Related FAQ 115_01: How to use an assembly symbol as a test variable?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_115_01.html

Related FAQ 115_02: How to test a function using #pragma inline_asm with the Renesas HEW compiler?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_115_02.html

Related FAQ 115_03: I cannot generate reference lists for variables defined in assembly.
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_115_03.html

Related FAQ 316_01: How to test an interrupt handler?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_316_01.html

Related FAQ 453_01: I cannot find a static function in the function list.
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_453_01.html

Related FAQ 102_08: How to test an inline function or a macro defined function?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_102_08.html

Related FAQ 301_02: How to test a function written in assembly?
https://www.en.gaio.co.jp/eng/support/user/faq/winams/faq_301_02.html

CoverageMaster Setup Manual

GAIO TECHNOLOGY CO., LTD. (JAPAN)

User Support / Technical Information
https://www.en.gaio.co.jp/eng/support/support.html

For technical support that cannot be resolved using the FAQs, create a support request using the following site:
https://www.en.gaio.co.jp/eng/support/support_entry.html
Please provide your user ID when requesting support.
* Note: a maintenance agreement is required for technical support.