

Renesas Compiler for SH

Abstract

This document describes the usage of the Renesas SH compiler which covers the SH family of microcontrollers. The compiler is integrated into the Renesas HEW and requires setting up a workspace within HEW in order to generate the necessary startup code for the respective SH device.

Please note: The Renesas compiler requires setting up a new workspace in order to build the test application. Follow the instructions below to create an appropriate workspace.

Table of Contents

Abstract	1
1 Introduction.....	2
2 HEW Workspace Setup.....	2
2.1 Step 1: Creating a new Workspace	2
2.2 Step 2: Building the Workspace Application	4
2.3 Step 3: Specify the Workspace within the Environment Editor (TEE).....	5
2.4 Step 4: Adapting the Makefile Template.....	5

1 Introduction

The Renesas HEW provides a workspace generator tool that should be used to generate the startup code for a specific SH microcontroller. The default settings prepared within the Tessy installation are for the SH1 microcontroller. If you are using another SH device (e.g. SH4) you will need to adapt the makefile template like described below.

2 HEW Workspace Setup

You need to create a new HEW workspace with the name “**ts_main**” in order to compile the test application with Tessy. The workspace may be saved somewhere on your hard disc. We would suggest to save the workspace within your project directory where you store other configuration related files.

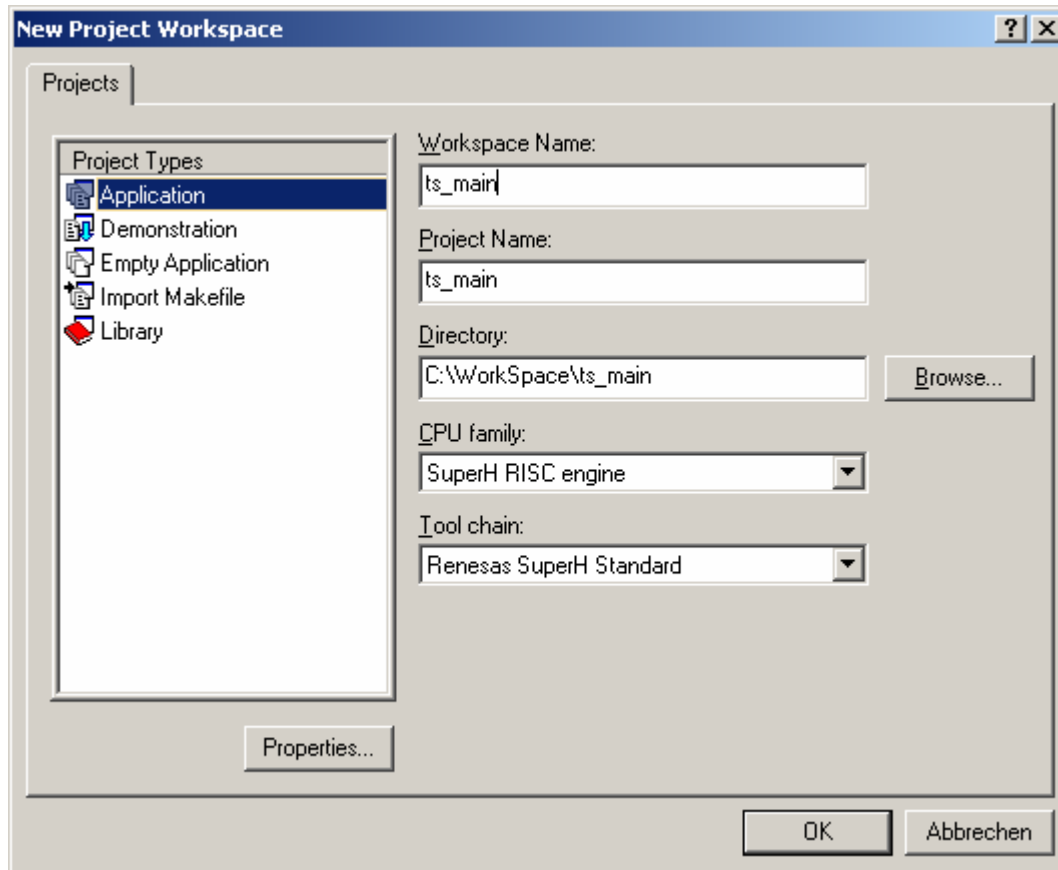
The generated workspace will contain the following files required for linking the test application:

- The standard C library
- The startup code consisting of several files defining the reset vector and other initialization code

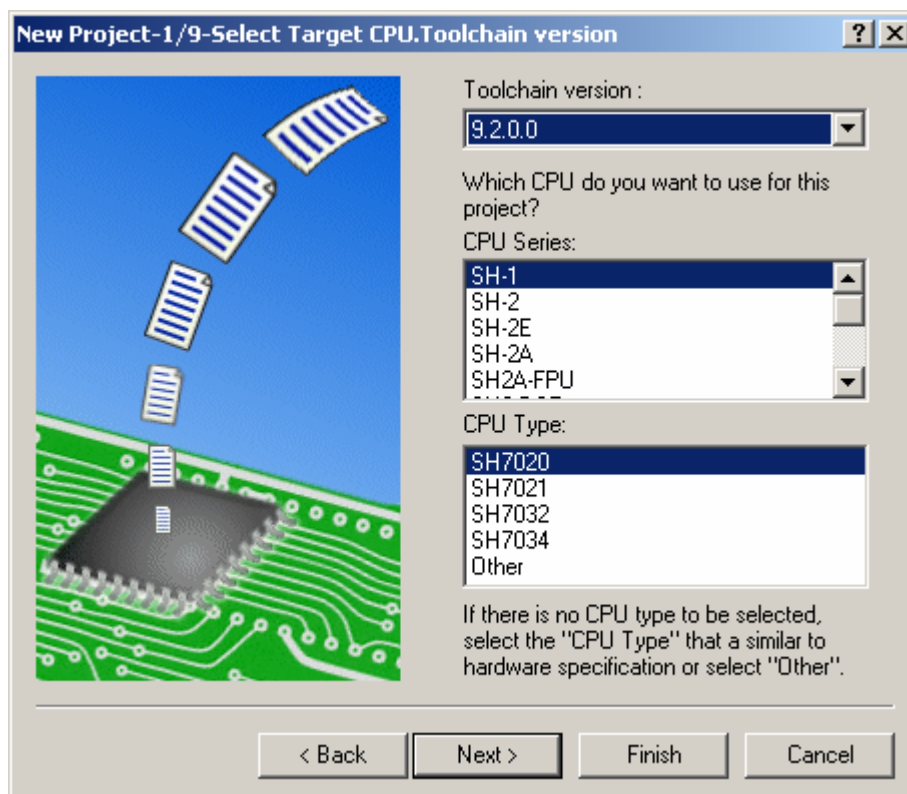
These files will be generated automatically from HEW when creating a new workspace.

2.1 Step 1: Creating a new Workspace

Choose **New Workspace** from the **File** menu and enter the following data into the dialog:



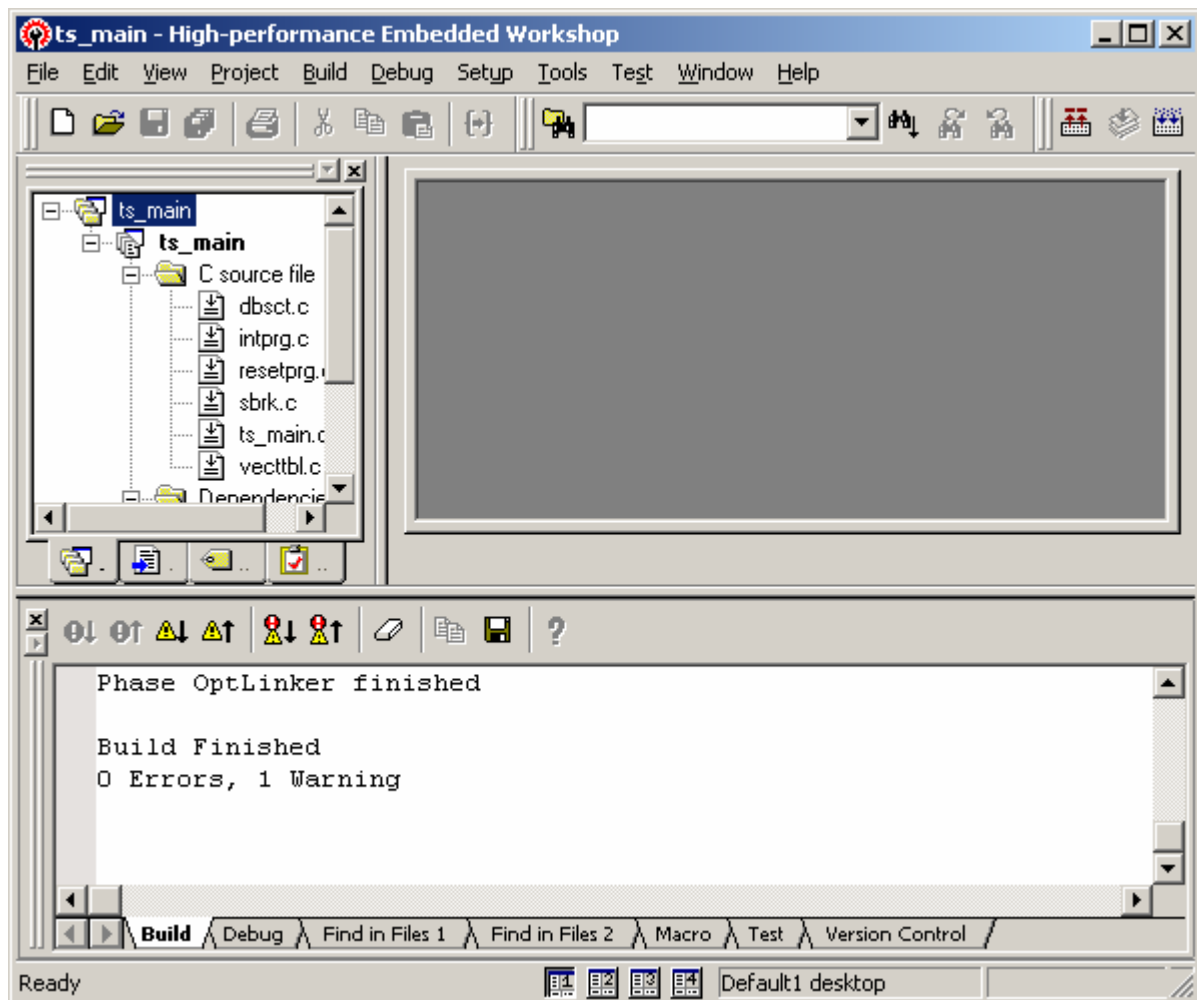
After pressing the OK button, you will see the following CPU selection dialog:



The next dialogs will guide you through the project creation wizard of HEW. Please refer to the HEW documentation for details about the necessary settings. Follow the instructions of the wizard until the project and workspace settings are complete and the new workspace and project will be opened.

2.2 Step 2: Building the Workspace Application

The next step is to choose **Build** from the context menu of the project. This compiles and links the sample application within the HEW workspace. The C library gets created and the startup code files are compiled. If everything went fine, you should not see any errors during compile and link. The workspace and project should look like follows after these actions:

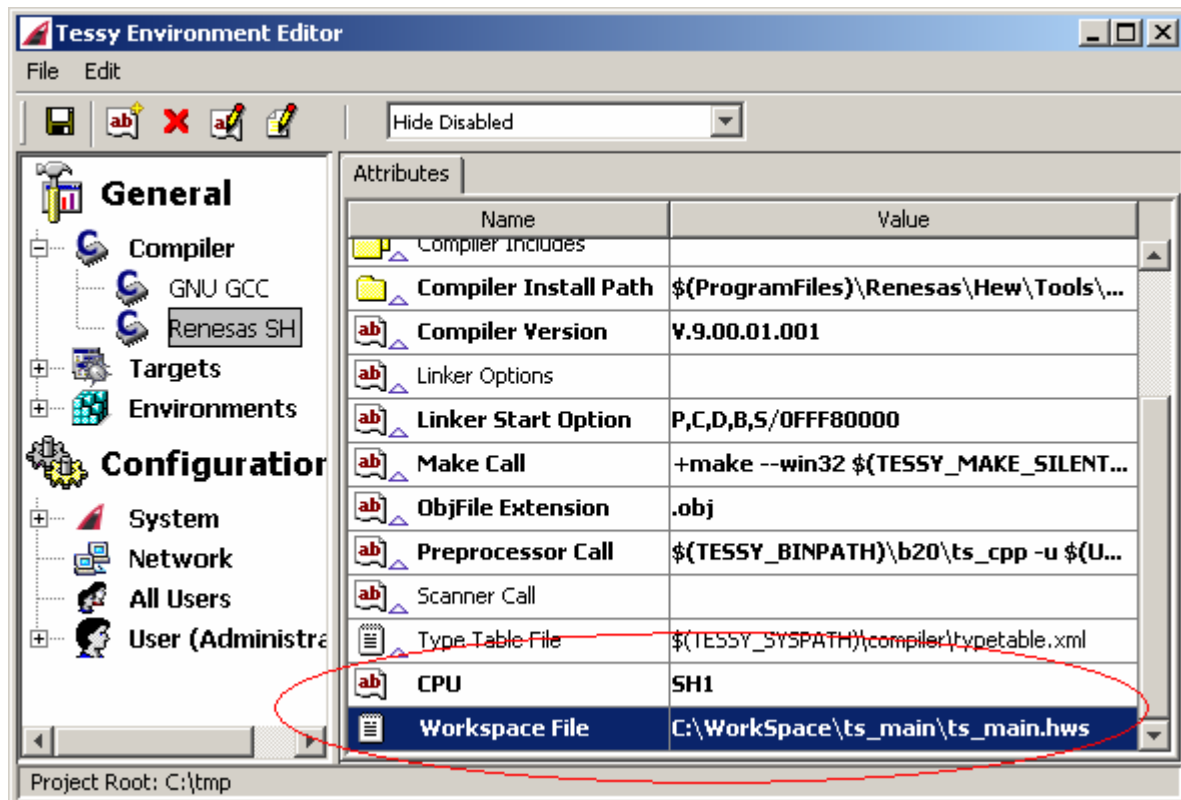


After completing these steps, save the workspace. The necessary files have been created now and you proceed with the next step below.

2.3 Step 3: Specify the Workspace within the Environment Editor (TEE)

You need to select your readily prepared workspace file within the Environment Editor TEE of Tessy like shown below.

*Please make sure, that the **CPU** attribute matches the CPU you selected during workspace creation.*

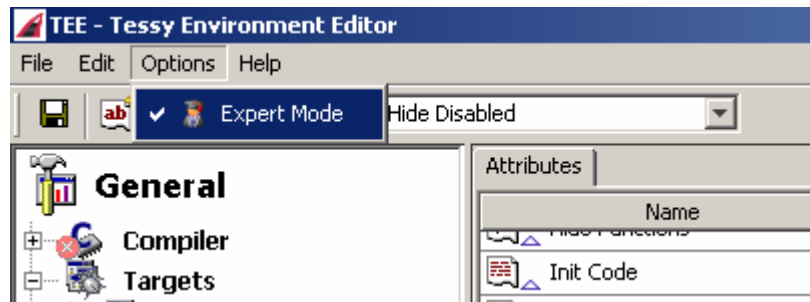


You may use different workspaces for different CPUs or different targets (i.e. simulator vs. emulator). In this case copy the existing system configuration to a configuration file (refer to the TEE online help for details) and change it to your preferred settings.

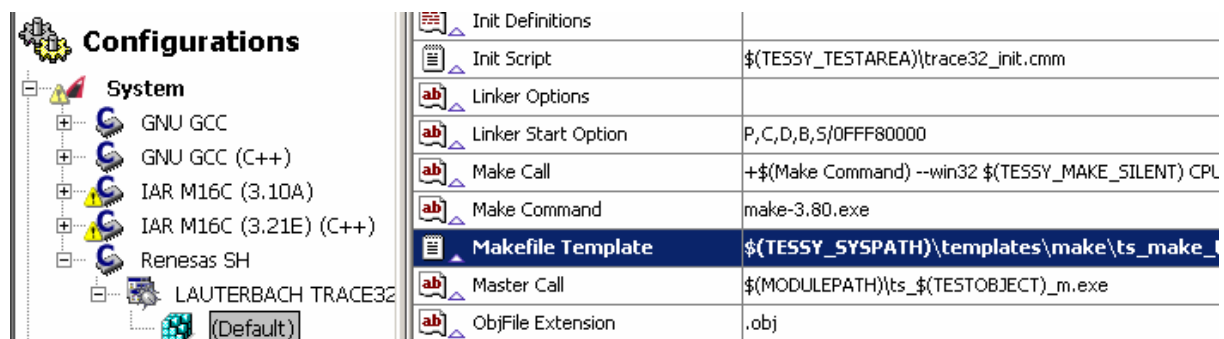
2.4 Step 4: Adapting the Makefile Template

The makefile template contains compiler options, the C library and startup files being used for linking the test application. The default template is prepared for the SH1 microcontroller. If you are using another controller or other settings for the compiler options, you need to adapt the makefile template.

The makefile template is specified within the attribute **Makefile Template** of the TEE. You need to switch on the expert mode in order to see this attribute within TEE. Choose **Expert Mode** from the **Options** menu of TEE to switch to this mode:



Please copy the default makefile template to your project directory if you make any changes. The changed file may then be selected within the attribute **Makefile Template**.



The following list of files may need to be adapted, depending on the names and number of files generated during step 1 above:

```

124 @echo ***** Linking Slave *****
125 @echo -noprelink > $(MODULE_PATH_DOS)\ts_${TESTOBJECT}.h1k
126 @echo -nomessage >> $(MODULE_PATH_DOS)\ts_${TESTOBJECT}.h1k
127 @echo -list="$(MODULE_PATH_DOS)\ts_${TESTOBJECT}.map" >> $(MODULE_PATH
128 @echo -nooptimize >> $(MODULE_PATH_DOS)\ts_${TESTOBJECT}.h1k
129 @echo -start=$(LINK_START_OPTION) >> $(MODULE_PATH_DOS)\ts_${TESTOBJEC
130 @echo -nologo >> $(MODULE_PATH_DOS)\ts_${TESTOBJECT}.h1k
131 @echo -input="$(WS_DEBUG_PATH)\resetprg.obj" >> $(MODULE_PATH_DOS)\ts_
132 @echo -input="$(WS_DEBUG_PATH)\vecttbl.obj" >> $(MODULE_PATH_DOS)\ts_
133 @echo -input="$(WS_DEBUG_PATH)\dbsct.obj" >> $(MODULE_PATH_DOS)\ts_
134 @echo -input="$(WS_DEBUG_PATH)\intprg.obj" >> $(MODULE_PATH_DOS)\ts_
135 @echo -input="$(WS_DEBUG_PATH)\sbrk.obj" >> $(MODULE_PATH_DOS)\ts_
136 @echo -input="$(MODULE_PATH_DOS)\ts_${TESTOBJECT}_s$(OBJECT_POSTFIX)"
137 @sh -c 'for f in $(S_SRC_OBJECTS); do echo -input="$$f" >> $(MODULE_PA
138 @echo -input="$(S_UC_OBJECT)" >> $(MODULE_PATH_DOS)\ts_${TESTOBJECT}.h
139 @echo -input="$(S_STUB_OBJECT)" >> $(MODULE_PATH_DOS)\ts_${TESTOBJECT}
140 @sh -c 'for f in $(S_CL_OBJECTS); do echo -input="$$f" >> $(MODULE_PAT
141 @sh -c 'for f in $(S_OBJECTS); do echo -input="$$f" >> $(MODULE_PATH D
142 @echo -library="$(WS_DEBUG_PATH)\ts_main.lib" >> $(MODULE_PATH_DOS)\t
143 @echo -output="$$@" >> $(MODULE_PATH_DOS)\ts_${TESTOBJECT}.h1k
144 @echo -end >> $(MODULE_PATH_DOS)\ts_${TESTOBJECT}.h1k
145 @echo -input="$$@" >> $(MODULE_PATH_DOS)\ts_${TESTOBJECT}.h1k

```

The list of startup code files may be different for other derivatives of the SH controller (e.g. SH4). In case of **SH4** you would need to add the file **vhandler.obj**. Simply duplicate the text line of another file and change the name accordingly.

Important note: *Be careful with any changes to the makefile template, because the makefile may easily get corrupted. We would suggest to make any changes only on copies of the original files.*

Please use an appropriate text editor that do not changed tab characters into spaces when saving!