

Customizing Makefile Templates

1 Abstract

This document describes the possible changes to the makefile templates in order to change compiler options, memory models or other customer specific settings. These changes will then apply to all modules with that certain compiler/target environment combination.

Table of Contents

1	Abstract	1
2	Introduction.....	2
2.1	Choosing the appropriate makefile template	2
2.2	Where to make changes	2
3	Common Changes	3
3.1	Compiler options	3
3.2	Linker options.....	3
3.3	Linker command file	4
3.4	Adding variables to the makefile.....	4

2 Introduction

For each compiler/target environment combination, there is a separate makefile template in the `sys\templates\make` directory of the TESSY installation. This template will be used to create the makefile necessary to build the test driver. You may change certain parts of the makefile template to incorporate your specific settings.

Please note: Be careful with any changes, since this could damage the makefile template and cause unexpected behavior of TESSY. Do not use editors that automatically replace tabs with blanks: This would also damage the makefile.

2.1 Choosing the appropriate makefile template

The name of the makefile template depends on the corresponding compiler, microcontroller and debugger/emulator to be used. For each such combination, there is a makefile template within the `sys\templates\make` directory of the TESSY installation. The following naming convention applies:

```
ts_make_<debugger>_<compiler>_<microcontroller>.tpl
```

As an example:

```
ts_make_hitop_keil_c166.tpl
```

2.2 Where to make changes

You may change the makefile templates in the following designated sections:

```
#BEGIN_DEFINES
Here you may add defines or change existing ones.
#END_DEFINES

#BEGIN_TARGETS
Here you may add rules or change existing ones.
#END_TARGETS
```

The **DEFINES** section holds the compiler and linker options. If you want to add your optimization settings or change the memory model, this would be the right location.

The **TARGETS** section contains all rules to build the test driver. For some compilers, there is a rule to create a linker command file, containing all the linker options, object files and libraries. You would need to changes this rule to make changes to the linker command file.

3 Common Changes

The examples described within the next sections will probably be the most common changes made to the makefile templates. If you need further assistance, please contact Razorcat technical support.

3.1 Compiler options

If you want to change the memory model or the optimizations used for compiling the test driver and your source files, you need to change the following **S_COMP_OPTIONS** entry within the makefile template (e.g. `ts_make_hitop_keil_c166.tpl`):

```
#
# SLAVE
# (You may change these settings to use your own compiler/linker opt
#
S_INCLUDES      := $(TESSY_SYS_DOS)\include\tessy\comm
S_ASM_INCLUDES := $(TESSY_KEILC166_HOME)\ASM
S_COMP_OPTIONS := SMALL DEBUG OT "(0)" MOD167 WARNINGLEVEL "(0)" INC
S_LINK_OPTIONS :=
```

These changes will be in effect with the next test driver generation. The makefile for your module will be generated from this template.

3.2 Linker options

There may be multiple locations, which you may want to change to enter your specific linker options. One define is the **S_LINK_OPTIONS** shown in the picture above. You may want to add some special libraries, then just add them to this define.

Other defines may be present, which specify the classes or memory sections to be applied. This depends on the compiler used. You should only make changes in this part, if you are familiar with the syntax and behavior of the respective compiler/linker.

```
CLASS_RANGE     := CLASSES { NCONST (200h-3FFFh), ICODE (200h-3FFFh)
SEC_RANGE       := SECTIONS (?C_INITSEC (200H), ?C_CLRMEMSEC)
LOFLAGS        := RESERVE (4-1FFh) IXREF $(CLASS_RANGE) $(SEC_RANGE
```

Other linker settings will be generated into in the linker command file described below.

3.3 Linker command file

Some linkers require a linker command file to be used to pass linker options. Refer to your compiler manual for details. An example for such a linker command file can be found in the `ts_make_hitop_keil_c166.tpl`:

```
$(MODULE_PATH_DOS)\ts_$(TESTOBJECT)_s.hex : $(MODULE_PATH_DOS)\ts_t_s.
    $(S_SRC_OBJECTS) \
    $(S_UC_OBJECT) \
    $(S_STUB_OBJECT) \
    $(S_C1_OBJECTS) \
    $(MODULE_PATH_DOS)\ts_sta
    $(S_OBJECTS)

@echo *
@echo ***** Linking Slave *****
@echo $(MODULE_PATH_DOS)\ts_t_s.obj, > $(MODULE_PATH_DOS)\ts_$(TEST
@sh -c 'for f in $(S_OBJECTS); do echo "$$f," >> $(MODULE_PATH_DOS)
@sh -c 'for f in $(S_SRC_OBJECTS); do echo "$$f," >> $(MODULE_PATH_
@echo $(S_UC_OBJECT), >> $(MODULE_PATH_DOS)\ts_$(TESTOBJECT).lin
@echo $(S_STUB_OBJECT), >> $(MODULE_PATH_DOS)\ts_$(TESTOBJECT).lin
@sh -c 'for f in $(S_C1_OBJECTS); do echo "$$f," >> $(MODULE_PATH_D
@sh -c 'for f in $(USER_LINK_OPTIONS); do echo "$$f," >> $(MODULE_P
@echo $(TESSY_KEILC166_HOME)\LIB\C167S.LIB >> $(MODULE_PATH_DOS)\ts
@echo TO $@ >> $(MODULE_PATH_DOS)\ts_$(TESTOBJECT).lin
@echo $(LOFLAGS) >> $(MODULE_PATH_DOS)\ts_$(TESTOBJECT).lin
```

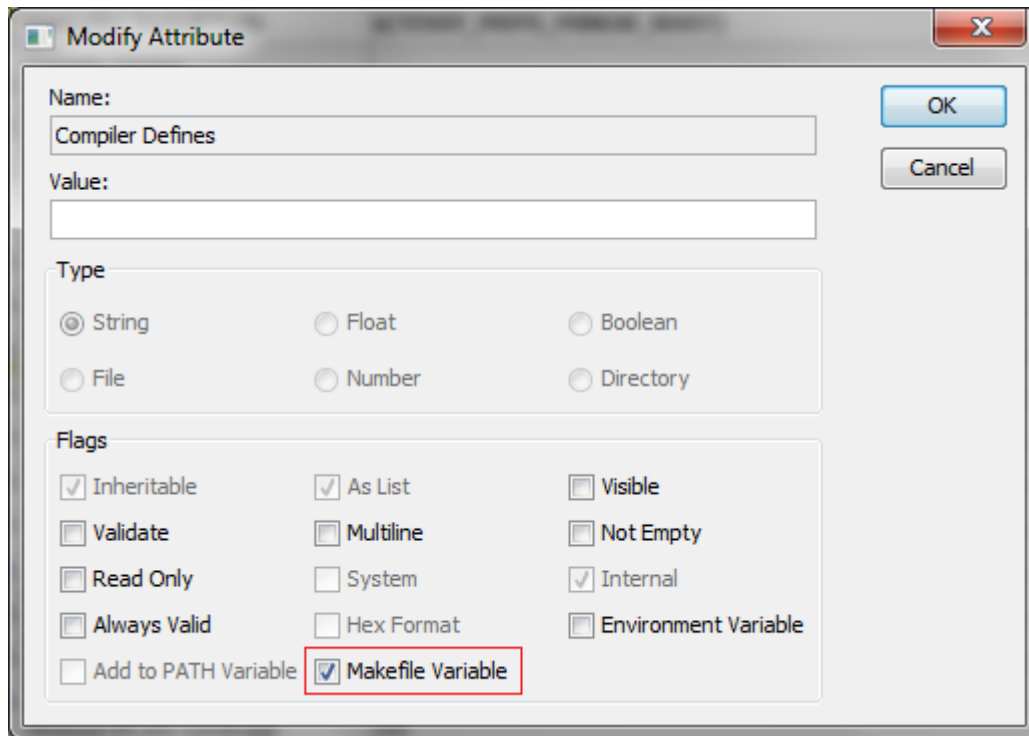
Before calling the linker to build the test driver binary, the linker command file `$(MODULE_PATH_DOS)\ts_$(TESTOBJECT).lin` will be generated from this makefile rule.

Please note: The linker command file will be generated using the `echo` command with the redirection operator `>` and `>>`. The first line needs to be written with the `>` to overwrite the old linker command file, all subsequent echos need to use the `>>` operator to append lines to the newly created file.

The defines from the upper section of the makefile template will be used to create this linker command file. If you need to change anything, you may want to add new defines in the defines section like described above and to use them in this linker command file rule.

3.4 Adding variables to the makefile

You may add own variables to the makefile within the environment editor (TEE). Each TEE attribute may be marked to be used within the makefile with the “Makefile Variable” flag as shown below:



Modify Attribute

Name:
Compiler Defines

Value:

Type

String Float Boolean
 File Number Directory

Flags

Inheritable As List Visible
 Validate Multiline Not Empty
 Read Only System Internal
 Always Valid Hex Format Environment Variable
 Add to PATH Variable Makefile Variable

OK
Cancel

All TEE attributes that are marked as “Makefile Variable” will be printed into the generated makefile.