

How to setup pre-build steps in CodeWarrior for Microcontrollers v10.x

1. Introduction

This document outlines the steps for setting up user-defined dependencies and pre-build steps in CodeWarrior for Microcontrollers v10.x.

An MC9S08QE128 CodeWarrior v10.5 project is used as an example to configure the pre-build settings from within the CodeWarrior IDE.

NOTE: The steps described in this document can be applied to any other architecture.

Contents

1. Introduction.....	1
2. Build steps settings.....	1
2.1. Create a dummy project.....	1
2.2. Pre-build steps	2
2.3. Solution for user-defined pre-build steps.....	3

2. Build steps settings

- Input the commands or a script to be executed before the execution of the build
- If needed, update related project settings

2.1. Create a dummy project

Create a new dummy project with the project wizard for MC9S08QE128.

To create a project:

Build steps settings

1. Select **File > New > Project... > Bareboard Project**.
2. Provide a name to the project and follow the steps in the project wizard.

2.2. Pre-build steps

This option identifies any steps that occur before the build takes place. The pre-build steps are executed only if the state of the main build is not up to date. An attempt to execute the main build occurs regardless of the success or failure of executing the pre-build step.

One or more commands are specified that execute immediately before the execution of the build. The build tools use the makefile data and the last-modification times of the files to decide which of the files need to be updated. The output of the pre-build steps can be saved into a file and used as a dependency to the build tools. However, this may incur some unwanted building effort if the pre-build commands only update the last-modification time of this file but its content. In such a case, all the source files will be rebuilt no matter if the source files are changed or not.

The MC9S08QE128 project created before is used to show you this problem and the method we take to work around it.

In this MC9S08QE128 project the pre-build commands create a user defined pre.prm file, which is in turn a dependency for the linker. The only thing changed in this project is the last-modification time of the pre.prm file created in the pre-build steps. The changes of the last-modification time of the pre.prm will cause a full rebuilding when a building is performed.

The following figure shows the command added to the MC9S08QE128 project for creating a user defined PRM file.

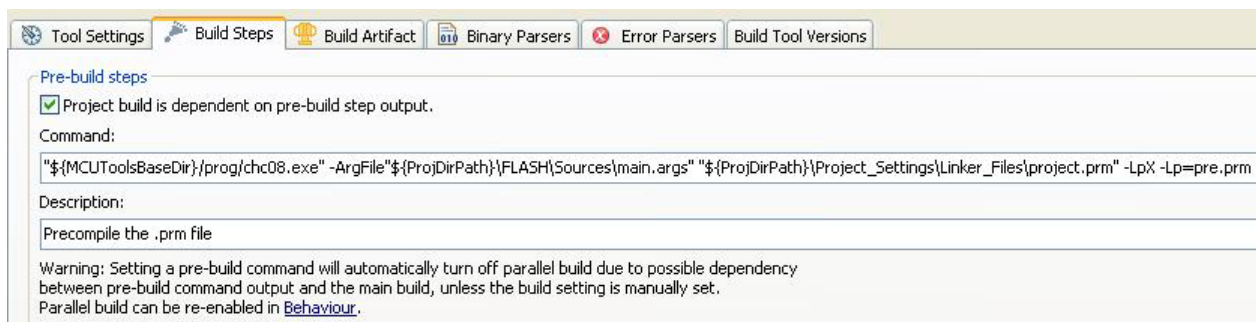


Figure 1. Build Steps

The following figure shows the input option of the S08 linker in the MC9S08QE128 project. The pre.prm file created before is used as the parameter file to the S08 linker.

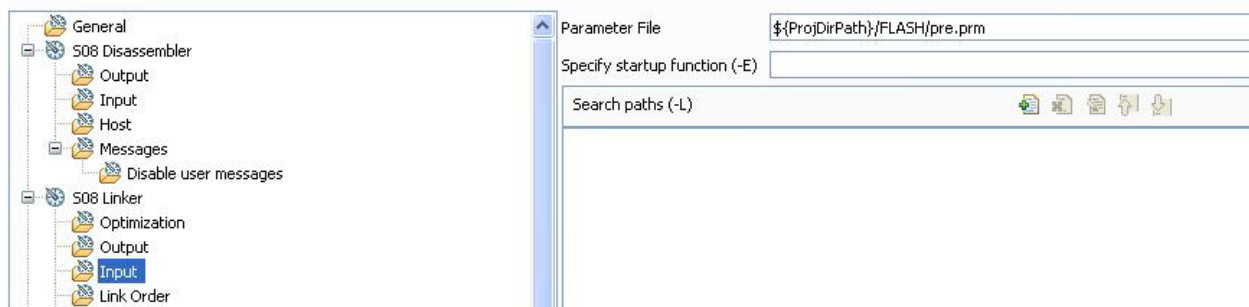


Figure 2. Input of S08 Linker

2.3. Solution for user-defined pre-build steps

In order to solve this problem, it would be required to avoid modifying the last-modification time of pre.prm if the content is the same. A script is executed in the pre-build phase to check if the content of pre.prm file is modified or not.

The first step is to create a script file, for example prebuild.bat:

```
""%1"\chc08.exe" -ArgFile"%2\FLASH\Sources\main.args"
"%2\Project_Settings\Linker_Files\Project.prm" -LpX -Lp=pre.prm.tmp

if not exist pre.prm goto differs

fc pre.prm.tmp pre.prm > nul
if errorlevel 1 goto differs

goto end

:differs
copy /y pre.prm.tmp pre.prm

:end
```

Then, set the pre-build command to

```
"${ProjDirPath}\prebuild.bat" "${MCUToolsBaseDir}/prog/" ${ProjDirPath}
```

Select **Properties > C/C++ Build > Settings > Build Steps**.

Uncheck the **Project build is dependent on pre-build step output** option

Build steps settings

Tool Settings | **Build Steps** | Build Artifact | Binary Parsers | Error Parsers | Build Tool Versions

Pre-build steps

Project build is dependent on pre-build step output.

Command:
`"${ProjDirPath}\\prebuild.bat" "${MCUToolsBaseDir}\\prog\\" "${ProjDirPath}"`

Description:
 Precompile the .prm file

Warning: Setting a pre-build command will automatically turn off parallel build due to possible dependency between pre-build command output and the main build, unless the build setting is manually set. Parallel build can be re-enabled in [Behaviour](#).

Figure 3. Pre-build steps

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions.

Freescale, the Freescale logo, CodeWarrior are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2014.