How to support multiple targets in CodeWarrior(Classic, Eclipse)/KDS

By Jennie Zhang

Contents

In some application, we hope one project can support multiple targets. Some typical examples: user needs different firmware to run on different PCB with different crystal on it; Or sometimes user needs one project support both debug and release mode; another widely used example is, user needs one project support multiple part numbers.etc.

To implement the multiple targets project, we will consider at least three aspects:

- Depend on the different targets, some files/codes may be included or not.
- Depend on the different targets, compiler/linker settings may be different.
- If bootloader involved, different target may link with different bootloader firmware.

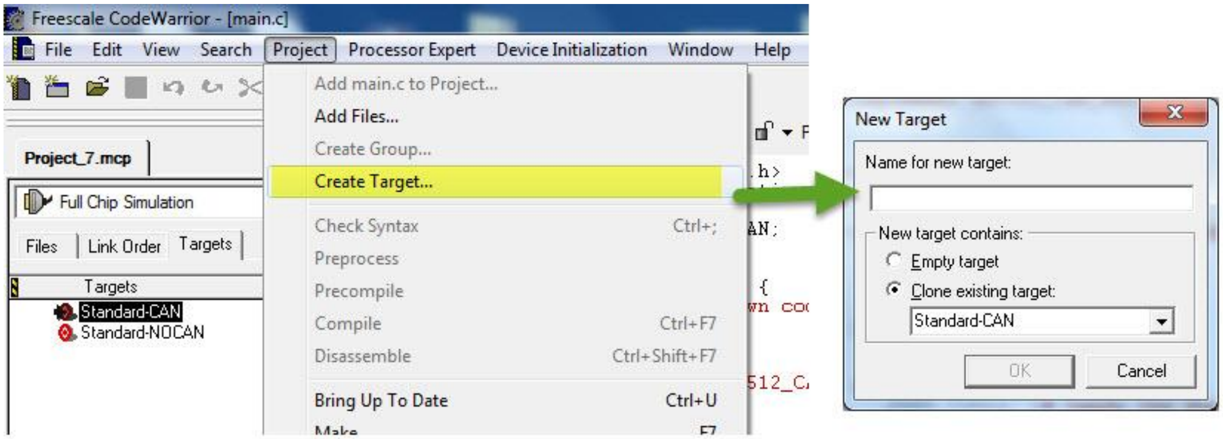
In the following article I will discuss this topic with Classic CodeWarrior and Eclipse CodeWarrior10.x/KDS separately.

1. Multiple Target Support in Classic CodeWarrior

   1.1 One file identifying multiple targets

   We will discuss it with example step by step (based on CodeWarrior for HCS12(X) v5.1):

   Step 1. Create a new target, the two targets name can be "Standard-CAN","Standard-NOCAN"



New Target

Name for new target:

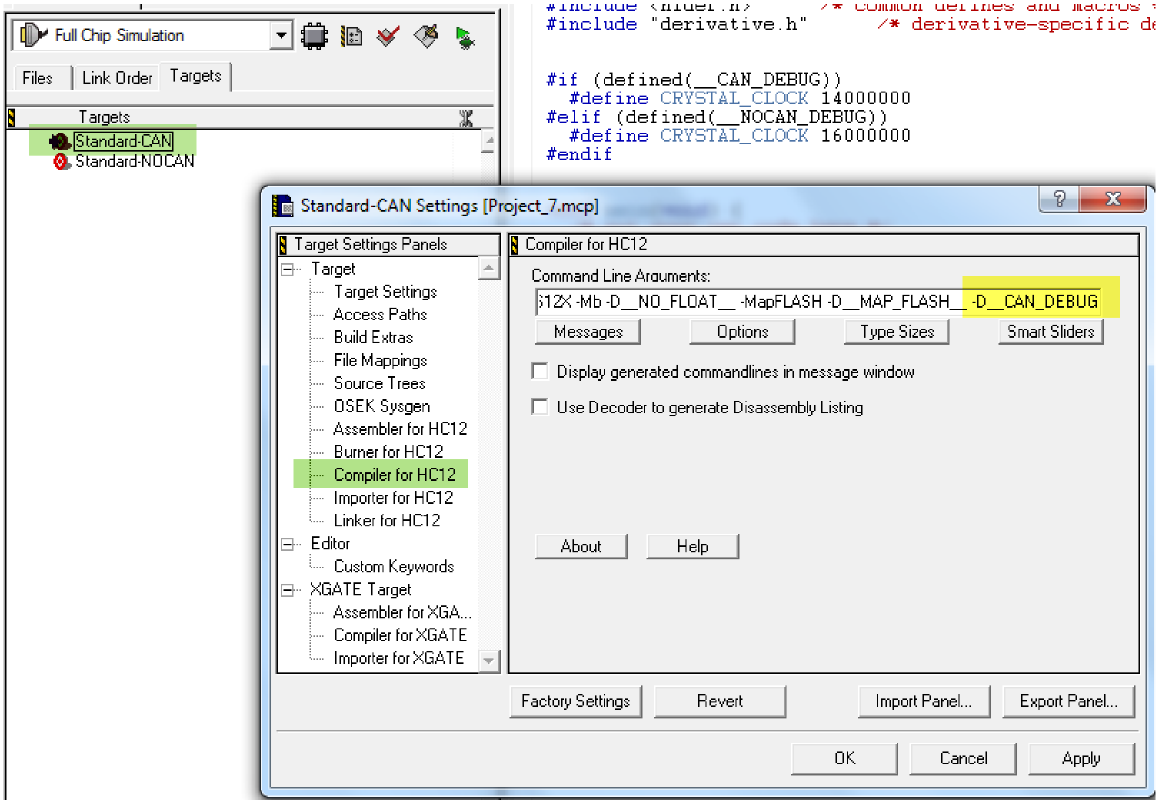
New target contains:

○ Empty target

● Clone existing target:

Standard-CAN

   Step 2. Set macro in each of the target compiler command line. For target Standard-CAN, define macro __CAN_DEBUG; for target Standard-NOCAN, define macro __NOCAN_DEBUG.



```
#include <hidef.h>      /* common defines and macros *
#include "derivative.h"      /* derivative-specific de

#if (defined(__CAN_DEBUG))
  #define CRYSTAL_CLOCK 14000000
#elif (defined(__NOCAN_DEBUG))
  #define CRYSTAL_CLOCK 16000000
#endif
```

Standard-CAN Settings [Project_7.mcp]

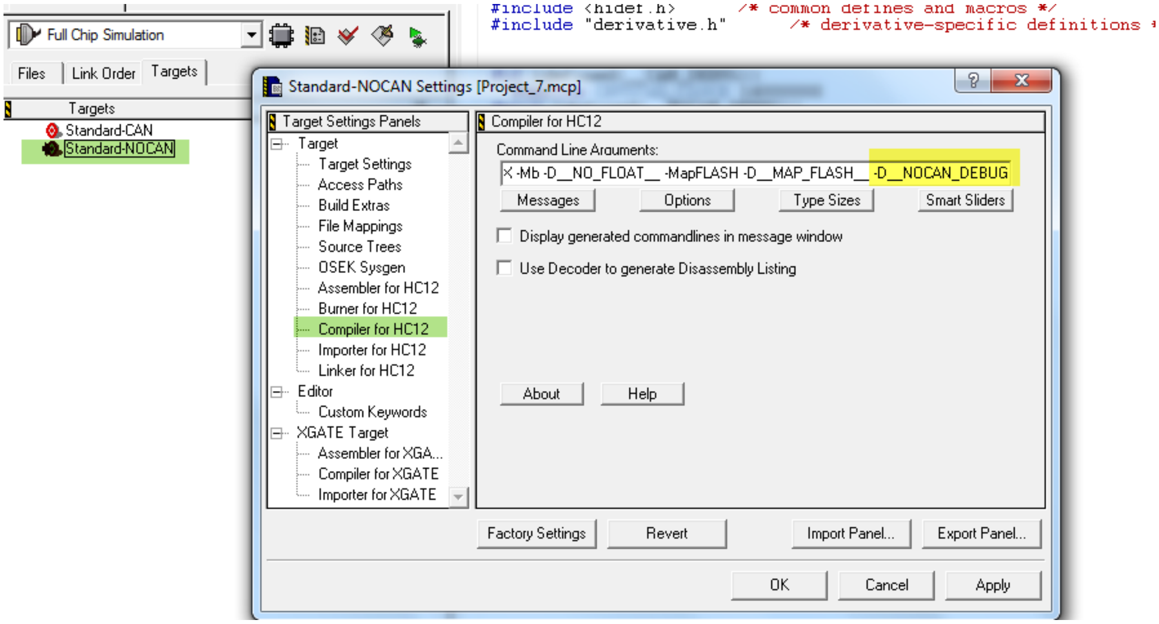
Compiler for HC12

Command Line Arguments: 12X -Mb -D__NO_FLOAT__ -MapFLASH -D__MAP_FLASH__ -D__CAN_DEBUG

□ Display generated commandlines in message window

□ Use Decoder to generate Disassembly Listing

Compiler command line setting for Standard-CAN target



```
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions *
```

Standard-NOCAN Settings [Project_7.mcp]

Compiler for HC12

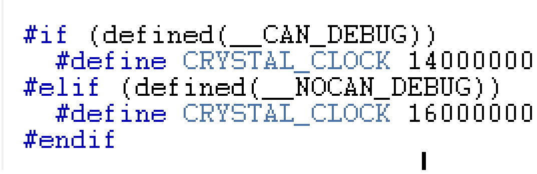
Command Line Arguments: X -Mb -D__NO_FLOAT__ -MapFLASH -D__MAP_FLASH__ -D__NOCAN_DEBUG

□ Display generated commandlines in message window

□ Use Decoder to generate Disassembly Listing

Compiler command line setting for Standard-NOCAN target

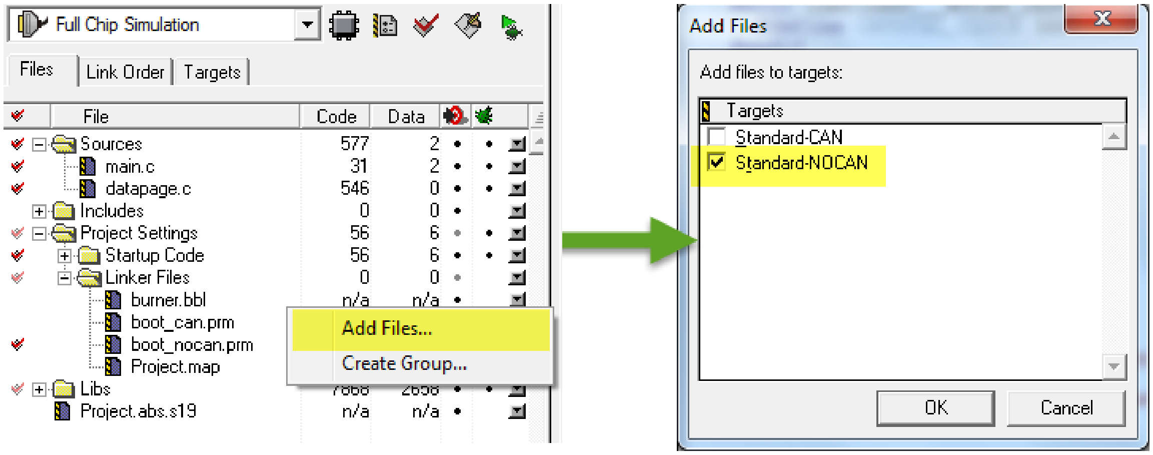
Step 3. in C code, use this code to identify different target



```
#if (defined(__CAN_DEBUG))
  #define CRYSTAL_CLOCK 14000000
#elif (defined(__NOCAN_DEBUG))
  #define CRYSTAL_CLOCK 16000000
#endif
```

1.2 One file for specific target only

Right click in project management window, “Add Files…”, choose the selected file we need to add, then “Add Files” windows will pop up: here, if the file is only valid to Standard-NOCAN target, we check “to Standard-NOCAN” only.

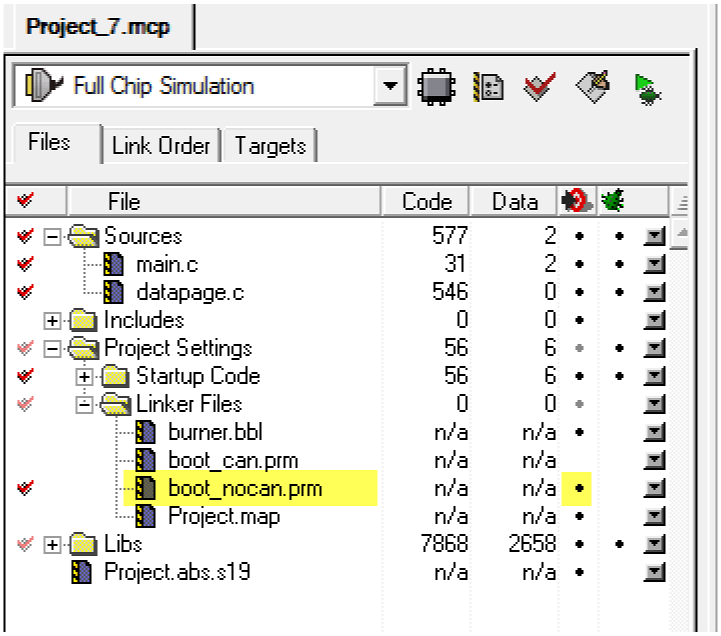


✓	File	Code	Data		
✓	Sources	577	2	•	•
✓	main.c	31	2	•	•
✓	datapage.c	546	0	•	•
	Includes	0	0	•	
✓	Project Settings	56	6	•	•
✓	Startup Code	56	6	•	•
✓	Linker Files	0	0	•	
	burner.bbl	n/a	n/a	•	
	boot_can.prm				
✓	boot_nocan.prm				
	Project.map				
✓	Libs	7668	2656	•	•
	Project.abs.s19	n/a	n/a	•	

Add files to targets:

Targets
- ☐ Standard-CAN
- ☑ Standard-NOCAN

After above step you will see the black dot show under the target column for Standard-NOCAN target only. This means boot_nocan.prm is only available for Standard-NOCAN target.



✓	File	Code	Data		
✓	Sources	577	2	•	•
✓	main.c	31	2	•	•
✓	datapage.c	546	0	•	•
	Includes	0	0	•	
✓	Project Settings	56	6	•	•
✓	Startup Code	56	6	•	•
✓	Linker Files	0	0	•	
	burner.bbl	n/a	n/a	•	
	boot_can.prm	n/a	n/a		
✓	boot_nocan.prm	n/a	n/a	•	
	Project.map	n/a	n/a	•	
✓	Libs	7868	2658	•	•
	Project.abs.s19	n/a	n/a	•	

The same method is also workable for .c, .asm, .prm,.h,etc file.

Thus we can set different prm file to different target. User can even link different s19 file in each target prm file. This is useful when different targets need to link different bootloader files. For example with below code in boot_nocan.prm file for Standard-NOCAN:

```
HEXFILE boot_nocan.s19
```

when build Standard-NOCAN target, boot_nocan.s19 will be merged in finial generated project s19 file Standard-NOCAN target only.

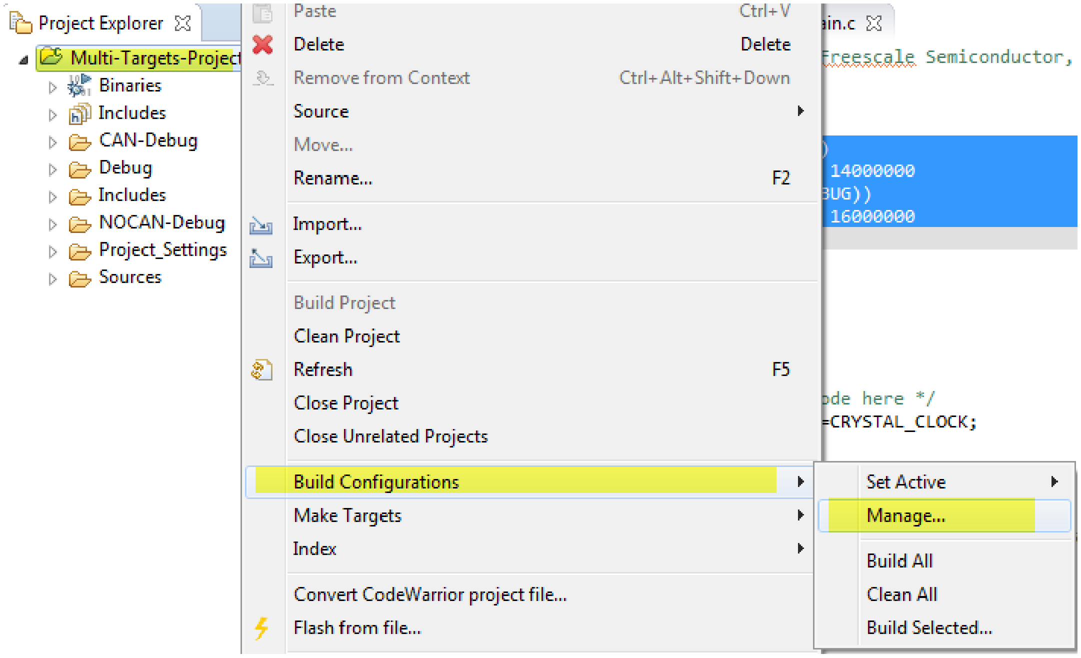
2. Multiple Build Configuration Support in Eclipse CodeWarrior and KDS

2.1 One file identifying Multiple build configurations

Eclipse CodeWarrior and KDS use “Build Configuration” which is similar with the “Target” in Classic CodeWarrior.

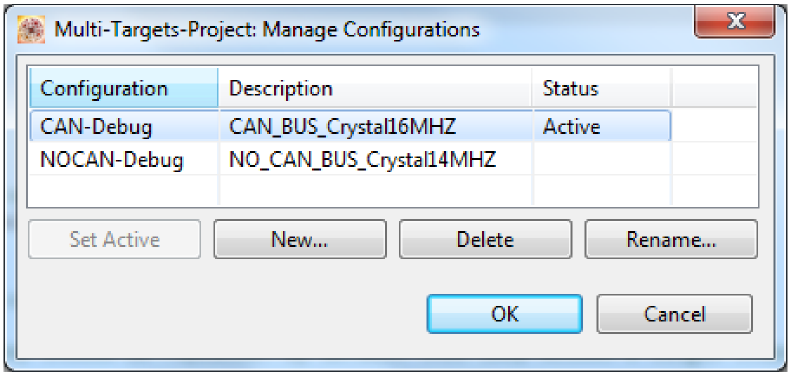
We will discuss it with example step by step (based on KDS3.2):

Step 1. Create a new build configuration. Right click on the project name, “Build Configurations”, “Manage…”, “New” to create a new Build Configuration.



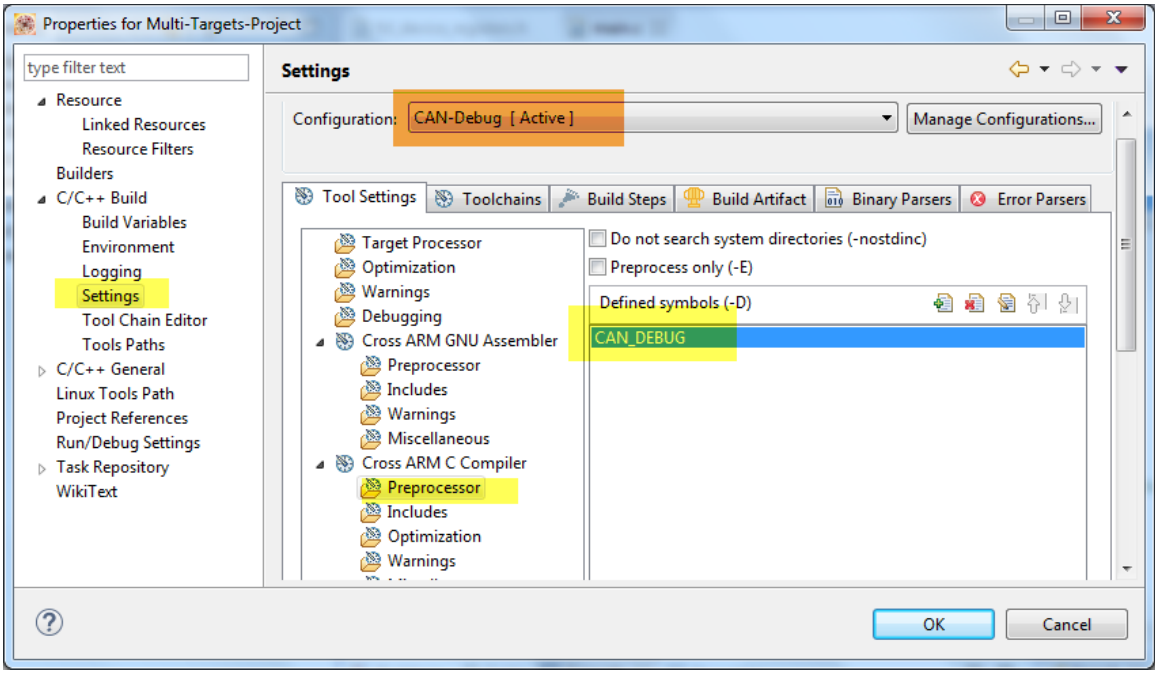
```
Freescale Semiconductor,
)
 14000000
BUG))
 16000000
ode here */
=CRYSTAL_CLOCK;
```

Now two build configurations are set well:



Configuration	Description	Status
CAN-Debug	CAN_BUS_Crystal16MHZ	Active
NOCAN-Debug	NO_CAN_BUS_Crystal14MHZ	

Step 2. Set macro in each of the target compiler command line. For Build Configuration CAN-Debug, define macro CAN_DEBUG; for Build Configuration NOCAN-Debug, define macro NOCAN_DEBUG.



Settings

Configuration: CAN-Debug [Active]

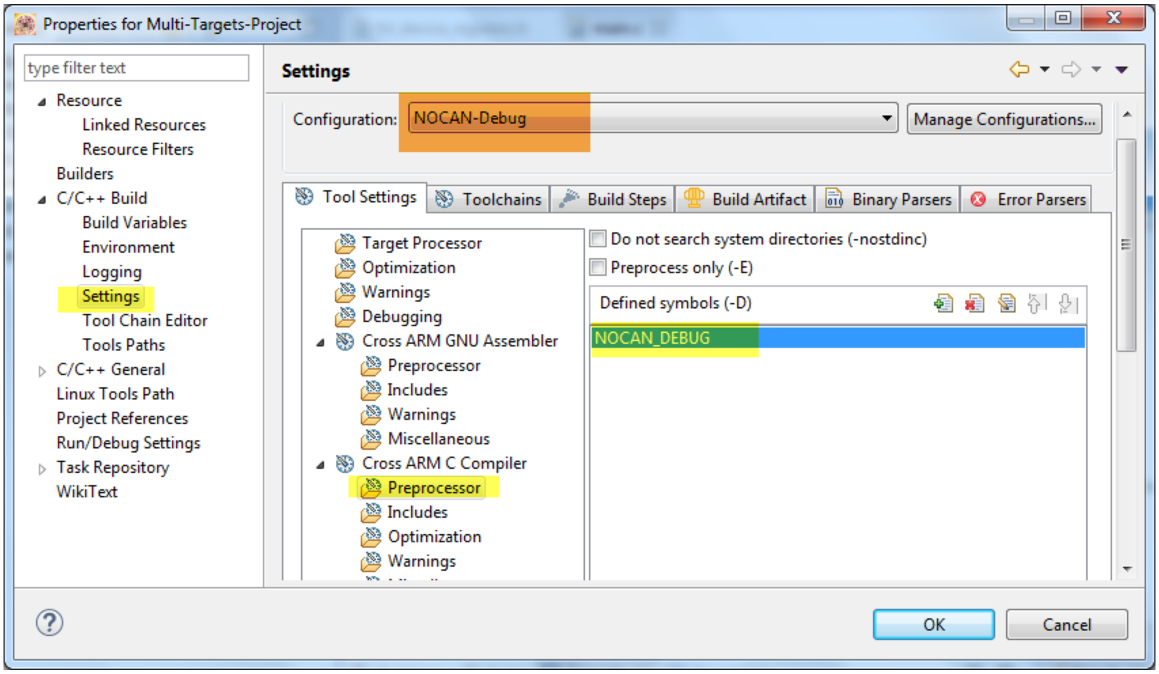
□ Do not search system directories (-nostdinc)

□ Preprocess only (-E)

Defined symbols (-D)

CAN_DEBUG

Preprocessor setting for “CAN-Debug” Configuration



Settings

Configuration: NOCAN-Debug

□ Do not search system directories (-nostdinc)

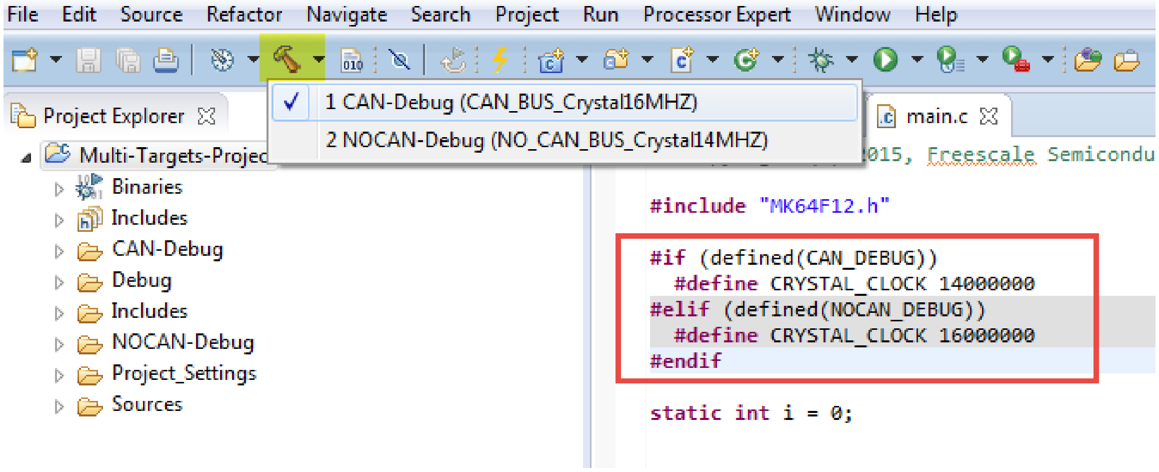
□ Preprocess only (-E)

Defined symbols (-D)

NOCAN_DEBUG

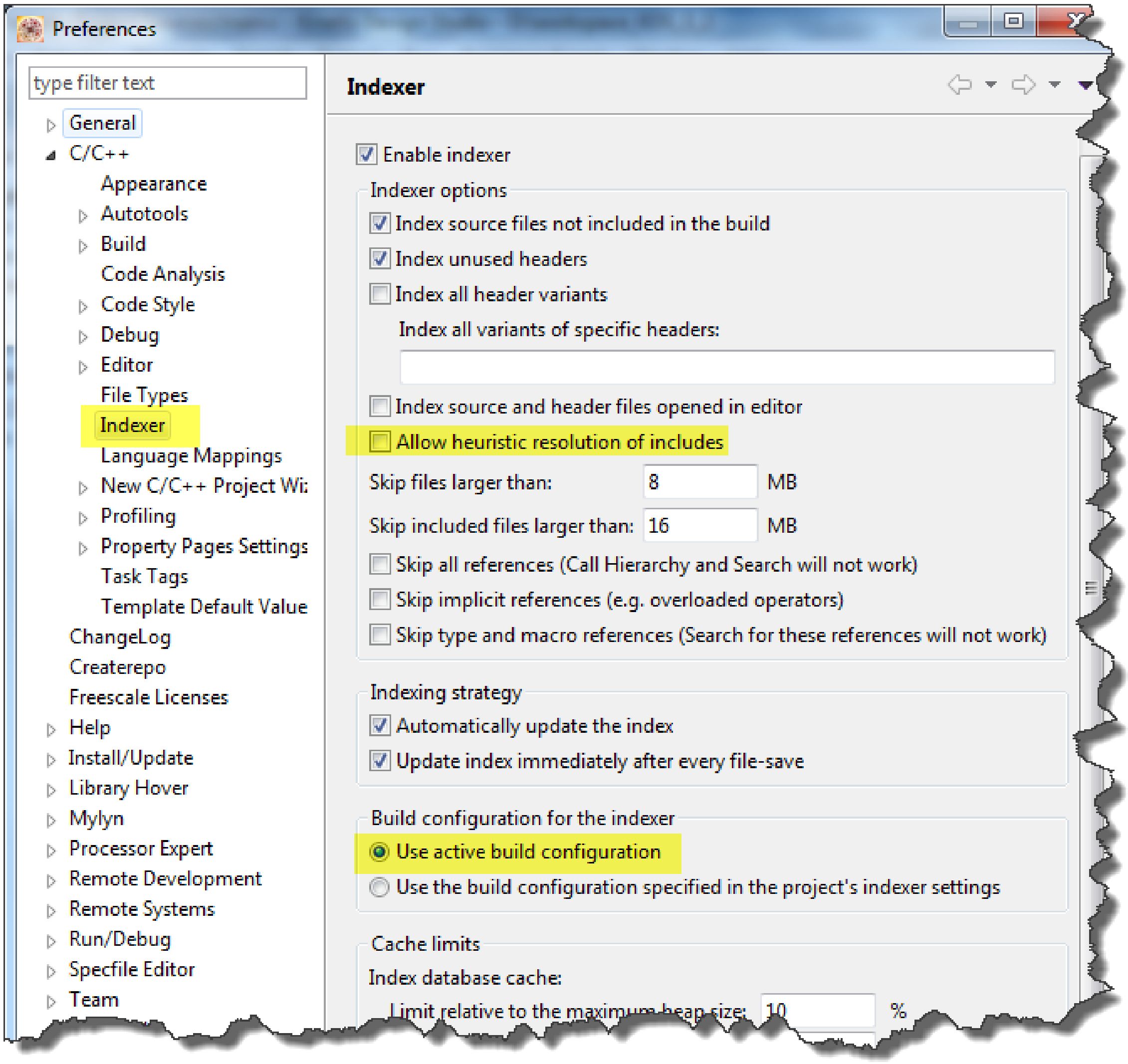
Preprocessor setting for “NOCAN-Debug” Configuration

Step 3. in C code, use following code to identify different build configuration in eclipse.



```
015, Freescale Semicondu

#include "MK64F12.h"

#if (defined(CAN_DEBUG))
  #define CRYSTAL_CLOCK 14000000
#elif (defined(NOCAN_DEBUG))
  #define CRYSTAL_CLOCK 16000000
#endif

static int i = 0;
```

Please note, to well work with this feature, please disable “heuristic resolution” and enable “use active build configuration in workspace preference setting:



Indexer

☑ Enable indexer

Indexer options

☑ Index source files not included in the build

☑ Index unused headers

□ Index all header variants

Index all variants of specific headers:

□ Index source and header files opened in editor

□ Allow heuristic resolution of includes

Skip files larger than: 8 MB

Skip included files larger than: 16 MB

□ Skip all references (Call Hierarchy and Search will not work)

□ Skip implicit references (e.g. overloaded operators)

□ Skip type and macro references (Search for these references will not work)

Indexing strategy

☑ Automatically update the index

☑ Update index immediately after every file-save

Build configuration for the indexer

● Use active build configuration

○ Use the build configuration specified in the project's indexer settings

Cache limits

Index database cache:

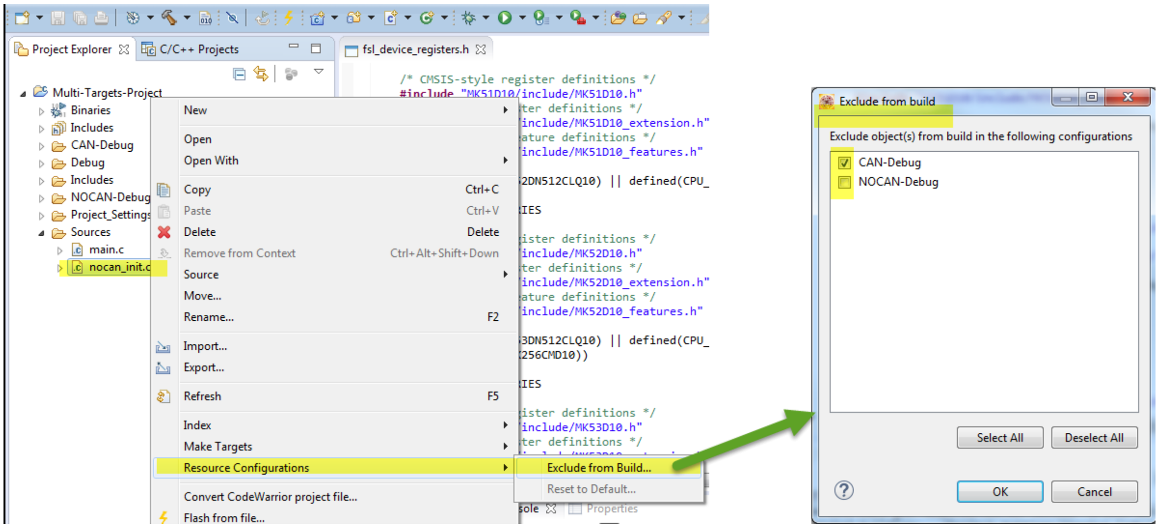
Limit relative to the maximum heap size: 10 %

For more information on this topic, See Article from Erich:

https://mcuoneclipse.com/2012/03/20/fixing-the-eclipse-index/

2.2 One file for specific build configuration only

By default, Eclipse CodeWarrior/KDS project build all the files existing in the project folder. If we want to exclude a file from one build configuration, for instance, exclude nocan_init.c from Build Configuration “CAN-Debug”, right click on the file, then choose “Resource Configurations”, “Exclude from build..” check “CAN-Debug”. Thus, file nocan_init.c is for build configuration “NOCAN-Debug” only.

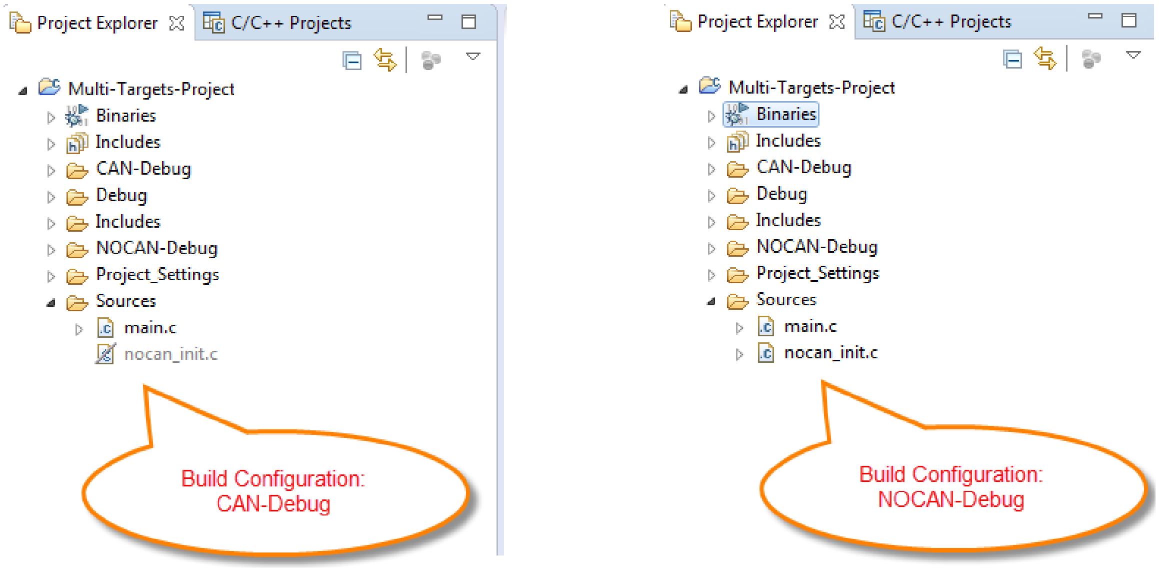


```
/* CMSIS-style register definitions */
#include "MK51D10/include/MK51D10.h"
ter definitions */
include/MK51D10_extension.h"
ature definitions */
include/MK51D10_features.h"

2DN512CLQ10) || defined(CPU_

IES

ister definitions */
include/MK52D10.h"
ter definitions */
include/MK52D10_extension.h"
ature definitions */
include/MK52D10_features.h"

3DN512CLQ10) || defined(CPU_
256CMD10))

IES

ister definitions */
include/MK53D10.h"
ter definitions */
```

Exclude object(s) from build in the following configurations

☑ CAN-Debug
☐ NOCAN-Debug

after that you will see nocan_init.c is gray out with CAN-Debug configuration. This means nocan_init.c is only available for NOCAN-Debug.



The same method is also workable for .c, .asm, .prm,.ld,.h file.

Thus we can set different build command line for different build configuration. User can even link different s19 file in Post-build. This is useful when different application build configuration needs to link different bootloader files. Just link the bootloader burning file in each build configuration command line accordingly.

For more information about this topic, pefer this article for how-to merge multiple s19 file with GNU objcopy:

https://mcuoneclipse.com/2013/10/29/s-record-manipulation-with-gnu-objcopy-and-burner-utility/