HYCON TECHNOLOGY

HY16F Series

IDE Software Optimum Instruction Manual

© 2015 HYCON Technology Corp.

目錄

1. INTRODUCTION 4

2. OPTIMUM INTRODUCTION AND SETTING 4

3. EXTRA NOTES FOR OPTIMIZATION SETTING APPLICATIONS 6

3.1 ADJUSTMENT PROBLEMS 6

3.2 INQUIRY FOR SECTIONS REMOVED 6

3.3. MEASURES TO PREVENT SOME PROGRAM CODES TO BE OPTIMIZED. 9

4. REFERENTIAL DOCUMENT 15

5. AMENDMENT RECORD 15

Notes:

1、 With the advancement of our product, content in this instruction manual might be amended without prior notification. Customers are welcomed to constantly download at our company's website http://www.hycontek.com

2、 Regarding to illustrations and applied circuits in this specification manual, our company is not responsible for problems arise from industrialownership in the third party.

3、 In situations where our product is applied individually, our company guarantees that its performance, typical application and function are consistent with conditions specified in the instruction manual. In situations where our product is applied in our client's product or equipment, our company does not guarantee the aforementioned conditions. We further recommend our clients to conduct sufficient evaluation and testing before doing so.

4、 Please pay extra note to application conditions in input voltage, output voltage, and load current, so that internal IC consumption does not surpass allowable consumption in the packaging. If our clients were to use our products under conditions where set values in the instruction manual were surpassed, even if immediately, our company is not responsible for the loss thus arise.

5、 Even though antistatic protective circuit is installed within this product, please do not exert excessive electrostatic which surpasses protective circuit performance.

6、 Without written permission, the product in the specification manual is not allowed to be applied in highly reliable electrical circuit, including instruments or devices affecting human bodies, such as medical instruments, disaster prevention instruments, vehicle instruments, loading instruments, and aircraft instruments.

7、 Our company has been dedicated to increasing quality and reliability in our products, thus possible failure rates present in all our semiconductor products. These failure rates might lead to some personal or fire accidents. Therefore, while designing products, please pay extra attention to redundancy design and adopt safe indicator, so as to prevent such accidents from happening.

8、 Without our company's permission, contents in this specification manual are prohibited to be transferred or copied for other purpose.

© 2015 HYCON Technology Corp.

1. Introduction

This article mainly introduces Andesight optimum settings and other relevant items.

2. Optimum Introduction and Setting

Setting Steps:

Upon opening project, please select project in the menu before choosing properties. Illustration below will show up. Next, select settings before choosing optimization. Users can choose the **required optimization levels within optimization.**

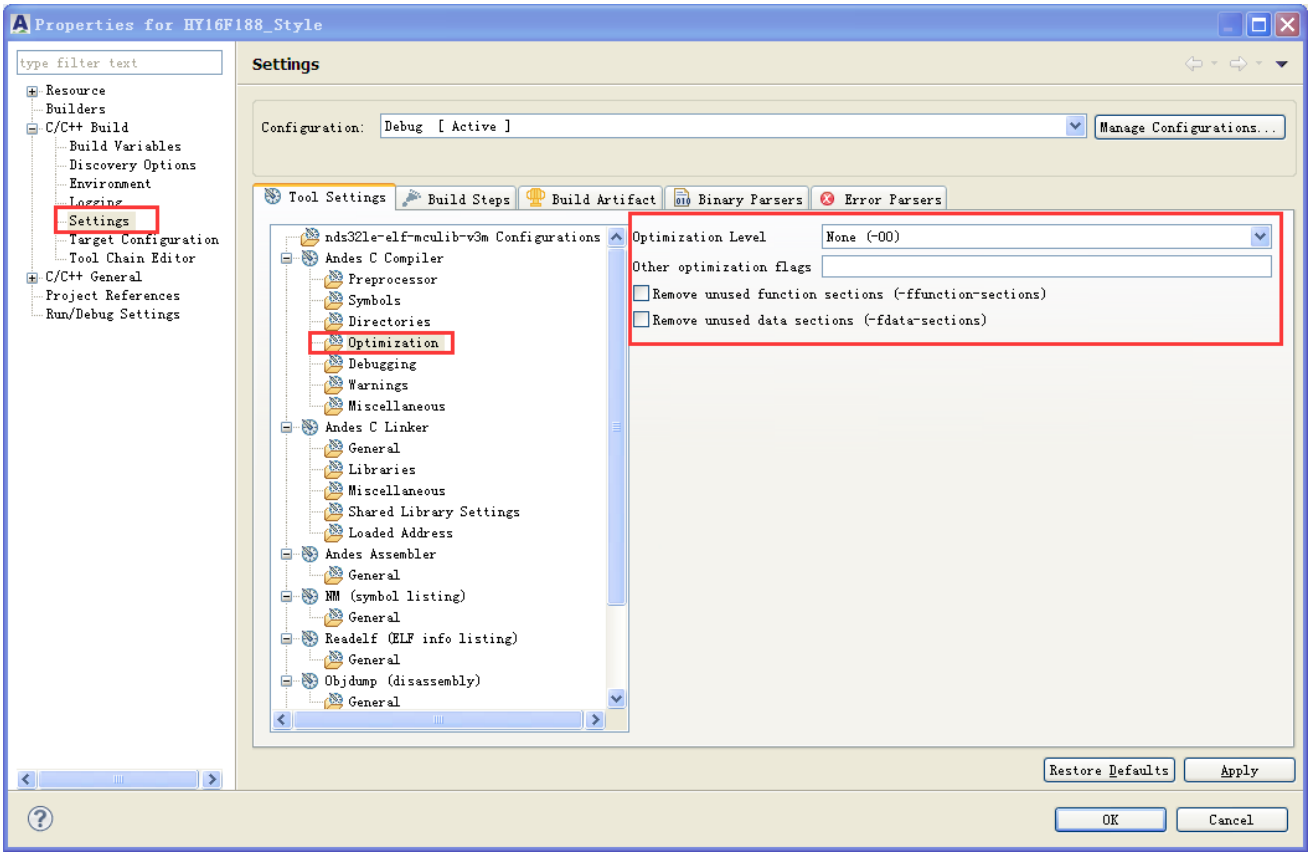


Illustration 1

Regarding to the definition in optimization level, please refer to the messages below.

According to occupied capacity and execution speed, Andesight has divided optimization levels into three categories.

Under general conditions, -Os3 and remove unused section will be selected for optimization in program code capacity, while –O3 and –funroll-loops will be selected for optimization in execution speed. (Please refer to illustration 2.)

(Optimize speed)-O/O1: GCC willexecite optimization in reducing code size and execution time. As to optimization items which severely affect compilation time, optimization in this specific level will not be implemented.

(Optimize speed more)-O2: Advanced level for –O. In this level, GCC will provide

© 2015 HYCON Technology Corp.

optimization in all supports, replacement of time with space excluded. Compiler will not use loop unwinding and Inline function. Compared to -O, this option speeds up performance in compilation time and generated code.

(Optimize speed most)-O3: -O2 provides optimization options. In addition to that options such as –finline-functions, -funswitch-loops, and –fgcse-after-reload are all designated. –O3 optimazationlevel increases speed in the generated executable document, but at the same time, it might also increase its size. Under certain circumstances, speed in program operation is reduced.

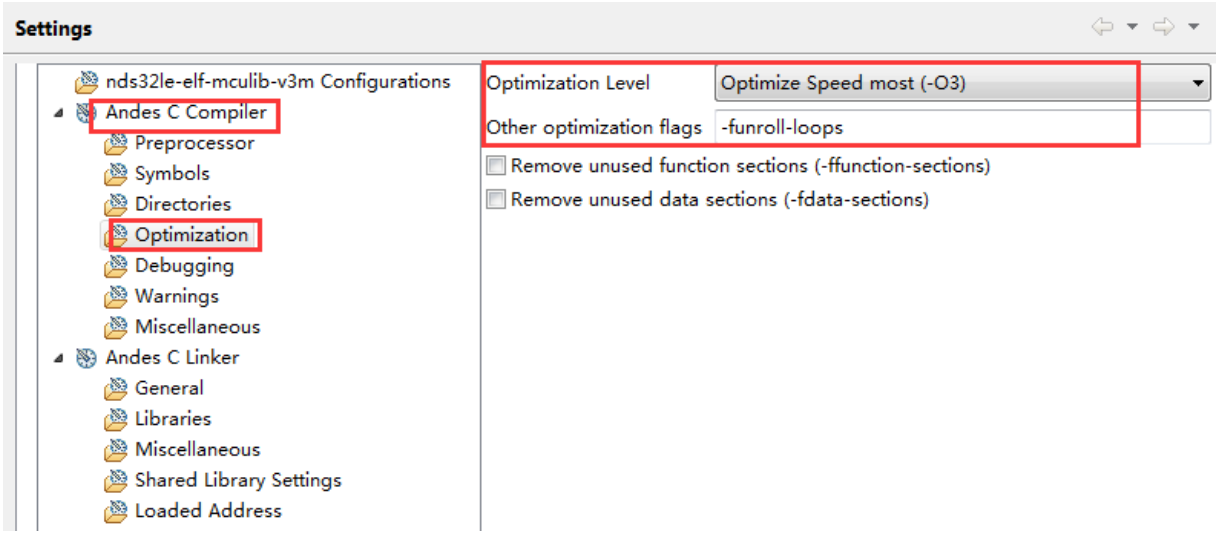
(Optimize size most)-Os3(-Os): This option is applied to optimize code size. –Os opens optimization options in slow-growing code sizes in all –O2 levels. At the same time, -Os also executes options in space occupation of optimization program. As a result, -Os can sometimes affect program performances. Thus, users can choose lower optimization level in place of –Os.

(Optimize size more)-Os2: Compared to –Os, options such as -mifc are used less frequently. Only part of the code optimization options is opened.

(Optimize size)-Os1: With lower optimization in code space, comes lower influence in code. Only part of the code optimization options is opened.

Illustration 2: Regarding to “Remove unused functions sections (-ffunction-sections)” and “Remove unused data sections (-fdata-sections)”, their function is to eliminate function and data not in use, so as to reduce spaces codes occupy. Please select them if required. In conditions where these two options are not in use, include directive, such as #include "DrvI2C.h", will be applied in the program. Regarding to functions not applied in the include directive, their spaces will also be occupied by compilation.
-funroll-loops can be added to other optimization flags as presented in illustration 2. Its function is to implement ascertained circulation expansion in circulation times under compilation and operation. Size in generated code will increase, while change in different code execution speed remains ambiguous.

Illustration 2



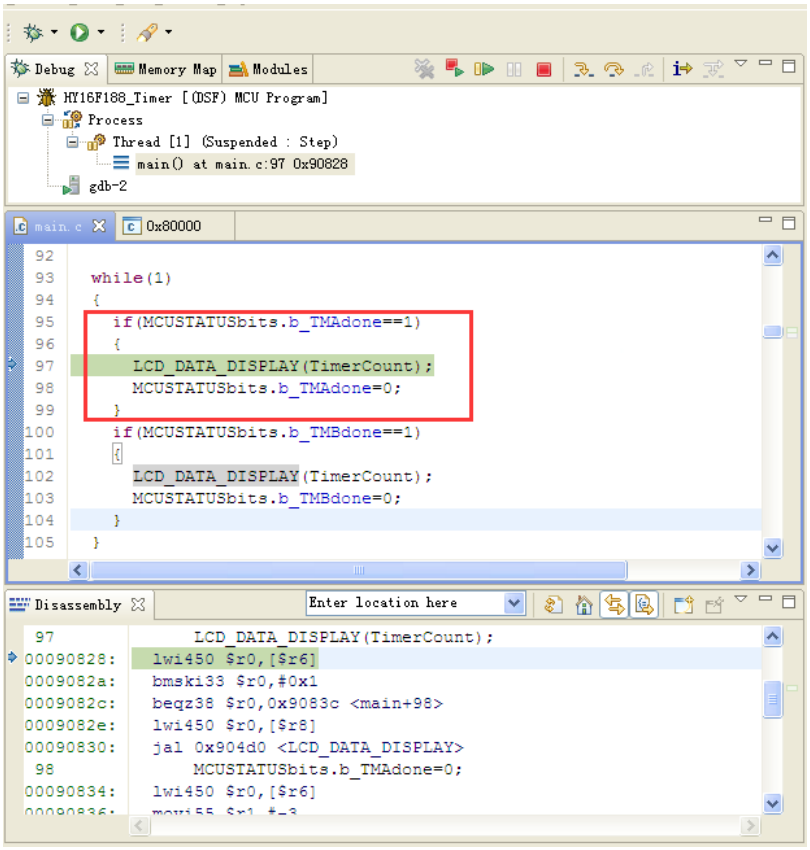
More detailed function introduction among respective optimization levels for reference:

https://gcc.gnu.org/onlinedocs/gcc-4.4.6/gcc/Optimize-Options.html#Optimize-Options
Andes_Programming_GuideDocument

3. Extra notes for optimization setting applications

3.1 Adjustment Problems

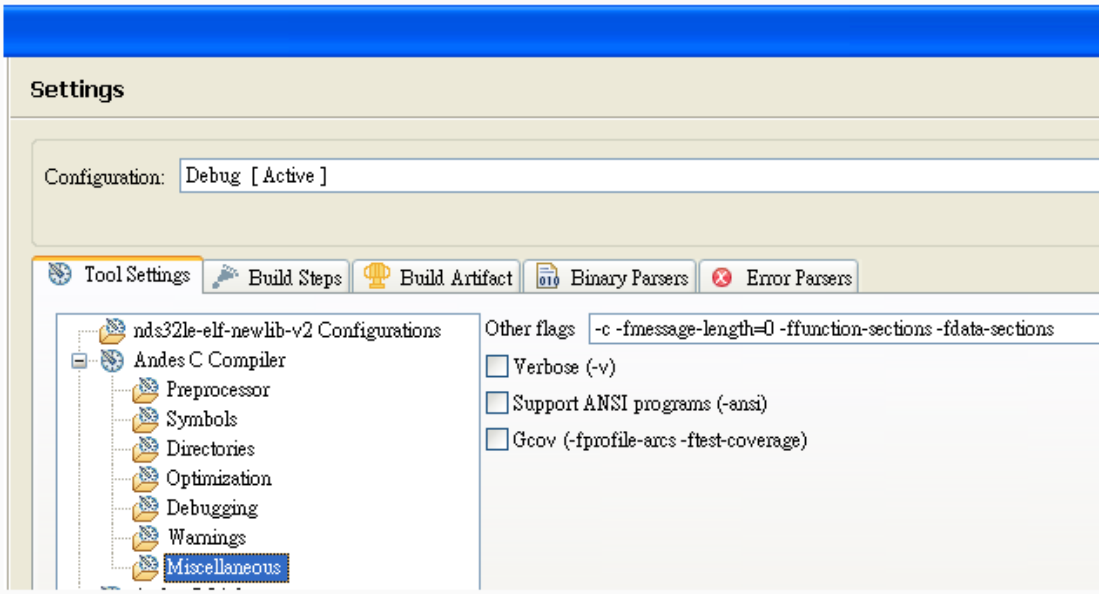
Due to program code optimization while applying aforementioned optimization functions, please pay extra attention to mismatched location in debug indicator, as presented in the red frame below. Since value in MCUSTATUSbits.b_TMAdone has been defaulted as 0, program within red frame belongs to redundancy and will be optimized. Even through debug indicator is matched to the red frame, through execution condition in disassembly orders, we can be notified that order within the red frame has not been executed. In another word, adjustment operational result has been correct.



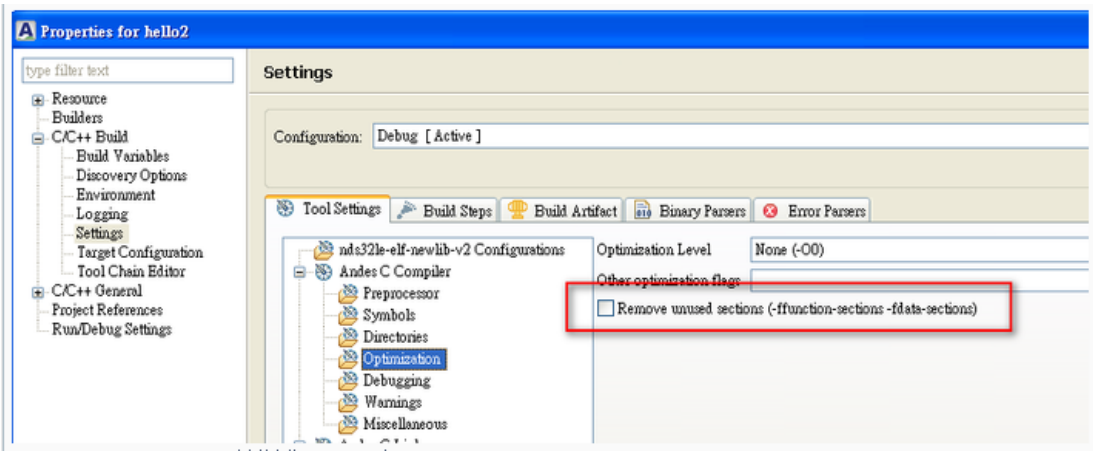
3.2 Inquiry for Sections Removed

Under –ffunction-section enable, please note whether applicable section has been removed by optimize. Usually, condition such as if/else can be calculated during compiling time. As a result, gcc considers it as dead code to be optimized. Here are some ways to make inquiry about which sections has been removed:

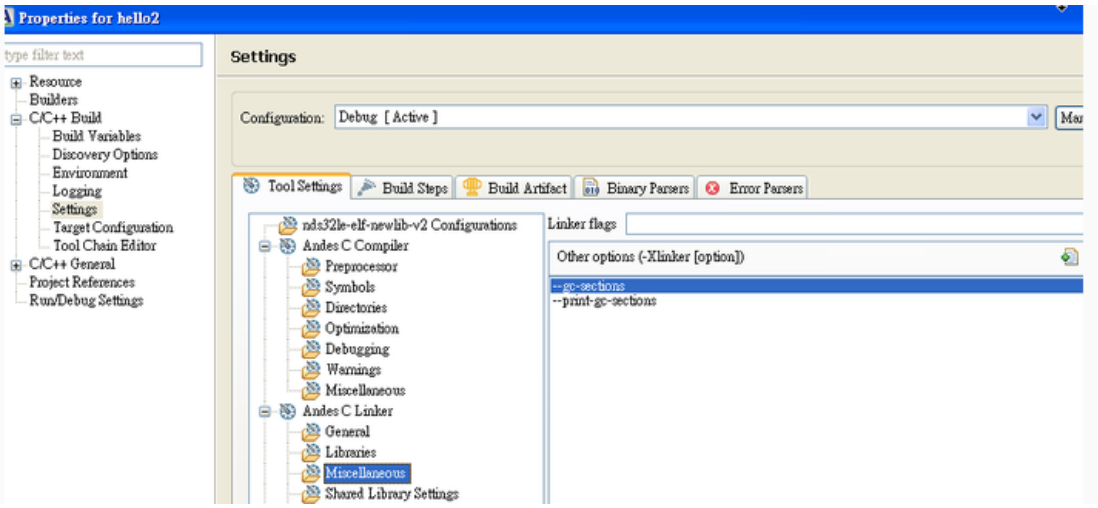
Please add -ffunction-sections -fdata-sections to Andes C Compiler/Miscellaneous.



2) Cancel the selection for Remove unused sections. By separating the original options before setting, remove unused sections will respectively become -ffunction-sections -fdata-sections and --gc-sections.



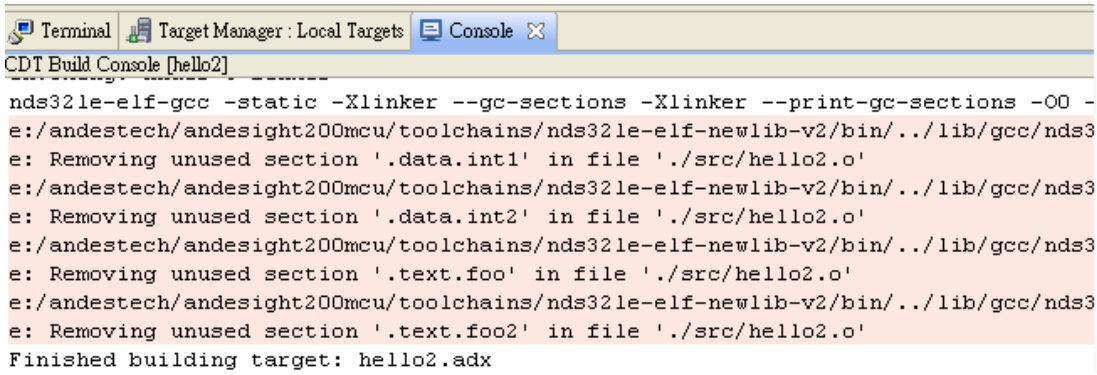
3) Please add the two options --gc-sections and --print-gc-sections to Andes C Linker/Muscellaneous. (Press the green cross button for adding.)



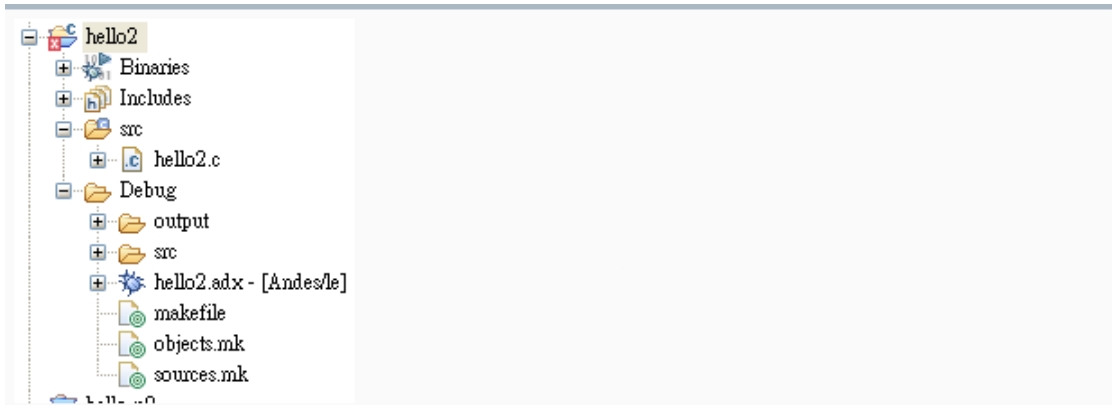
4) My Original Code

```
#include <stdio.h>
#include <stdlib.h>

int int1=34;
int int2=56;
void foo(void);
void foo2(void);

int main(void)
{
   puts("!!!Hello World!!!"); /* prints !!!Hello World!!! */
   return EXIT_SUCCESS;
}

void foo()
{
   puts("foo"); /* prints !!!Hello World!!! */
}

void foo2()
{
   puts("foo2"); /* prints !!!Hello World!!! */
}
```

5) Build code, int1 int2 foo1 foo2 in the original version have been eliminated, as presented below:



6) Simply ignore the red x if it were to appear in project, since message has been output. It can still generate result normally, with .adx document output.



3.3. Measures to prevent some program codes to be optimized.

Referential Language：http://gcc.gnu.org/onlinedocs/gcc/Funct ... agmas.html
Two writing methods are presented below. A warning demonstrating language is not supported in this machine will appear in the second writing method, when in fact its result is correct.
Note: Regarding to setting optimization levels in single function, “Os1” and “Os2” are not supported. Since “Os1” and “Os2” are defined through Andes, not standardized GCC optimization levels.

Writing Method 1: Regarding to only one function, add().
__attribute__((optimize("O0")))// setting optimization levels belongs to no optimization.

int add (int a, int b)

© 2015 HYCON Technology Corp.

```
{
int x = a;
int y = b;
return x + y;
}

int main ()
{
int r = 1;
int a = r;
int b = r;
func ();
return 0;
}
```

Writing Method 2: Codes included will not be optimized.

Note! This writing method need not to be limited to function range, any code segment can be selected.

```
#pragma GCC push_options
#pragma GCC optimize ("O0")//Optimum level is "O0"
int add (int a, int b )
{
int x = a;
int y = b;
return x + y;
}
#pragma GCC pop_options

int main ()
{
int r = 1;
int a = r;
int b = r;
func ();
```

© 2015 HYCON Technology Corp.

```
return 0;
}
```

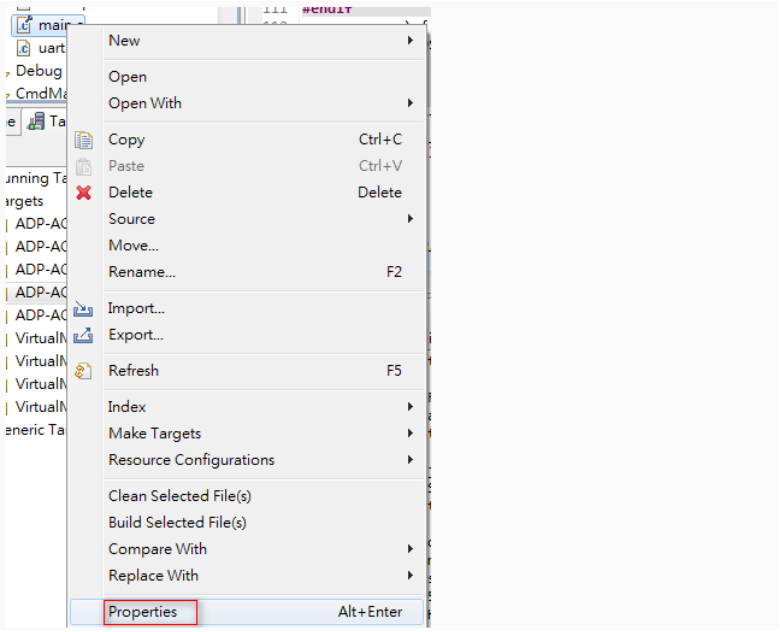
Note ：#pragma GCC push_options
#pragma GCC pop_options

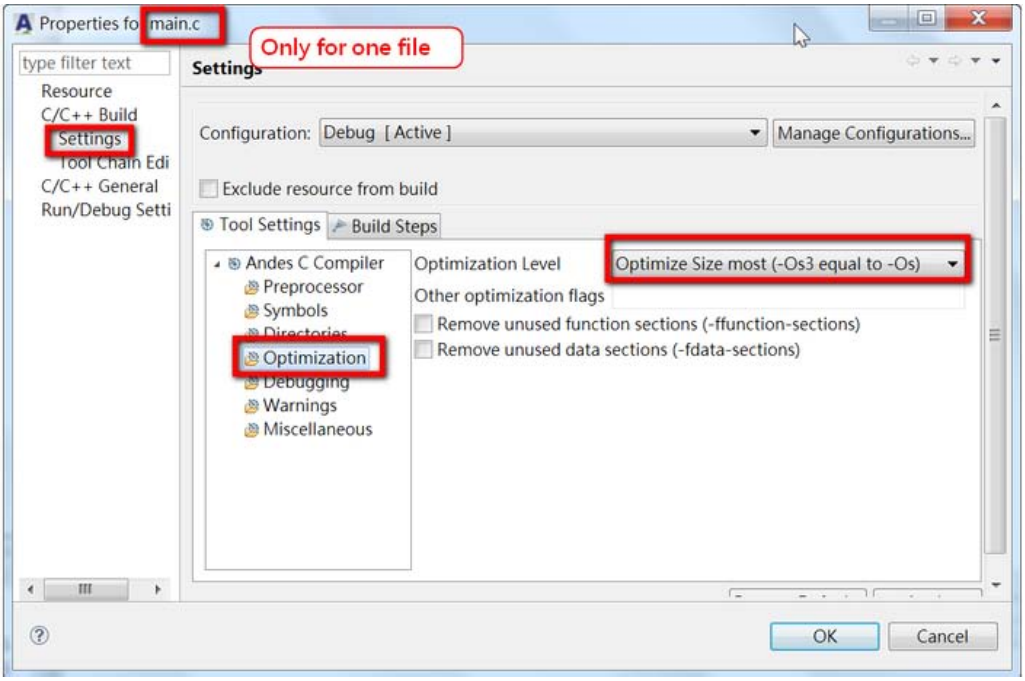
These two lines indicate that original option will be pushed inward. For example, regarding to –Os, the one at the back will pop –Os back and restore the original optimize setting.

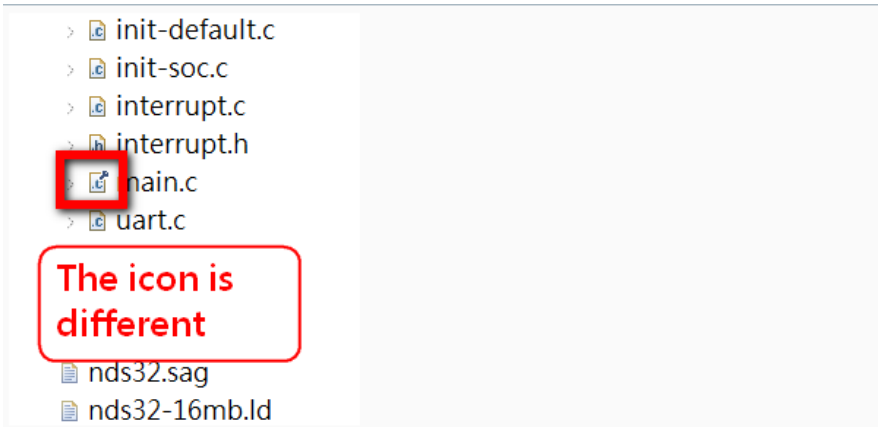
3.4 Partial Code Optimize Methods

3.4.1 Single Document Setting Optimization

Upon clicking the right button on original code document, please execute optimization level setting according to the illustration below.


New
Open
Open With
Copy Ctrl+C
Paste Ctrl+V
Delete Delete
Source
Move...
Rename... F2
Import...
Export...
Refresh F5
Index
Make Targets
Resource Configurations
Clean Selected File(s)
Build Selected File(s)
Compare With
Replace With
Properties Alt+Enter


Properties fo main.c
Only for one file
type filter text
Resource
C/C++ Build
Settings
Tool Chain Edi
C/C++ General
Run/Debug Setti
Settings
Configuration: Debug [Active]
Manage Configurations...
Exclude resource from build
Tool Settings
Build Steps
Andes C Compiler
Preprocessor
Symbols
Optimization
Debugging
Warnings
Miscellaneous
Optimization Level
Optimize Size most (-Os3 equal to -Os)
Other optimization flags
Remove unused function sections (-ffunction-sections)
Remove unused data sections (-fdata-sections)
OK
Cancel


init-default.c
init-soc.c
interrupt.c
interrupt.h
main.c
uart.c
The icon is different
nds32.sag
nds32-16mb.ld

3.4.2 Partial Code Setting Optimization Levels

Referential Languages

http://gcc.gnu.org/onlinedocs/gcc/Funct ... agmas.html

Codes are presented below:

#pragma GCC push_options

#pragma GCC optimize ("O0")Optimization level is “O0” which can be set into other levels except for Os1 and Os2.

// code ...// Program Code

#pragma GCC pop_options

Note: #pragma GCC push_options

#pragma GCC pop_options

These two lines indicate that original option will be pushed inward. For example, regarding to –Os, the one at the back will pop –Os back and restore the original optimize setting.

Single Function Setting Optimization Levels

Execute setting for single function optimization levels. Code below is set as “Os”.

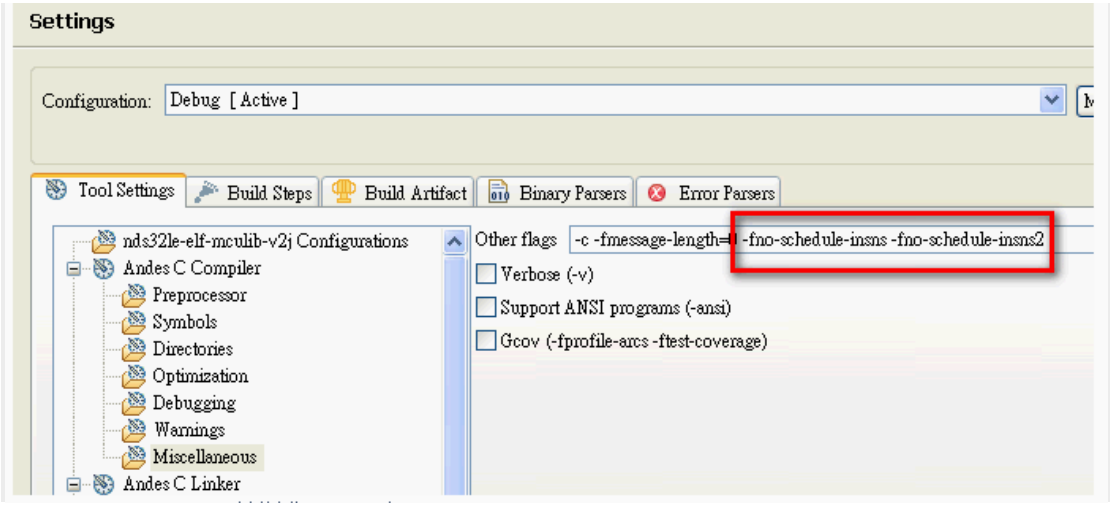
```
void __attribute__ ((optimize ("Os"))) __cpu_init()
{
unsigned int tmp;
/* turn on BTB */
tmp = 0x0;
__nds32__mtsr(tmp, NDS32_SR_MISC_CTL);

/* Set PSW INTL to 0 */
tmp = __nds32__mfsr(NDS32_SR_PSW);
tmp = tmp & 0xfffffff9;
/* ....*/
return;
}
```

© 2015 HYCON Technology Corp.

Note: Regarding to setting optimization levels in single function, “Os1” and “Os2” are not supported. Since “Os1” and “Os2” are defined through Andes, not standardized GCC optimization levels.

3.5 Measures to Prevent Code Order to be Switched Upon Optimization

In order to prevent optimization from changing program code order, two options can be added: -fno-schedule -insns -fno-schedule-insns2



3.6 Other Items to Be Noted

(1)“volatile” shall be added to variable and Register relevant to hardware. Please add the keyword volatile to prevent variables from being optimized. Take the below programs for example:

XBYTE[2]=0x55;
XBYTE[2]=0x56;
XBYTE[2]=0x57;
XBYTE[2]=0x58;

Compiler optimization function will only consider XBYTE[2]=0x58, with the former three sentences being ignored and combined into one sentence. If volatile were to be inserted, compiler will gradually execute compilation and generate the machine code corresponding to four code .

(2)Once c code announces volatile is being added to the variable, volatile should also be added to variable at extern.

© 2015 HYCON Technology Corp.

(3)While amending variables used by other functions in the interruption service program, it is suggested to add volatile. Once optimization function has been opened, register will be borrowed for variable visiting, instead of visiting variable address directly. Result in variable address might be changed without reflecting onto the register.

(4)In multithreading application, volatiles are suggested to be added to variables being shared by multiple tasks.

4. Referential Document

Andes_Programming_Guide_v1.5_PG009_V2.1

Partial contents are from: http://forum.andestech.com

5. Amendment Record

Greater differences in this document are described below, with punctuation and font being excluded from the description range.

Version	Page Number	Amendment Summary	Date
V01	All	Initial Issuing	2015/03/11

© 2015 HYCON Technology Corp.