



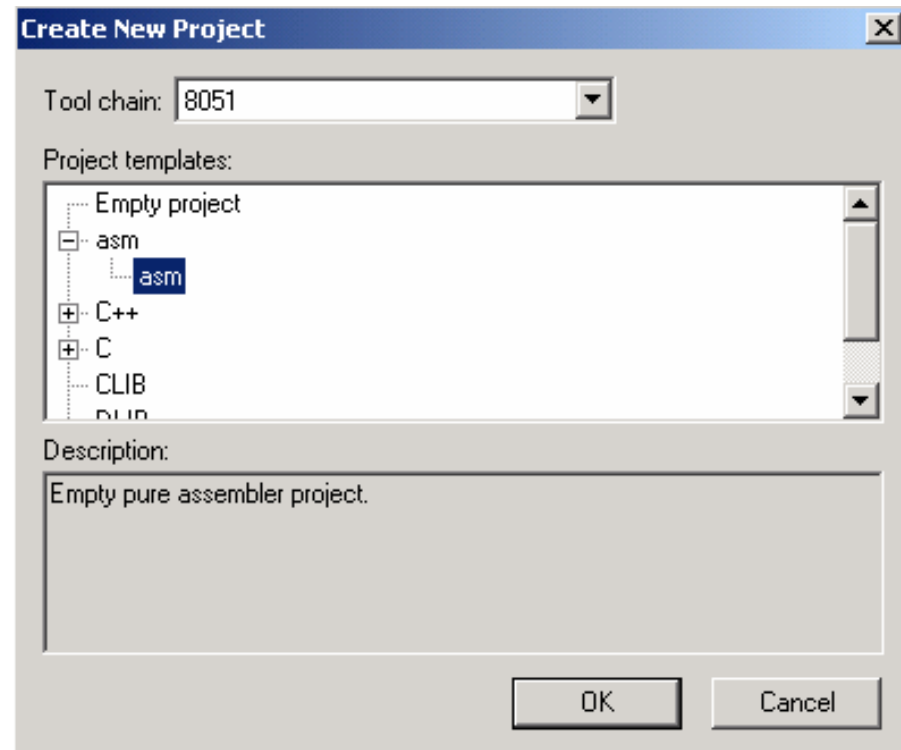
**IAR WORKBENCH FOR 8051**  
**PART2**

**CONTENTS**  
Using Assemble  
Creating library  
Using library

## For assembly project...

- Make a new project of asm type
- Assembly source file .s51
- Select appropriate microcontroller derivative
- Change the general options for assembly project
- Do not include CLIB or DLIB libraries
- Change the program entry option in linker option tab
- Generation of list file and map files (optional)

Project > Create New Project  
Select asm  
OK  
And save the project file

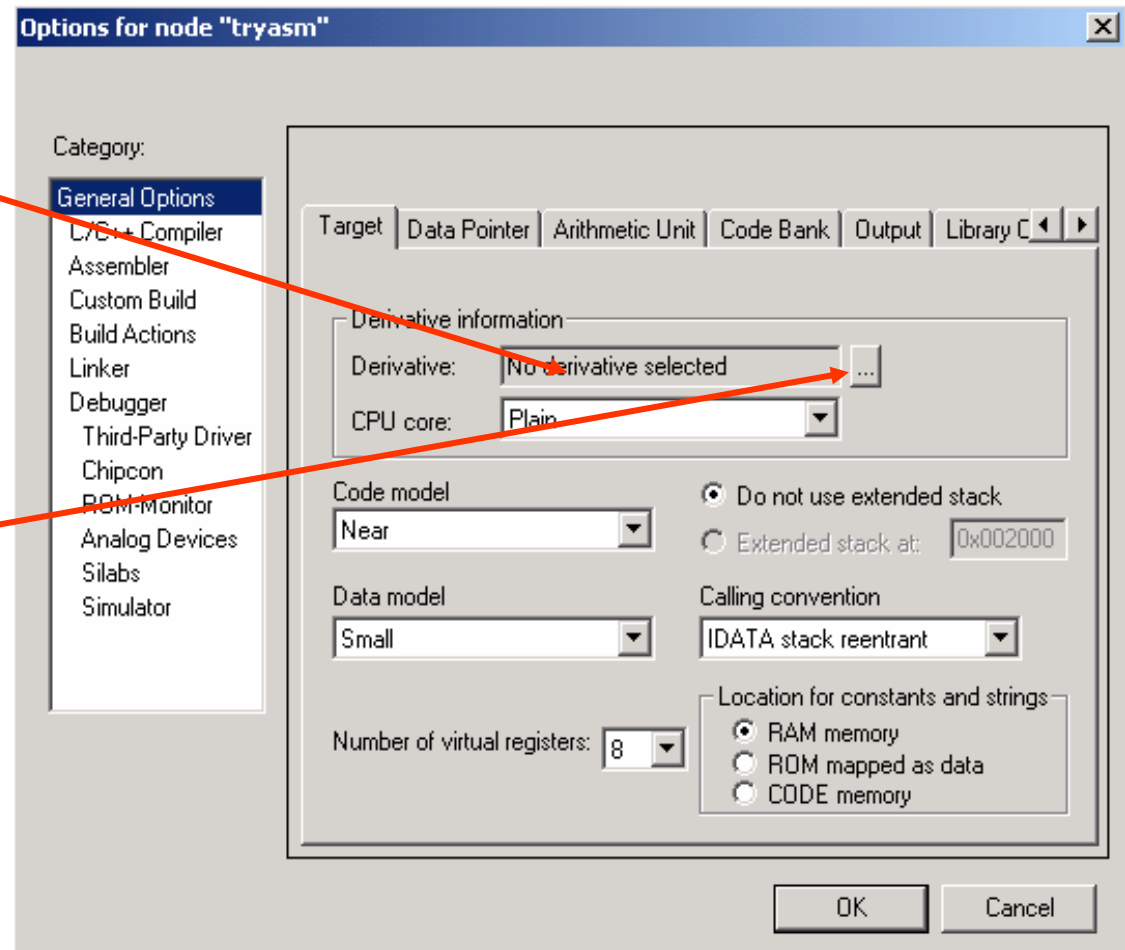


# Asm project: option settings

Project general options > Target Tab > Select microcontroller directive

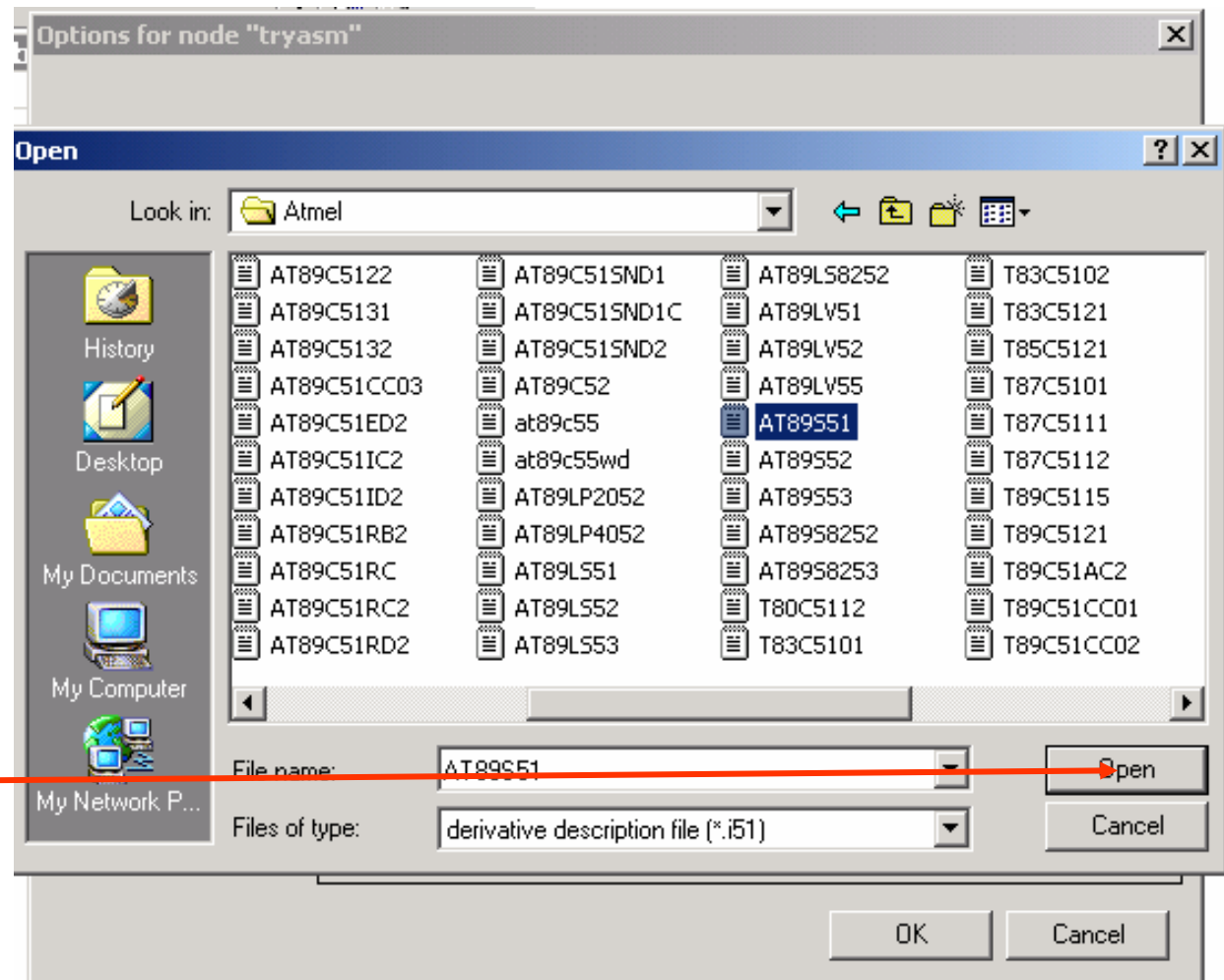
By default no derivative is selected

To Select Derivative  
Click here



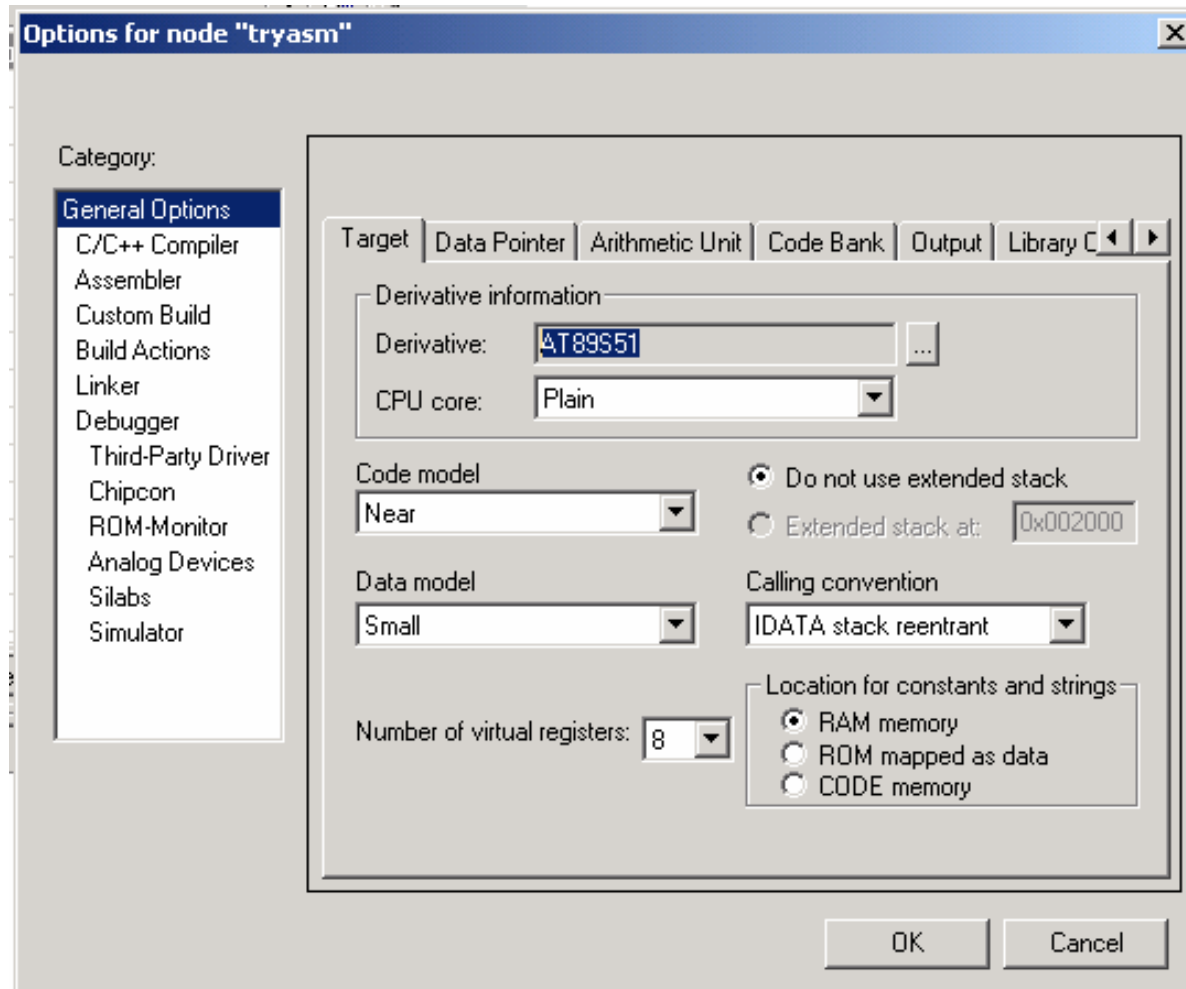
# Asm project: select directive

Suppose AT89S51



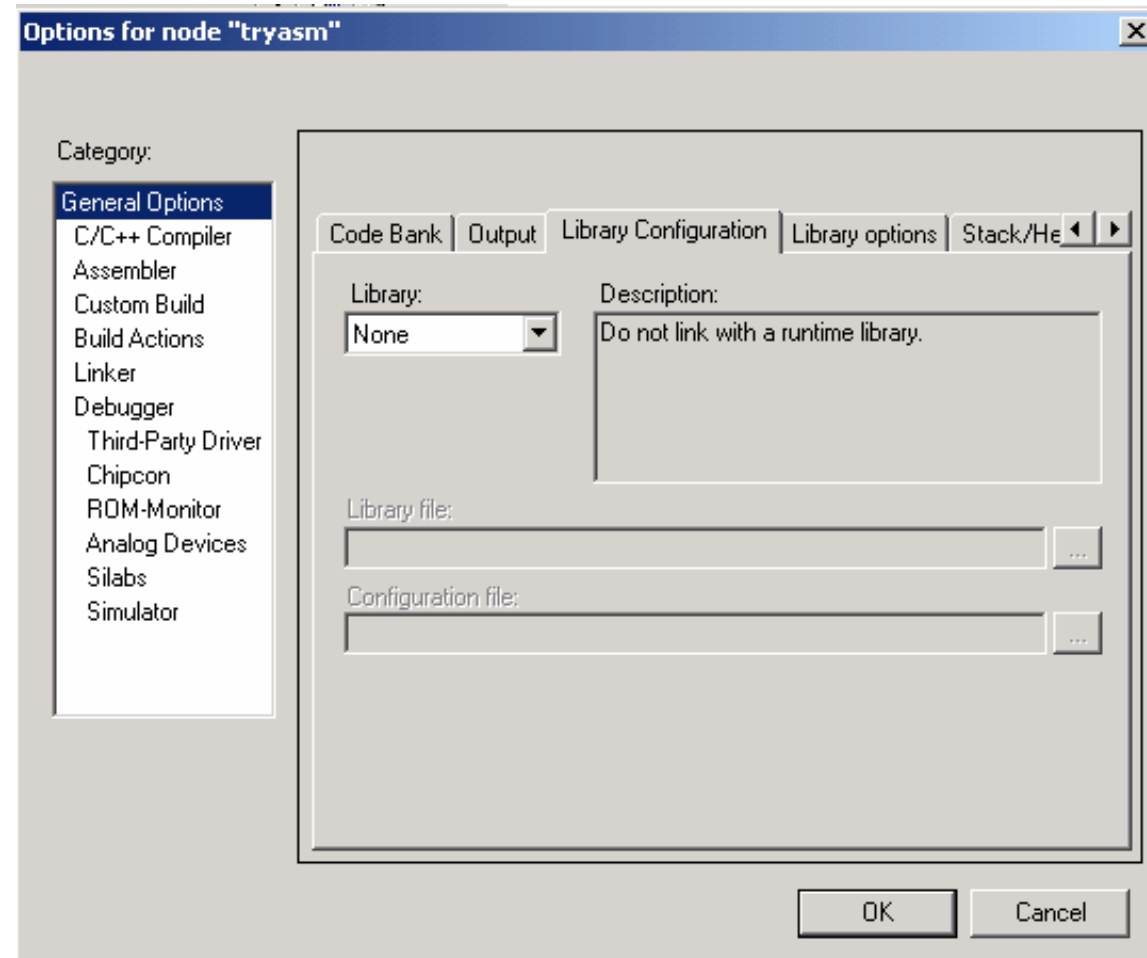
Click open

# Asm project: derivative selected



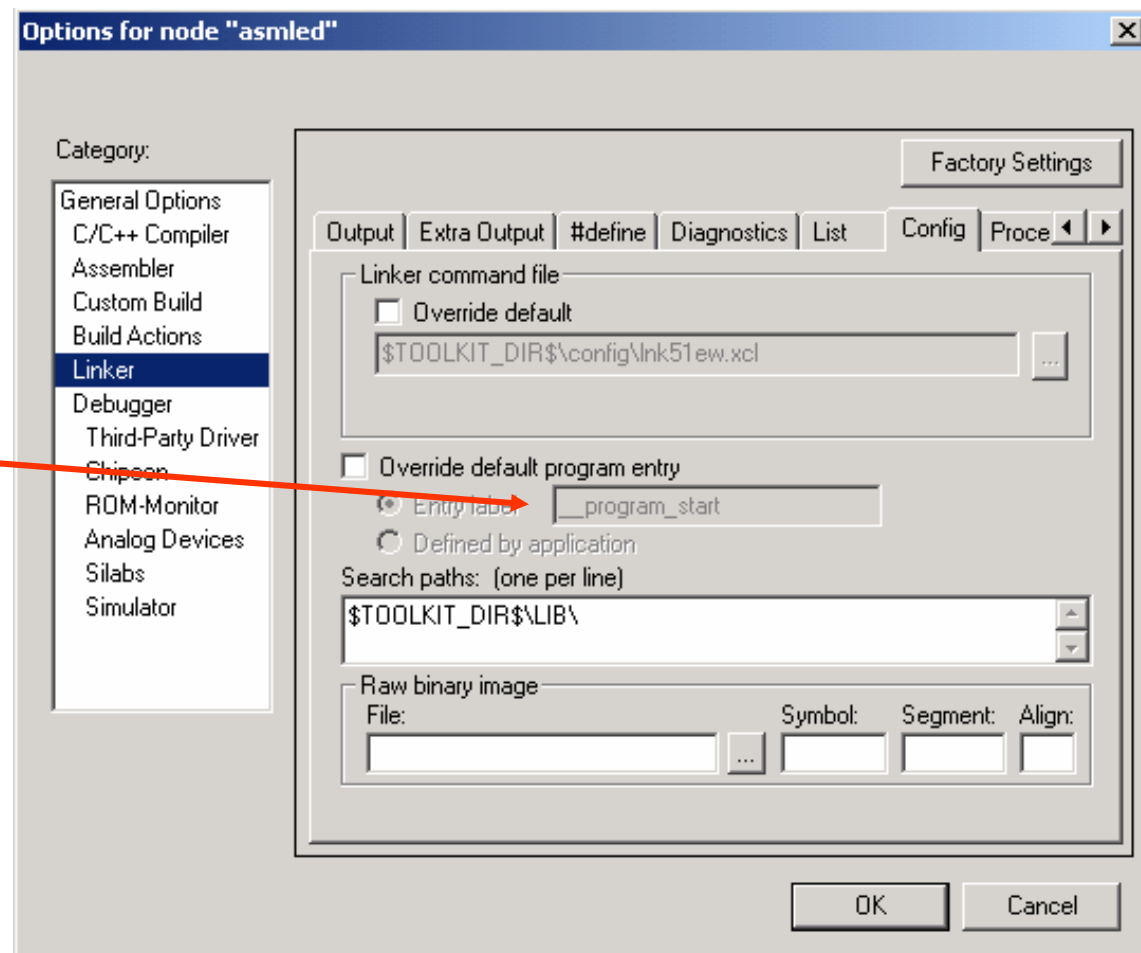
# Asm Project: Not to include library

It is None  
by default in  
asm project templates



# Asm Project: program entry option 1

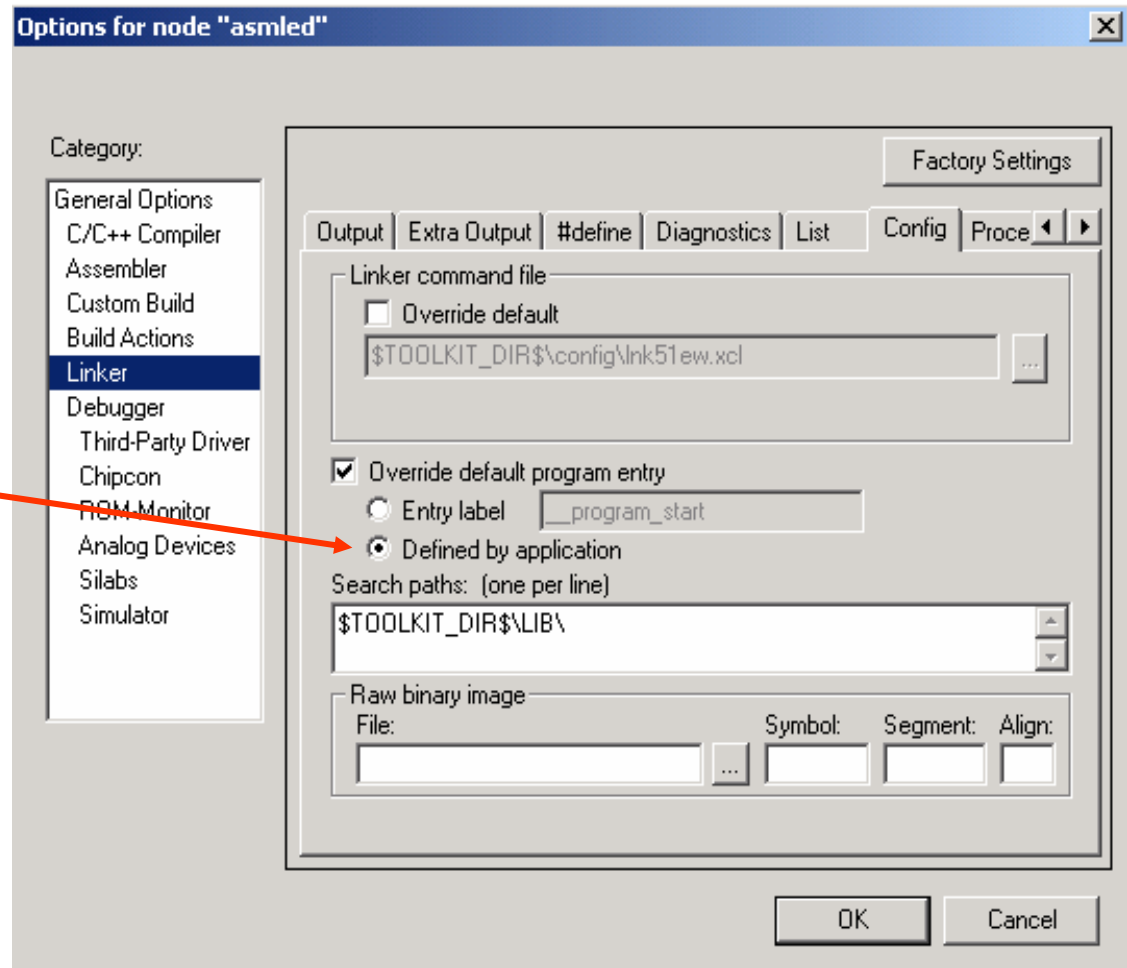
Default Setting



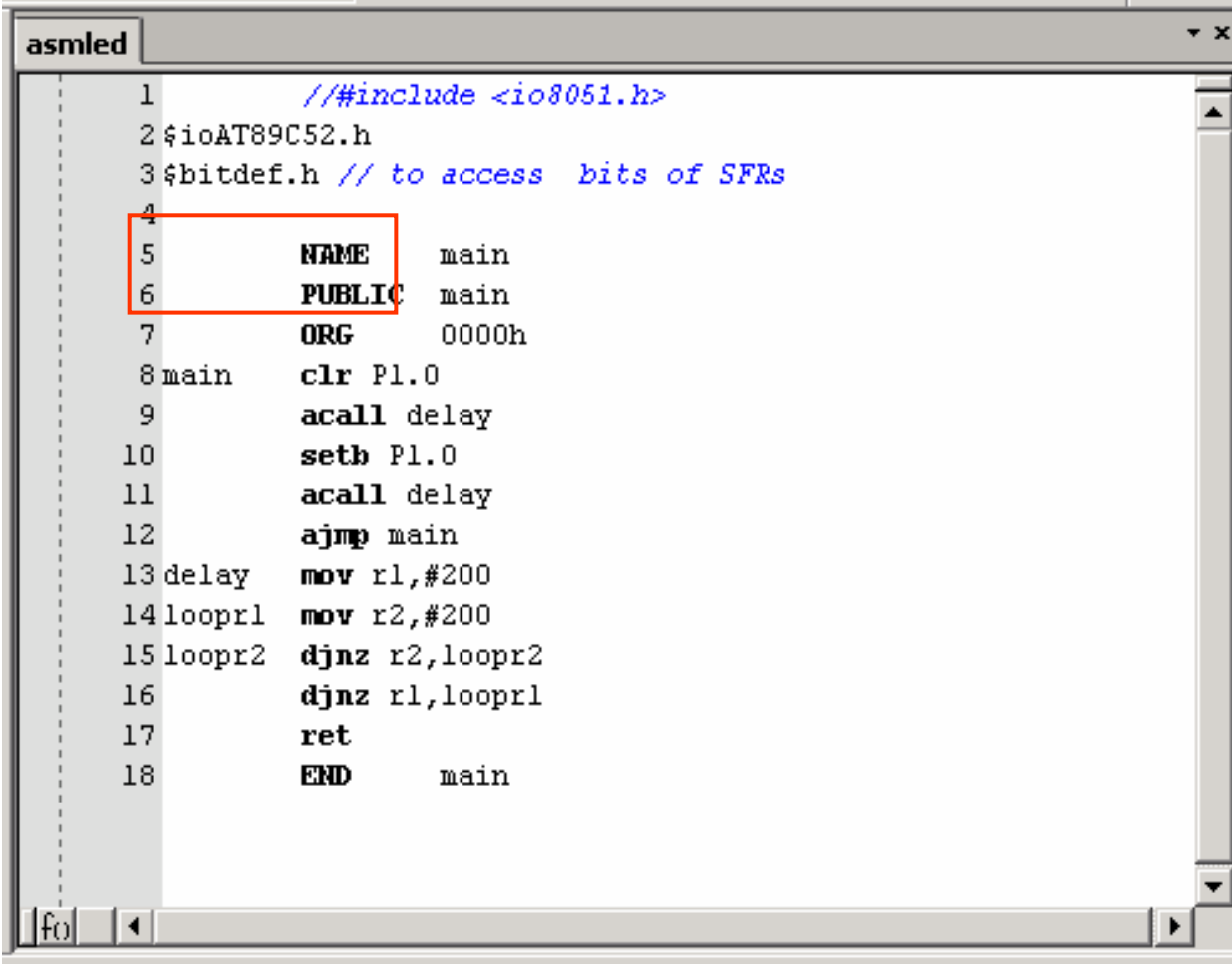


# Asm Project: program entry option 2

Change setting



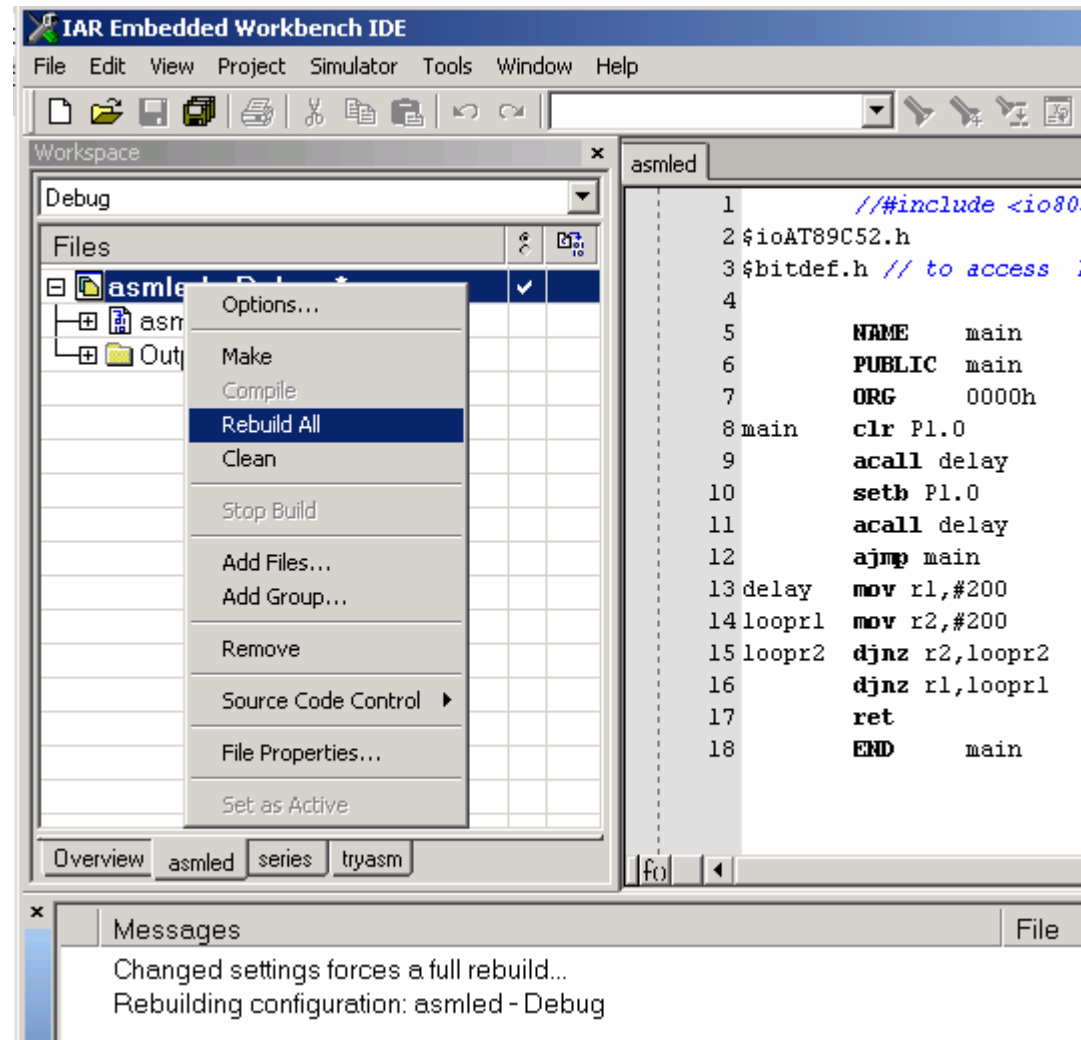
# Asm Project: Include header files



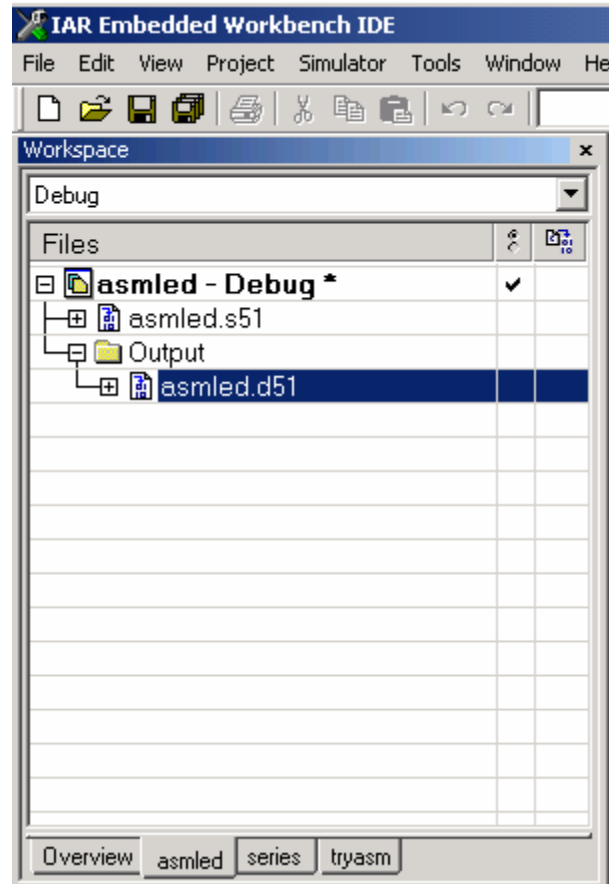
```
asmled
1      //#include <io8051.h>
2 $ioAT89C52.h
3 $bitdef.h // to access bits of SFRs
4
5      NAME    main
6      PUBLIC  main
7      ORG    0000h
8 main  clr P1.0
9      acall delay
10     setb P1.0
11     acall delay
12     ajmp main
13 delay mov r1,#200
14 loopr1 mov r2,#200
15 loopr2 djnz r2,loopr2
16     djnz r1,loopr1
17     ret
18     END    main
```

# Asm Project: Build program

Right click project  
> Rebuild All



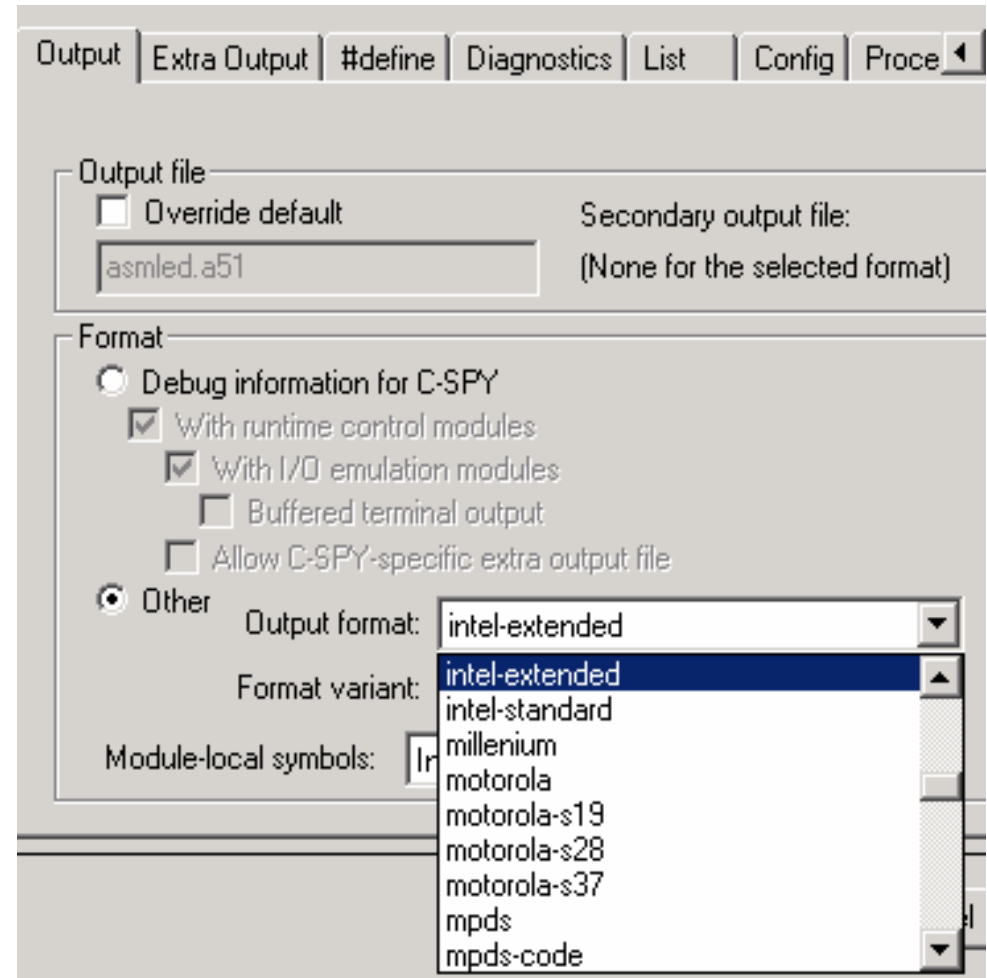
# Asm Project: Output file



# Output Formats

IAR Xlinker generate 30 O/P formats

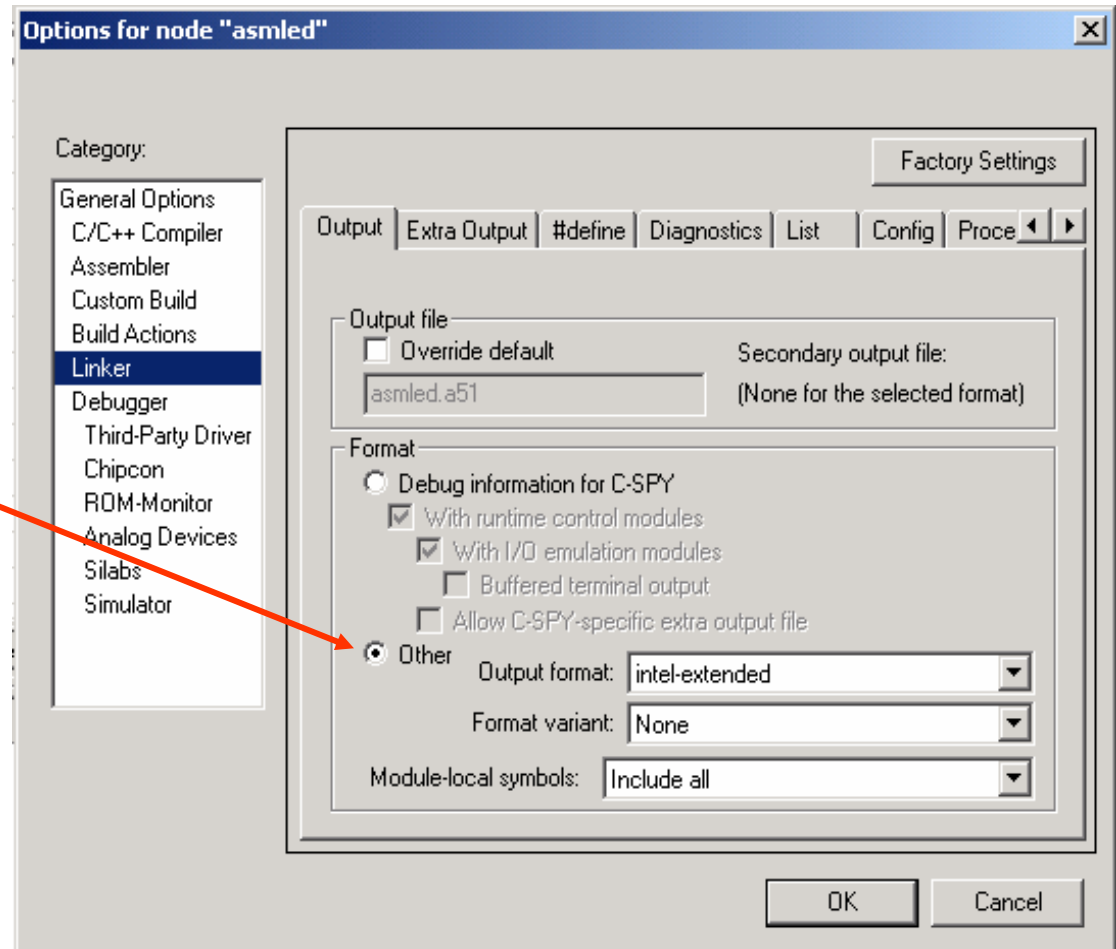
- UBROF (IAR proprietary)
- Pentica
- Intel Standard (HEX)
- Motorola etc



# Output Formats: Generation of Hex file

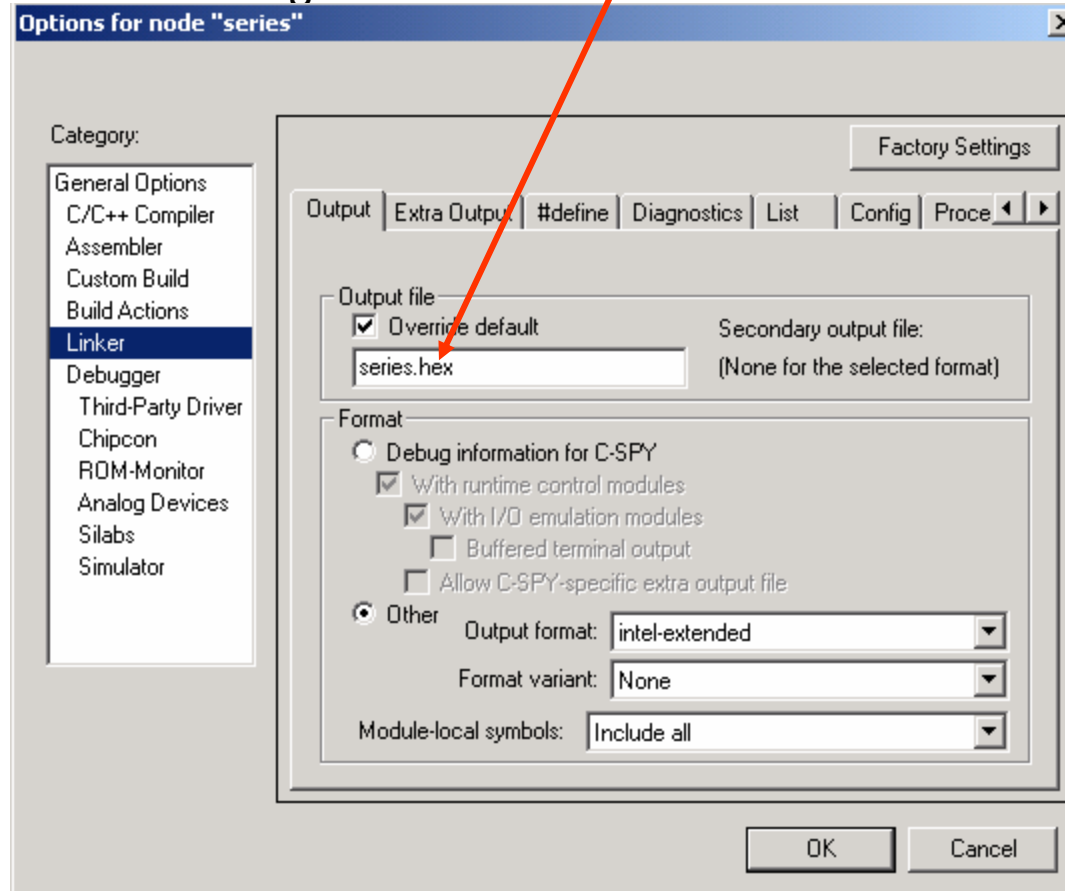
## Project options

- > Linker
- > output
- > Formats



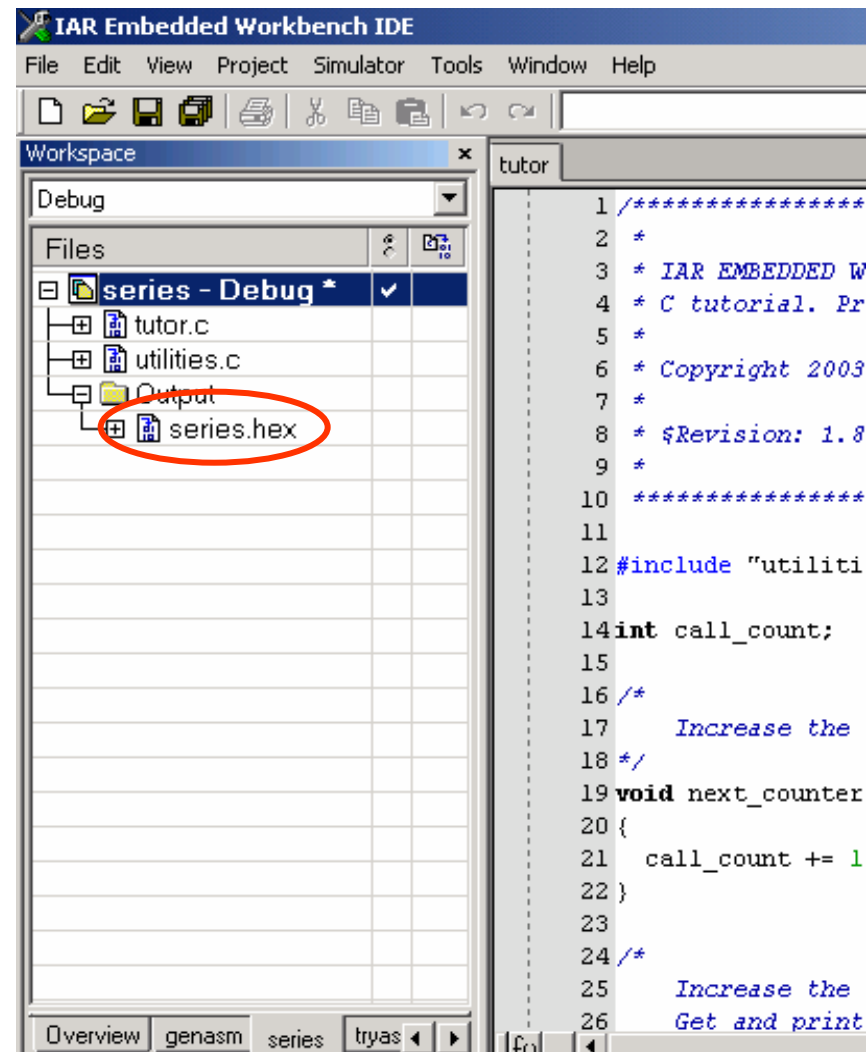
# Output Formats: Generation of Hex file

Change default name to .hex name



# Output Formats: Generation of Hex file

- Rebuild the project
- check the hex file at output



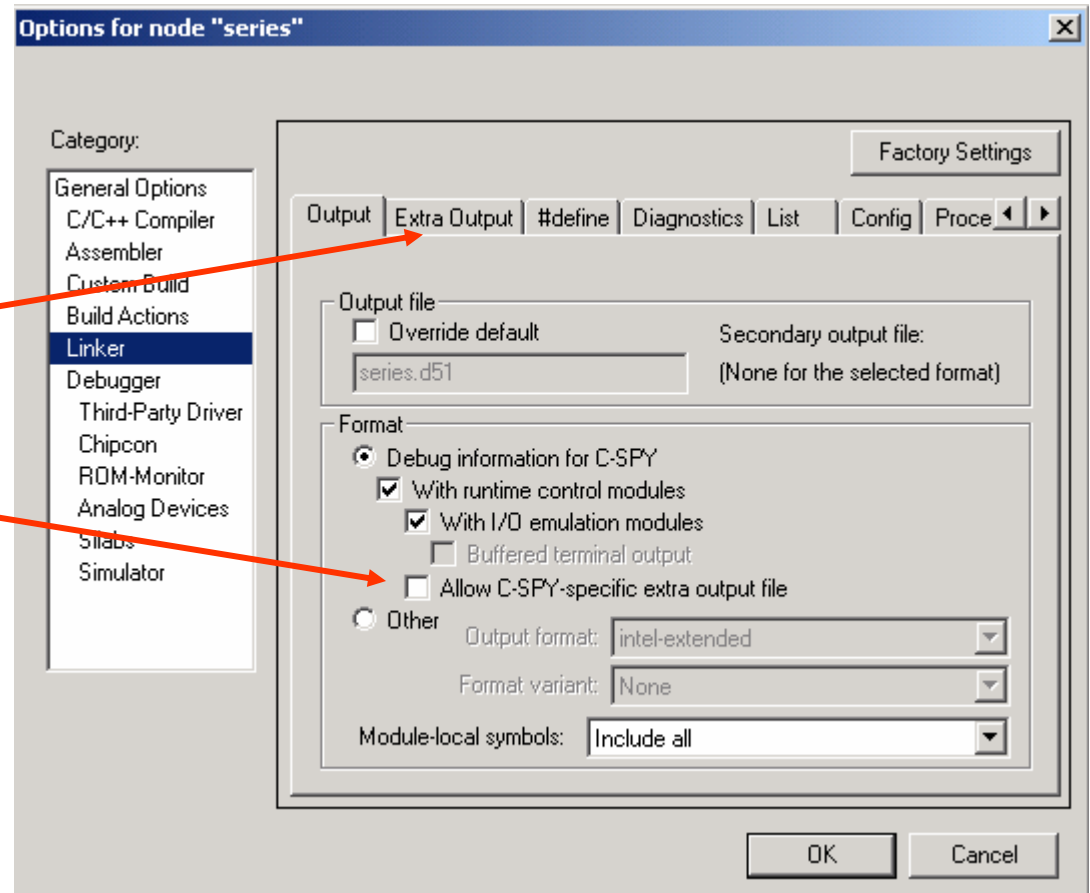


# Output Formats: Generation of both debug and hex file-1/3

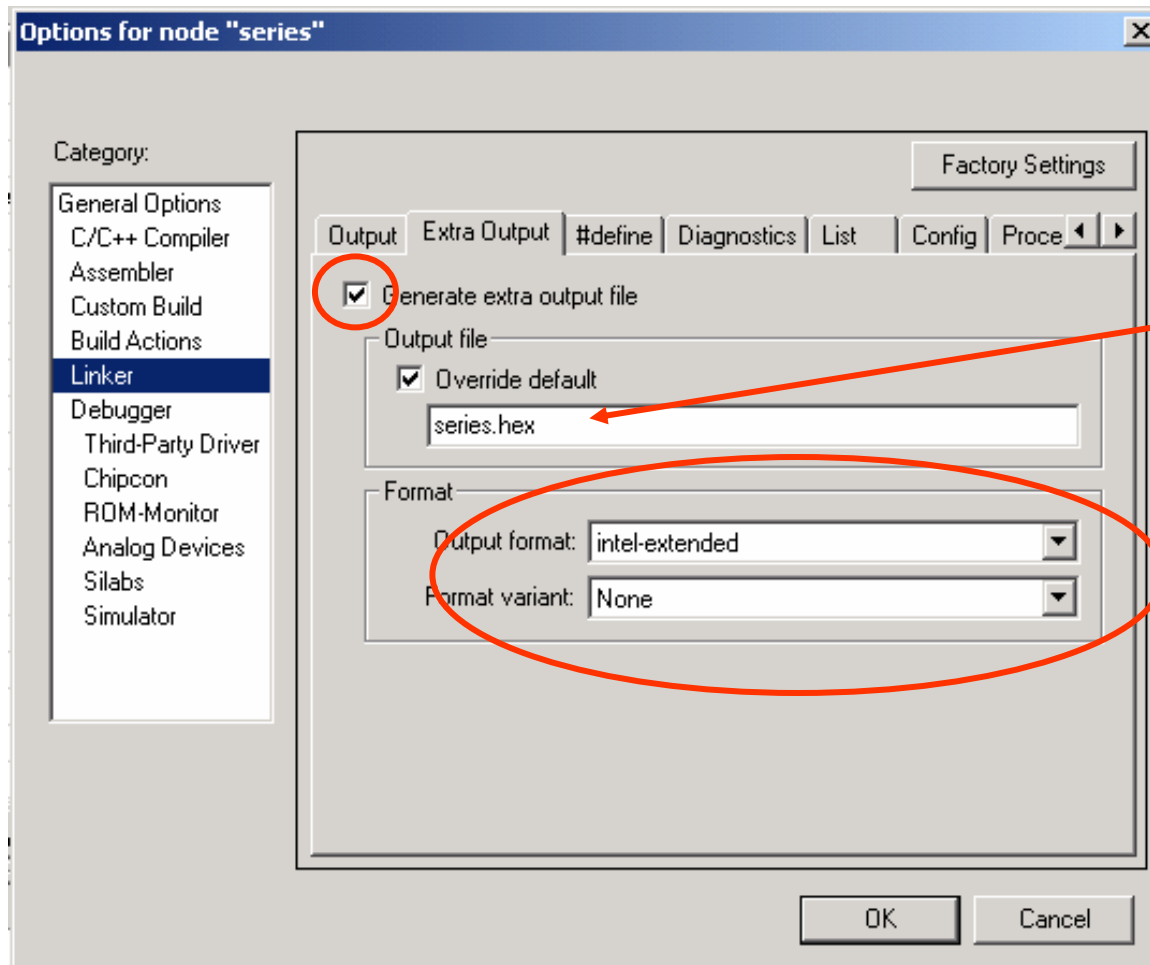
Here is setting for extra o/p

Enable extra output file

By default it is disable



# Output Formats: Generation of both debug and hex file-2/3

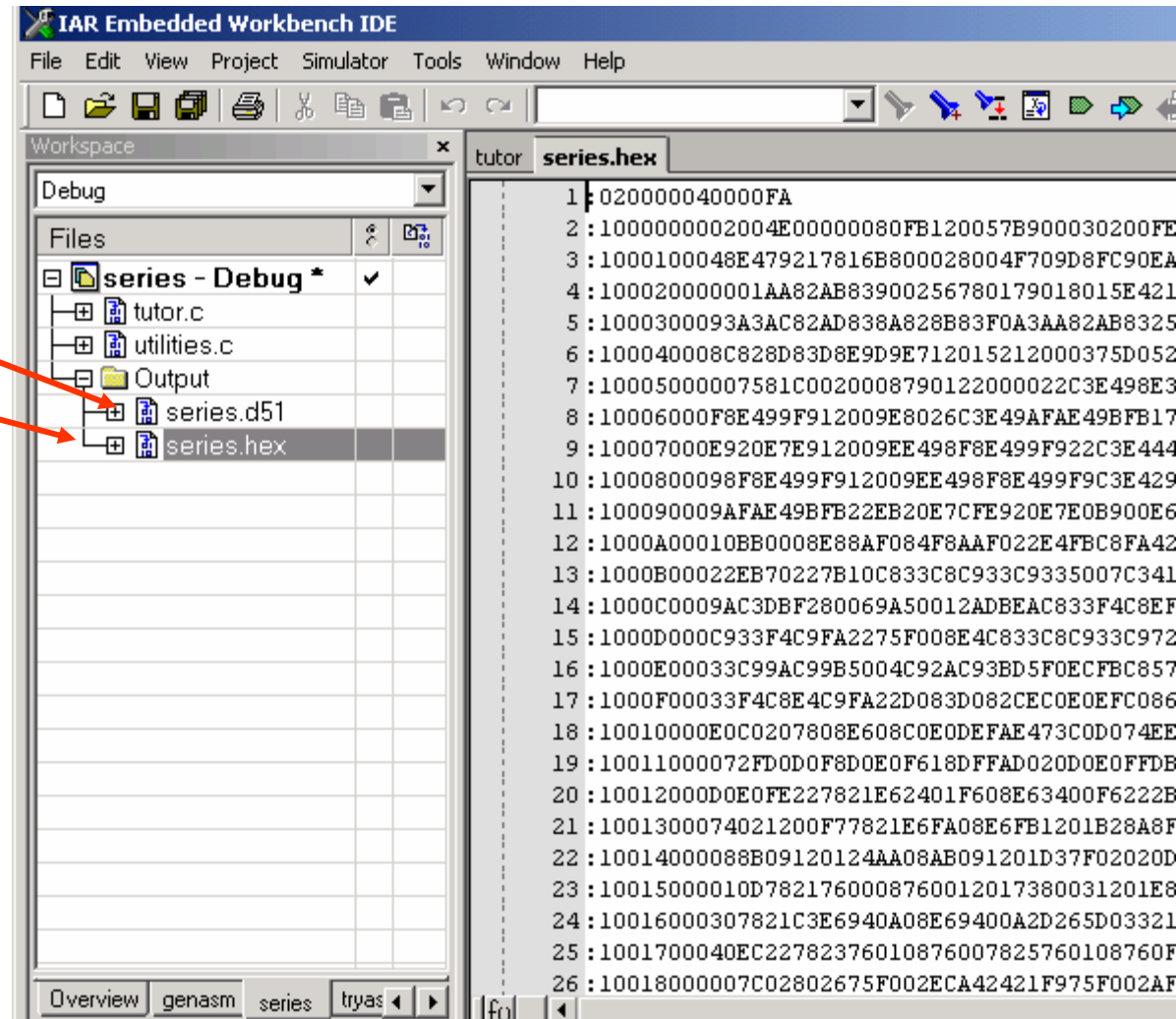


Change default name

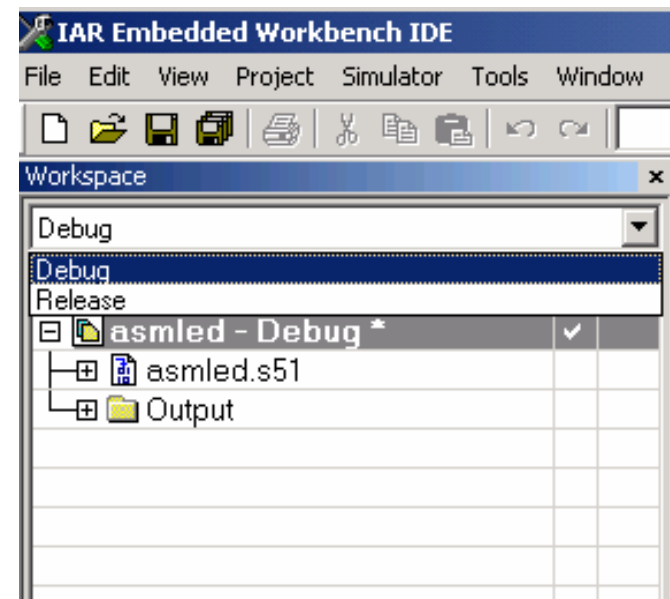
# Output Formats: Generation of both debug and hex file-3/3

Rebuild the project

Debug file  
Hex file



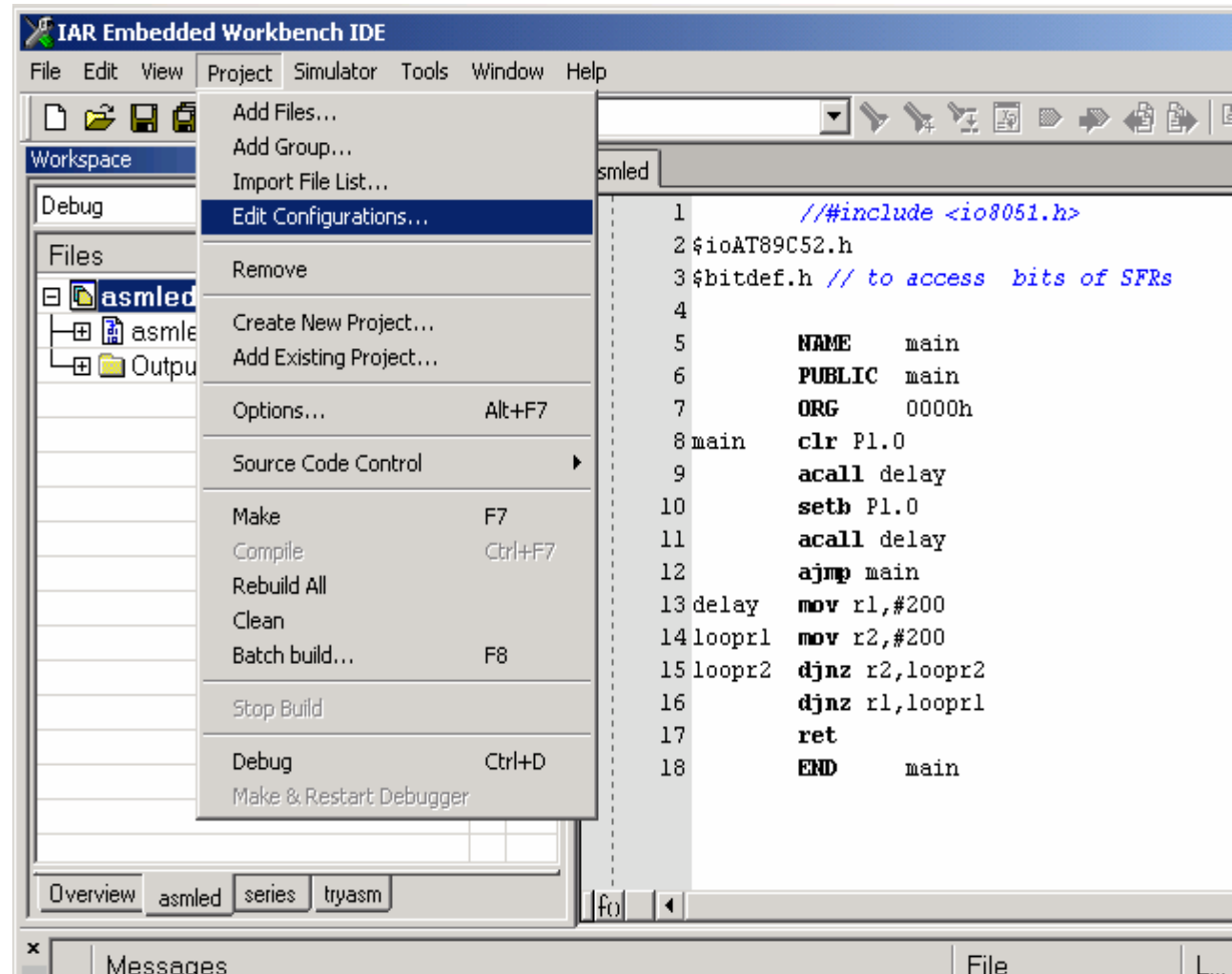
- Two project configurations (settings for particular o/p formats)
  - Debug (O/P for debug)
  - Release (O/P for download)



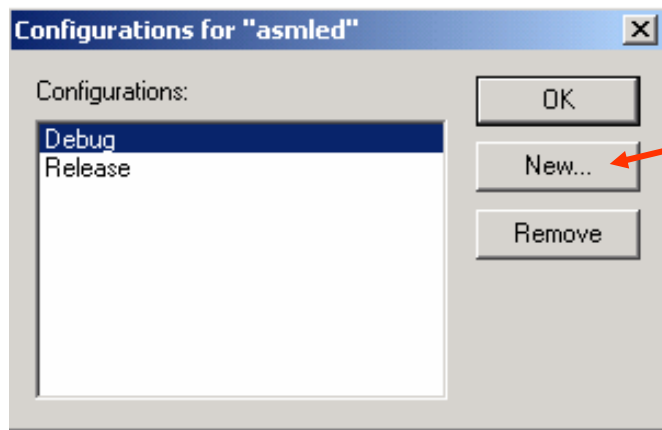
Any new configuration can also be added

# New project configuration-1/3

Project  
> Edit Configuration

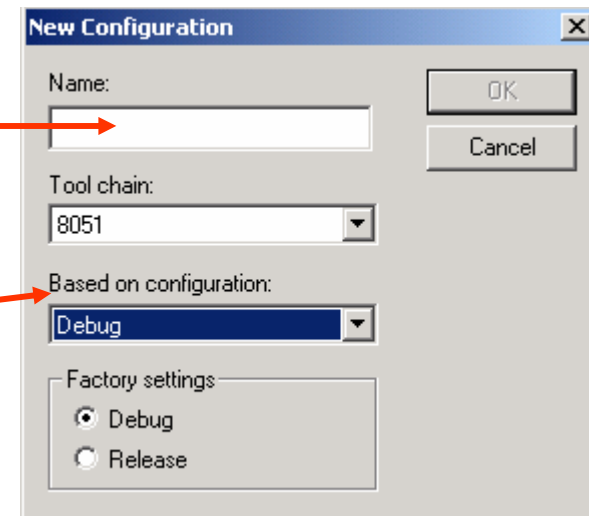


# New project configuration-2/3



Click new

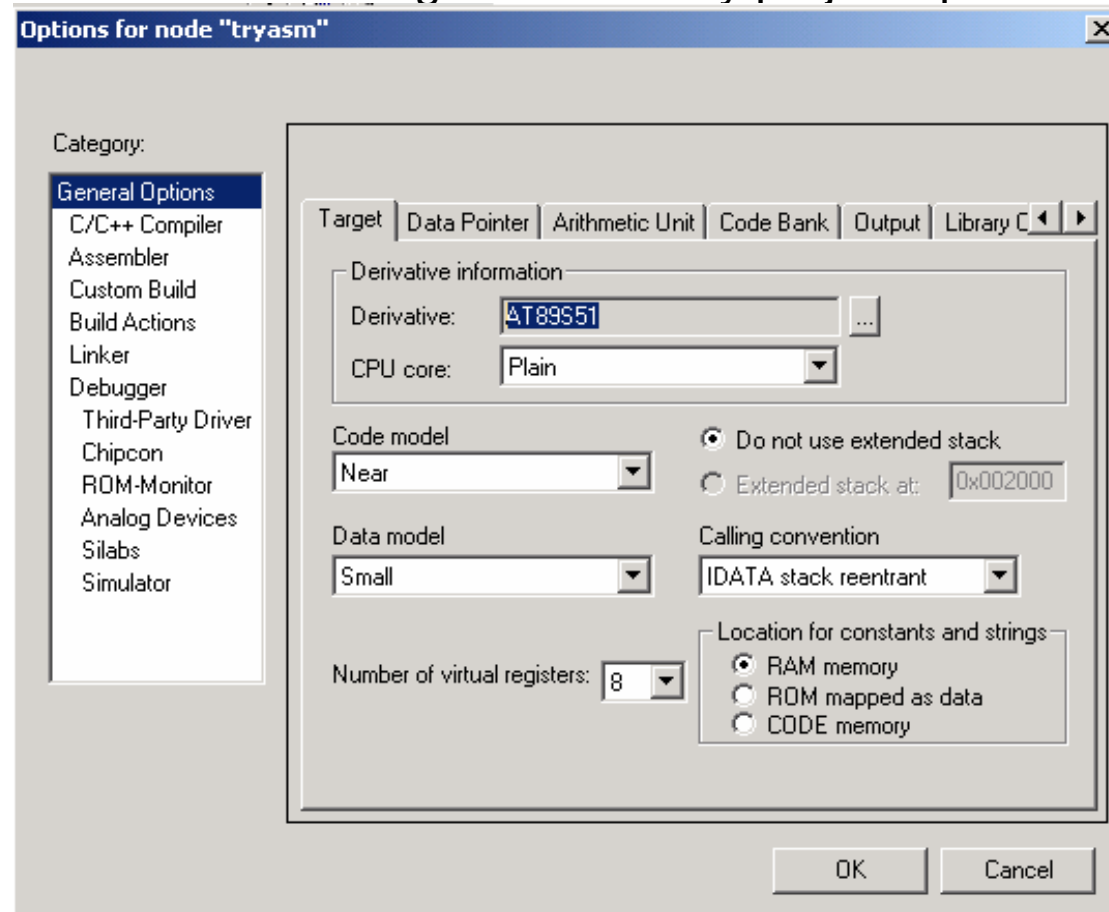
Enter the name of configuration



Select base configuration

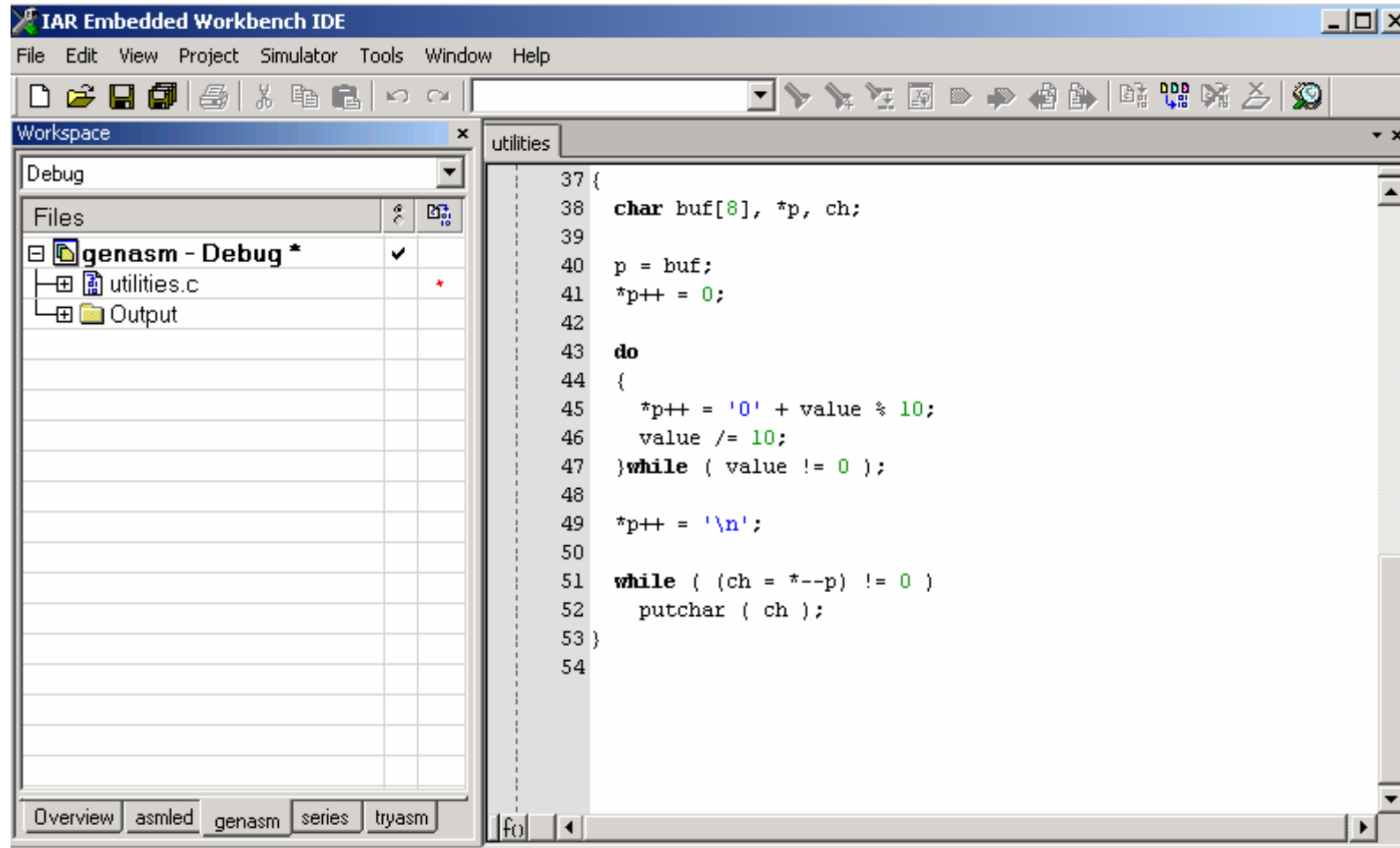
# New project configuration-3/3 Customize

## Customize configuration modify project options



# Generation of assembly file from C file -1/5

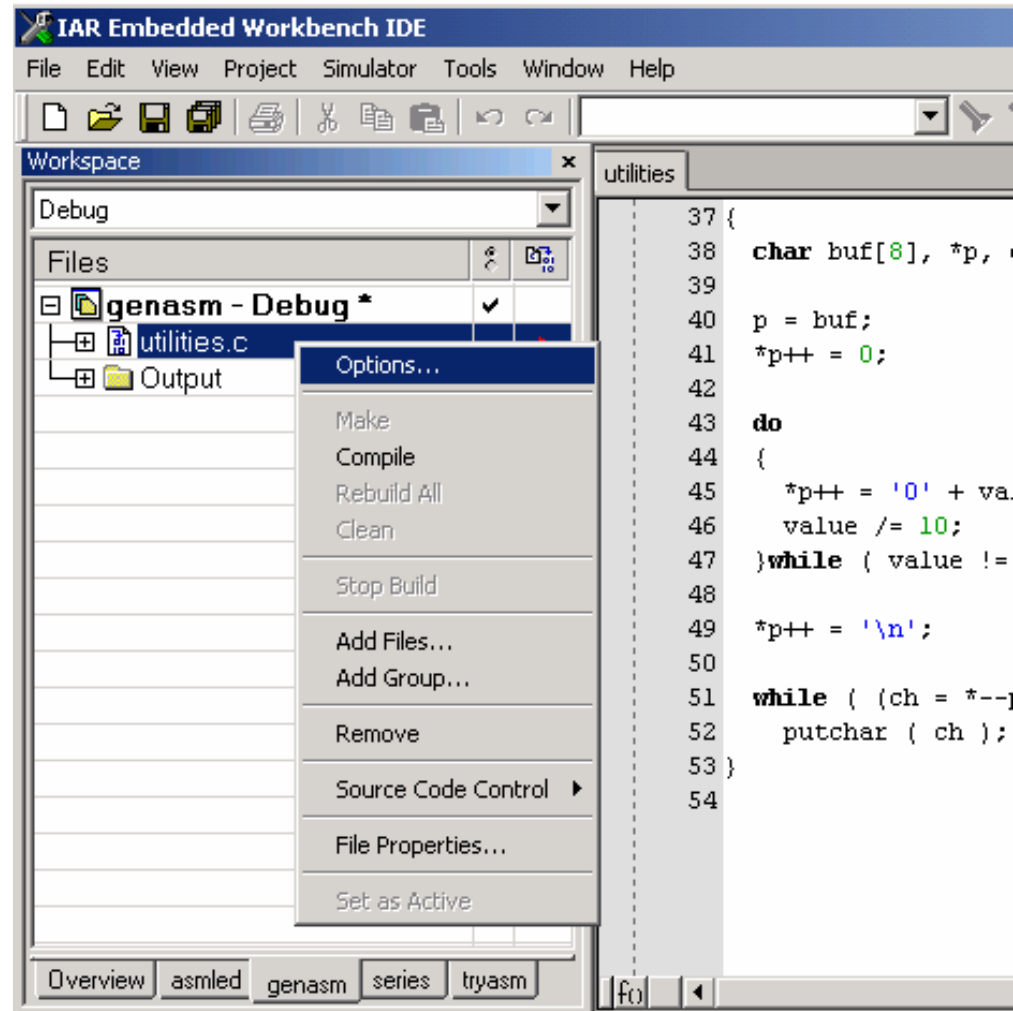
Create a new project name as “**genasm**”, add utilities.c file in the project



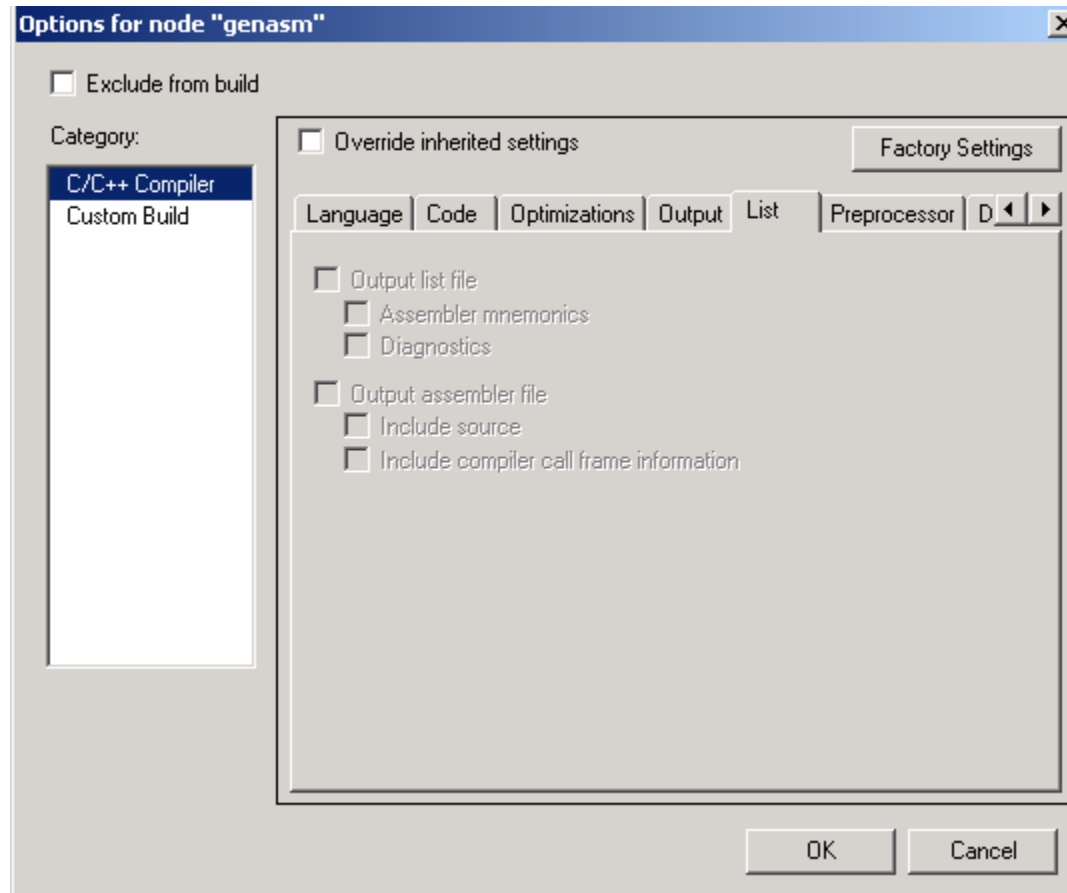


# Generation of assembly file from C file -2/5

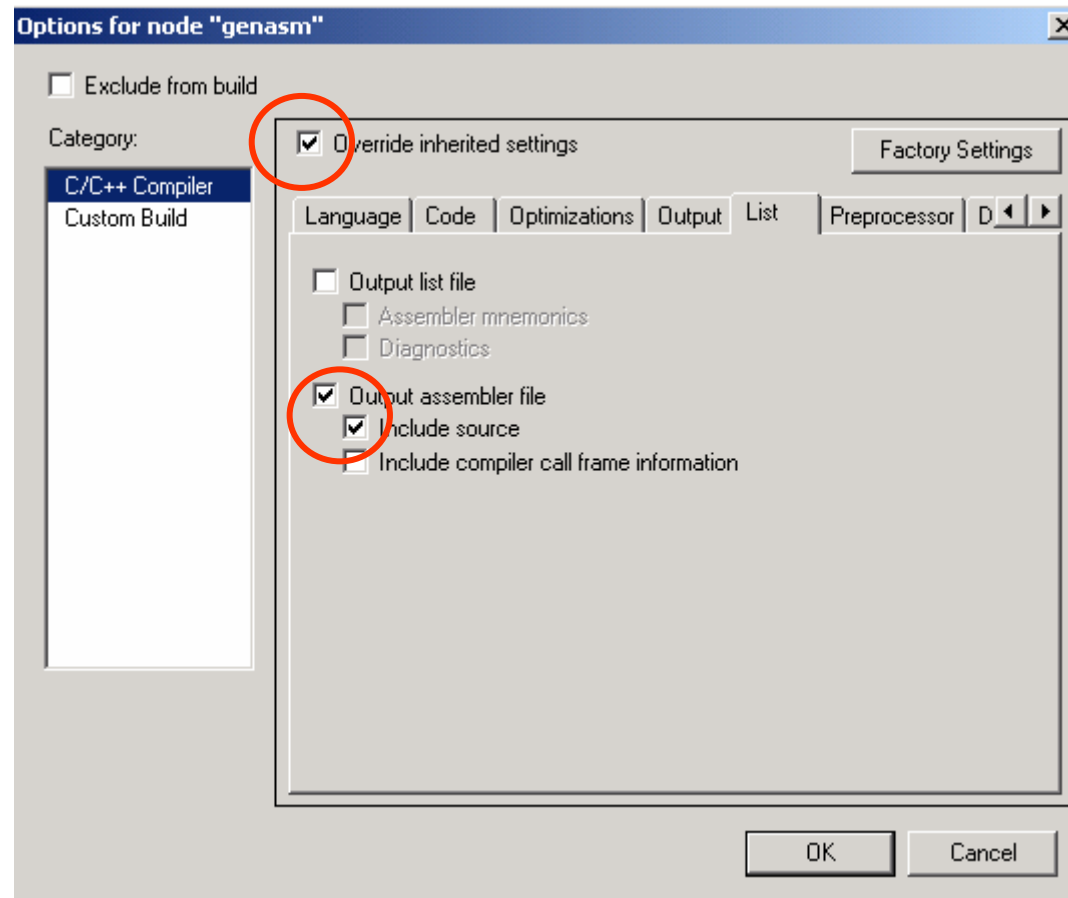
Right click the “utilities.c” file  
Select Options



Override the default option for utilities.c file in order to generate assembly file



Override the default option for utilities.c file in order to generate assembly file



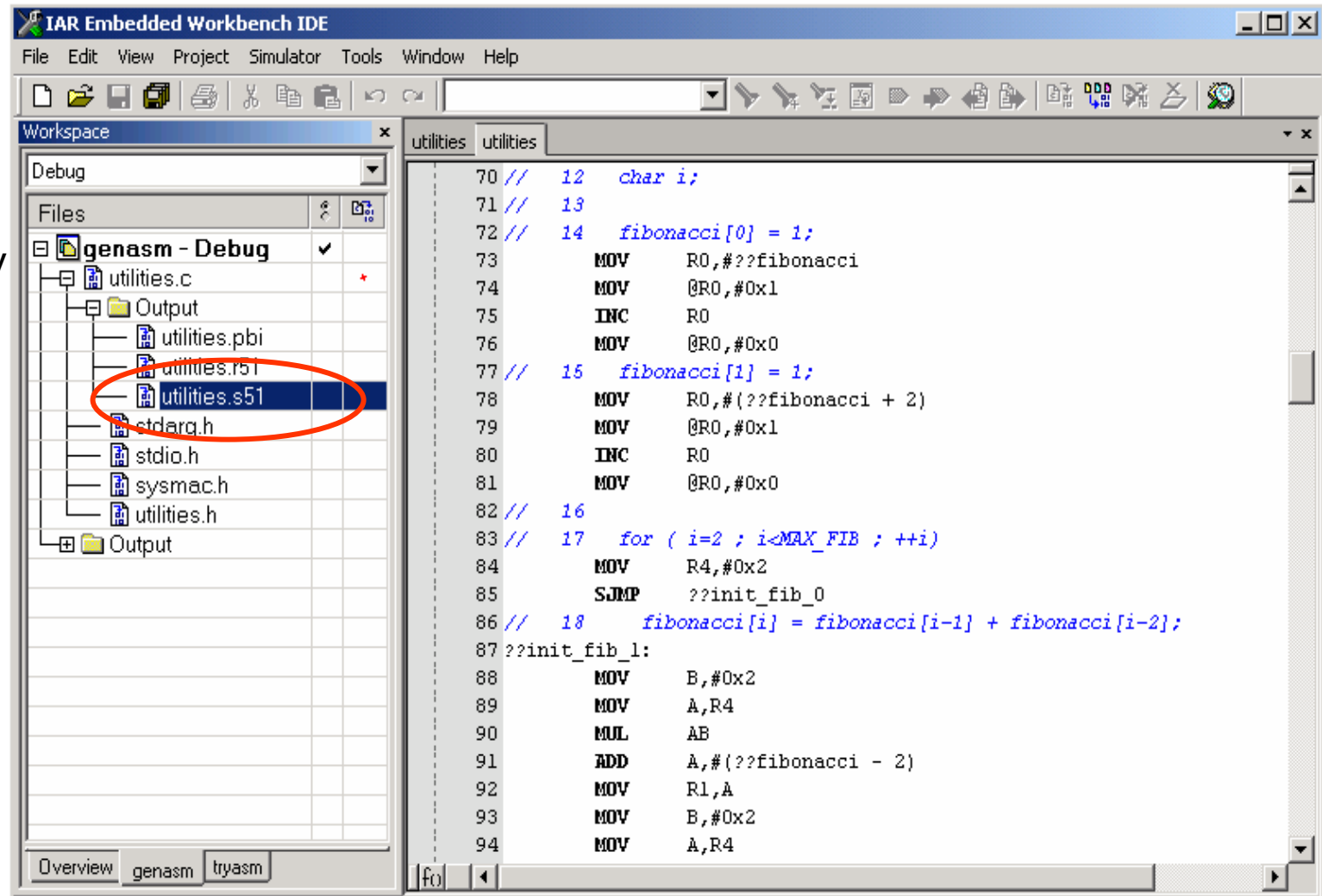
## Generation of assembly file from C file -5/5

Compile Project

Asm file will be Available in

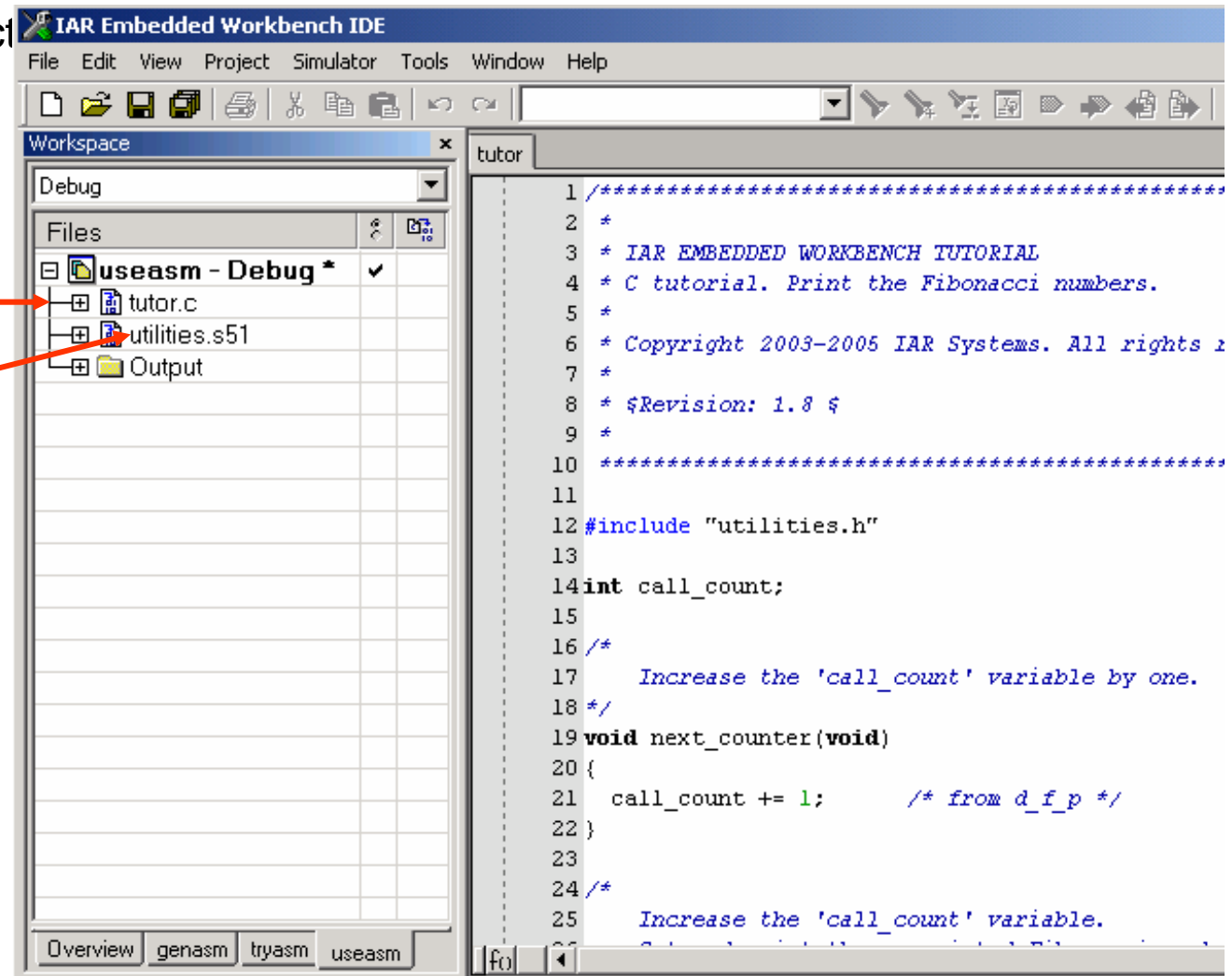
- >Project directory
- >Ist directory

O/P file is utilities.s51



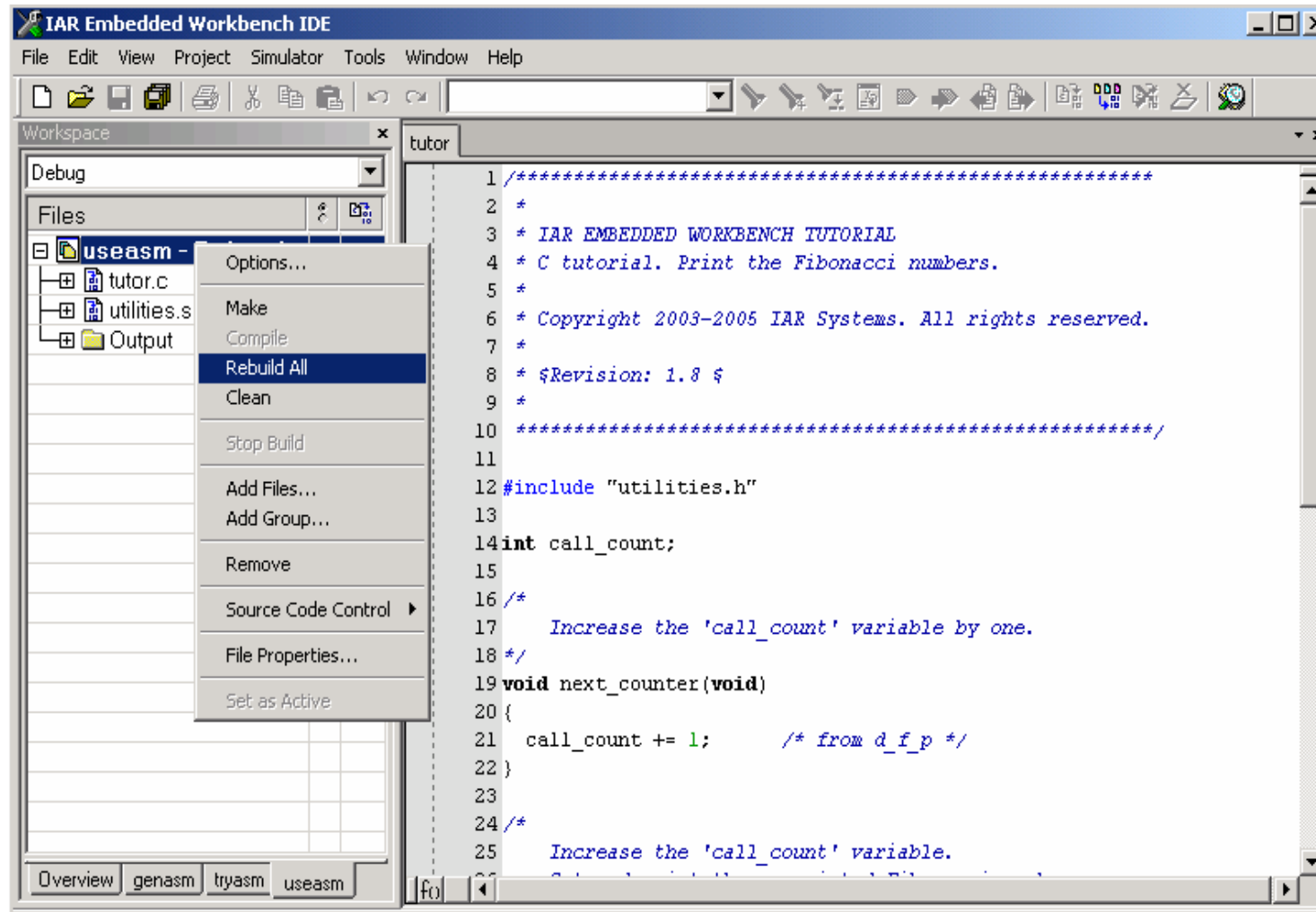
## Mixing asm file with c file-1

- Start a new empty project
- Name it as “useasm”
- Add tutor.c file
- Add utilities.s51 file



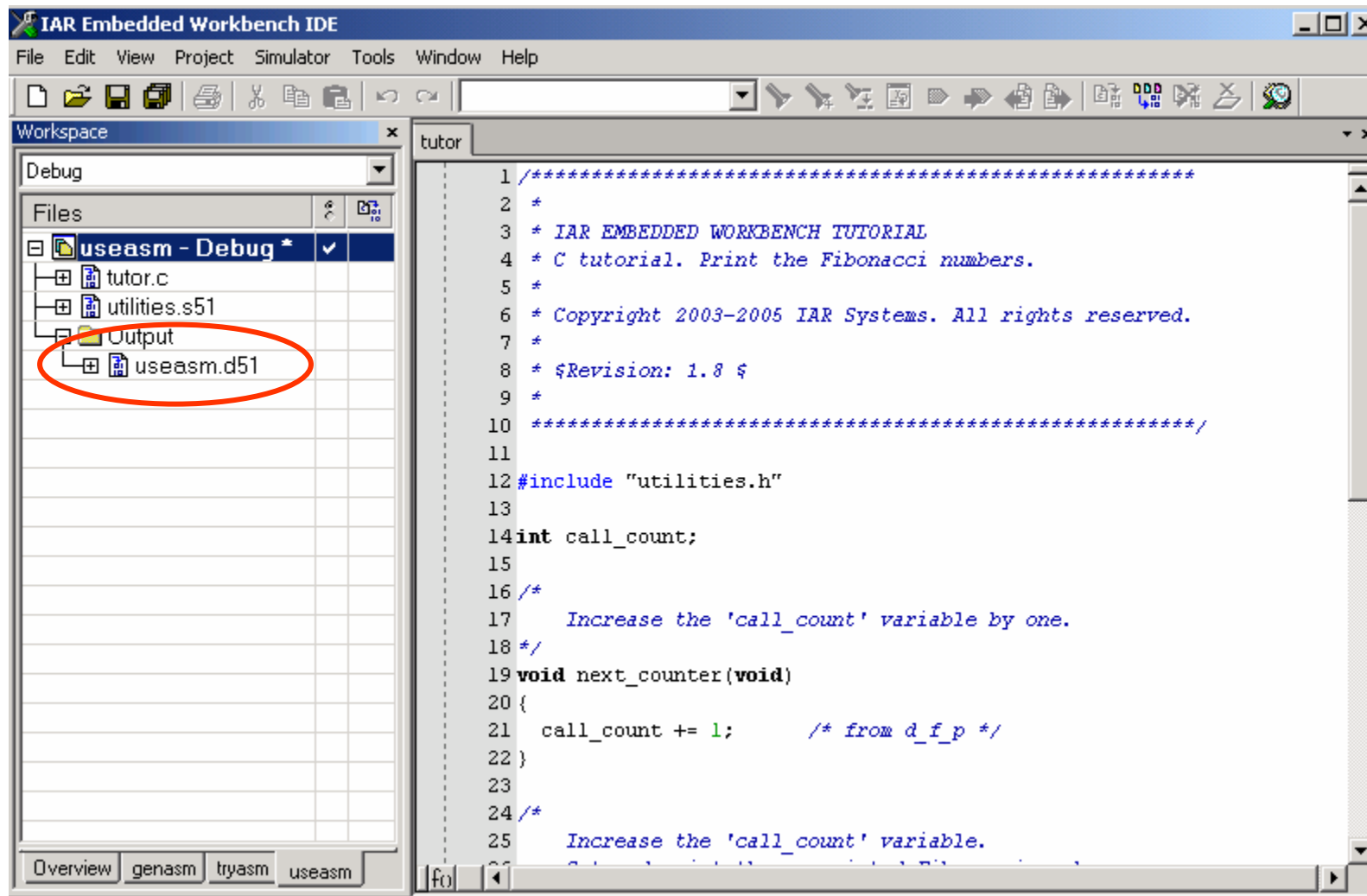
## Mixing asm file with c file-2

### Rebuild project



## Mixing asm file with c file-3

### O/P file will be generated



Generation of Library Modules  
and  
Working with Library



## Tools for library

IAR XLIB : Library builder

Two values are assigned in R1 and R2 register

**Max** routine is called to load maximum value in R1 register

**Min** Routine is called to load minimum value in R1 register

Files used

Main.s51 : mail assembly file

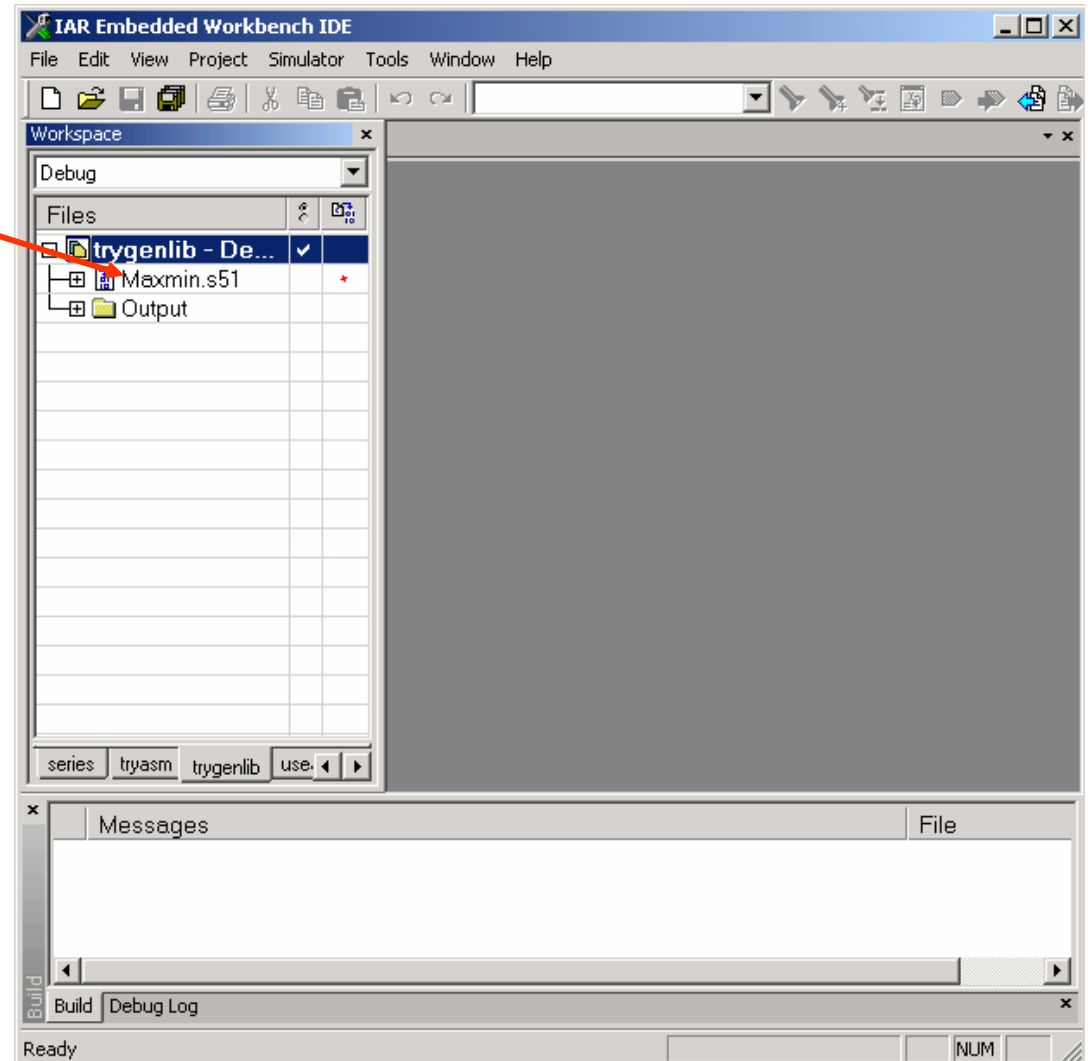
Maxmin.s51: subroutines for max and min subroutines

```
Maxmin *
22      RSEG    NEAR_CODE
23 max:
24      MOV     A,R1
25      CLR     Cy
26      SUBB    A,R2
27      JNC     max_in_rl
28      MOV     A,R2
29      MOV     R1,A
30 max_in_rl:
31      ; Return to caller, result in R1
32      RET
33      ; End of the 'max' module
34      ENDMOD
35      ; -----
36      ; This is the 'min' module
37      ; -----
38      MODULE  min
39      ; Make 'min' accessible to other modules
40      PUBLIC  min
41      ; Subroutine 'min'
42      RSEG    NEAR_CODE
43 min:
44      MOV     A,R1
45      CLR     Cy
46      SUBB    A,R2
47      JC      min_in_rl
48      MOV     A,R2
49      MOV     R1,A
50 min_in_rl:
51      ; Return to caller, result in R1
52      RET
53      ; End of the 'min' module
54      END
```

```
Maxmin Main
10 *****/
11 ; Label this program
12 NAME main
13
14 ; Some needed definitions
15 EXTERN max
16
17 ; locate a jump to program start in the reset vector
18 COMMON INTVEC
19 LJMP main
20
21 ;-----
22 ; Code for the main program
23 ;-----
24
25 RSEG NEAR_CODE
26
27 main:
28 MOV R1,#0x20 ; Parameter #1: a simple constant
29 MOV R2,#0x30 ; Parameter #2: another constant
30 LCALL max ; return MAX(R1,R2) in R1
31
32 ; The simulator end-of-program label
33 ?C_EXIT:
34 exit:
35 LJMP exit
36
37 END
38
```

## Creating library module 1/4

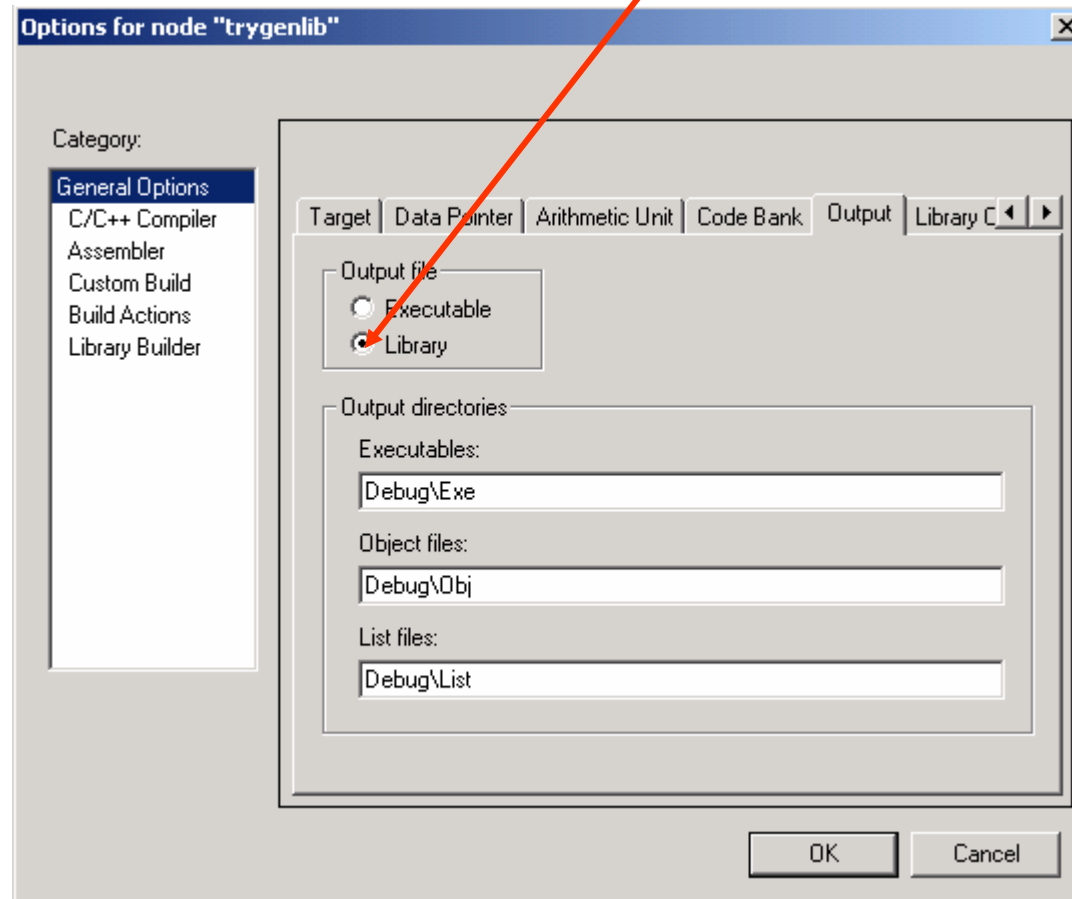
- Make a project (**trygenlib**) and add maxmin.s51 in the project,
- File location is tutor directory



Set following options

- derivative AT89S51
- No standard library (CLIB and DLIB)

Set Output generation option to Library

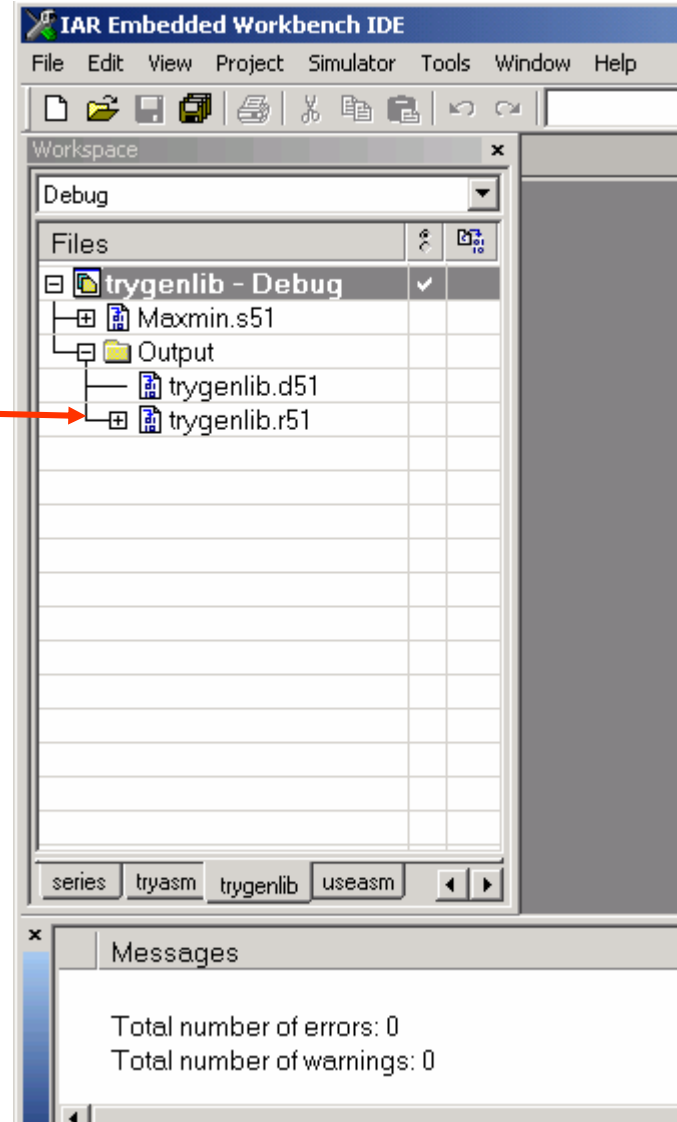


# Creating library module 4/4

Build the project

Library File

File will be available in  
**EXE** directory



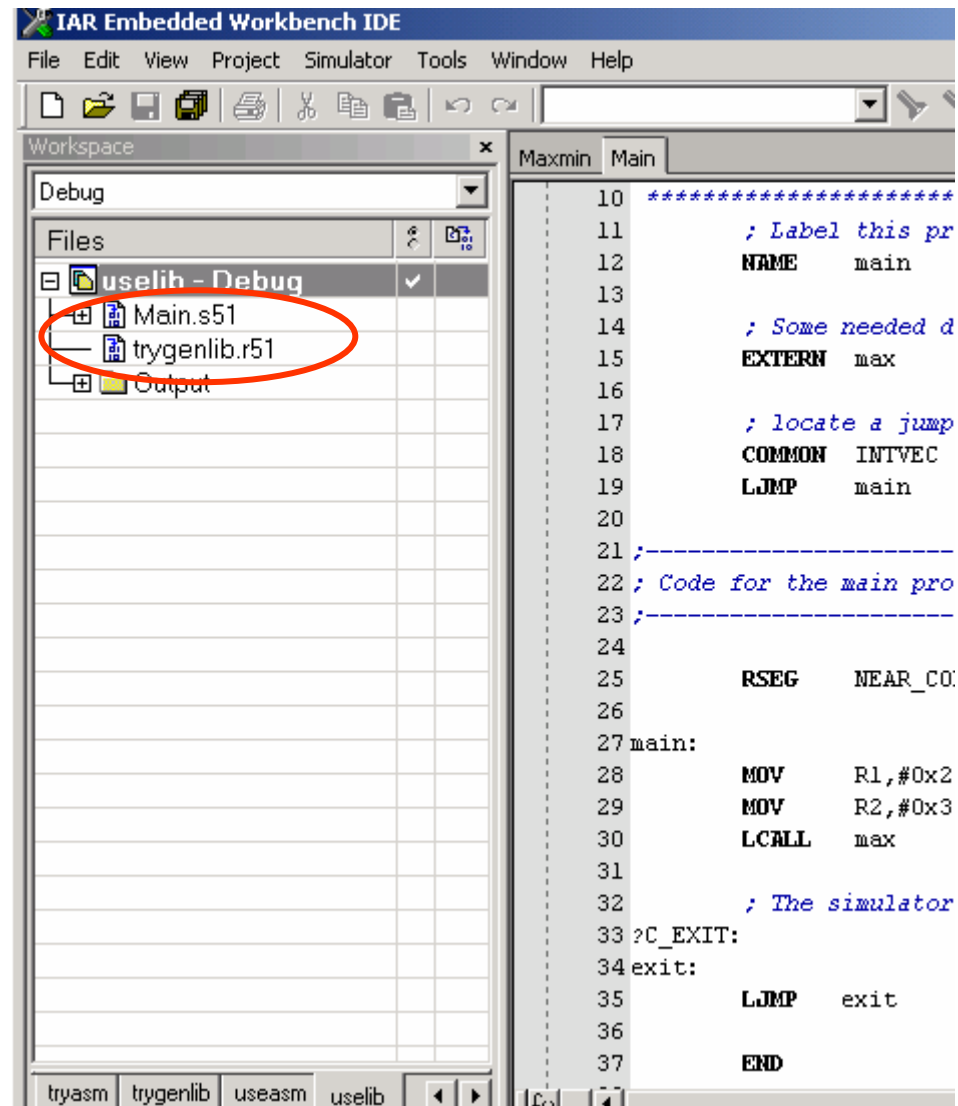


## Using library module 1/2

- Make a project (**uselib**) and add following files in the project
  - main.s51 (main asm file)
  - trygenlib.r51 (library module)

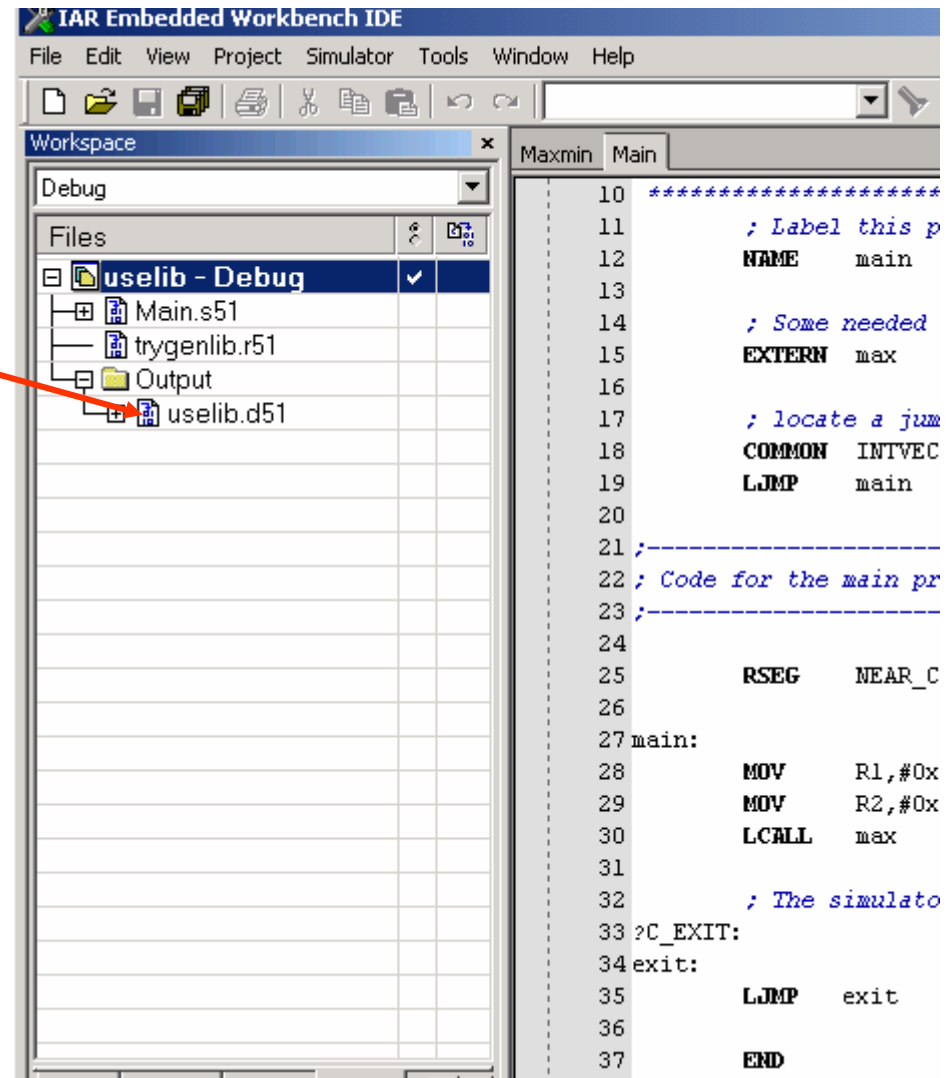
Project options will be following

- derivative AT89S51
- Program entry as defined in application
- disable standard libraries



## Using library module 2/2

- Build the project
- output file will be generated



The screenshot shows the IAR Embedded Workbench IDE interface. The 'Workspace' window displays a project named 'Debug' with the following file structure:

- Files
  - Main.s51
  - trygenlib.r51
  - Output
    - uselib.d51

An orange arrow points from the text 'output file will be generated' to the 'uselib.d51' file in the 'Output' folder.

The main editor window shows assembly code for the 'Main' window:

```
10 *****
11 ; Label this p
12 NAME main
13
14 ; Some needed
15 EXTERN max
16
17 ; locate a jum
18 COMMON INTVEC
19 L JMP main
20
21 ;-----
22 ; Code for the main pr
23 ;-----
24
25 RSEG NEAR_C
26
27 main:
28 MOV R1,#0x
29 MOV R2,#0x
30 LCALL max
31
32 ; The simulato
33 ?C_EXIT:
34 exit:
35 L JMP exit
36
37 END
```

<http://www.embeddedcraft.org>

**EmbeddedCraft**  
crafting of intelligent systems

---