Getting Started with the IAR Embedded Workbench for FPSLIC

Getting Started with the IAR Embedded Workbench™ for FPSLIC™

Application Note

Introduction

The purpose of this application note is to guide new users through the design and development of C/Assembly code for FPSLIC when using an IAR Embedded Workbench (IAREW). This application note provides information on all the settings in IAREW that should be selected for size and speed efficient embedded designs. Options that are disabled when the FPSLIC is selected as the target processor are not described in this application note. This application note assumes that IAREW is already installed on the host system (for those systems without IAREW, 30 day evaluation version can be downloaded from the IAR™ website (www.iar.com), and the full version can be purchased by contacting IAR sales at info@iar.com).

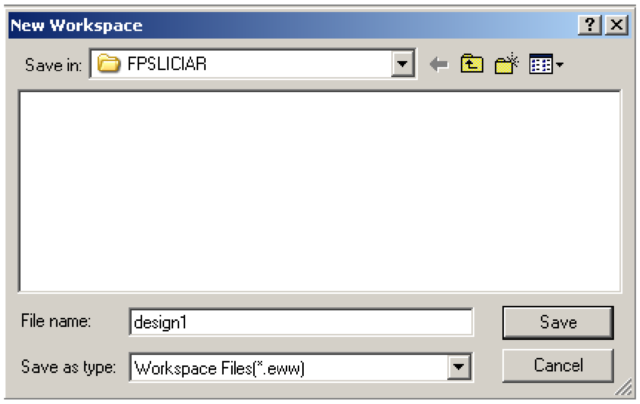
Creating a New Project

This section explains how to create a new project in IAREW. To create a new project, first create a new workspace.

1. Open IAR Embedded Workbench:
   Start > All Programs > IAR Systems > IAR Embedded Workbench for Atmel AVR v3 Evaluation Version
2. Access the New Workspace dialog box:
   File > New > Workspace

The New Workspace dialog box is displayed.

Figure 1. New Workspace Dialog Box



File name: design1

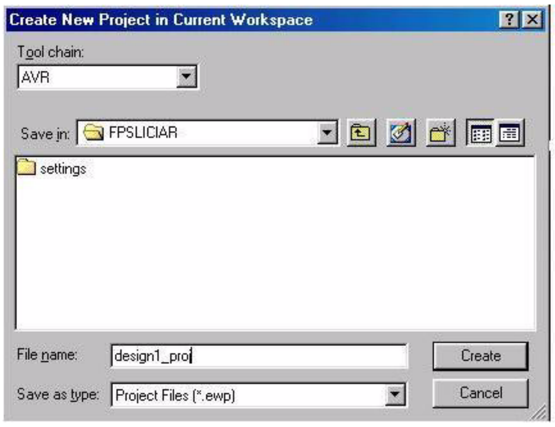
Save as type: Workspace Files(*.eww)

3483A–FPSLI–9/04

3. Create a workspace directory (e.g., C:\FPSLICIAR), type `design1` in the *File name* text box, and then click *Save*.
4. Create a new project in current workspace:
   Project > Create New Project

   By default, the new project should be created in the same directory as that of the workspace: C:\FPSLICIAR. Make sure that the workspace directory name (e.g., FPSLICIAR) is displayed in the *Save in* pull-down box. IAREW version 3.10a allows for multiple projects to be created and filed in the same workplace directory in order to maintain project integrity.
5. AVR is the only option present in the *Tool chain* drop-down list. This application note assumes that the IAREW downloaded is EWAVR, which supports only an AVR controller core. IAR provides different tool chains for different processors.

Figure 2. Create a New Project



Settings in Project > Options for FPSLIC

This section covers the project settings to choose in order to generate size- and speed-efficient debuggable embedded designs. One can choose the options for node design either by selecting *Project* > *Options* or by right-clicking the debug project in the workspace window.

Option Categories

There are six categories of options:

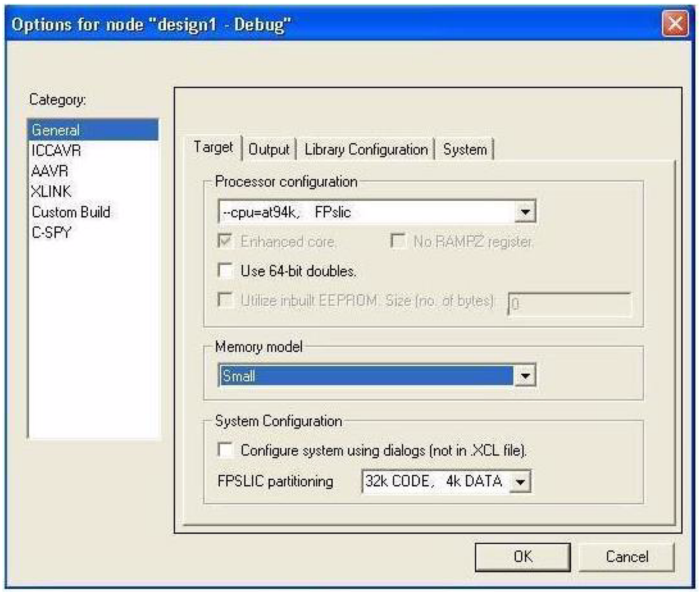
- "General Category" on page 3
- "ICCAVR Category" on page 7
- "XLINK Category" on page 18

General Category

The General category has four tab pages.

- "Target Options Tab Page" on page 3
- "Output Options Tab Page" on page 4
- "Library Configuration Tab Page" on page 5
- "System Tab Page" on page 6

Target Options Tab Page


Options for node "design1 - Debug"

Category:
General
ICCAVR
AAVR
XLINK
Custom Build
C-SPY

Target | Output | Library Configuration | System

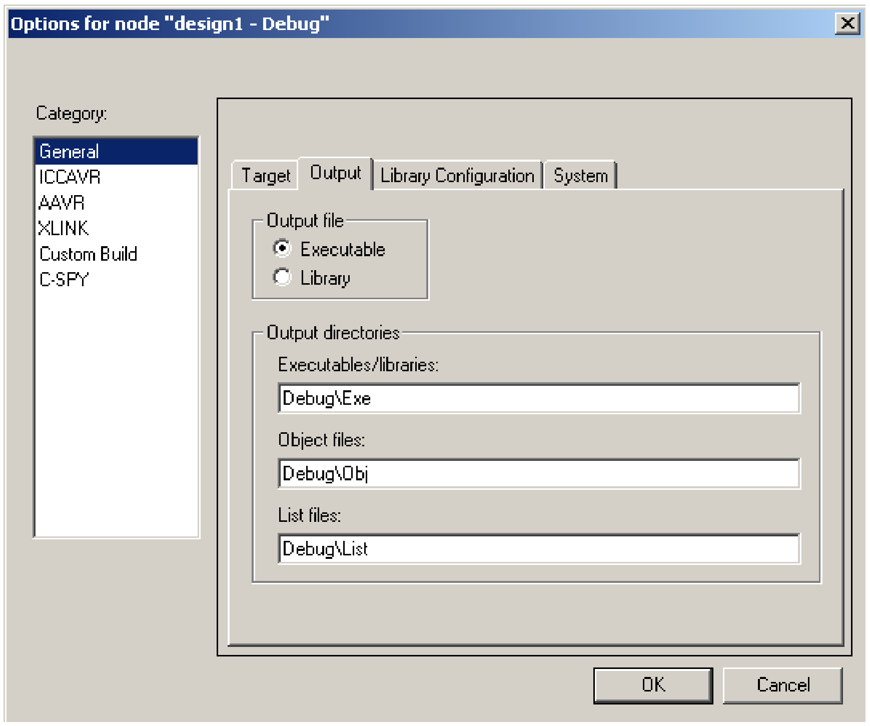
Processor configuration
--cpu=at94k, FPslic
☑ Enhanced core. ☐ No RAMPZ register.
☐ Use 64-bit doubles.
☐ Utilize inbuilt EEPROM. Size (no. of bytes): 0

Memory model
Small

System Configuration
☐ Configure system using dialogs (not in .XCL file).
FPSLIC partitioning 32k CODE, 4k DATA

- *Processor Configuration*
  - Select **--cpu=at94k, FPslic** from the drop-down list box. Choice of processor configuration determines the availability of memory model option.
  - *Enhanced core*: (Disabled)
  - *No RAMPZ register*: (Disabled)
  - *Use 64-bit doubles*: (Enabled) **Select/deselect check box according to the design requirements**. Use this option to force the compiler to use 64-bit doubles instead of 32-bit doubles, which is the default.
  - *Utilize inbuilt EEPROM. Size (no. of bytes)*: (Disabled) Available to processors with built-in EEPROM.
- *Model:* (Enable. Default.) The default, **Small***, is displayed* from the drop-down list box. Use this option to select the memory model for the project.
- *System Configuration:* Select the system configuration of the selected device.
  - *Configure System using dialogs (not in .XCL file)*: (Enabled. Do not select.) Selecting this box overrides the preferred Linker file (XCL) option and configures the system using the linker dialogs instead. (See "System Tab Page" on page 6 for more information.)
  - *FPSLIC Partitioning*: (Enabled. Select.) Select ***Max: 32k CODE, 4k DATA*** from the drop-down list box. Make selection depending on project memory requirements (corresponding to the hardware setup).

Output Options Tab Page


Options for node "design1 - Debug"

Category:
- General
- ICCAVR
- AAVR
- XLINK
- Custom Build
- C-SPY

Target | Output | Library Configuration | System

Output file
- [✓ Executable]
- Library

Output directories

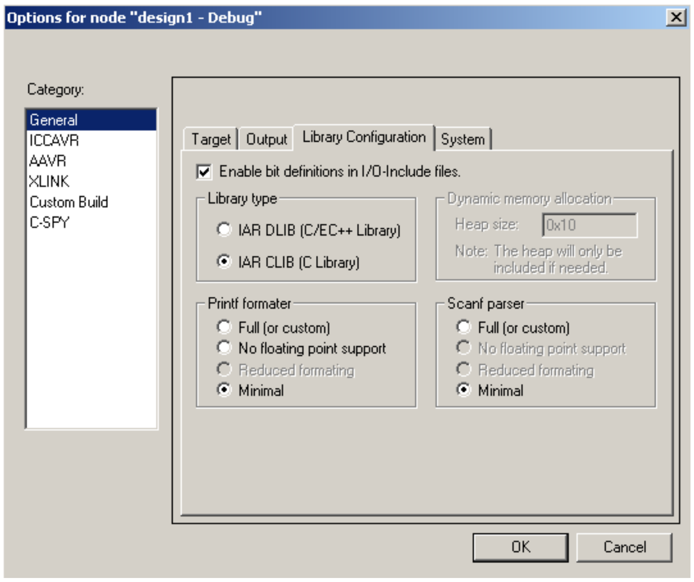
Executables/libraries: Debug\Exe

Object files: Debug\Obj

List files: Debug\List

- *Output File:* (Enabled)
  - Select **Executable** to generate an executable debug file.
  - When *Library* is selected, the XAR category is available, but the XLINK category will not be listed. As a result of the build process, XAR library will create a library output file.
- *Output Directories:* (Enabled) The user can specify the type of output file (executable or library). The user can also specify the destination directories for executable files, object files, and list files. The figure shows what the default setting are when the Output File selected is Executable.
  - *Executables/libraries*: ***(Debug\Exe).*** Use this option to override the default directory for executable files
  - *Object files*: ***(Debug\Obj).*** Use this option to override the default directory for object files
  - *List Files*: ***(Debug\List).*** Use this option to override the default directory for list files

Library Configuration Tab Page


Options for node "design1 - Debug"

Category:
- General
- ICCAVR
- AAVR
- XLINK
- Custom Build
- C-SPY

Target | Output | Library Configuration | System

[✓] Enable bit definitions in I/O-Include files.

Library type
- ( ) IAR DLIB (C/EC++ Library)
- (●) IAR CLIB (C Library)

Dynamic memory allocation
- Heap size: 0x10
- Note: The heap will only be included if needed.

Printf formater
- ( ) Full (or custom)
- ( ) No floating point support
- ( ) Reduced formating
- (●) Minimal

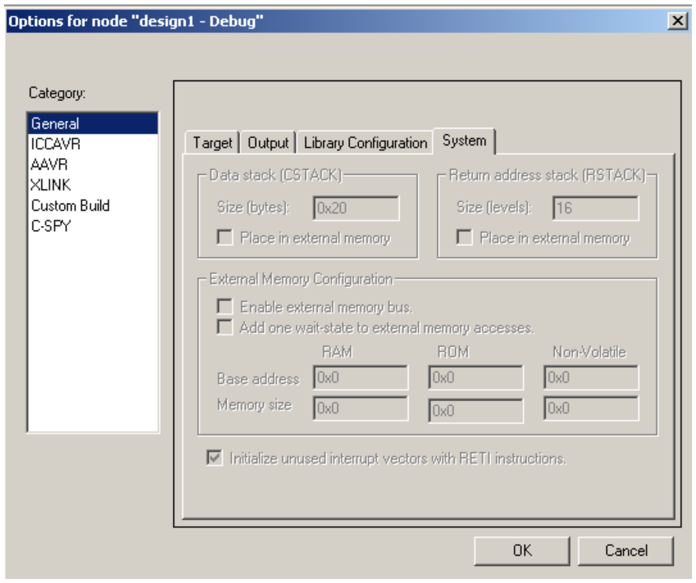
Scanf parser
- ( ) Full (or custom)
- ( ) No floating point support
- ( ) Reduced formating
- (●) Minimal

OK | Cancel

- ***Enable bit definitions in I/O-include files:*** (Enabled. Select.) Select this box to enable the definition of I/O register bit names in the device header files
- *Library type:* (Enabled. Default.) Select a runtime library type: IAR DLIB (C/EC++ Library) or **IAR CLIB (C Library)**. The choice of library type will affect a few other options such as include paths for AVR IAR C/EC++ Compiler and IAR XLINK Linker.
- *Dynamic memory allocation:* (Disabled) Dynamic memory allocation sets the allocated heap size in bytes.
- *Printf formatter/Scanf parser:* (Enabled. Default.) **Minimal** is the default for both. The full versions of printf/scanf formatters are large, and provide facilities that are not required by most embedded applications. To reduce memory consumption, smaller and alternative versions of formatters are provided in the IAR DLIB library.

System Tab Page

Use this tab page to specify system settings if the preferred *Configure System using dialogs (not in .XCL file)* setting is selected in the "Target Options Tab Page" on page 3. If the preferred setting is selected, all of the options on this tab page are disabled because they are included in the .XCL file.


Options for node "design1 - Debug"

Category:
- General
- ICCAVR
- AAVR
- XLINK
- Custom Build
- C-SPY

Target | Output | Library Configuration | System

Data stack (CSTACK)
Size (bytes): 0x20
[ ] Place in external memory

Return address stack (RSTACK)
Size (levels): 16
[ ] Place in external memory

External Memory Configuration
[ ] Enable external memory bus.
[ ] Add one wait-state to external memory accesses.

	RAM	ROM	Non-Volatile
Base address	0x0	0x0	0x0
Memory size	0x0	0x0	0x0

[✓] Initialize unused interrupt vectors with RETI instructions.

OK | Cancel

- *Data stack (CSTACK)/Return address stack (RSTACK):* (Disabled) Size of data stack and return address stack.
- *External Memory Configuration:* (Disabled) Available to only those processors with external memory interface availability.
- *Initialize unused interrupt vectors with RETI instructions:* (Disabled) Unused interrupt vectors can be filled with RETI instruction. This is useful to catch and ignore interrupts that are not handled by application. Note that this option will conflict with the *Fill unused code memory option* which is on the XLINK Processing tab page (see page 24).

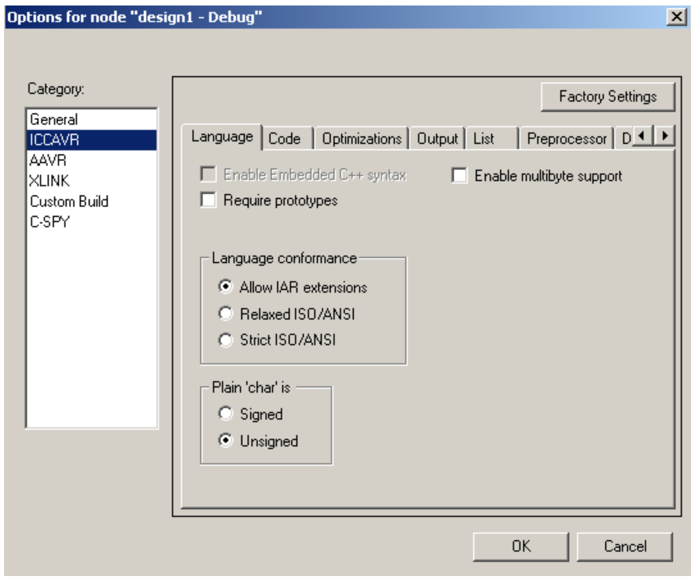
ICCAVR Category

The ICCAVR category has eight tab pages:

- "Language Tab Page" on page 7
- "Code Tab Page" on page 8
- "Optimizations Tab Page" on page 9
- "Output Tab Page" on page 10
- "List Tab Page" on page 11
- "Preprocessor Tab Page" on page 12
- "Diagnostics Tab Page" on page 12
- "Extra Options Tab Page" on page 13

Language Tab Page

Language options enable the use of target-dependant extensions to the C or embedded C++ language.



Enable Embedded C++ syntax: [ ]
Enable multibyte support: [ ]
Require prototypes: [ ]

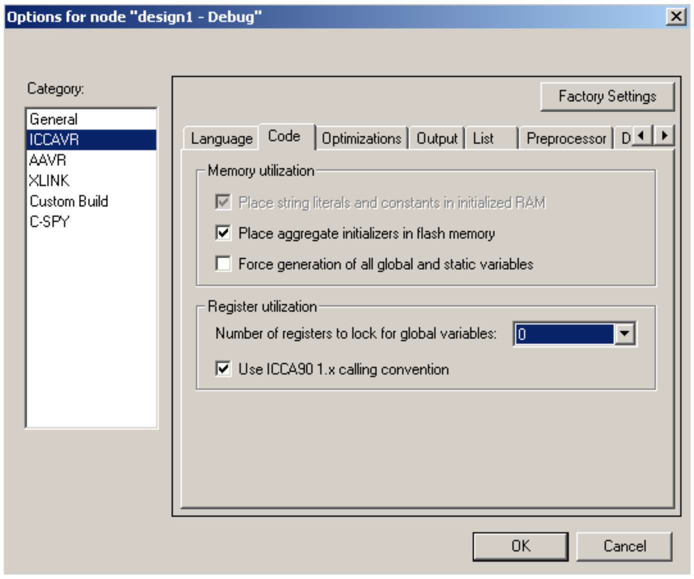
Language conformance
[✓ Allow IAR extensions]
Relaxed ISO/ANSI
Strict ISO/ANSI

Plain 'char' is
Signed
[✓ Unsigned]

- *Enable Embedded C++ syntax:* (Disabled)
- *Require prototypes:* (Enabled. Do not select.) This option forces the compiler to verify that all functions have proper prototypes. Using this option means that code containing any of the following will generate an error: A function call of a function with no declaration, or with a Kemighan & Ritchie C declaration. A function definition of a public function with no previous prototype declaration. An indirect function call through a function pointer with a type that doesn't include prototype.
- *Enable multibyte support:* (Enabled. Do not select.)
- *Language conformance:* (Enabled) Language extensions must be enabled for the IAR AVR compiler to be able to accept AVR-specific key words as extensions to the standard C or embedded C++ language.
  - ***Allow IAR extensions*** is enabled by default.
  - Relaxed ISO/ANSI disables IAR extensions but does not adhere to strict ISO/ANSI.
  - Select Strict ISO/ANSI option to adhere to strict ISO/ANSI C standards.
- *Plain 'char' is:* (Optional) Default is ***Unsigned***; select ***Signed*** if the design requires that explicit defined variables be treated as signed.

Code Tab Page

The code option determines the usage of segments and registers.


Options for node "design1 - Debug"

Category:
General
ICCAVR
AAVR
XLINK
Custom Build
C-SPY

Factory Settings

Language | Code | Optimizations | Output | List | Preprocessor | D

Memory utilization
☑ Place string literals and constants in initialized RAM
☑ Place aggregate initializers in flash memory
☐ Force generation of all global and static variables

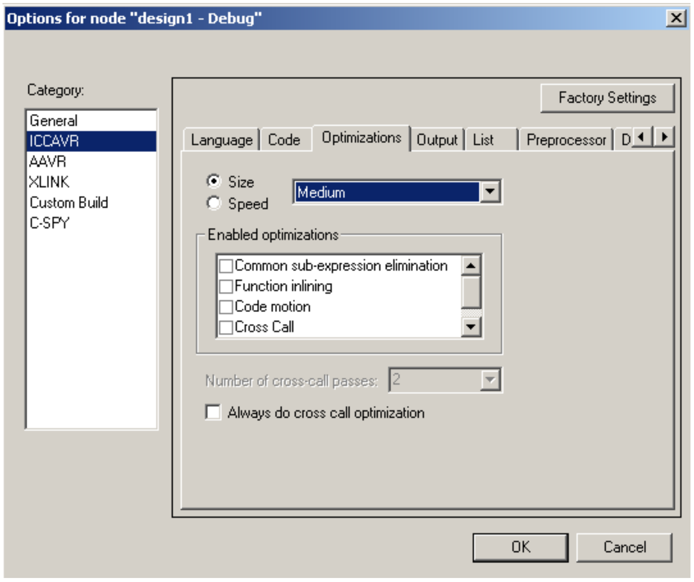
Register utilization
Number of registers to lock for global variables: 0
☑ Use ICCA90 1.x calling convention

OK | Cancel

- *Memory Utilization*
  - *Place string literals and constants in initialized RAM* (Disabled.) This option is used to override the default placement of constants and literals.
  - ***Place aggregate initializers in flash memory*** (Enabled. Default.) Even though this option is selected, initializers are placed as mentioned in the XLINK Command Line (XCL) file.
  - *Force generation of all global and static variables*: (Enabled. Do not select). Use this option to apply __root extended keyword to all global and static variables. This will make sure that variables are not removed by the XLINK linker.
- *Register Utilization*
  - *Number of registers to lock for global variables:* (Enabled. Default.) ***0*** is the selected default. This option is used to lock registers that are to be used for global register variables.
  - ***Use ICCA90 1.x calling conventions:*** (Enabled. Default.) This option is provided for backward compatibility. It makes all functions and function calls use the calling conventions of A90 IAR compiler, ICCAR90.

Optimizations Tab Page

Optimization options determine the type and level of optimization for generation of object code.


Options for node "design1 - Debug"

Category: General, ICCAVR (selected), AAVR, XLINK, Custom Build, C-SPY

Tab: Optimizations

(● Size / ○ Speed): Medium

Enabled optimizations:

- ☐ Common sub-expression elimination
- ☐ Function inlining
- ☐ Code motion
- ☐ Cross Call

Number of cross-call passes: 2

☐ Always do cross call optimization

- *Size and Speed:* (Enabled) Select **Size** and **Medium**. The AVR IAR compiler supports two optimization models-size and speed – at different optimization levels.
  - None: No Optimization
  - Low: Fully Debuggable
  - Medium: Heavy optimization can make the program flow hard to follow during debug
  - High: Full Optimization
- *Enabled Optimizations:* (Enabled. Do not select any of the options.) Choose from the list depending upon project requirements. The IAR compiler supports the following types of optimizations.
  - *Common sub expression elimination*: Redundant re-evaluation of common sub-expressions is by default eliminated at optimization level high. This optimization normally reduces both code size and time (the resulting code may however be difficult to debug).
  - *Function inlining*: Function inlining means that a simple function, whose definition is known at compile time, is integrated into the body of its caller to eliminate the overhead of the call. This optimization, which is performed at optimization level high, normally reduces execution time but increases code size (resulting code may also be difficult to debug). The compiler decides which functions to inline. Different heuristics are used when optimizing for speed.
  - *Code motion*: Evaluation of loop invariant expressions and common sub-expressions are moved to avoid redundant re-evaluation. This optimization, which is performed at optimization level high, normally reduces the code size and execution time (the resulting code may however be difficult to debug). This option has no effects at optimization levels none or low.
  - *Cross call*: This optimization creates subroutines of common code sequences, and is performed as a size optimization at level high. Although it can drastically reduce the code size, this option increases the execution time

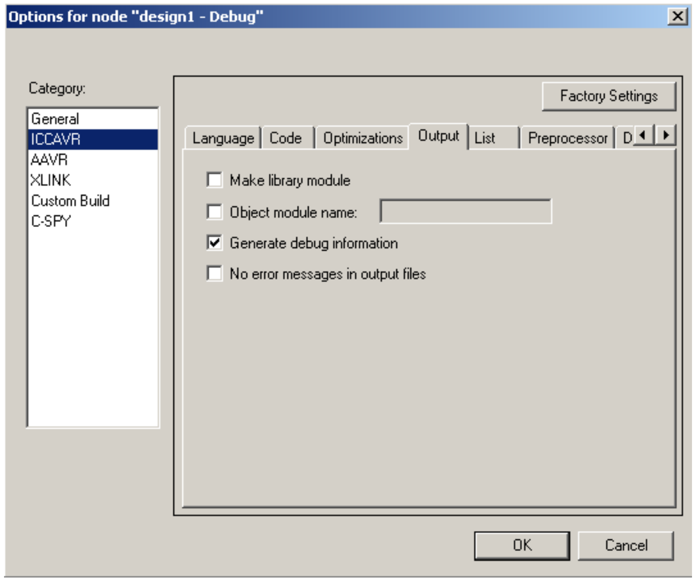
as well as the internal return data stack, RSTACK usage. Avoid this option, if the target processor has a hardware stack or small RAM-based internal return stack segment, RSTACK.

– *Clustering of variables*: When clustering variables is enabled, static and global variables are arranged so that variables that are accessed in the same function are stored close to each other. This makes it possible for the compiler to use the same base pointer for several accesses. Alignment gaps between variables can also be eliminated.

- *Number of cross-call passes:* (Disabled) Use this option to decrease the RSTACK usage by running the cross-call optimizer N times, where N can be from 1 to 5. This option is enabled when the Always do cross call optimization check box is selected.
- *Always do cross call optimization:* (Enabled. Do not select.) Use this option to force the compiler to run the cross call optimizer, regardless of the optimization level. The cross call optimizer is otherwise only run at the high size optimization level.

Output Tab Page

The output option determines the output file format of the compiled file, including the level of the debugging information in the object code.



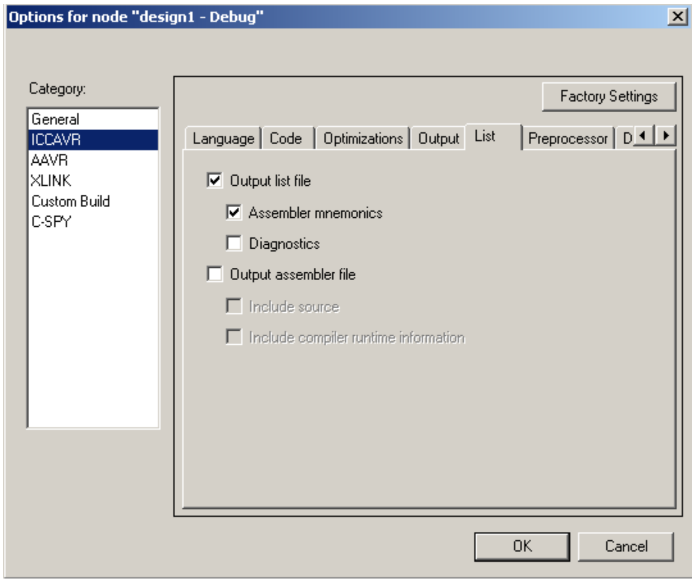
☐ Make library module
☐ Object module name:
☑ Generate debug information
☐ No error messages in output files

- *Make library module:* (Enabled. Do not select.) By default compiler generates program modules, which are always included during linking. Use this option to make a library module that will only be included if it is referenced in user application. Select Make library module option to make the object file be treated as a library module rather than as a program module.
- *Object module name:* (Enabled. Do not select.) Normally, the internal name of the object module is the name of the source file, without a directory name or extension. Use this option to set the object module name explicitly.
- **Generate debug information**: (Enabled. Default.) This option causes the compiler to include additional information in the object modules that is required by C-SPY and other symbolic debuggers. Uncheck this box if you do not want the compile to generate the debug information. Note that enabling this option increases the size of the object file.

- *No error messages in output files:* (Enabled. Do not select.) This option minimizes the size of application object file by excluding messages from the UBROF files. A file size decrease of up to 60% can be expected by excluding the messages information. Note that this option does not affect the code generation; it does not perform any optimizations.

List Tab Page

The list options determine whether a list file is produced, and what information is included in the list file. Normally the compiler does not generate a list file, the user must select one of the following options in order to generate a list file.

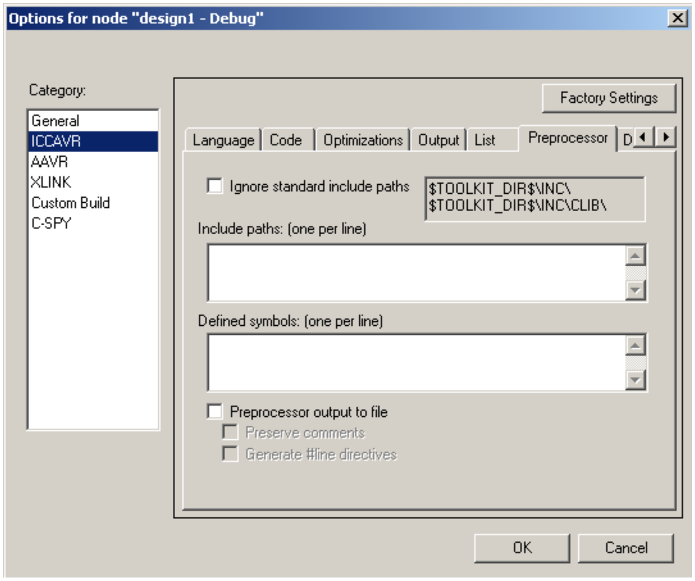


[✓] Output list file
[✓] Assembler mnemonics
[ ] Diagnostics
[ ] Output assembler file
[ ] Include source
[ ] Include compiler runtime information

- ***Output List File:*** (Enabled) Generates a list file.
- ***Assembler mnemonics:*** (Enabled) Includes assembler mnemonics in the list file.
- *Diagnostics:* (Enabled. Do not select.) Includes diagnostic information in the list file.
- *Output assembler file:* (Enabled Do not select.) Generates an assembler list file.
- *Include source:* (Disabled) Includes source code in the assembler file.
- *Include compiler runtime information:* (Disabled) Includes compiler-generated information for runtime model attributes, call frame information, and frame size information.

Preprocessor Tab Page

This option lets the user define symbols and include paths for use by the compiler.



Ignore standard include paths

$TOOLKIT_DIR$\INC\
$TOOLKIT_DIR$\INC\CLIB\

Include paths: (one per line)

Defined symbols: (one per line)

Preprocessor output to file

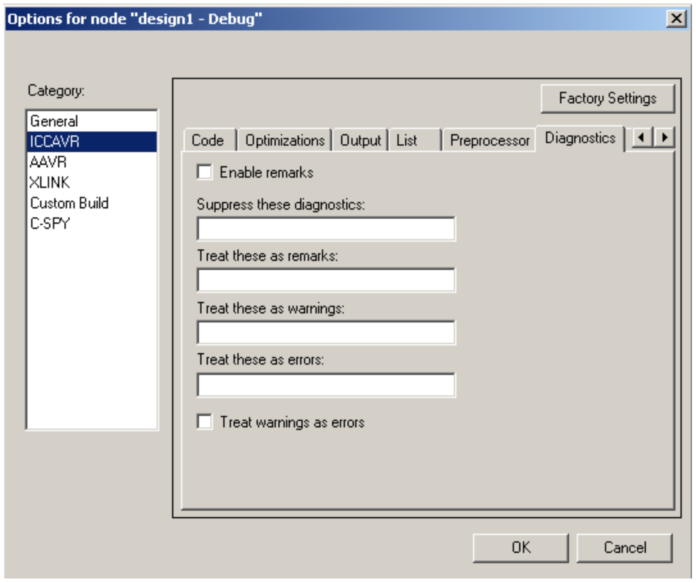
Preserve comments

Generate #line directives

- Options in this tab are not useful for FPSLIC, hence no descriptions are provided.

Diagnostics Tab Page

The diagnostics options determine how diagnostics are classified and displayed. Use the diagnostic option to override the default classification of the specified diagnostics. The diagnostics can't be suppressed for fatal errors, and fatal errors can't be reclassified.



Enable remarks

Suppress these diagnostics:

Treat these as remarks:

Treat these as warnings:

Treat these as errors:

Treat warnings as errors

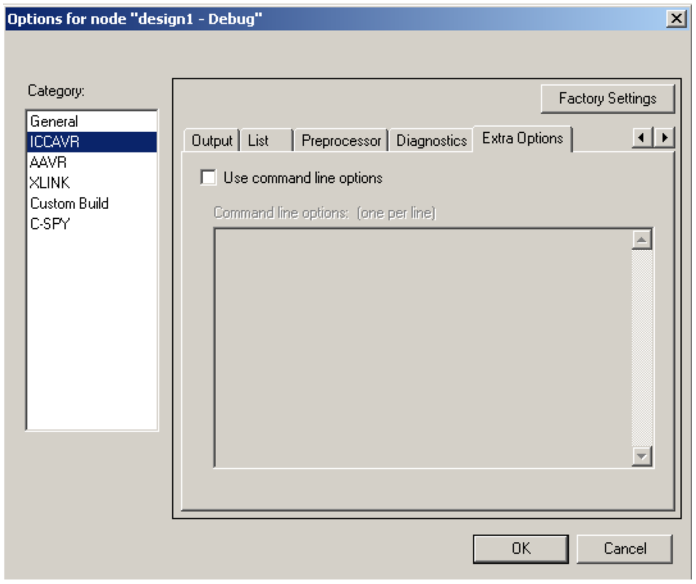
- *Enable remarks:* (Enabled. Do not select.) The least diagnostic messages are called remarks. A remark indicates a source code construct that might cause strange behavior in the generated code. Select this option for the compiler to generate remarks.

- *Suppress these diagnostics:* (Enabled. Do not select.) This option suppresses the output of the diagnostics for the tags that the user specifies.
- *Treat these as remarks:* (Enabled. Do not select.) A remark is the least severe type of diagnostic message. It indicates a source code construct that might cause strange behavior in the generated code. Use this option to classify diagnostics as remarks.
- *Treat these as warnings:* (Enabled. Do not select.) A warning indicates an error or omission that is of concern, but which will not cause the compiler to stop before compilation is completed. Use this option to classify diagnostics messages as warnings.
- *Treat these as errors:* (Enabled. Do not select.) An error indicates a violation of the C or Embedded C++ rules of such severity that object code will not be generated, and the exit code will be non-zero. Use this option to classify diagnostic messages as errors.
- *Treat all warnings as errors:* (Enabled. Do not select.) Use this option to make the compiler treat all warnings as errors. If the compiler encounters an error, object code will not be generated.

Extra Options Tab Page

Additional command line options for the AVR IAR C/EC++ compiler can be specified here.

Note: This option is not supported by the GUI.



- Options in this tab are not useful for FPSLIC, hence no descriptions are provided.

AAVR Category

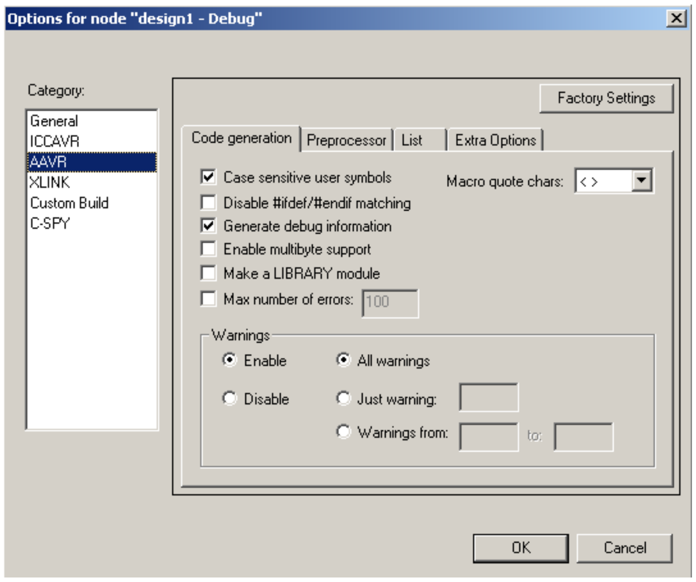
Should only be used in cases where the design is developed using assembly code.

The AAVR category has four tab pages.

- "Code Generation Tab Page" on page 14
- "Preprocessor Tab Page" on page 15
- "List Tab Page" on page 16
- "Extra Options Tab Page" on page 17

Code Generation Tab Page

The code generation options control the assembler output.


Options for node "design1 - Debug"

Category: General, ICCAVR, AAVR, XLINK, Custom Build, C-SPY

Code generation | Preprocessor | List | Extra Options

- [✓] Case sensitive user symbols
- [ ] Disable #ifdef/#endif matching
- [✓] Generate debug information
- [ ] Enable multibyte support
- [ ] Make a LIBRARY module
- [ ] Max number of errors: 100

Macro quote chars: < >

Warnings: [✓ Enable] Disable; [✓ All warnings] Just warning: Warnings from: to:

- ***Case sensitive user symbols:*** (Enabled. Default.) Case-sensitivity is on. This means that, for example, "LABEL" and "label" refer to different symbols. The user can deselect this box to turn case sensitivity off, in which case "LABEL" and "label" will refer to the same symbol.
- *Disable #ifdef/#endif matching:* (Enabled. Optional.) Use this option to disable the matching between #ifdef and #endif matching.
- ***Generate debug information:*** (Enabled. Default). This option must be selected, if users want to use a debugger with the application.

Note: This option is selected by default in debug project, where as not in a release process.

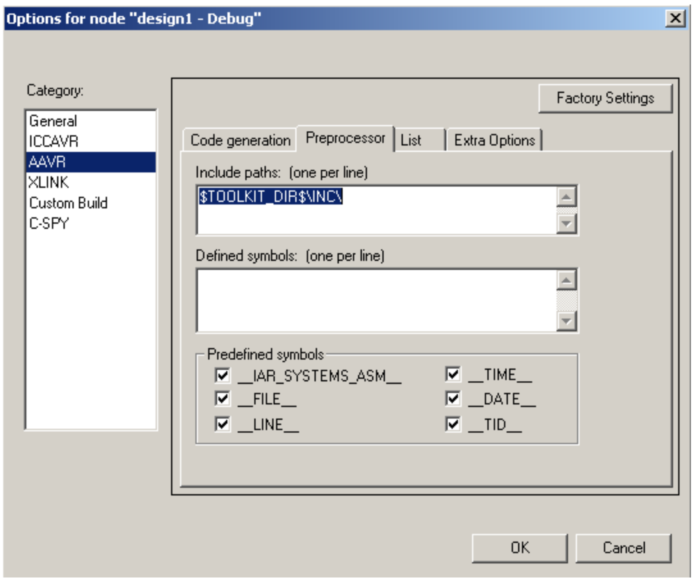
- *Enable multibyte support:* (Enabled. Do not select.) By default multibyte characters can't be used in C/Embedded C++ source code. If this option is used, multibyte characters in the source code are interpreted according to the host computer default settings for multibyte support.
- *Make a library module:* (Enabled. Do not select.) This option is used to make a library module that will only be included if it is referenced in your application.
- *Max number of errors:* (Enabled. Do not select.) By default, the maximum number of the errors reported by assembler is 100. This option lets the user increase or decrease the default.
- ***Warnings:*** (Enabled. Default.) The assembler displays a warning message when it finds an element of the source code that is legal, but probably the result of a programming error. By default, ***All Warnings*** are enabled. The Warnings option lets the user enable only some warnings, or to disable all or some warnings. Use the

Warnings radio buttons and entry fields to specify which warnings to enable or disable.

- ***Macro quote chars:*** (Enabled. Default. < >). This option sets the characters used for the left and right quotes of each macro argument.

Preprocessor Tab Page

This option lets the user define include paths and symbols, and remove the pre-defined symbols in assembler.



Include paths: (one per line)
$TOOLKIT_DIR$\INC\

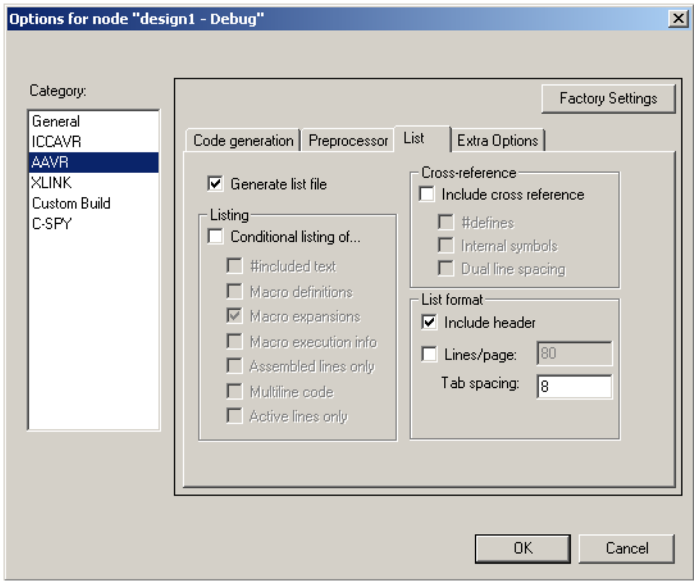
Defined symbols: (one per line)

Predefined symbols
☑ __IAR_SYSTEMS_ASM__ ☑ __TIME__
☑ __FILE__ ☑ __DATE__
☑ __LINE__ ☑ __TID__

- *Include paths (one per line):* (Enabled. Default. **$TOOLKIT_DIR$\INC**) By default the assembler searches for #include files in the current working directory. Include option lets the user to specify the names of directories that the assembler will also search, if it fails to fine the file.
- *Defined symbols:* (Enabled. Optional.) The defined symbols option is useful for conveniently specifying a value or choice that would otherwise be specified in the source file. This option has the same effect as a #define statement at the top of the source file.
- *Predefined symbols:* (Enabled. Default.) By default, the assembler provides certain pre-defined symbols. This option lets the user to undefine such a predefined symbol to make its name available for application use. To define a symbol, deselect it in the *Predefined symbols* list.

List Tab Page

List options are used for making the assembler generate a list file, for selecting the list file contents, and to generate other listing type output.



- [✓] Generate list file
- Listing
  - [ ] Conditional listing of...
  - [ ] #included text
  - [ ] Macro definitions
  - [✓] Macro expansions
  - [ ] Macro execution info
  - [ ] Assembled lines only
  - [ ] Multiline code
  - [ ] Active lines only
- Cross-reference
  - [ ] Include cross reference
  - [ ] #defines
  - [ ] Internal symbols
  - [ ] Dual line spacing
- List format
  - [✓] Include header
  - [ ] Lines/page: 80
  - Tab spacing: 8

- *Generate list file:* (Enabled. Optional.) Select if C language-based development was used (see "AAVR category:" on page 26). By default, the assembler does not generate a list file. The assembler generates a listing when this option is selected.

Note: If the users want to save the list file in another directory than the default directory for list files, use the Output directories option in General category. When Generate list file is selected Listing, Cross-reference, and List format options become available.

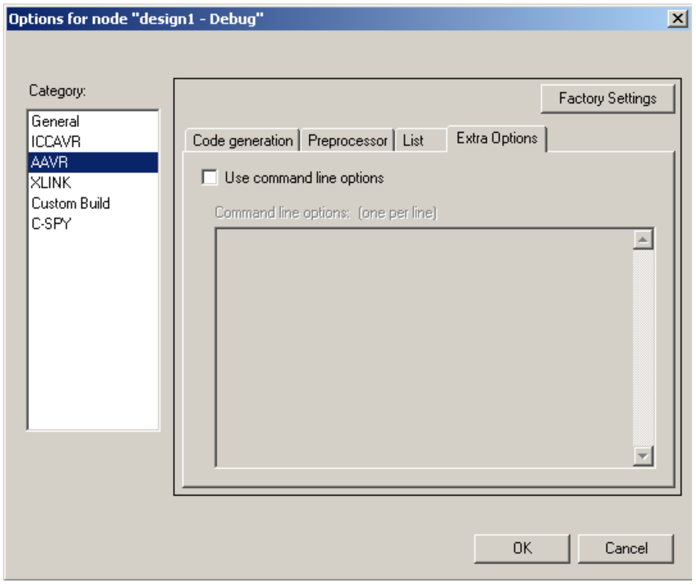
- *Listing:* (Enabled. Select.) Use the ***Conditional listing of*** option to select what type of information to include in the list file.
  – #ncluded text: (Enabled. Optional.) Includes #includes files in the list file.
  – Macro definitions: (Enabled. Optional.) Includes macro definitions in the list file
  – Macro expansions: (Enabled. Optional.) Includes macro expansions in the list file.
  – Macro execution info (Enabled. Optional.) Prints macro execution information on every call of a macro
  – Assembled lines only (Enabled. Optional.) Excludes lines in false conditional assembly sections from the list file
  – Multiline code (Enabled. Optional.) Lists the code generated by directives on several lines if necessary
  – Active lines only (Enabled. Optional.) Includes active lines only
- *Cross-reference:* (Enabled. Do not select.) The include cross-reference option causes the assembler to generate a cross reference table at the end of the list file.
- *List format:* This option lets the user specify details about the format of the assembler list file.
  – *Include header:* (Enabled. Do not select.) The header of the assembler list file contains information about the product version, date and time of assembly, and the command line equivalents of the assembler options that are used.

– *Lines/page:* (Enabled. Do not select.) The default number of lines per page is 44 for the assembler list file. This option is used to set the number of lines per page, with in the range of 10 to 150.
– ***Tab spacing:*** (Enabled. 8 character Default). This option is used to change the number of character positions per tab space, ranging from 2 to 9. By default the assembler sets eight character positions per tab space.

Extra Options Tab Page

Additional command line options for the AVR IAR assembler can be specified here.

Note: This option is not supported by the GUI.



- Options in this tab are not useful for FPSLIC, hence no descriptions are provided

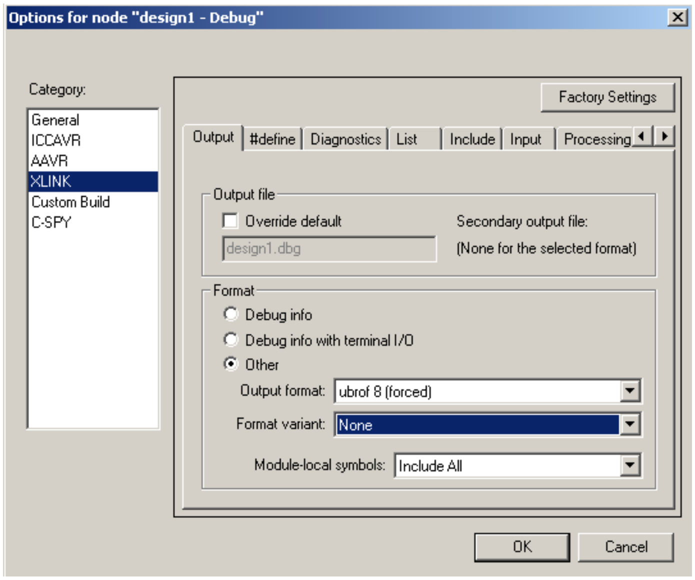
XLINK Category

The XLINK category has eight tab pages.

- "Output Tab Page" on page 18
- "#define Tab Page" on page 19
- "Diagnostics Tab Page" on page 20
- "List Tab Page" on page 21
- "Include Tab Page" on page 22
- "Input Tab Page" on page 23
- "Processing Tab Page" on page 24
- "Extra Options Tab Page" on page 24

Output Tab Page

Output options are used to specify the output format and the level of debugging information included in the output file.



Output file
[ ] Override default
design1.dbg
Secondary output file:
(None for the selected format)

Format
( ) Debug info
( ) Debug info with terminal I/O
(•) Other
Output format: ubrof 8 (forced)
Format variant: None
Module-local symbols: Include All

- *Output file:* (Enabled. Default. ***source filename.dbg***) This option can be used to specify the file name or file name extension other than the default.
- *Override default:* (Enabled. Optional.) This option lets the user specify the name of the XLINK output file. If the name is not specified, the linker will use the name with project.dbg (/project.d90) extension. If the user selects a format that generates two output files, the file type that is specified will only affect the primary output.
- *Format:* These options determine the format of the output file generated by the IAR XLINK linker. The output file is used as input to either a debugger or as an input for programming the target system.

  Note: The IAR systems proprietary output format is called UBROF (Universal Binary Relocatable Object Format)

  Note: In debug project, Debug info with terminal I/O is selected by default. In Release project, Motorola is selected by default.

  – *Debug info:* (Enabled. Do not select.) This option creates a UBROF output file, with a d90 filename extension.

  – *Debug info with terminal I/O:* (Enabled. Do not select.) This option is equivalent to the Debug info option but also includes debugging information for terminal I/O, which means that stdin and stdout are redirected to the Terminal I/O window.

– **Other:** (Enabled. Select.)

  Note: AVR Studio 4.08 can't debug the output file generated by selecting these options.

– *Output format:* (Enabled. Select.) The default output file generated is of ubrof 9 format. Use the Output format drop-down list to select the appropriate output **ubrof 8 (forced)**.

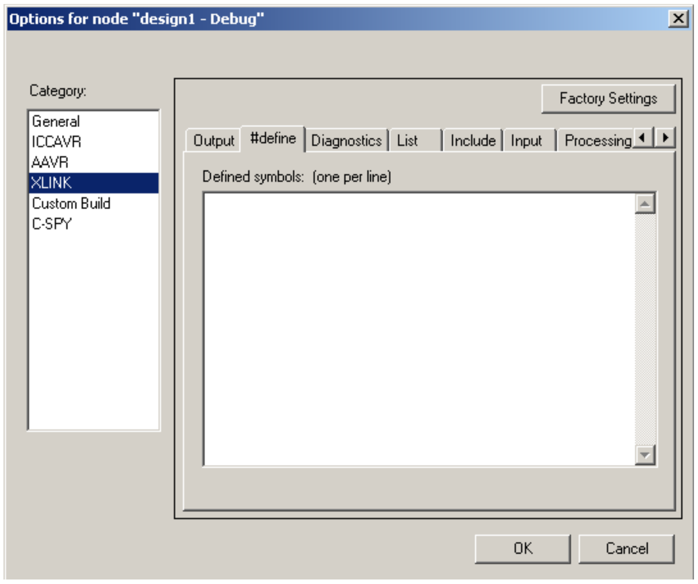
– *Format Variant:* (Enabled. Select.) Select **None** from the pull-down list box. If applicable, use Format variant to select variants available for some of the output formats.

– *Module-local symbols:* (Enabled. Select.) Select ***Include All*** from the drop-down list box. Use this option to specify if local (non-public) symbols should be included in the input module by the IAR XLINK linker. If suppressed, local symbols will not appear in the listing cross-reference and they will not be passed onto the output file. The user can choose to ignore just the compiler-generated local symbols, such as jump or constant labels. Usually these are only of interest when debugging at assembler level.

  Note: Local symbols are only included in files if they were compiled or assembled when the appropriate option is specified.

#define Tab Page

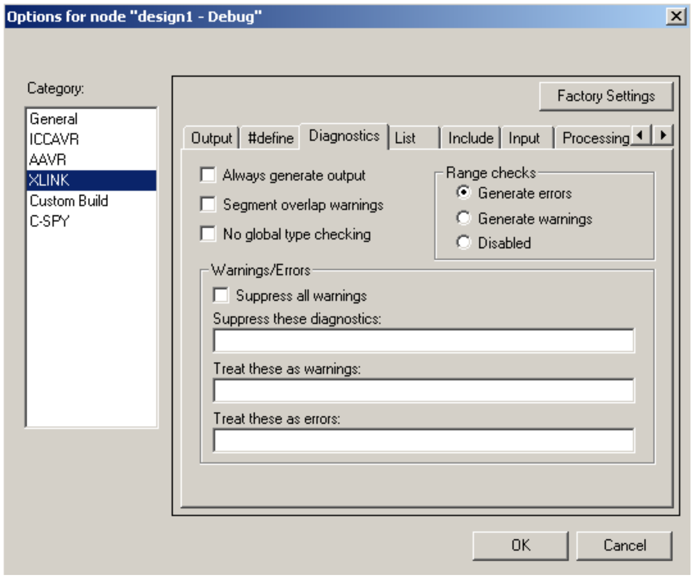
The #define option lets the user define symbols.



- *Define symbol (one per line):* (Enabled. Do. not select.) Use this option to define absolute symbols at link time. This is especially useful for configuration purposes. Any number of symbols can be defined in a linker command file. The symbol(s) defined in this manner will be located in a special module called ?ABS_ENTRY_MOD, which is generated by the linker.

Diagnostics Tab Page

This option lets the user determine the error and warning messages that will be generated by the IAR XLINK linker.



Always generate output [ ]
Segment overlap warnings [ ]
No global type checking [ ]
Range checks: [✓ Generate errors] Generate warnings Disabled
Warnings/Errors
Suppress all warnings [ ]
Suppress these diagnostics:
Treat these as warnings:
Treat these as errors:

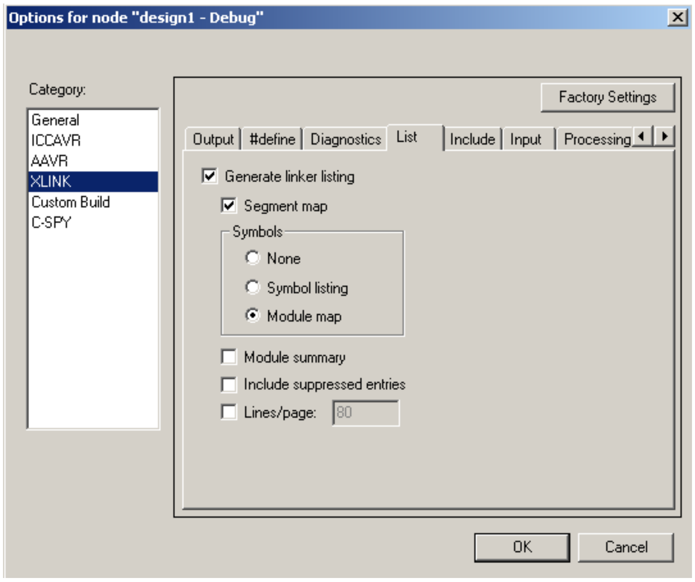
- **Always generate output:** (Enabled. Select.) Use this option to generate an output file even if a non-fatal error was encountered during the linking process. For example, a missing global entry or a duplicate declaration.
- *Segment overlap warnings:* (Enabled. Do not select.) Use this option to reduce the segment overlap errors to warnings, making it possible to produce cross-reference maps.
- *No global type checking:* (Enabled. Do not select.) Use this option to disable type checking at the link time.
- *Range checks:* (Enabled) Use this option to specify the address range check.
  - ***Generate errors:*** (Enabled. Default.) An error message is generated
  - *Generate warnings:* (Enabled. Do not select.) Range errors are treated as warnings
  - *Disabled:* (Enabled. Do not select.) Disables the address range checking

  Note: If an address is relocated outside of the address range of target CPU (code, external data, or internal data address), an error message is generated.
- *Warnings/Errors:* By default, the IAR XLINK Linker generates a warning when it detects that something may be wrong, although the generated code might still be correct. The Warnings/Errors options allow the user to suppress or enable all warnings, and to change the severity classification of errors and warnings.
  - *Suppress all warnings:* (Enabled. Do not select.) This option suppresses all warnings.
  - *Suppress these diagnostics:* (Enabled. Do not select.) This option suppresses the output of diagnostics for the tags that are specified. For example, the warnings w117 and w177 can be suppressed by entering `w117,w177` in the text box.
  - *Treat these as warnings:* (Enabled. Do not select.) Use this option to specify errors that should be treated as warnings instead of errors. For example, error 106 can be treated as a warning by entering `e106` in the text box.

- *Treat these as errors:* (Enabled. Do not select.) Use this option to specify warnings that should be treated as errors instead. For example, to make warning 26 be treated as an error, enter `w26` in the text box

List Tab Page

This option determines the generation of an XLINK cross-reference listing.


Options for node "design1 - Debug"

Category: General, ICCAVR, AAVR, XLINK (selected), Custom Build, C-SPY

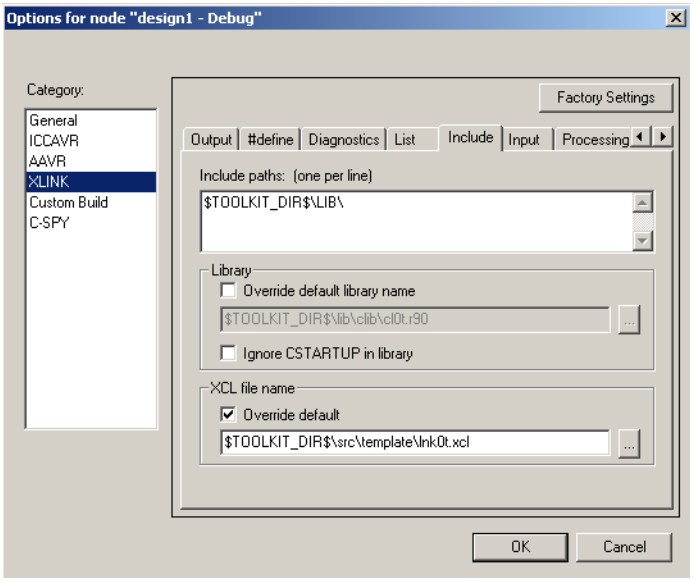
Tabs: Output, #define, Diagnostics, List (selected), Include, Input, Processing

[✓] Generate linker listing
[✓] Segment map
Symbols: ( ) None, ( ) Symbol listing, (●) Module map
[ ] Module summary
[ ] Include suppressed entries
[ ] Lines/page: 80

- **Generate linker listing:** (Enabled. Default.) Causes the linker to generate a listing.
- **Segment map:** (Enabled. Default.) Use this option to include a segment map in XLINK listing file. The segment map will contain a list of all the segments in dump order
- *Symbols:* (Enabled.)
  – None: (Enabled. Do not select.) Symbols will be excluded from the linker listing.
  – Symbol listing: (Enabled. Do not select.) An abbreviated list of every entry (global system) in every module. This entry map is useful for quickly finding the address of a routine of data element.
  – **Module map:** (Enabled. Select.) A list of all segments, local symbols, and entries (public Symbols) for every module in the application.
- *Module summary:* (Enabled. Do not select.) Use this option to generate a summary of the contributions to the total memory use from each module. Only modules with non-zero contributions to memory are listed.
- *Include suppressed entries:* (Enabled. Do not select.) Use this option to include all segment parts in a linked module in the list file, not just the segment parts that were included in the output. This makes it possible to determine exactly which entries were not needed.
- *Lines/page:* (Enabled. Do not select.) Use this option to set the number of lines per page for the XLINK listings, ranging from 10 to 150.

Include Tab Page

This option lets the user specify the linker command file and to set the include path for linker command file.


Options for node "design1 - Debug"

Category: General, ICCAVR, AAVR, XLINK, Custom Build, C-SPY

Tabs: Output, #define, Diagnostics, List, Include, Input, Processing

Include paths: (one per line)
$TOOLKIT_DIR$\LIB\

Library
☐ Override default library name
$TOOLKIT_DIR$\lib\clib\cl0t.r90
☐ Ignore CSTARTUP in library

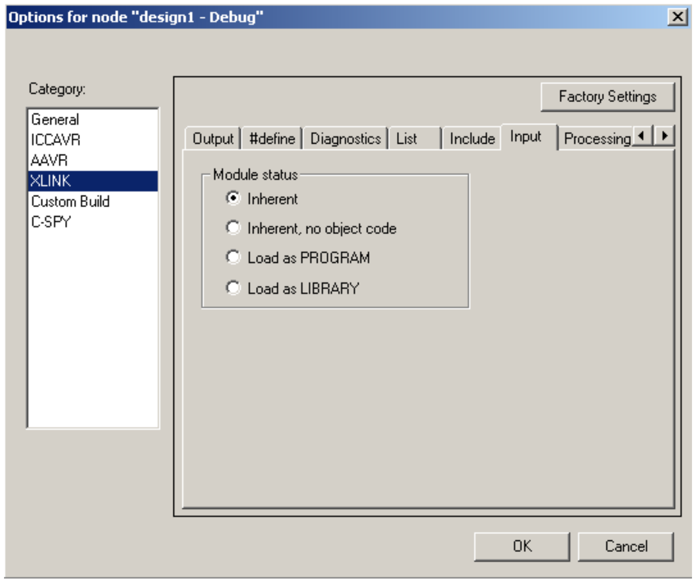
XCL file name
✓ Override default
$TOOLKIT_DIR$\src\template\lnk0t.xcl

- *Include paths:* (Enabled. Default. ***$TOOLKIT_DIR$\LIB***). This option specifies the names of the directories which XLINK will search if it fails to find the object files to be linked in the current working directory.
- *Library:*
  – *Override default library name*: (Enabled. Do not select.) A default library file is selected automatically, based on the user's choice of IAR CLIB (C Library) or IAR DLIB (C/EC++ Library) runtime library. User can override this by selecting Override default library name, and then specifying an alternative library file.
  – *Ignore CSTARTUP in library:* (Enabled. Do not select.) When the user selects this option, all modules in the library will be treated as library modules, even if they have not been assembled or compiled as library modules.
- *XCL file name:*
  – **Override default:** (Enabled. Select). The default linker command file is selected automatically for the chosen Target settings in the General category. The user can override this file by selecting Override default option (***lnkat94k-32s.xcl*** is the default XLINK command file in case of FPSLIC).

  Note: It is advisable to always check this option and provide the .xcl file that is modified by the user. This will allocate the target memory according to the user's requirements (e.g., copy lnkat94k-32s.xcl file into the Workspace directory and modify accordingly, and override the path).

Input Tab Page

Input options allow the user to specify explicitly, how input modules are treated when linked.

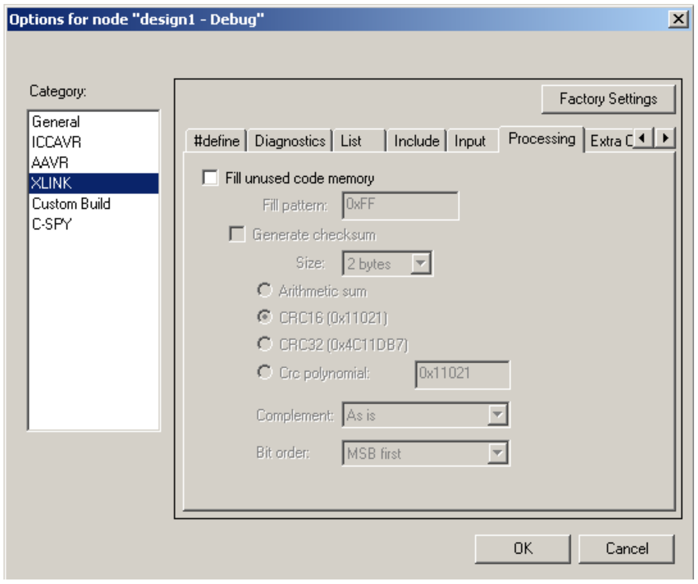


Module status
- (●) Inherent
- ( ) Inherent, no object code
- ( ) Load as PROGRAM
- ( ) Load as LIBRARY

- *Module status:* The default status of the input files, whether they are specified as PROGRAM or LIBRARY, is defined in the files themselves.
  - ***Inherent:*** (Enabled. Default.) When this option is selected, the file will use the status defined in the file itself. Inherent is used to link the files normally, and generate output file.
  - *Inherent, no object code:* (Enabled. Do not select.) Use this option to empty-load specified input files; they will be processed normally in all regards by the linker, but output code will not be generated for these files. One potential use of this option is in creating separate output files for programming multiple EPROMs. This is done by empty-loading all input files except the ones that the user wants to appear in the output files.
  - *Load as PROGRAM:* (Enabled. Do not select.) Use this option to temporarily force all of the modules within the specified input files to be loaded as if they were all Program modules, even if some of the modules have the Library attribute. This option is particularly suited for testing library modules before they are installed in a library file, because the option will override an existing library module with the same entries. In other words, XLINK will load the module from the specified input file rather than from the original library.
  - *Load as LIBRARY:* (Enabled. Do not select.) Use this option to temporarily cause all of the modules within the specified input files to be treated as if they were all Library modules, even if some of the modules have the Program attribute. This means that the modules in the input files will be loaded only if they contain an entry that is referenced by another loaded module. If CSTARTUP was modified, this option is particularly useful when testing CSTARTUP before installing it in the library file, because this option will override the existing program module CSTARTUP.

Processing Tab Page

This option lets the user to specify details about how the code is generated.



- [ ] Fill unused code memory

Fill pattern: 0xFF

- [ ] Generate checksum

Size: 2 bytes

( ) Arithmetic sum
(●) CRC16 (0x11021)
( ) CRC32 (0x4C11DB7)
( ) Crc polynomial: 0x11021

Complement: As is

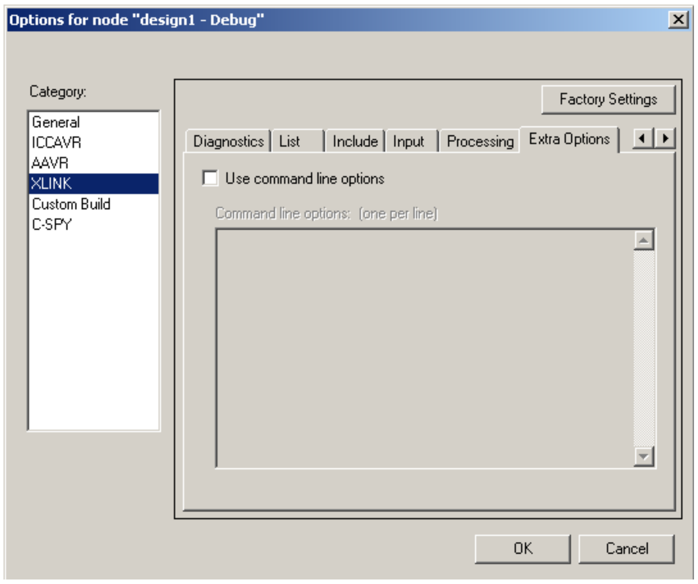
Bit order: MSB first

- Options in this tab are not useful for FPSLIC, hence no descriptions are provided.

Extra Options Tab Page

Additional command line options for the IAR XLINK linker can be specified here.

Note: This option is not supported by the GUI.



- [ ] Use command line options

Command line options: (one per line)

- Options in this tab are not useful for FPSLIC, hence no descriptions are provided.

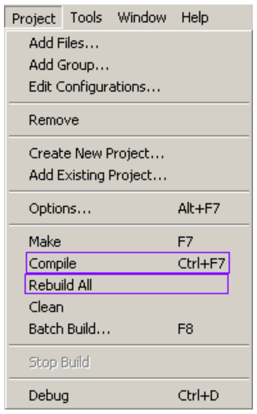
XAR, Custom Build and C-SPY categories are not useful in the case of FPSLIC, hence these options are not described in this application note. Refer to IAR Embedded Workbench user guide for detailed descriptions of these options.

Compiling, Linking and Building the Code

Once all of the options are selected, compiling, linking, and building the project will generate the .d90/.dbg file, which can be used to debug the code using either IAR C-SPY debugger or Atmel AVR Studio debugger.

- To compile, select the C/C++ file and then select *Compile* from *Project* menu option.
- To build, select *Rebuild All* from the *Project* menu option (which will link the compiled object code with library includes and build all together to generate output .dbg file).

The output file will be generated in the Debug\Exe directory that is present in the Workspace directory (C:\FPSLICIAR). The object, list the files can be found in Debug\Obj, Debug\List folders respectively.



Ready to Debug

The output file that is generated can be used to debug the code using industry standard debuggers, in this case either C-SPY or AVR Studio. There are a number of output file formats (e.g., UBROF, Extended-Intel, hp, elf/dwarf etc.) In case of FPSLIC the file format is UBROF8 (force).

Quick Reference

Following is a quick reference for creating and building a new project:

1. File > New > Workspace (file name)
2. Project > Create a new project
3. Tool Chain > AVR
4. Project > Add Files or File > New (Select Source/Text)
5. Project > Options
6. General category:
   - Select **--cpu=at94k**, FPSLIC as target processor with memory model **Small** on the *Target* tab page.
   - Check the **Executable** radio button on the *Output* tab page.
   - Check **Enable bit definitions in I/O-include files** on the L*ibrary Configuration* tab page.
   - Leave the rest of the General category options at their default.
7. ICCAVR category:
   - Set **Size** or **Speed** optimizations as per the requirements (medium in general) on the *Optimizations* tab page.
   - Check **Output list file** and **Assembler mnemonics** on the *List* tab page
   - Leave the rest of the ICCAVR category options at their default.
8. AAVR category:
   - Leave AAVR category at default settings in case of C language-based development. Otherwise, select **Generate list file** on the *List* tab page.
   - Leave the rest of the AAVR category options at their default.
9. XLINK category:
   - Select **Other** and choose **ubrof 8 (forced)** as the *Output format* and select **None** as the *Format variant* in *Output* tab page.
   - Check **Always generate output** on the *Diagnostics* tab page.
   - Check **Override Default** in *XCL file name option* and **include the pat**h **of custom-made XLINK command file** on the *Include* tab page.
   - Leave the rest of the XLINK category options at their default.
10. Select each C file individually, select Project > Compile.
11. Select PROJECT_NAME – Debug in workspace window,
    select Project > Rebuild All.

This procedure produces a PROJECT_NAME.dbg file, which can be used to run co-verification simulations using System Designer, or used as a standard debug file by AVR Studio (Atmel recommended) or C-SPY (IAR) debuggers.

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature