NOHAU CORPORATION

EMUL–ARM™

ARM Compilers

March 6, 2003

Contents

About This Guide

The EMUL–ARM is a PC-based hardware debugger for the ARM™ Core (currently ARM7 and ARM9 cores). Seehau is the name of the user interface of EMUL-ARM – Seehau and EMUL-ARM is often used interchangeably.

This guide helps you to get started with using compilers that work with EMUL-ARM – those are currently:

- ARM
- IAR
- MicroCross GNU

Note that Seehau has a "Project Manager" that allows compilation and linking from within Seehau. Seehau can thus be used as IDE.

Terminology

- **Compiler** – in this document, we use the term "compiler" loosely for compiler, linker and possibly an integrated development environment (IDE).
- **Embedded C++** – a subset of the C++ programming language, which is intended for embedded systems programming. An industry consortium, the Embedded C++ Technical Committee, defines it.
- **IDE** – Integrated Development Environment – a graphical user interface that allows easy access to compiler, linker and often debugger. The IDE typically allows defining "projects", which define compiler switches, files to compile etc. An example of an IDE is Visual C++.

History

Initial release.

1 General Topics

About Optimization

Please note that when compiler optimization is turned on, this will often cause source stepping to seem not to be working. This is because the relationship between source code and assembly code is often drastically changed with optimization. Consider a “for” loop which could be implemented without branches – i.e. the statements are repeated instead of jumping back to them.

For this reason, we suggest turning all optimization off when debugging.

Load File Format

Currently supported load file format is “Elf/Dwarf” on all compilers.

Thumb Mode

EMUL-ARM automatically detects c functions that contain 16-bit Thumb code. Note that this means that there is no auto detection for Thumb mode in code already on target board and load file formats that do not give information – for instance Motorola S-Record and Intel Hex. For these cases Thumb regions has to be configured manually – select “Config|Emulutor”, and go to the “Map Config” tab, then “Add” memory ranges that are in Thumb Mode.

Additionally, Thumb mode is not automatically detected in assembly code for all compilers. See each compiler below.

2 Compilers

The ARM Compiler

ARM has two compilers – the older "Software Development Toolkit" (SDT) and the newer "ARM Developer Suite" (ADS). EMUL-ARM works with both.

ARM – Supported Versions

We currently support:

- SDT 2.51 – this is an older version of the ARM compiler.
- ADS 1.01, 1.1 and 1.2.

However, Seehau will probably work with SDT 2.5x and ADS 1.x in general.

ARM – STD 2.51 Settings

Modify project settings by selecting Project | Tool settings for *project*xxxx.

Compiler settings – pick the compiler use and then set:

- Language and Debug tab – Enable Debug table generation = checked.
- Language and Debug t tab – Optimization level = Minimum or Most.

Linker settings – Select Armlink and then select set:

- General tab – Include Debug information = checked.
- Output tab – Initially select ARM ELF image format.
- Entry and Base tab – for initial attempts you can set program location by entering the address in Read-Only text box under Base of Image. (Note this may not be where the program actually starts.)

ARM – ADS 1.x Settings

Modify project settings by clicking the "Settings Button" in the project (*.mcp) window.

Compiler settings – when settings is open, expand the "Language Settings" and for each of the different compilers used set:

- Debug/Opt tab – Enable Debug table generation = checked.
- Debug/Opt tab – Optimization level = Minimum or Most.

Linker settings – when settings is open, expand the "Linker", select ARM Linker and set:

- Options tab – Include Debug information = checked.
- Output tab – for initial attempts you can set program location by selecting Link-type=Simple, and RO Base to desired address. (Note this may not be where the program actually starts.)

ARM SDT – Thumb Mode

Thumb mode in c/c++ functions is recognized automatically. Thumb mode in assembly must be configured manually.

ARM ADS – Thumb Mode

Thumb mode is fully detected in both c/c++ and assembly.

ARM – Notes

There are two different compiler executables for ARM and Thumb – so each c-file will be either ARM or Thumb.

The IAR Compiler

The IAR Embedded Workbench (EW):

- Supports C/Embedded C++.
- Runs under Windows 98/ME/NT4/2000/XP.

IAR – Supported Versions

- Version 3.21a.

However, Seehau will probably work with other 3.x versions as well. An earlier version required an updated XLINK that was available for download at http://www.iar.com.

IAR – Settings

Compiler settings – to set optimization, select menu option “Project | Options”, category is ICCARM, and the “Code” tab and set:

- Optimizations = Size, None.

Linker settings – select menu option “Project | Options”, category is XLINK, click the “Output” tab and set:

- Format = Other.
- Format variant = ARM compatible.

IAR – Thumb Mode

Thumb mode in c/c++ functions is recognized automatically. Thumb mode in assembly must be configured manually.

IAR – Notes

IAR EW comes with complete source code for a number of popular target boards. Look in the “Src” directory where the IAR EW is installed, and consult the “readme.txt” file. Currently supported boards include Atmel AT91EB40, AT91EB40A, AT91EB42, AT91EB55, AT91EB63 and OKI JOB671000.

The IAR linker will remove (optimize away) unused functions and variables. One way to avoid this is to use the proprietary keyword __root. One effect is that if there is no startup code that calls the main() function, there will be no code generated. However, the IAR C-library will call main by default.

The MicroCross GNU Compiler

The GNU compiler comes in many shapes and forms – there are commercial distributions from companies like MicroCross and Red Hat, or you can "make" your own by download from internet. Additionally, KEIL – a traditional compiler vendor, now uses GNU as a compiler, and they have built an EDI around it (still in Beta).

We currently claim to support the MicroCross GNU distribution, only. However EMUL-ARM ***should*** work with other distributions as well. For information on the GNU compiler in general, please visit: http://www.billgatliff.com.

MicroCross GNU – Supported Versions

- GNU X-Tools version 2.0 – our copy reports to be "GNU C 2.95.2".

MicroCross just released a new product a new product called "Visual X-Tools". However, it should initially be base on the same compiler, and is therefore supported. Visual X-Tools adds a new IDE to the GNU compiler and is based on the SlickEdit code editor.

MicroCross GNU – Settings

Compiler settings – on the command line, use following switches:

- -g – generate debug information.

Compiler settings – on the command line, do **not** use following switches:

- -mapcs_xxx – where "xxx" means anything (support the default calling convention only).
- -mtpcs_xxx – where "xxx" means anything (support the default calling convention only).

Linker settings – there are no required command line switches.

MicroCross GNU – Thumb Mode

Thumb mode in c/c++ functions is recognized automatically. Thumb mode in assembly must be configured manually.

MicroCross GNU – Notes

There are two different compiler executables for ARM and Thumb – so each c-file will be either ARM or Thumb.

MicroCross GNU – Source Step Into Libraries

The MicroCross GNU std c libraries are built with debug information. This means when doing a source step into for instance “sprintf()”, Seehau will ask for the location of the source code. Depending on debugging style, this can be irritating. There are two solutions for this:

1. First, make sure you have the source code. Then use menu “Config | Environment” and the “Paths” tab on the popup to add the path to the source code.

2. Use the “strip.exe” utility that comes with the MicroCross tools to remove symbols from the c library: “**strip –g libfile.a**”. The ‘-g’ switch cause only debugging symbols to be stripped. Otherwise all symbols (including globals and entry points) are stripped and the library becomes unusable.