Learn the skills you need to be a professional embedded developer with this hands-on training course

The embedded world is changing: standardization on **ARM**, the dramatic increase in development tools based on **Eclipse** and **GNU**, the emergence of the **Internet of Things**...

Enroll in one of the Atollic professional training courses and get a fast track to the knowledge and expertise you need to succeed in ARM Cortex-M development!

Learn everything worth knowing about the Cortex-M cores and the GNU compiler and debugger tool chain. You will also learn the ins- and outs- of the ECLIPSE IDE, and how to use it for advanced Cortex-M development and debugging.

Book this Training

Atollic AB
Science Park Jönköping, Sweden
www.atollic.com

atollic

Professional Development Series

refresh your skills

Cortex-M and Eclipse/GNU Training Course

A 3-day, hands-on course packed with the information you need to succeed in ARM Cortex-M development.

The training you need for successful ARM® development.

ARM Cortex-M and GNU tools are strong forces in the embedded space, consolidating the industry and providing advanced new capabilities. Thousands of developers are migrating to Cortex-M and/or the GNU toolchain, not knowing what they need to learn beforehand to ensure project success.

ARM Cortex-M has been adopted by almost all of the leading microcontroller manufacturers. Embedded development environments using Eclipse with the GCC compiler and GDB debugger are everywhere and present a real threat to the established development tool companies because of their versatility and performance.

This training course is designed to give you a detailed understanding of the Cortex-M device architecture and the ARM-backed CMSIS standards along with detailed training on how to use Eclipse/GNU-based tools to develop with ARM Cortex devices.

Who is this course for?

This course is for embedded developers planning to use Cortex-M and GNU ECLIPSE tools and who are migrating from another architecture or who want to take the next step and become experts in their field.

Pre-requisites

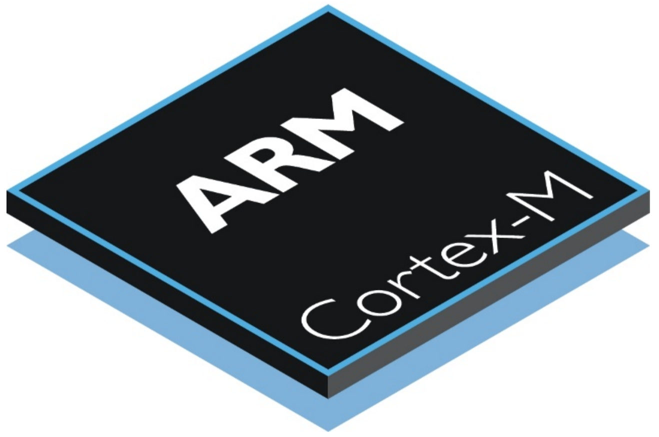
You do not need any previous experience in ARM cores, the GNU tools or the ECLIPSE IDE. But we do assume you know how to program in C and have some embedded systems development experience.

This is a workshop course with hands-on training

Each participant will need to bring a laptop. Necessary development software, documentation and target hardware is included in the course material. The training course is developed and delivered by experienced engineers.

Hands-on training. Learn and then try it for yourself!

We have designed this course to give you hands-on experience with the subject matter. This is how engineers learn. Each course module combines lecture and try-it-yourself exercises. We want you to come away from this course with the knowledge and experience to successfully develop a project with any ARM Cortex-M device.



Book this Training

DAY 1

MORNING

Understand the Cortex-M cores and CoreSight hardware debug architecture
- Cortex-M0/Cortex-M0+
- Cortex-M3/Cortex-M4/Cortex-M7
- Cortex-M23/Cortex-M33

Associated software standards
- CMSIS-HAL
- CMSIS-DSP
- CMSIS-RTOS
- CMSIS-PACK

AFTERNOON

The ECLIPSE IDE
- Perspectives and views
- Projects and workspaces
- Advanced editing and code navigation
- Basic building and debugging
- Build & debug configurations
- Extensibility and 3rd party plugins

Dual-target: PC & embedded tools in one IDE

Several practical exercises throughout the day:
- Demo of CMSIS-HAL, CMSIS-DSP, CMSIS-RTOS and CMSIS-PACK
- Eclipse/CDT demos and exercises
    - Overview ECLIPSE <-> CDT <-> GNU <-> gdb-server (for debug)
    - Projects and workspaces
    - Perspectives and views
    - Editing, code navigation, code structure visualization, refactoring, code formatting, coding style, ... (editing and code views)
    - Build settings and basic building (build views)
    - Debug settings and basic debugging (debugger views)

DAY 2

MORNING

The GNU tools
- The big picture
- ARM Cortex-M startup code
- Writing interrupt handlers
- Linker and linker configuration files
- GCC compiler options
- GCC #pragmas
- GCC inline assembly
- Binary utilities (gas, objdump, ...)
- Newlib and Newlib nano runtime libraries
- Syscalls.c and printf() redirection
- GDB, gdb-servers and the JTAG probe
- GDB scripting

AFTERNOON

Cortex-M advanced debugging
- Instruction tracing (ETM/ETB/MTB)
- Real-time event- and data tracing (SWV/SWO/SWD)
- Live variable watch
- Hard fault crash analysis

Several practical exercises throughout the day
- Create a simple example with power-on-reset startup code, interrupt vector table, linker file and empty main().
- Write an interrupt handler
- Use #pragma and inline assembler (use objdump to inspect binary code)
- Use newlib nano and redirect printf()
- Write a gdb script
- Use SWV, live variable watch and crash analyzer

DAY 3

MORNING

Using an RTOS
- Views and structures
- Development including use of FreeRTOS
- Debugging FreeRTOS using Kernel-awareness
- Considerations when using an RTOS in a larger development team

AFTERNOON

System engineering and team collaboration
- Version control system integration
- Practical considerations
- Bug database integration
- Continuous integration/Nightly build

Several practical exercises throughout the day
- Add FreeRTOS to a simple demo project
- Use tasks, semaphores, message boxes
- Task aware debugging
- Connect to Git or Subversion in Bitnami stack
- Use version control features
- Connect to Trac or Bugzilla in Bitnami stack
- Use bug database features
- Use Hudson in Bitnami stack
- Use headless build/nightly build features

Organized by LOGIC TECHNOLOGY