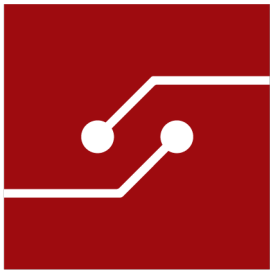

eclipseina

List of Embedded Trainings (Online and In-house) by Eclipseina GmbH

Status June 2023

Content

Training Dates and Registration

All documents concerning training content and training dates are provided on our website:

https://eclipseina.com/registration

Training dates

All current dates can be found on our website: https://eclipseina.com

Registration

Please either use the internet template or send us an e-mail to: training@eclipseina.com

Please include the following data:

- Training title:
- Training date:
- Name of participant:
- Participant's contact details:
- Company:
- Billing address:

For trainings with certificate examination, please provide the following information as well:

- Exam participation desired
- ASQF membership and membership number

Software Architecture

Software Architecture for Embedded Systems

Description

Participants will obtain knowledge about the tasks carried out by software architects as well as the tools and methods they use. Using architecture principles as a basis, participants will learn about the key architecture methods for embedded systems and how to apply these in projects. After completing the seminar, participants will be able to develop and document structured software architectures. Participants will apply and expand their knowledge with the help of a universal and practical example.

Target Group

Software architects, software developers, software project managers and system architects

Prerequisites

Knowledge of embedded software development Experience in working with a modelling language would be an advantage (e.g. UML or ROOM)

Training Content

- Basic Principles of Software Architecture (1st day)
- What is software architecture?
- The role of architects and the interfaces they use
- Software architecture requirements
- UML for documenting software architectures
- Designing software architectures
- Architecture patterns for embedded systems
- The tools used by software architects
- Advanced Principles of Software Architecture (2nd day)
- Architecture patterns for embedded software
- Communication and implementation models
- Description of structures and properties
- Component-based development
- Abstraction and automation with model-based software development
- Demonstration of tools for model-based software development

Duration

3 days

Price

1.790 € plus VAT per participant

---

freeRTOS in Theory and Practice

Description

This seminar provides practical knowledge and understanding of real-time kernel usage, and answers questions about potential advantages and trade-offs. By learning from an experienced trainer, participants can take home a working knowledge of freeRTOS and the ability to use it effectively in their own embedded development project.

This three-day training class uses hands-on exercises combined with well-chosen instructions to illustrate the concepts of a real-time kernel. Examples using freeRTOS form a series of practical coding exercises designed to bring you quickly up to speed. The concepts and commands necessary to make effective use of freeRTOS are described through a combination of theory and hands-on-training.

Target Group

Software engineers, field engineers, (project) managers

Prerequisites

- Familiarity with embedded C concepts and programming
- Ability to develop software using C syntax
- Ability to use basic embedded compiler and debug tools
- Basic knowledge of embedded processor development boards
- Training Content

Training Content

Introduction

- freeRTOS overview, market position, the „free“ in freeRTOS, understanding the freeRTOS license, software architecture, features
- LPCXpresso IDE download and installation, importing an example workspace

Task Management

- Tasks - Creation, states, priorities, the idle task, deletion

- LPCXpresso Scheduling - Determinism, multitasking, endless loop, cyclic executives, issues with interrupts, non-preemptive, prioritized preemptive, rate monotonic, deadline, cooperative, hybrid

Queue Management

- Creation, sending, receiving

Interrupt Management

- Deferred interrupt processing, interrupt handlers, interrupt safe functions, task with interrupt synchronization, efficient queue usage even from within an interrupt, interrupt nesting

Resource Management

- Mutual exclusion, critical sections, suspending/locking the scheduler, mutexes, priority inversion, priority inheritance, deadlock, gatekeeper tasks

Memory Management

- Resource constrained memory allocation schemes, determining the amount of free memory remaining

Trouble Shooting

- avoiding bugs and how to find those you did not avoid

freeRTOS-MPU

- User vs. privileged mode, access permissions, defining MPU regions, linker configuration, practical usage tips

The freeRTOS Download

- Files and directories, demo apps, data types and coding style

Method and Training Materials

Presentation and practical examples with host (Laptops with Ubuntu 14.04.x LTS) and target system (e.g. Beagle Bone Black Rev. C – http://beagleboard.org/BLACK). These electronic devices will be provided during the training. There will be one workspace for two participants. A customized version of this training, freeRTOS on LPCXPresso 1769, is offered by freeRTOS.

Duration

3 days

Price

1.790 € plus VAT per participant

Embedded Security

Embedded & IoT Security Training – Full Lifecycle

Description

You will learn what it takes to secure connected embedded devices. Starting with the big picture, you will be introduced to security best practices as well as technical challenges like managing secrets or the integrity on an embedded platform on a productive scale.

Concepts such as security by design as well as security aspects for all phases of the product's lifecycle (design, development, production, maintenance, decommissioning) are explained using lot's of examples and best practices.

You will learn the essentials about commonly used cryptography – Why do you need which crypto primitives to reach your security goals.

We will focus specifically on aspects of embedded devices and derive answers to many questions such as:

- Where should I start with security in my project?
- What are current regulations demanding in respect of cybersecurity?
- What standards can be taken as reference?
- How to protect keys, IP, or firmware on an embedded device?
- What are attackers capable of?
- Why should anyone hack my YouNameIt®?
- How is key provisioning and onboarding done in practice?
- How to deal with post quantum cryptography issues?
- What are good ways to keep embedded systems updated?

Practice is very important for us and we are happy to answer questions during the training. Some aspects are discussed and explained using an embedded Linux device sending data to an IoT backend.

If requested, an additional workshop or training day with the focus on certain branches like Automotive, Agriculture, Industrial Automation, Energy (Solar Inverters, Batteries, Energy Management Systems) can be planned.

Target Group

product managers, embedded developers, system architects, connectivity architects, IoT system administrators

Prerequisites

No programming skills are needed, but a background in embedded systems or security topics is helpful.

No special tools are needed, but a notebook with network card is recommended to be used to interact with the practical demonstrators.

We recommend to take the e-learning courses on information security https://embedded-academy.com/en/courses/information-security-en/ and cryptography https://embedded-academy.com/en/courses/cryptography/ and for employees in Automotive we also recommend Automotive cybersecurity https://embedded-academy.com/en/courses/automotive-cybersecurity-en/ before the training.

Training Content

How to secure any Thing?

- Security Introduction, Assets, Security Goals
- Hacker: Motivation, Classification, Tools, Real Life Examples
- Security Engineering – Security in the Product Lifecycle
- Introduction to Risk Assessment
- Security Best practices

Standards and Regulations

- Ongoing regulatory activities in the EU (e.g. RED, CRA, NIS2)
- Overview of available standards and regulations with a focus on embedded systems and IoT
- Practical deep dive into EN 3030645 Cybersecurity for Consumer IoT
- Deriving and explaining security requirements

Cryptographic Toolbox

- Module about WHY we need crypto, not how it works
- What security goals can be achieved with which cryptographic functions
- Security Hash Functions
- Symmetric cryptography
- Asymmetric cryptography
- Attack Options
- Certificates and PKIs

Trust and Crypto in Embedded Hardware

- Crypto Accelerators
- Trusted execution environment
- Secure key storage (SE, HSM, TPM, …)
- Secure Boot concept

Safe and Secure Software Update Concepts for Embedded devices

- Local, network and OTA (over the air)
- Update System Requirements
- Update types for embedded Linux devices
- Software Signing and validation
- Open Source Update System examples (RAUC, SWupdate, hawkBit)

Key and Device Provisioning and Onboarding

- Key Provisioning Challenges
- Provisioning Options (JITR, JITP, Batch Provisioning)
- Zero Touch – Automated device registration and onboarding

Duration

2 days

Price

1.290 € plus VAT per participant

C and C++ Programming

C++11 and C++14

Description

In the seminar “C ++ 11 and C ++ 14” you will get to know the innovations and differences that programming in C ++ 11 and C ++ 14 brings with it. The changes in the core language are presented, multithreading is discussed and the standard library with all its options is discussed.

All these changes lead to more security and make daily programming easier. The prerequisite for this is of course that you have familiarized yourself with the innovations. The seminar participants are supported by an experienced trainer who is passionate about the subject. A large number of examples illustrate the innovations and the numerous exercises in the seminar ensure that you can later use it independently in your own projects.

After the seminar, participants will not only be able to recognize the changes and differences between C ++ and modern C ++, but also to use C ++ 11 and C ++ 14 efficiently and confidently in their everyday work.

Target Group

Experienced software developers

Prerequisites

You need a laptop with a C++ compiler (at least C++11)

Training Content

Core language

- Improved usability
- Design of classes
- Rvalue references and move semantic
- Generic programming

- Extended data concepts and user-defined literals

Multithreading

- The C++11 memory model
- Atomic data types
- Threads
- Sharing of data
- Thread-local data
- Condition variables
- Tasks

Standard library

- Regular expressions
- Type-Traits
- Random numbers
- Time library
- Reference wrapper
- The new containers
- New algorithm
- bind and function

Methods and Materials

Theory and practice with many exercises to help you get the source code and sample solutions.

Duration

3 days

Price

1.790 € plus VAT per participant

C++ for embedded applications – a deep dive

This 3-day training introduces the participants to C++ programming techniques that have proven themselves especially in the embedded area, but can basically be used anywhere. The goal is the consolidation of C++ knowledge, in particular to features of the language which are relevant for embedded systems and to convey a feeling for the runtime costs and the memory

overhead of various features of C++, especially in the area of object orientation. Code examples according to the C++03/C++98 standard are used throughout the course since many compilers for embedded systems only support C++03 (e.g. IAR, Hitex, ARM to v5). Adaptations to more modern language features are explained where appropriate. Also details about RTOS or multi-core programming are (primarily) not part of this course.

Target Group

Embedded software developers

Prerequisites

Basic C++ knowledge is a prerequisite. A laptop with an installation of VirtualBox is needed for exercises.

Training Content

Day 1

- Generic programming using C++ Templates
- Traits and Policies
- Function pointers
- Delegates
- Smart Pointer

Day 2

- STL Container and Alternatives
- Object Alive Check
- Dynamic Allocation of Memory
- Memory leaks and -manager

Day 3

- Multithreading
- Thread
- Synchronization
- Factory Design Pattern
- Singleton Design Pattern

Methods and Materials

Code examples according to the C++03/C++98 standard

Duration

3 days

Price

1.790 € plus VAT per participant

Embedded Linux

Embedded Linux Debugging/Tracing/Profiling

Participants will be given a basic overview of debugging/tracing/profiling facilities with (embedded) GNU/Linux. Hands-on exercises provide participants with the necessary practical experience to choose the right tool for their debugging needs.

Target Group

Software architects, software developers, software project managers, system architects

Prerequisites

Theoretical and practical knowledge in how to use Embedded Linux is assumed.

Training Content

1st Day: Debugging

- Debugging with easy methods: e.g. errno, shell debugging, lsof, netstat, procfs, sysfs, debugfs, syslog, ltrace…
- Gdb and friends: e.g. gdb, gdbserver, gdb and threads, core dump, zero pointer, log segmentation faults, crash …
- TOP and friends: top, latencytop, powertop, powerdebug, iotop, atop, htop
- Boot-up time optimization: grabserial, bootgraph, bootchart

2nd Day: Profiling, tracing and relevant tools

- Profiling and Tracing: e.g. time, gcov, gprof, oprofile, systemtap, perf, ftrace, trace printk, kernelshark…
- Tools for debugging, profiling and tracing – the Yocto-Project: e.g. Eclipse plugin, tcf-agent, User space Debugging, perf, ftrace…

Methods and Training Materials

Lecture and practical examples with host (laptops with Ubuntu 14.04.x LTS) and target system (e.g. Beagle Bone Black Rev.C – http://beagleboard.org/BLACK). The devices will be provided during the training. You receive a reference- and workbook, one Beagle Bone Black Rev.C plus one standard FTDI 3.3 V to USB cable. After the training you will get access to a download link with a prepared Docker image and examples.

Duration

2 days

Price

1.290 € plus VAT per participant

Embedded Linux in Theory and Practice

Linux is an extremely powerful tool to work with. The aim of this training is to make participants familiar with the basic concepts of embedded Linux as well as provide them with information on how to work with Linux. The advantages and disadvantages of Linux are discussed, as well as the necessary components for building an embedded GNU/Linux system. At the end of the seminar, participants will know where to obtain relevant components and how to configure, translate and install those. Further, explanations on the different license models used within the Linux environment are provided. By looking at various hands-on examples, participants will learn how to build an embedded GNU/Linux system from mainline components. Additionally, possible ways to achieve support in case of any uncertainties will be one of the topics.

Target Group

Project manager, software, hardware and system engineers

Prerequisites

Basic knowledge of how to use Linux (Ubuntu), familiarity with embedded C concepts and programming, the ability to develop software using C-Syntax, also basic knowledge of embedded hardware (Eval Boards) is of advantage.

Training Content

1st day: basics

- Introduction into GNU/ Linux: History, licenses, standards,working with open source, spelunking, Unix, philosophy
- Characteristics of Embedded Linux:
    - Embedded Systems
    - Comparison of Embedded Linux vs. Desktop Linux
    - Identifying and solving dysfunctionalities
    - Portability
    - Building for the target: tool-chains, C-Libraries
- Eval Board – Beagle Bone Black:
    - Booten (generic) or rather the Beagle Bone Black
    - Partitioning or rather formatting of SD cards
    - Partition with boot-loader, kernel, filling of rootfs
    - Configuring serial consoles
    - Booting a board with Linux/GNU

2nd day: installing and configuring the Host and the Target

- Installing of the toolkit, the NFS server, tftp server
- U boot: check out, configuration, cross compiling, installation
- Flattened device tree
- GNU/Linux kernel: uImage, check out, configuration, cross compiling, installation, kernel modules
- Root File System
- Adjustments: Adding network support to the board (U boot scripting, network support in U boot, customized kernel with network support)
- Rootfs via NFS
- Init (Sys-V, Upstart, Initng, Systemd), Bootgraph, Bootchart

3rd day: kernel modules, an overview of drivers and debugging

- Kernel module: Hello Kernel, module-init-tools, Kconfig, Kbuild, out of tree, in tree
- Device driver: Device nodes, writing a character driver, registration, initialization, Miscellaneous Character driver
- An overview of debugging / profiling / tracing:
    - Simple debugging tools: lsof, ltrace, strace, proc, top, netstat, syslog
    - Further debugging tools: gdb and target gdb, gdbserver, kgdb/kdb and agent-proxy, JTAG
    - Profiling: time, gprof, gcov, oprofile
    - Tracing: race, kernelshark, LTTng
    - Multiple other tools: top, latencytop, powertop, powerdebug, crash, systemtap

Methods and Training Materials

Lecture and practical examples with host (laptops with Ubuntu 14.04.x LTS) and target system (e.g. Beagle Bone Black Rev.C – http://beagleboard.org/BLACK). The devices will be provided during the training. You receive a reference- and workbook, one Beagle Bone Black Rev.C plus one standard FTDI 3.3 V to USB cable.

Duration

3 days

Price

1.790 € plus VAT per participant

Embedded Linux - Theory/Practice and Debugging/Tracing/Profiling

This training is a combination of Embedded Linux in Theory and Practice and Embedded Linux Debugging/Tracing/Profiling.

To get more information please click on the links to the individual trainings.

Duration

5 days

Price

2.590 € plus VAT per participant

The Yocto Project - A Summary

The Yocto-Project (YP) represents an Open Source project which provides templates, tools and modes to generate embedded products independent of Linux based systems and their hardware architecture. Even experienced GNU/Linux-users might be unfamiliar with how to accommodate their workflow to YP and whether they need it at all. This seminar will give answers and show the most important elements which define the Yocto Project.

The seminar's aim is to impart the essentials necessary for using YP, based on existing knowledge in GNU and Linux.

Target Group

The workshop is aimed at software engineers, development engineers, system engineers, testers, administrators, engineers and other parties interested in the YP, with a solid knowledge of Embedded GNU/Linux.

Prerequisites

- Basic knowledge of using a GNU/Linux system (e.g. Ubuntu) as an end user in user space
- Basic knowledge of a command line shell
- Basic knowledge of user/kernel space programming with GNU/Linux
- Intermediate C programming knowledge

Training Content

Day 1: Basics

- Introduction into the History of Unix/Linux, Standards
- Embedded Specifics: Embedded Linux vs. Desktop Linux, Cross-/Native tool chains, Build-systems, C-libraries
- Eval Board
- Introduction into the Yocto Project
- The Yocto auto builder

Day 2: The YP-Work flow and Bitbake

- Configurations and sources
- Building processes: source fetching, patching, configure, compile, install, Pseudo, examples of recipes, Output Analysis/Packaging, image and SDK generation
- Customizing images: Intro, local.conf, IMAGEFEATURES, custom .bb files and package groups
- Bitbake's history and syntax
- Bitbake Debugging:
    - Debug Level
    - Finding recipes/images/package groups
    - BitBake environment/tasks/logging
    - Build/force specific task cleansstate, stamp invalid explanation, Devshell
    - Dependence-Explorer
    - BitBake with graphic wrapper

Day 3: Layers, Kernel and Application Development Toolkit (ADT)

- Layers Intro, Bitbake-layers tool, Yocto-layer tool

- Board Support Package (BSP) Intro, system development workflow, developer's handbook (bsp-tool)
- Kernel Intro, workflow system development, Kernel's developer handbook (defconfig, defconfig + configuration fragment, in tree kmod, out of tree kmod, fdt)
- Application Development Toolkit (ADT) Intro, Cross-Development tool chain, Sysroot, ADT-Eclipse
- Yocto plug-in, the QEMU emulator, user space tools
- Install ADT and tool chains (to use ADT-Installer Cross-tool chain tarball)

Day 4: Debugging, Tracing and Profiling

- Debugging: gdb, gdb Remote-Debugging, (gdb Remote) Debugging with Eclipse, (remote) perform with Eclipse
- Tracing and Profiling: perf, gprof, gcov, strace, race, systemtap, oprofile, LTTng + Eclipse (data visualization)
- Package management: working with packages, IPK, creating a package-feed, Installation of software suit with opkg on hardware
- Licensing, adding a customized license, Open-Source-License-Compliance
- Devtool, demonstrating the creation of meta-layers for a real project (meta-cfengine)

Methods and Training Materials

Lecture and practical examples with host (laptops with Ubuntu 14.04.x LTS) and target system (e.g. Beagle Bone Black Rev.C – http://beagleboard.org/BLACK). The devices will be provided during the training. You receive a reference- and workbook, one Beagle Bone Black Rev.C plus one standard FTDI 3.3 V to USB cable.

After the training you will get access to a download link with a prepared Docker image and examples.

Duration

4 days

Price

2.290 € plus VAT per participant

Embedded Linux Refresher and Introduction to the Yocto Project

This five-day training combines theory with hands-on exercises in order to introduce Embedded Linux and the Yocto Project.

After refreshing all the necessary topics to Embedded Linux (2 days) you will be provided with an understanding of the essentials to utilize the Yocto Project (3 days). After intruding you to Yocto in general we'll see how a BSP/framework maintainer would use the Yocto Project as well as developers who might not even want/need to know they are using it.

It answers frequently asked questions like:

- What is GNU/Linux?
- Why use upstream?
- Where to get u-boot/the kernel from? How to configure/build/install it?
- How does interprocess communication work and what to use/avoid?
- Is it really necessary to use another version of the tool-chain/libraries/packages for each and every GNU/Linux project and on top of that to follow a different work-flow each time?
- Can you ensure that the development environment is identical for all developers/suppliers and that you can still produce identical builds like today in 10+ years from now?
- Can the YP help you with Open Source license audits or do you prefer a copyright troll instead?
- ... and much more.

Target Group

You already use GNU/Linux for your projects and have probably heard about the Yocto Project, but did not dare to have a closer look into it, or had difficulties using it. You don't know whether and how your daily workflow can be accomodated in the YP and generally find the YP rather complicated. Why do we need all this since up to know everything was (supposedly) much easier? After the training you should be able to decide whether you need the YP or not. The workshop is aimed at software-, development-, system engineers, testers, administrators, engineers and other parties interested in the YP, with a minimal knowledge of Embedded GNU/Linux.

Prerequisites

- Basic familiarity with using a GNU/Linux system (e.g. Ubuntu) as an end user in user space
- Basic familiarity with a command line shell
- Basic knowledge of user/kernel space programming with GNU/Linux
- Intermediate C programming knowledge

- It's sufficient to know how to build the GNU/Linux kernel, kernel drivers in/out of tree and the fdt from the kernel side of things to follow the Yocto training which we'll try to quickly cover in the first two days.

Training Content

- Introduction
  - Introduction | History
- Eval Board
  - Eval Board Introduction
  - Booting
    - How does Linux boot on a PC
    - How is an Embedded System different?
  - Booting the target
    - Boot Sequence
    - SD card partitions
- Stuff needed
  - U-Boot
  - U-Boot: Fancy Stuff
  - U-Boot: Get/Configure/Build/Install
  - U-Boot: Commands
  - Fdt
  - Kernel
  - Kernel: Get/kbuild
  - Kbuild
  - Kernel: Configure/Build/Install
  - Kernel: fdt
  - Kernel: modules
- Kernel Modules
  - ...can be
  - init/exit
  - Licensing
  - tainted module/kernel
  - EXPORT_SYMBOL()
  - out of tree .ko makefile
  - module-init-tools
  - put module in kernel tree
  - build and install
  - load it
  - parameter passing

    - Access TCB
- Character Driver
    - Device Files
        - Intro
        - device types
        - major/minor
        - Architecture
    - Driver Kernel Interface
    - Device Driver
        - Intro
        - Registration
        - Initialization
        - Open Release
    - Misc. Char Drivers
- User Space Debugging
    - Debugging: Simple Tools
        - lsof, ltrace, strace,...
        - procfs, top, netstat, syslog,...
    - Debugging: Advanced Tools
        - What's a Debugger?
        - target gdb
        - gdb remote debugging
- Kernel Debugging
    - Debugging Intro
    - KGDB/KDB
    - JTAG
- Process IPC
    - IPC Intro
        - Unix/Linux architecture
        - What's on OS?
        - What's a scheduler?
        - Linux scheduler
        - Linux priorities
        - Linux scheduler(s)
        - Linux scheduling classes
        - Process/Task/Thread
        - errno
        - fork()
        - Process termination

        - Process states
        - Zombies
        - More about Processes
        - Watch a process
    - Simple IPC
        - shell redirection
        - shelling out
        - tempfiles
    - IPC Generic
    - IPC
        - Message passing vs. shared memory
    - Advanced IPC
        - Pipes
        - Signals
        - Interrupted System Calls
        - POSIX.4 Message Queues
        - Semaphores Introduction
        - Mutex
        - Semaphores
        - Shared Memory
        - Sockets
        - select
            - self-pipe trick
        - Other IPC mechanisms
    - IPC techniques to avoid
- Real-time
    - prerequisites
        - Kernel vs. User Space
        - Toolchain
        - Program Sections
        - Interrupts
        - Reentrant Code
    - Real-Time Intro
        - Time/Utility Functions
        - What is Real-Time?
        - Determinism
        - What is hard real-time?
    - Real-Time Linux
    - Degrees of Real-Time behavior

    - Dual Kernel
        - Xenomai
        - Measurement results
    - Real-Time Myths
- Yocto Introduction
    - What is Yocto?
    - Why use the YP?
    - What is the YP?
    - Some tools under the YP umbrella
        - Poky
        - BitBake
        - OE-Core
        - Metadata
- The Yocto Autobuilder
    - Intro
        - What is the Yocto Autobuilder?
        - Docker container (pull, launch container)
    - Yocto build environment without Docker/Yocto Autobuilder
- The YP Workflow
    - Intro
    - Workflow
        - OE architecture
    - Configuration
        - User Configuration
        - Features
            - Machine Features
            - Distro Features
            - Combined Features
            - Image Features
        - Recipe Versioning
            - Intro
            - Hyphens
            - SCM-based
            - Pitfalls
            - Devel/Stable
            - Overrides
        - Metadata (Recipes)
        - Machine (BSP) Configuration
        - Distribution Policy

    - Sources
    - Build
        - Source fetching: do_fetch, do_unpack
        - Patching
        - Configure/Compile/Install
        - Pseudo
        - recipetool
        - Examples of Recipes
            - Single .c File Package
            - Autotooled Package
            - Splitting App. in Multiple Packages
        - Output Analysis/Packaging
        - Image Generation
        - SDK Generation
    - Customizing Images
        - Intro
        - local.conf
        - IMAGE_FEATURES
        - custom .bb files - inherit core-image
        - custom .bb files - based on core-image-minimal
        - custom packagegroups
- BitBake
    - History
    - Syntax
        - Variable Expansion
        - Variable Assignment
        - Pre-/Append
        - Removal (Override Style Syntax)
        - Variable Flag Syntax
        - Conditional Syntax (Overrides)
    - Debugging
        - BitBake debugging
        - find recipes
        - find images
        - find packagegroups
        - BitBake Environment
        - BitBake logs
        - Re-BitBake stuff
            - force build/specific task

            - cleansstate
            - invalidate stamp
        - Devshell
        - Dependencies
        - Packages
        - Killall Bitbake
        - BitBake with ncurses wrapper
        - Tools/Tweaks
    - Cleaning
        - Cleaning to gain disc space
        - Cleaning to rebuild
- Layers
    - Intro
    - bitbake-layers tool
    - yocto-layer tool
- BSP
    - Intro
    - System Development Workflow
    - BSP Developer's Guide
        - bsp-tool
- Kernel
    - Intro
    - System Development Workflow
    - Kernel Development Manual
        - defconfig + configuration fragment
        - in tree kmod
        - out of tree kmod
        - fdt
- Software Development Kit
    - Software Development Kit
        - Intro
        - Cross-Development Toolchain
        - Sysroot
        - The QEMU Emulator
        - Eclipse Yocto Plug-in
        - Performance Enhancing tools
    - Installing SDKs & Toolchains
    - Cross-Toolchains/SDKs
        - Intro

        - Building a Cross-Toolchain installer
        - Using the Standard SDK
        - Cross-Toolchain+Makefile
        - Cross-Toolchain+Autotools
        - Autotooled lib + App., recipes
        - Extensible SDK
- Package Management
    - Software updates
    - Working with packages
        - IPK
        - creating a package feed
        - installing a package with opkg on the target
- Licensing
    - Intro
    - Add custom license to the YP
    - Open Source License Compliance with the YP
- Devtool
    - Intro
    - Add recipe/Build/Deploy
    - Create/Add layer
    - Finish
    - Modify/Update-Recipe
    - Build/Run
    - Build Image

Methods and Training Materials

Lecture and practical examples with host (laptops with Ubuntu 14.04.x LTS) and target system (e.g. Beagle Bone Black Rev.C – http://beagleboard.org/BLACK). The devices will be provided during the training. You receive a reference- and workbook, one Beagle Bone Black Rev.C plus one standard FTDI 3.3 V to USB cable.

After the training you will get access to a download link with a prepared Docker image and examples.

Technical requirements to attend a remote/online training

- (ship-it/web/host/target/phone) e-mail address to get login credentials
- (web) screen sharing/audio/video/whiteboard/chat/Q&A: https://www.bigmarker.com requires this: https://rlbl.me/bm-req.

- backup: (web)/audio (phone) conference call: https://www.turbobridge.com/international.html
- (host/target) shell: port 22 not blocked: something like: ssh @vlabx.dyndns.org
- backup: (host/target) shell via browser: port 443 not blocked: something like: https://vlabx.dyndns.org

Duration

5 days

Price

2.590 € plus VAT per participant

Software Test

ISTQB® - Certified Tester Foundation Level – CTFL (EN)

Beschreibung

In this basic training, software testing tasks, methods and techniques are taught in accordance with the internationally standardized syllabus.

You will get a comprehensive overview of all activities related to software testing, starting with the test process, planning, designing and implementing the test and ending with test execution and test management. Basic aspects of tool support are also considered.

After the seminar, you will be able to select and apply important test case design procedures and know all the steps of software testing from the module to the acceptance test as well as the connections with the overall project.

This training is conducted by the accredited training provider sepp.med

Target Group

Software developer, Software tester, Test analysts, Test designer, Test manager, Quality manager

Prerequisites

Software test experience is of advantage

Training Content

- Basics of Software Testing
- General test process
- Static analysis
- Structural and functional testing
- Testing in the software life cycle
- test management
- Tool support

Certificate

The examination by iSQI takes place on request on the last day of the course. After passing the exam, the participant receives the certificate ISTQB® Certified Tester – Foundation Level.

Duration

3 days

Price

Online: 1.290 € plus VAT per participant
On site: 1.440 € plus VAT per participant
Examination fee: 200 € plus VAT per participant

RISCV, System C, TML2 und Virtual Prototype Primer

RISCV - An Introduction

This 1-day introduction to RISCV introduces the concepts which are standardized by the RISCV foundation. It will also provide an overview over the most common SW tool in the RISCV tool chain and the most common open source hardware platforms. Some demonstrations will be used to show how to get started with a RISCV based development flow.

Target Group

Architects, software and hardware developers, specifically for embedded systems

Prerequisites

-

Training Content

RISCV overview

- History
- Role of RISCV foundation
- Standardization within the RISCV foundation
- Platform standardization organizations, e.g. Open Hardware, Chips Alliance

RISCV HW introduction

- Unprivileged instruction set architecture (ISA)
- Privileged Architecture
- Extensions

RISCV development tools

- SW development tools (compiler, debugger, …)
- Emulation and prototyping

Open source RISCV platforms

- RocketChip based HW elements
- Pulp Platform based HW elements

Demonstration

Duration

1 day

Price

790 € plus VAT per participant

SystemC / TLM2 Primer

This 3-day training introduces the SystemC C++ class library and the TLM 2.0 modeling standard. It is intended for engineers who are new to SystemC or those with an interest in learning SystemC for modeling purposes. The participants will learn how to write, compile, execute and debug system and hardware descriptions with SystemC and will receive thorough coverage of the concepts of the Accellera/IEEE TLM 2.0 modeling standard. This course is a mix of lectures and exercises.

Target Group

Developers in the area of HW/SW co-design

Prerequisites

Fundamental C/C++, SystemC and TLM2 knowledge is a prerequisite. A laptop with permissions to install software is required.

Training Content

Introduction to SystemC

- Core library basics
  - Modules& communication (channels, ports, and exports)
  - Simulation kernel: scheduler, events, and event queues
- Modeling behavior
  - Method processes
  - Dynamic and static thread processes
  - Hierarchy creation and Simulation semantics
- Core library elements
  - SystemC data types
  - Debugging and tracing aids
  - Primitive channels
- User defined channels
  - Custom constructors

Introduction to the IEEE TLM 2.0 Standard

- TLM 2.0 Overview
  - Interfaces, sockets, generic payload, and protocol
- Generic payload overview

- Interfaces
  - Transport (blocking interface and non-blocking)
  - DMI
  - Debug
- Sockets
  - Initiator and Target
  - Socket Binding
  - Hierarchy, Multi-connect
- Convenience Sockets
  - Simple Sockets
  - Tagged Sockets and multi-passthrough Sockets
- Generic Payload In-depth
  - Byte Enable, Streaming, and endianness
  - Memory Management
  - Generic Payload Extensions (and exercise)

SystemC based Standards and Libraries

- SystemC Verification Library
- Control and Configuration Interface (CCI) for SystemC
- SystemC Unified Verification Methodology (SystemC UVM)

Duration

3 days

Price

2.370 € plus VAT per participant

Virtual Prototype Primer

This 1-day training introduces the participants to the foundations of Virtual Platform (VP) modeling and deployment. It is intended for engineers who are new to or interested in learning about VP modeling.

The participants will learn how to write and debug VPs as well as which features are typically needed for a successful VP deployment. The focus is on providing an overview over different components, tools and techniques. The training is agnostic with regards to tool or IP vendors. This course is a mix of lecture and exercises. The exercises will be shown as demos but the

student will have access to all material in order to follow along or repeat exercises on their own.

Target Group

Developers in the area of HW/SW co-design

Prerequisites

Fundamental C/C++, SystemC and TLM2 knowledge is a prerequisite.

A laptop with permission to install software is required.

Training Content

Introduction

- What is a VP?
    - Virtual prototype versus Virtual platform
- How do VPs fit into the SoC development cycle?

Modeling techniques

- Behavioral/untimed
- Functional/loosely timed
- Cycle-accurate/approximately timed
- Register-transfer-level

Standards

- C++/SystemC,/TLM2
- Register modeling e.g. IP-XACT, RDL
- Productivity libraries, e.g. SCML
- Commonly used open source libraries

IP for VP development

- Typical building blocks of a VP
    - ISS, Peripherals, Interconnects
- Commercial 3rd party IPs, e.g. ARM, Synopsys, Cadence
- Make versus open source versus buy

Tools for VP development

- Commercial tools, e.g. Virtualizer, SoC Designer, VLAB Works

- Open source tools
- Connections to other environments
- Pitfalls

Putting it all together

- Setting up an example VP
- Debug & analysis
- Debugging the VP itself
- Debugging SW/FW components running on a VP
- Infrastructure, e.g. test-driven development (TDD), continuous integration (CI)
- Configuration

Addressing typical requirements for VP deployment

- Simulation speed
- Scalability, e.g. extensibility, documentation, usage model, packaging
- Observability, e.g. TLM recording, logging, data visualization
- Accuracy

Duration

1 day

Price

790 € plus VAT per participant

Measurement Technology

Strain gauges - Understanding and interpreting measurements in practice

Description

Keywords: Loads on construction components, forces, torque, stress, strain.

Based on professional experience, strain itself is usually of minor interest in strain gauge measurements. Therefore, the training will (unlike most courses on this topic) focus on the effect of load which causes finally strain. After excursions into the basics of mechanics and

elastostatics, there will be an introduction into the technique of resistance strain gauges: from principle to types, circuitry, special features, limits to errors and uncertainty.

As the trainer has long-standing practical experience in the field of strain gauge measurements, the training is supplemented by many performed measuring tasks. By a broad scope of examples as well as “lessons learned”, you will delve into the topic. The theory will be deepened by a practical part with an application of a strain gauge.

Target Group

This compact training is meant for everyone who wants to understand how it comes to strain and how to interpret the results of strain gauges measurements.

Prerequisites

There are no special prerequisites for this training.

Duration

1 day

Price

790 € plus VAT per participant