Tuesday (10/8/2019)

Session 1 (10:00 AM – 11:50 AM): Hands-On Workshop: Getting Started with S32K

Software Tool used – Eclipse

Hardware – Micro-Controller, ARM Cortex – M

* Configuration settings in Eclipse for the targeted Hardware with Debug session, password protected debug session
* IRQ handler routine
* GNU compiler settings for code optimization
* Debugging code the right way
  + Run the program
  + View the variables tab to locate the address of the variable
  + Fetch the address in disassembly window
  + Correlate the c source
* Things to consider not to hit Hard Fault. Hard Fault is the scenario where the Hardware hangs on power up.
  + Try to avoid non-mapped memory address
  + Avoid illegal pointers
  + Get the interrupt routine right
    - Clear the interrupt
    - Set the interrupt
    - Set the priority

Session 2 (1:30 PM – 2:20 PM): CAN Secure

* A brief session about securing different aspect of Hardware / software. The talk was more focused around CAN bus but the idea can be well extended to any other component.
* Secure by Design Vs Security as an afterthought, you know the answer, Secure by Design obviously.
* The challenge in designing a secure system is to find the right balance b/w risk and cost. The debate here is not whether hacker will break the system but is how much time and resources a hacker needs in order to break the system.
* One of the questions we need to ask ourselves is whether there exists any link in our system/architecture using which hackers can exploit our system.
* Prevention of all types of attacks :
  + Local attacks : attack using the system hardware
  + Remote attacks: attack using the software / remote application
* Secure following four aspects
  + Secure Interface – Incoming messages
  + Secure Gateway – Information inside the system
  + Secure Network – Exchange of data / communication Link
  + Secure Processing – Secure boot
* Threats to be handled
  + Spoof- authentication
  + Information Disclosure – Confidentiality
  + Tampering – Data Integrity
  + Denial of Service – Flooding
  + Repudiation
  + Elevation of Privilege – Authorization
* Key Management
  + Symmetric
  + Asymmetric

Session 3 (2:30 PM - 3:30 PM):i.MX 8 MCUs with Safe Assure®-Certified Features –Camera, Video, Audio, and Trusted Environment

* The talk was more focused on Leveraging Trusted Environment to secure different software/Hardware components as well as different functionalities for e.g. Camera, Video, and Audio etc.
* Securing the entire system using Hardware would be reliable but a costly approach, not much reusable / scalability.
* They have come up with an architecture where they have managed to separate safety & open source processes and run them. There is a way to exchange data/calls between the trusted world application and normal world applications.
* They did not discuss much about how they have achieved these architectures. The main objective of the presentation was to bring out these solutions to the market.

Session 4 (4:00 PM – 4:50 PM): Yocto Project TM 101

* The talk was focused on how to use different commands/features of bitbake to build an image of a suitable operating system for a targeted hardware. This would be helpful for customization of our operating system.
* Basic housekeeping rules while building OS. How to create and name directories, how to create different versions and how to maintain those versions, how to speed up the build process (normally a build process takes about 4-8 hours depending upon the configuration of the host machine). The idea is to create cache files and re-use these cache files while building another image. Remember that you need to build the image every time there is small change. Every time you issue “bitbake” command, it fetchs files from the remote server. The idea here is to guide “bitbake” to only search the changed files from remote server. The unchanged files could be made available by those cahes.
* Basic housekeeping rules while writing the image (OS) to SD card. “dd” command is used to write image (OS) to the SD Card . “fsync” or “sync” command should be issued at the end of the writing process. Linux actually writes the image (OS) to the SD Card only when “fsync” or “sync” command is issued. Remember “dd” only copies the image (OS) to the SD Card, it does not actually write image (OS) to SD Card.
* Basic rules to handle multiple builds
* How QEMU Emulator can be utilized to run the targeted architecture without actual hardware in place.

Wednesday (10/9/2019)

Session 1 (9:10 AM – 10:00 AM): Yocto Project TM Advanced

* The main focus of the talk was on how to build different layers of the image (OS) for the targeted hardware
* Different layers
  + Board Support Package - Machine layer
  + Networking
  + Generic Drivers (USB, Wi-Fi, etc.)
  + 3rd party drivers
  + Framebuffer (drm)
  + Qt support
* Different naming conventions for folders and use of git to achieve version control.

Session 2 (10:10 AM – 11:00 AM): Altia: Future Architecture of Vehicle Cockpit Displays with Multiple Inter-Operating Systems

* The main focus of the talk was on how to achieve high quality of Display, seamless integration of hardware components running on different operating system (Linux, Android, etc.)
* How to present the critical message on the Display without distracting the driver to avoid panic or sudden change in behavior of the driver which could cause fatal incidents.
* More focus on automotive & what products and services are available with Altia. More integration focused.
* Design of control logic. Which component/hardware to give control of the display
* Proper hardware resource allocation.

Session 3 (11:10 AM - 12:00 PM): S32 Design Studio Tools – S32 for ARM®, S32DS for e200 Overview of Tools

* Main focus was on tool selection for targeted hardware, toolchain selection for the targeted hardware, know your architecture to best utilize its capability.
* More light on tools for different architecture developed using Eclipse.
* We can follow similar workflow to develop SDKs for our customers using Eclipse & qt creator.

Session 4 (1:30 PM – 2:20 PM): PCIe® Overviews

* Main focus was on signal Integrity Issues that should be taken care while working with high speed bus
* Current state of PCIe® affairs in Industry. I feel that NXP are much more behind than other companies like TE Connectivity in this business.
* Shed some light of how exactly PCIe® works
* Expected date to roll out PCIe 4.0 and 5.0
* Taking advantage of PCIe interface
* How data integrity is maintained in High Speed Bus.
* Effect of Resistance, Inductance and Capacitance on the bus (micro strip and strip line)
* Effect of Vias on speed of communication and integrity of data
* Burst mode communication and its effects
* Use of differential pairs for reducing Common Mode Noise

Session 5(2:30 PM - 3:30 PM)

* Automotive Ethernet , different IEEE standards uses in automotive industry
* Deep discussion about Open Standard Interconnect (OSI) Layer and how automotive industry is using it.
* Did not discuss much about POE architecture though.
* Discussion on how automotive industry could not use IEEE standard cable architecture directly due to increase in cost and less availability of space in automotive
* Variation of OSI layers to fit the need of the targeted architecture/system
* Security related to Ethernet, role of SSL and TSL.
* Discussion of different Network topology.

Session 6 (4:00 PM – 4:50 PM): Arm: Open Source Compute Libraries to Accelerate Algorithms on ARM ® Cortex® - M Processers

* Discussion of nomenclature of the processors and how they represent their capabilities. For example; ARM Cortex-A = Application processor, can run graphics and rich operating system (Linux, Windows), ARM Cortex –R = runs on Real time operating system, ARM Cortex – M = Micro-controller architecture, can run bare metal operating system or no operating system.
* Discussion of ARM Development studio software tool. Can be used to develop APIs for targeted architecture.
* Discussion about Digital Signal Processing (DSP) Libraries, Neural Network Libraries (NN)
* Discussion on role of Hand-optimized code. Myth – “compiler does all the work to generate suitable assembly code. Compiler optimization flags can be used to generate highly optimized code”. Reality “Compiler always does not generated highly optimized code. If the code is time optimized, there is a high chance it is not memory optimized and vice-versa”. Programmer needs to step in to write that portion of code in assembly so that time and memory optimization can be achieved together.
* Discussion on zero-overhead loop capability.