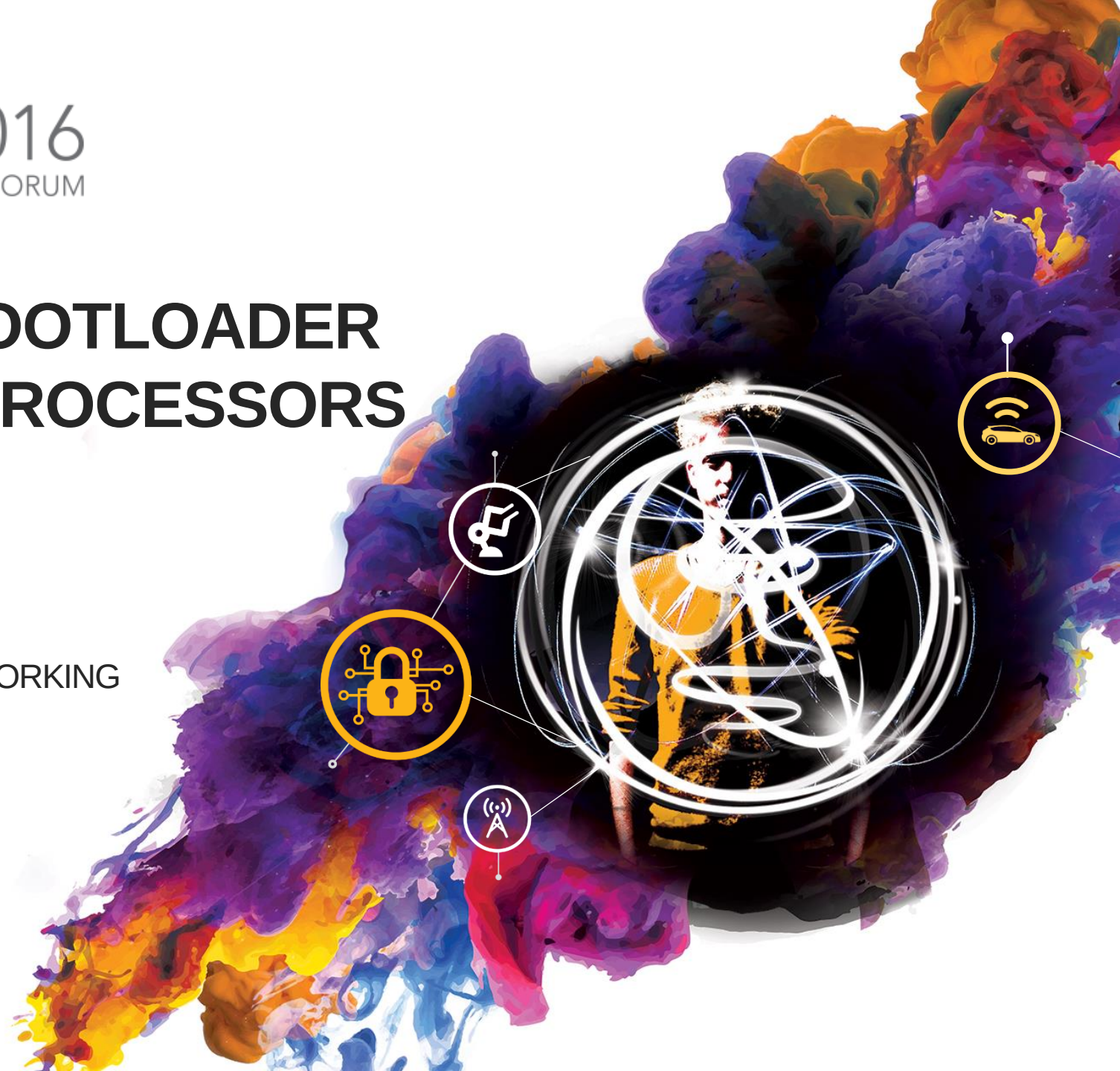
FTF 2016
TECHNOLOGY FORUM

INTRODUCING UEFI BOOTLOADER ON QorIQ LS SERIES PROCESSORS

FTF-DES-N1839

BHUPESH SHARMA
TEAM LEAD, UEFI FIRMWARE, DIGITAL NETWORKING
FTF-DES-N1839
MAY 17, 2016



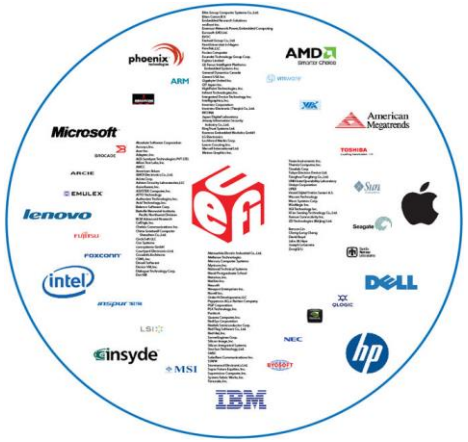
AGENDA

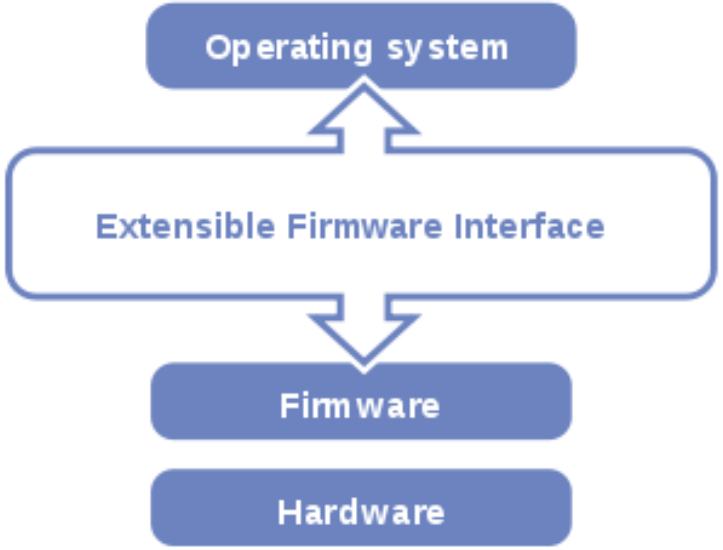
- UEFI Firmware Overview
- UEFI Boot Process On ARM®-based SOCs
- UEFI & Other Bootloaders
- Why UEFI?
- UEFI On QorIQ LS Series SoCs
- UEFI + GRUB2 + CentOS distro on LS Series SoCs

UEFI FIRMWARE OVERVIEW

What is UEFI?

- UEFI is a community effort by many companies in the personal-computer industry to modernize the booting process.
- UEFI stands for ***Unified Extensible Firmware Interface***
- UEFI specification defines a new model for the interface between operating systems and platform firmware.



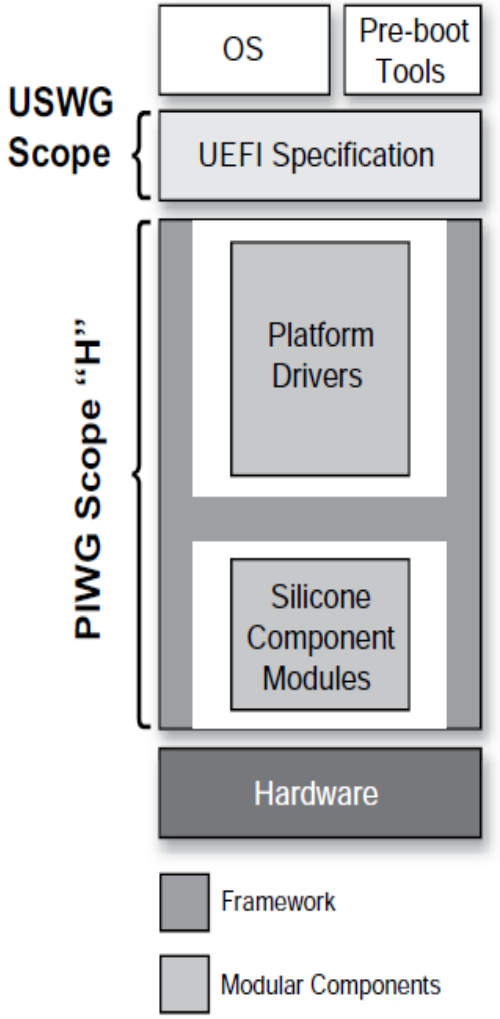


UEFI Specifications – A Community Effort

Who creates and manages the UEFI Specifications?

- *UEFI Specification Working Group (USWG)*
  - Creates the UEFI specification, describing a firmware-to-OS interface analogous to BIOS software interrupts and the BIOS data area (BDA).

- *Platform Initialization Working Group (PIWG)*
  - Creates the PI specifications, intended to promote interoperability between firmware components provided by different silicon and firmware vendors.



UEFI Source Code – A Community Effort

Tianocore EDK2 is a feature-rich, cross-platform firmware development environment for the UEFI and PI specifications

Governance

Tianocore governance updated

New stewards announced for Tianocore:

- Andrew Fish – Apple
- Leif Lindholm – Linaro
- Michael Kinney – Intel

Community Manager

- Tony Mangefeste

UEFI Overview – FAQs

- **How do UEFI specifications differ from BIOS?**
  - BIOS is typically used to refer to an Intel® Architecture firmware implementation rooted in the IBM PC design.
  - UEFI is *processor architecture-agnostic*, supporting x86, x64, ARM and Itanium.

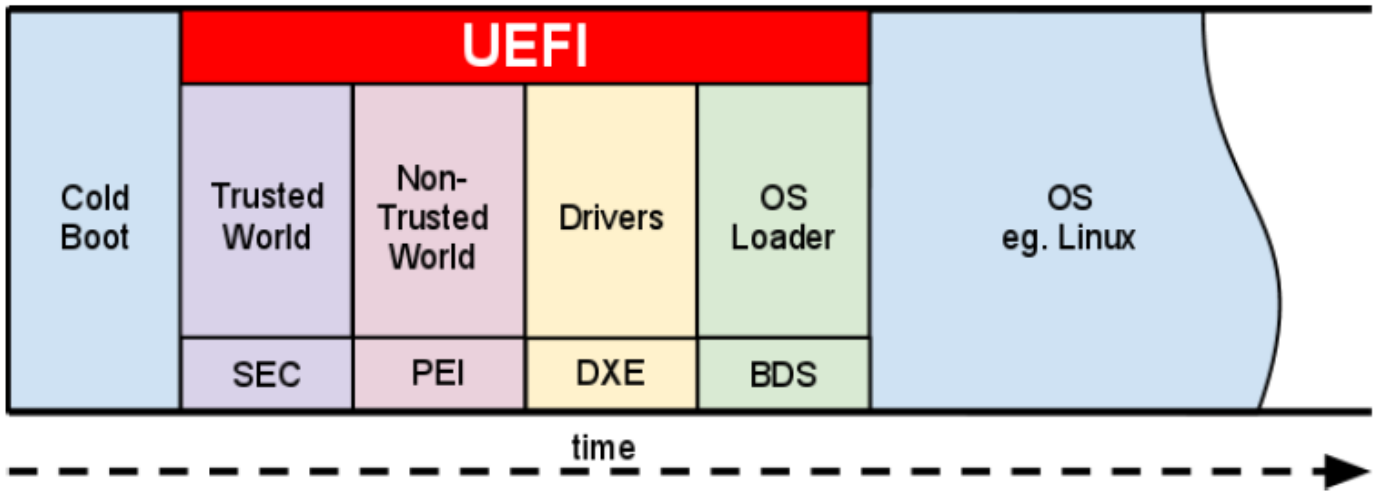
- **Do UEFI specifications completely replace the BIOS?**
  - UEFI specifications define an interface between firmware that initializes the platform and the OS.
  - BIOS refers to a specific implementation of such a firmware.
  - UEFI specifications define an interface in which the implementation of UEFI performs the equivalent of the BIOS.

- **What is the relationship between EFI and UEFI?**
  - UEFI specification is based on the EFI 1.10 specification published by Intel®, with corrections and changes managed by the UEFI Forum.

UEFI BOOT PROCESS ON ARM- BASED SOCs

UEFI Boot Process – A Top-level View

- **UEFI power-on flow**
  - Follows traditional UEFI boot sequence through SEC through BDS
    - Further details in slides to follow
- **Not all UEFI phases are essential**
  - SEC phase can be skipped if a BootROM or Security Firmware is used before UEFI starts.



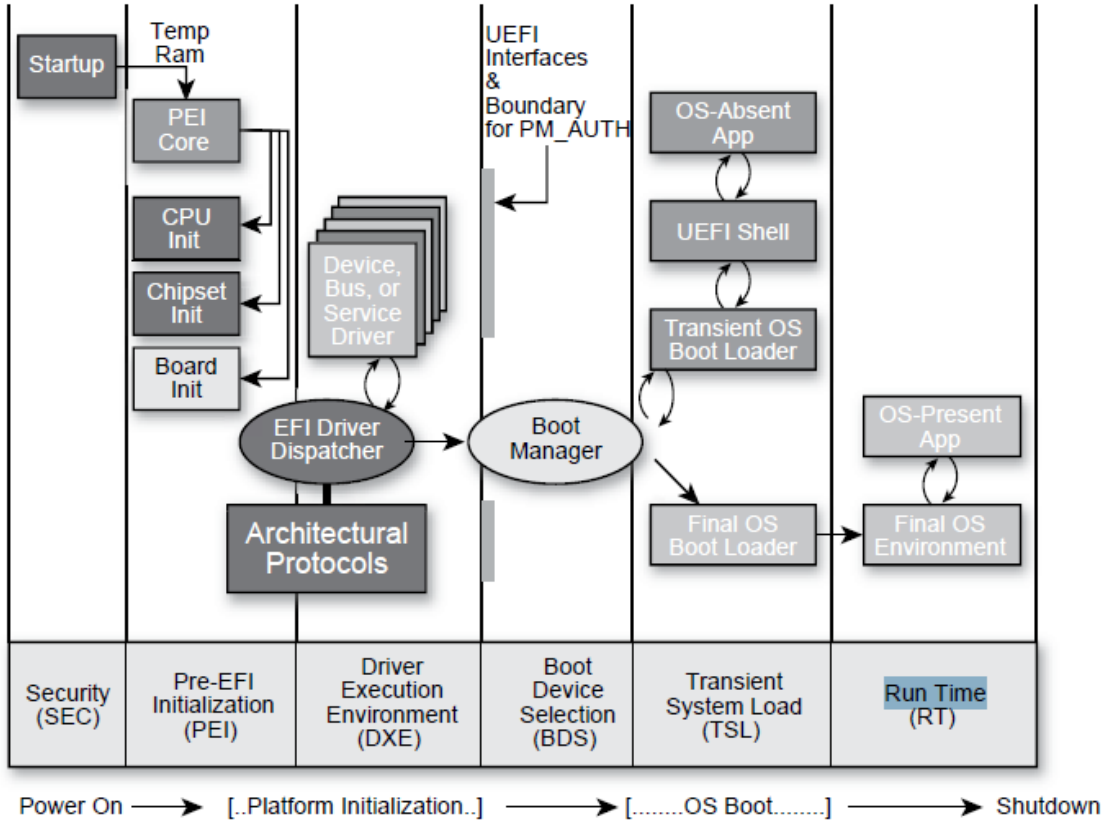
SEC	**SEC**ure Code	*xip*
PEI	**P**re-**EFI** **I**nitialisation	*xip -> relocated*
DXE	**D**river e**X**ecution **E**nv.	*relocated*
BDS	**B**oot **D**evice **S**election	*relocated*

UEFI Boot Process – Detailed View

UEFI boot process on ARM SoCs involves 6 prominent stages

- **SEC** phase:
  - 1st PI phase run after POR.
  - Platform and Processor architecture dependent phase.
  - Creates a temporary memory store.
  - Serves as the root of trust in the system.

- **PEI** Phase:
  - 2nd PI phase.
  - Does CPU and board initialization/platform configuration.
  - Initializing permanent memory – DDR.
  - Discover and launch DXE core and convey platform information to it.
  - It start in XIP and later relocated to system memory.

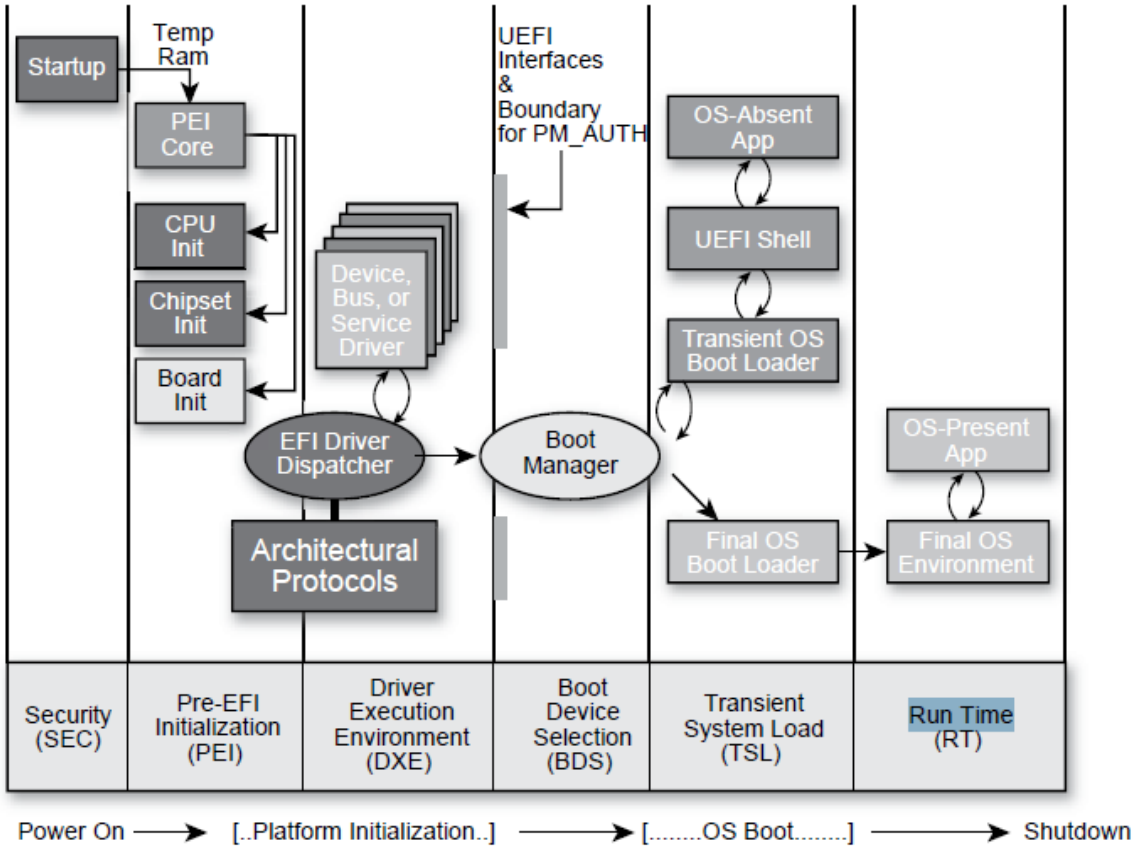


UEFI Boot Process – Detailed View

UEFI boot process on ARM SoCs involves 6 prominent stages

- **DXE** Phase:
  - Bulk of booting occurs in this phase
  - DXE phase code, is loaded and executed from system memory
  - Discovers and executes DXE drivers in the correct order.
  - Provides software abstractions for system services, console devices, and boot devices

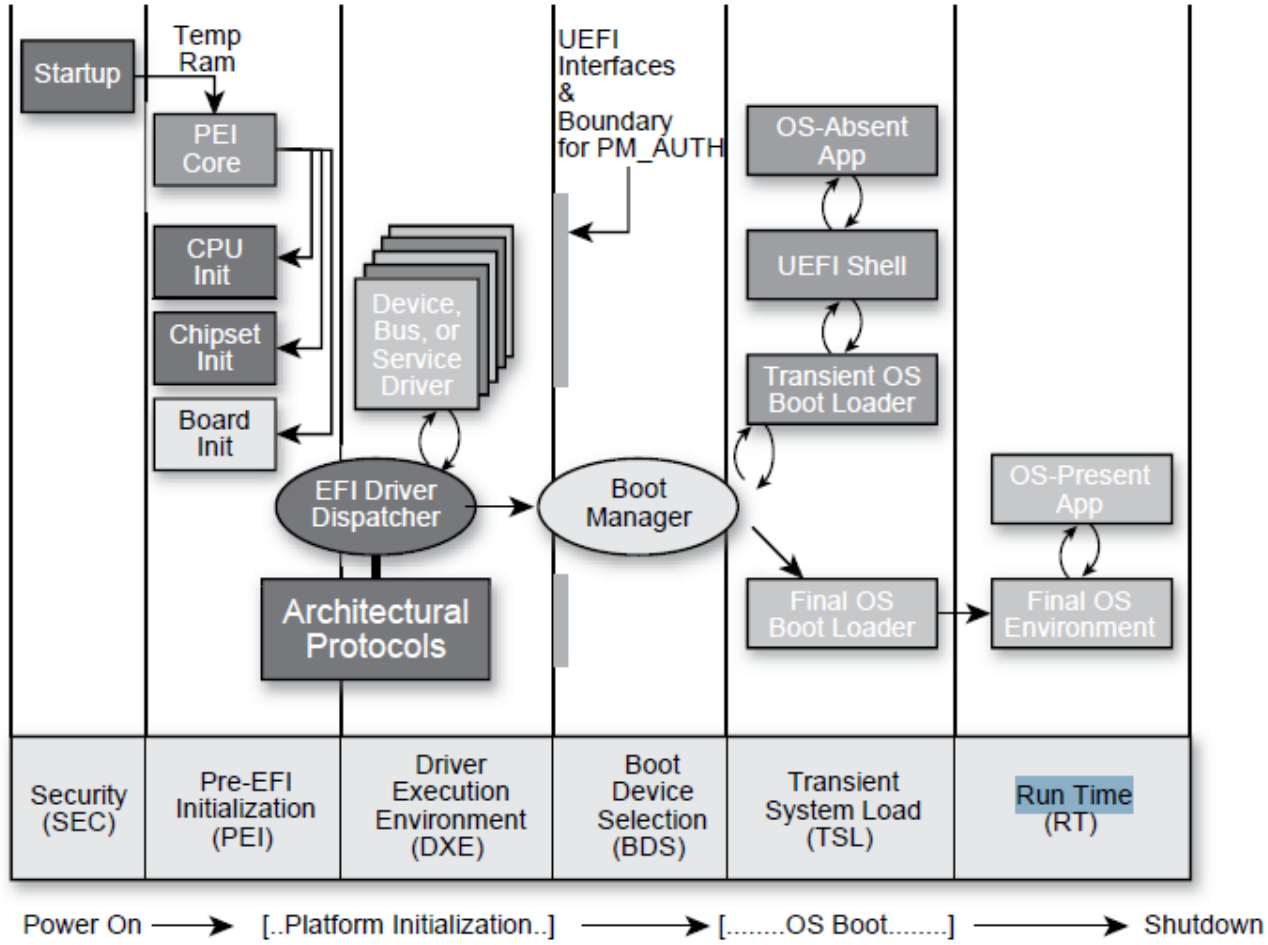
- **BDS** Phase:
  - How and from where you want to boot OS
    - Initializing console devices
    - Loading device drivers
    - Attempting to load and execute boot selections.



UEFI Boot Process – Detailed View

UEFI boot process on ARM SoCs involves 6 prominent stages

- **OS** Phase:
  - *TSL* Phase
    - TSL is the 1st stage of the boot process where the OS loader is an EFI application.
  - *RT* Phase (afterlife)
  - Outside the booting phase
    - Run Time Services are available even after the Operating System boots and takes over the system.
    - E.g. Real Time Clock service.

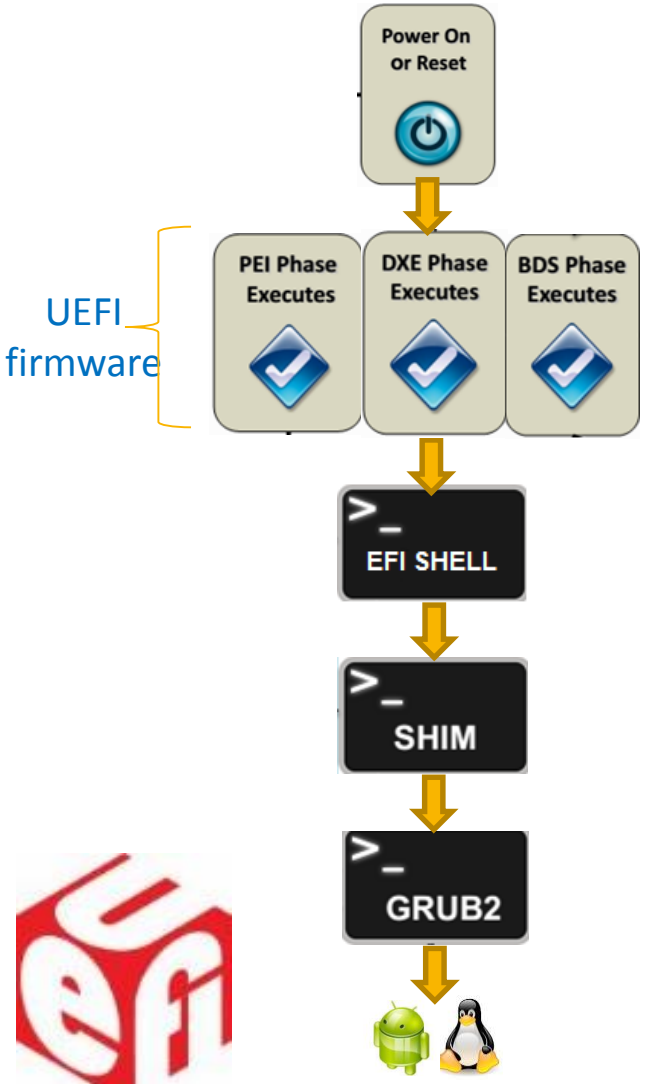


UEFI & OTHER BOOTLOADERS

UEFI is Not a Bootloader

- **UEFI is essentially a firmware**
  - It can chainload various bootloaders (for e.g. GRUB2 or LILO) to boot OS variants.
- **Comparison with u-boot bootloader**

Feature	UEFI over u-boot
Boot-time	✓ UEFI is usually faster than u-boot. ✓ Employs delayed slave probing to achieve faster boot-time.
Footprint	✓ UEFI is compressed and hence smaller.
OS which it can boot	✓ UEFI is essentially agnostic to the OS it loads. ✓ UEFI can support Linux, Windows, Mac and Android
Specifications	UEFI provides a forum-controlled specification

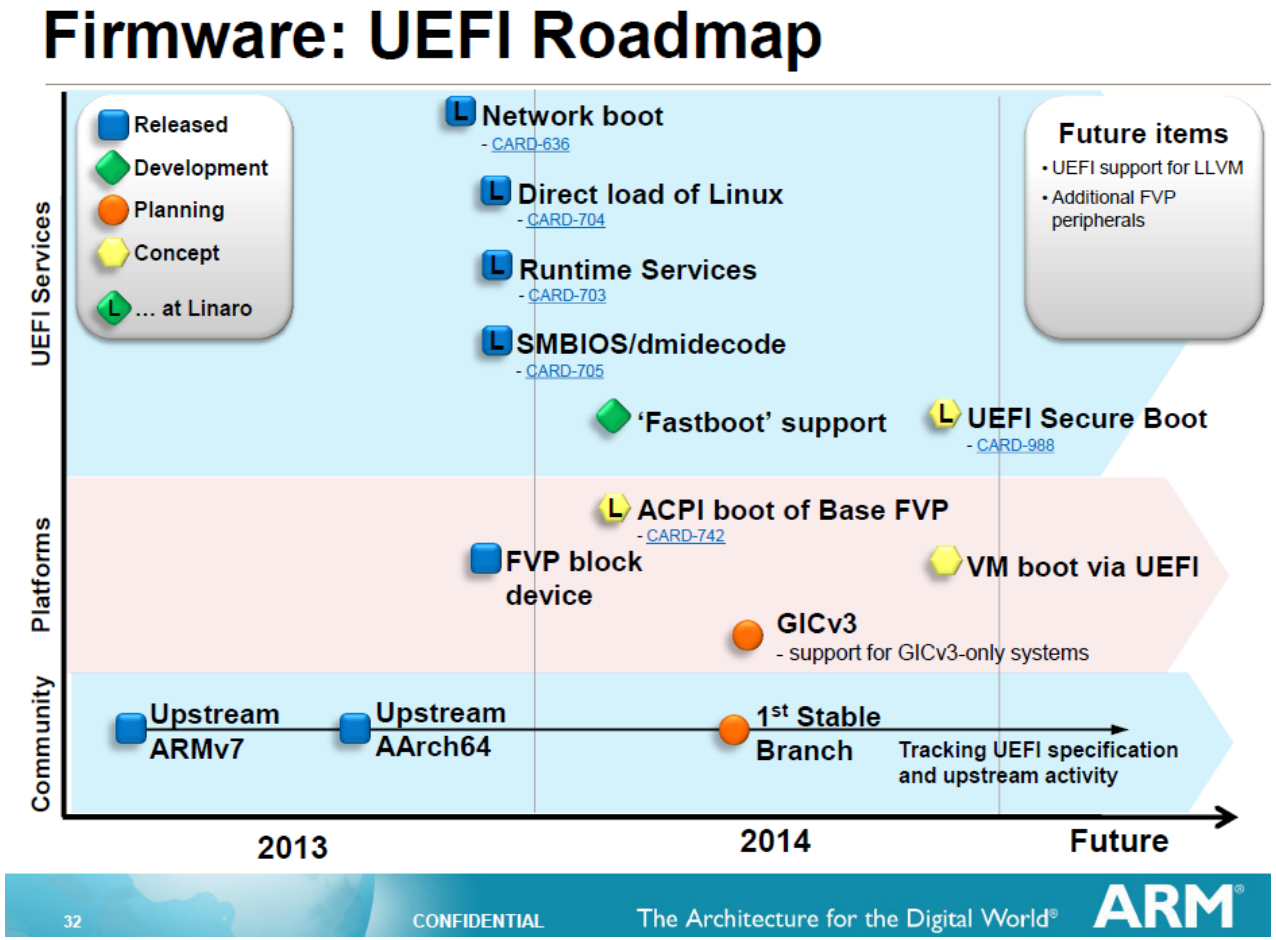


WHY UEFI?

ARM/Linaro Roadmap

- ARM/Linaro plan to support UEFI as the default boot firmware on ARM-based SoCs.

- Juno (ARMv8 based platform from ARM) already uses the UEFI ecosystem.

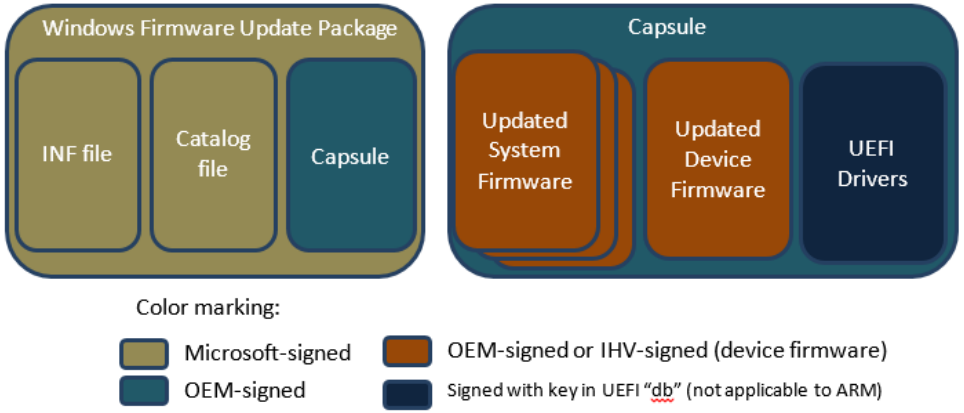


UEFI – Differentiating Features

Firmware Update

- Secure **Run-time** variable services.
  - UEFI specifications defines *Firmware Management protocol* which offers interfaces to:
    - validate, read and write firmware.

- UEFI supports
  - verifying the firmware binary integrity.
  - updating the firmware.
  - verifying update is successful.
  - In case of a failure a rollback can be performed.



UEFI – Differentiating Features

Bare Metal Provisioning

- Needs a *no-touch*, automated installation medium
  - Repurpose / Configure / Recover.

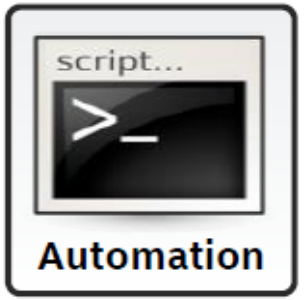
- UEFI supports
  - reading an OS image from the network and writing to the local system board storage.
  - usages of the Human Interface Infrastructure (HII), which is a forms-based mechanism for configuration.
  - mandatory persistent storage in the platform by way of the UEFI Variable interface.

Bare Metal Provisioning

- Pre-Boot Networking
- IPv4, IPv6 TCP/UDP
- PXE, iSCSI, HTTP, FTP

Deployment

- Boot Device Selection
- Boot Order control
- OS install & recovery



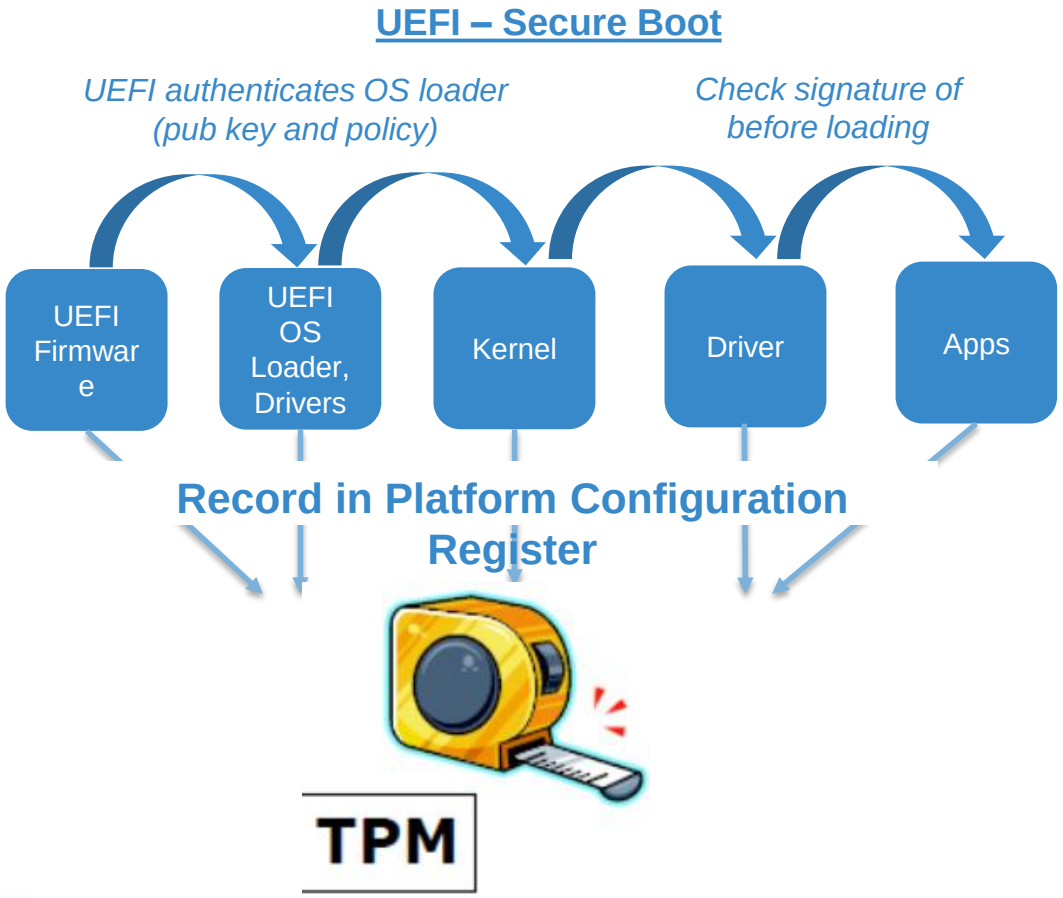
Automation

- UEFI Shell
- Scripting language

UEFI – Differentiating Features

Secure and Measured Boot

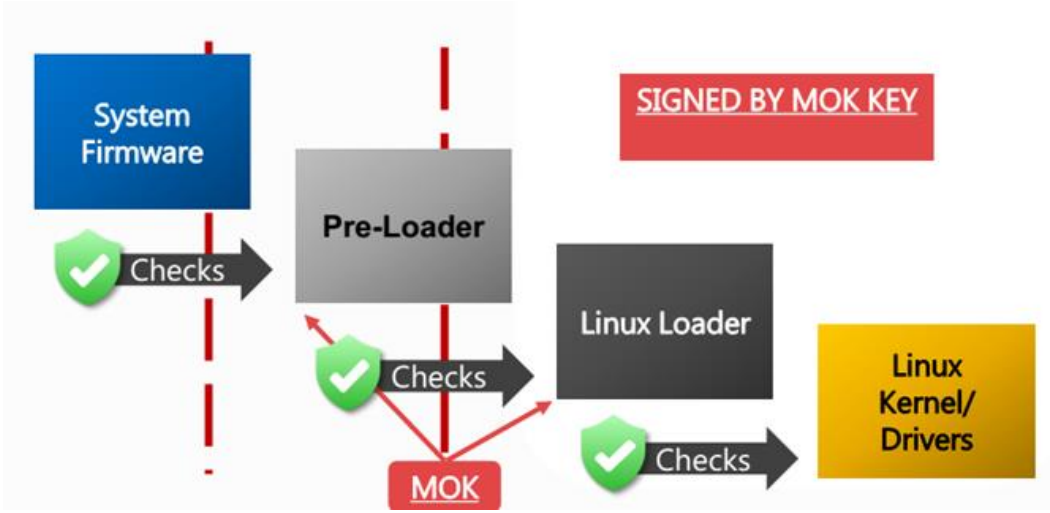
- Secure Boot - UEFI
  - Defined a policy for Image loading
  - Cryptographically signed
  - Private key at signing server
  - Public key in platform
- Measured Boot -Trusted Computing Group (TCG)
  - Trusted Platform Module (TPM)
  - Isolated storage and execution for Logging changes, attestation



UEFI – Differentiating Features

Secure Boot

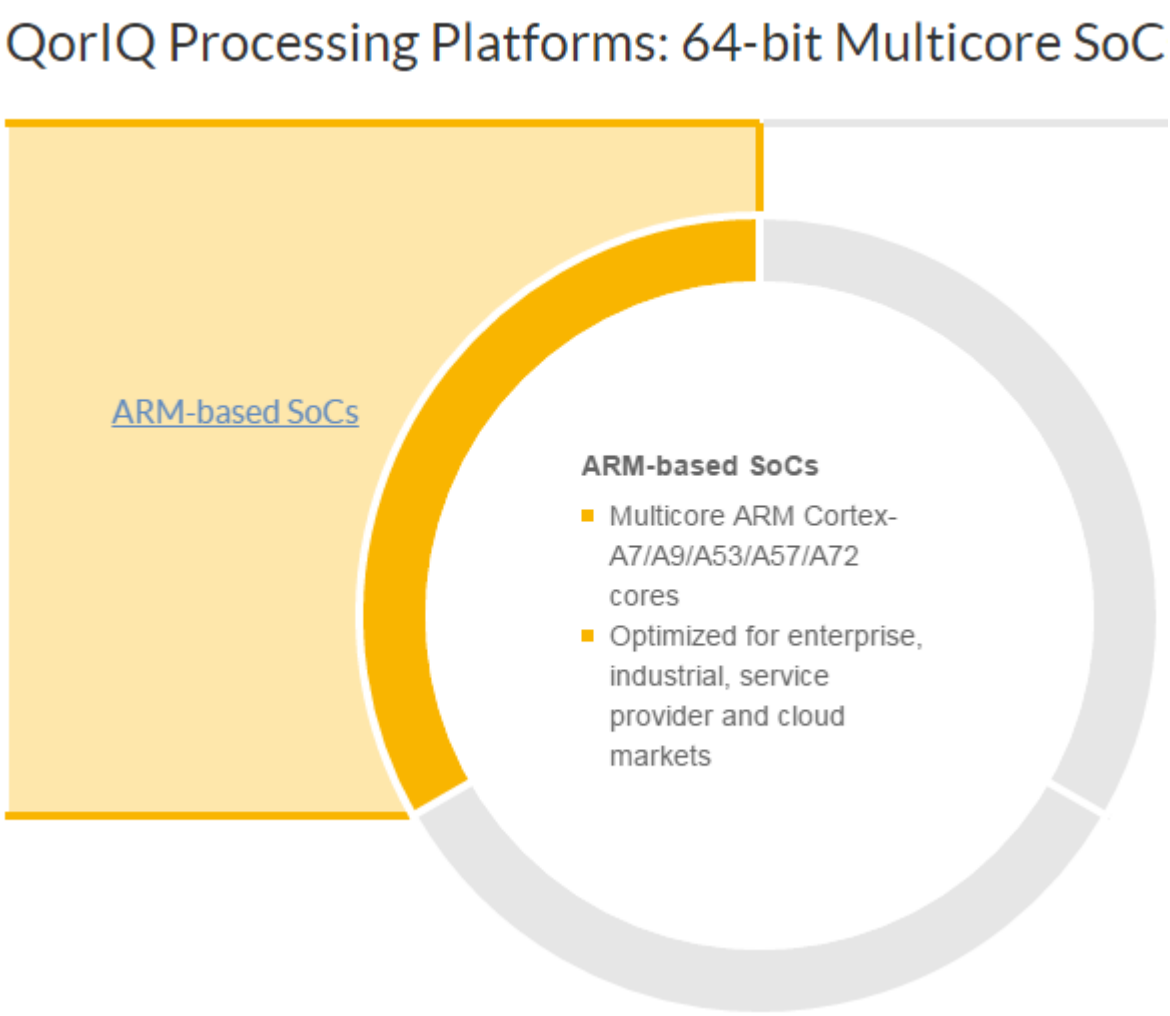
PK	Platform Key – Root key
KEK	Key Exchange Key - List of Cert. Owners with db, dbx, dbt and dbr update privilege
db	If signed by key in ***db***, driver/loader can Run
dbx	If signed by key in ***dbx***, driver/loader forbidden
dbt	If signed by key in ***dbt***, Check cert's timestamp
dbr	If signed by key in ***dbr***, loader can Run for recovery



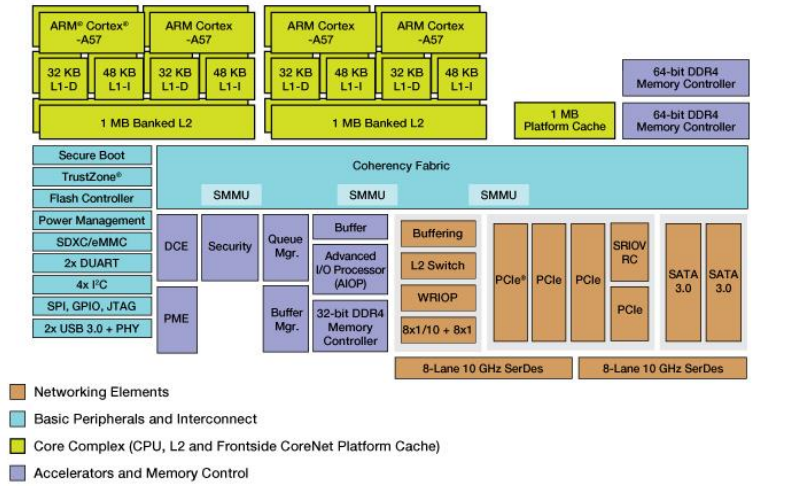

```
FS0:\> grubaa64.efiOpen '.' Success
FSOpen: Open '\grubaa64.efi' Success
FS0:\> Open '\grubaa64.efi' Success
FSOpen: Open '\grubaa64.efi' Success
FSOpen: Open '\grubaa64.efi' Success
The image doesn't pass verification: VenHw(837DCA9E-E874-4D82-B29A-23FE0E23D1E2,003E000A00000000)/HD(1,MBR,0x00000000,0x3F,0x21FC0)/\
rubaa64.efi
InstallProtocolInterface: 5B1B31A1-9562-11D2-8E3F-00A0C969723B 795E6C00
Loading driver at 0x000781A3000 EntryPoint=0x000781A3400
Loading driver at 0x000781A3000 EntryPoint=0x000781A3400
InstallProtocolInterface: BC62157E-3E33-4FEC-9920-2D3B36D750DF 795E0D18
Unloading driver at 0x000781A3000
Command Error Status: Security Violation
```

UEFI ON QORIQ LS SERIES SOCs

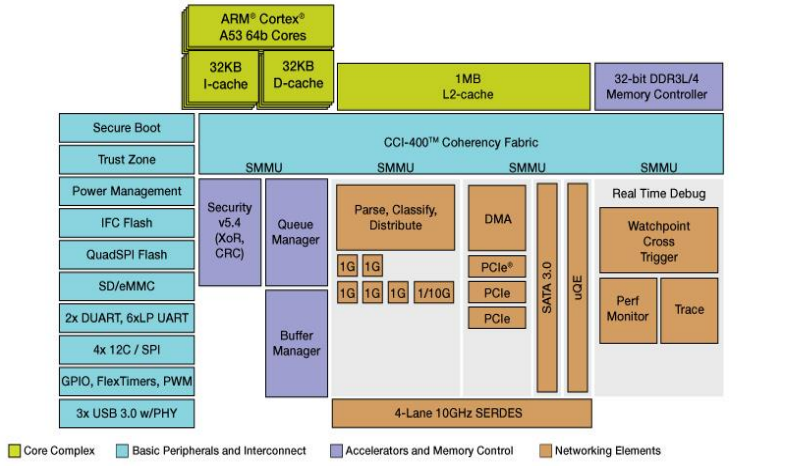
QorIQ LS Series Processors Based on ARM Technology



QorIQ LS2085A Processor Block Diagram

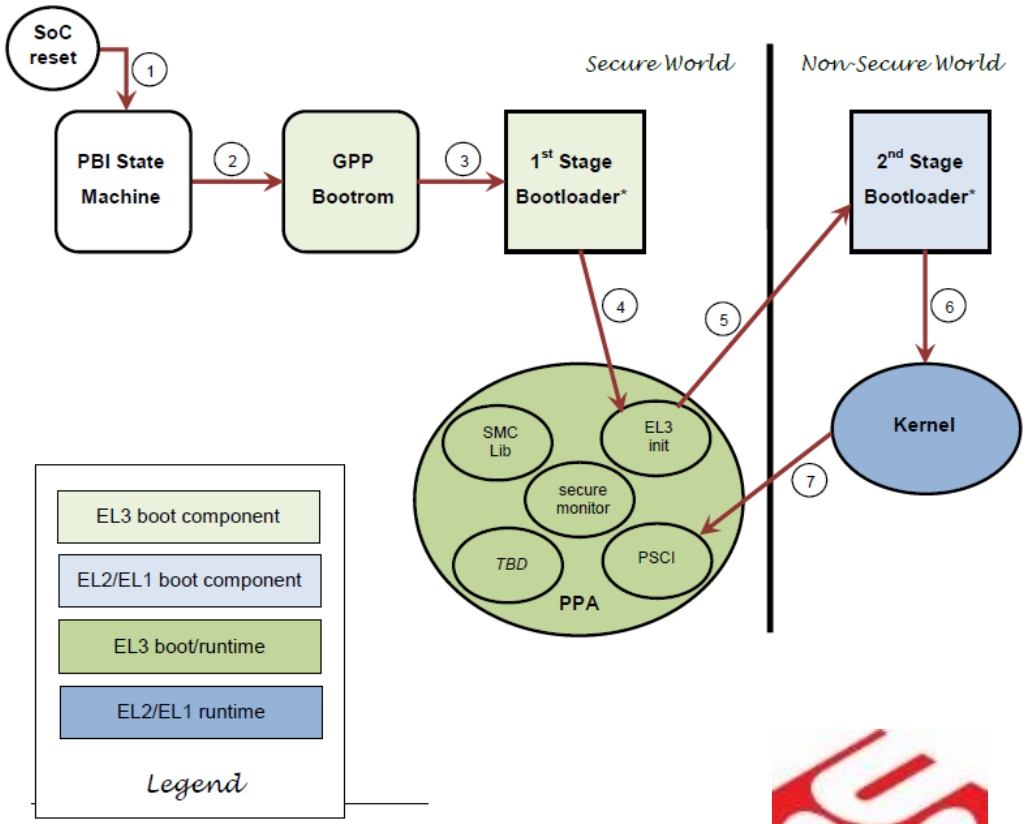


QorIQ LS1043A Processor Block Diagram



UEFI – Bootflow on a QorIQ LS Series SoCs

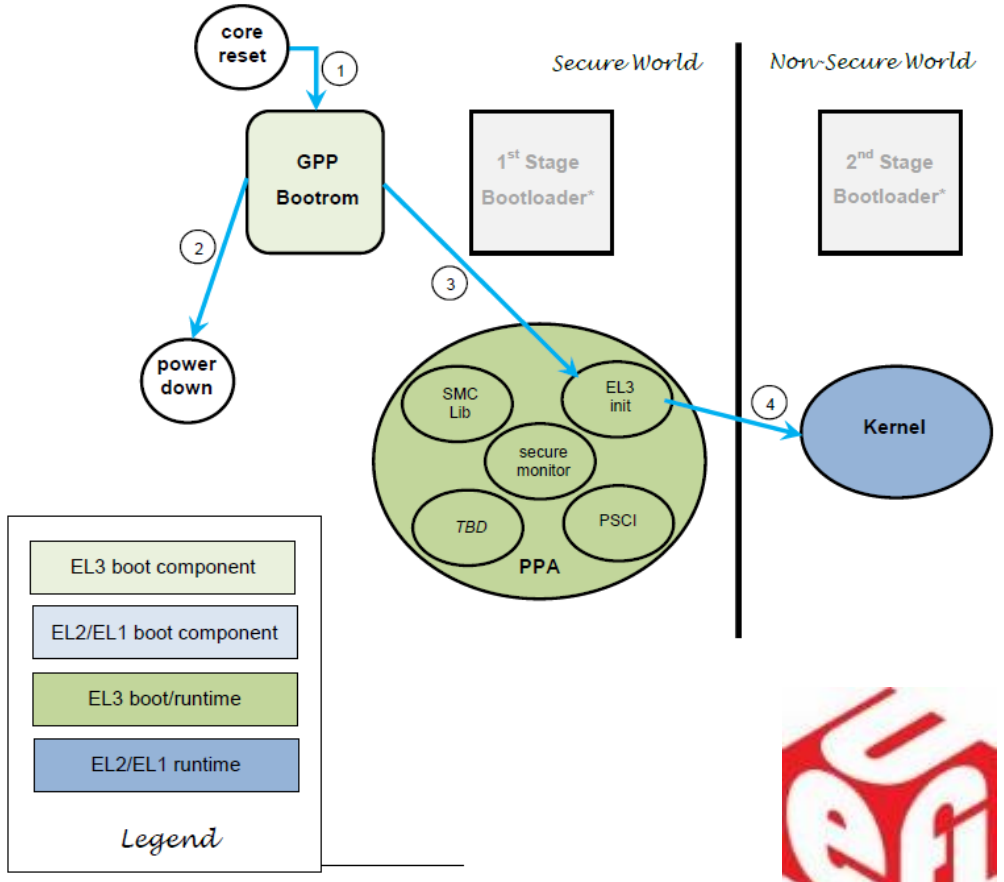
- Execution begins in the PBI State Machine.
- After PBI, execution starts with bootcore in GPP bootrom
- Bootcore branches to 1st stage bootloader running in EL3
- Bootcore in 1st stage bootloader branches to EL3 init code in PPA
- When bootcore completes EL3 init, it branches to 2nd stage bootloader in EL2
- Bootcore in 2nd stage bootloader branches to Linux kernel in EL2
- Kernel calls PSCI (cpu_on) to release secondary cores



UEFI – Bootflow on a QorIQ LS Series SoCs

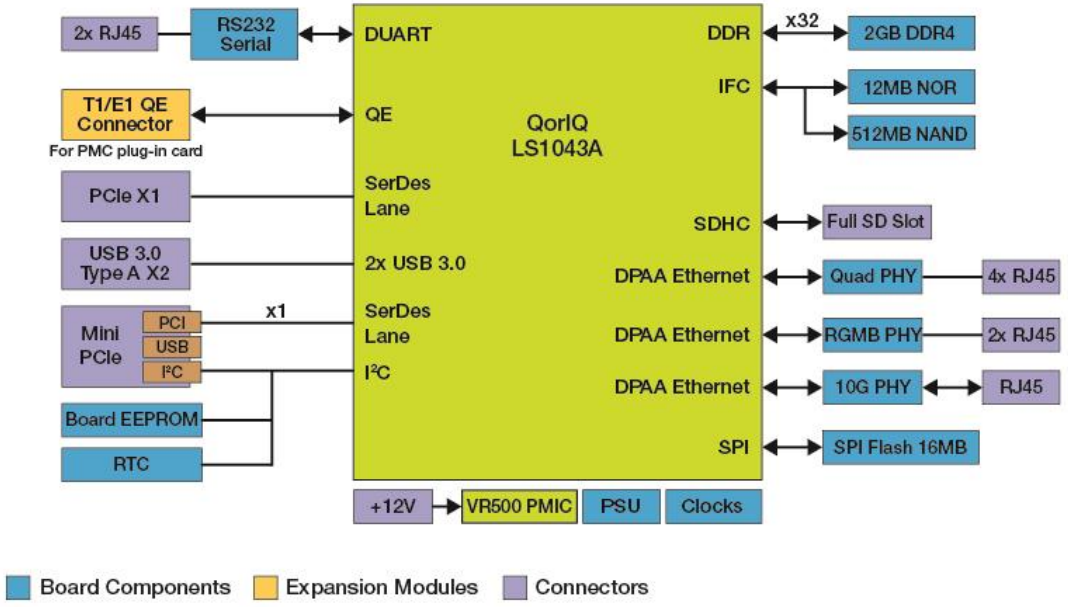
Secondary core Execution Path

1. Execution starts in the GPP bootrom when secondary core released from reset.
2. If core is marked to be disabled, core enters power-down sequence in bootrom.
3. Cores not disabled branch to EL3 init code in PPA.
4. Upon completion of EL3 init, cores branch to start address at EL2 in kernel

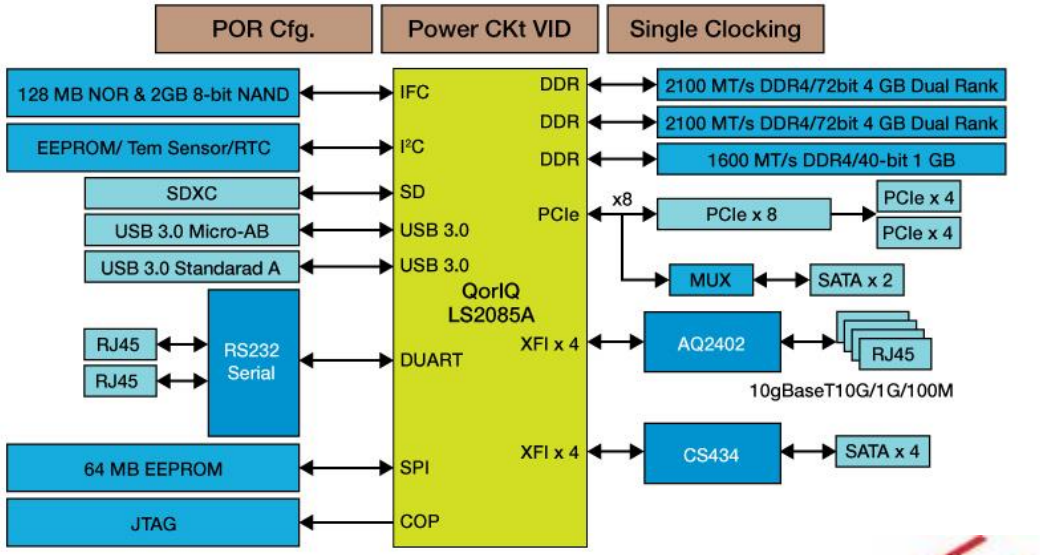


UEFI – Support available for LS1043A & LS2085A RDB Boards

QorIQ LS1043A RDB Block Diagram



QorIQ LS2085A RDB Block Diagram

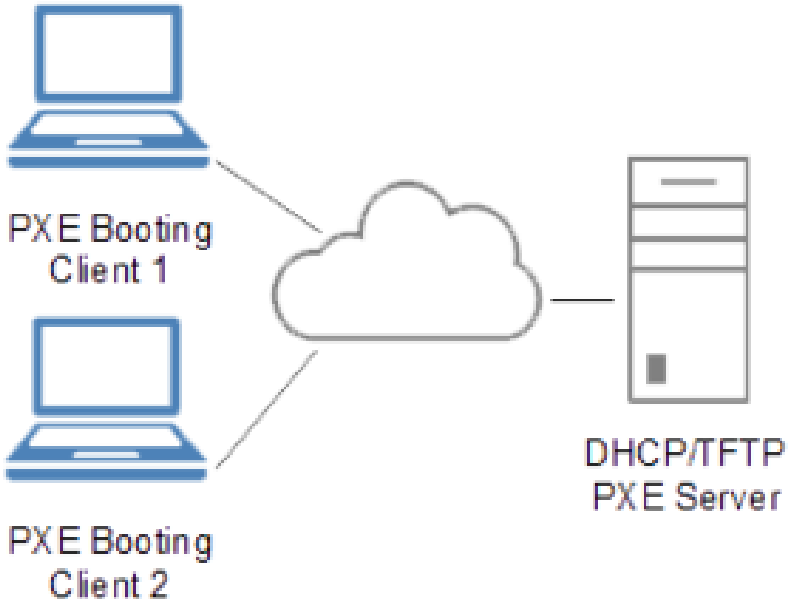


UEFI + GRUB2 + CENTOS DISTRO ON QORIQ LS SOCs

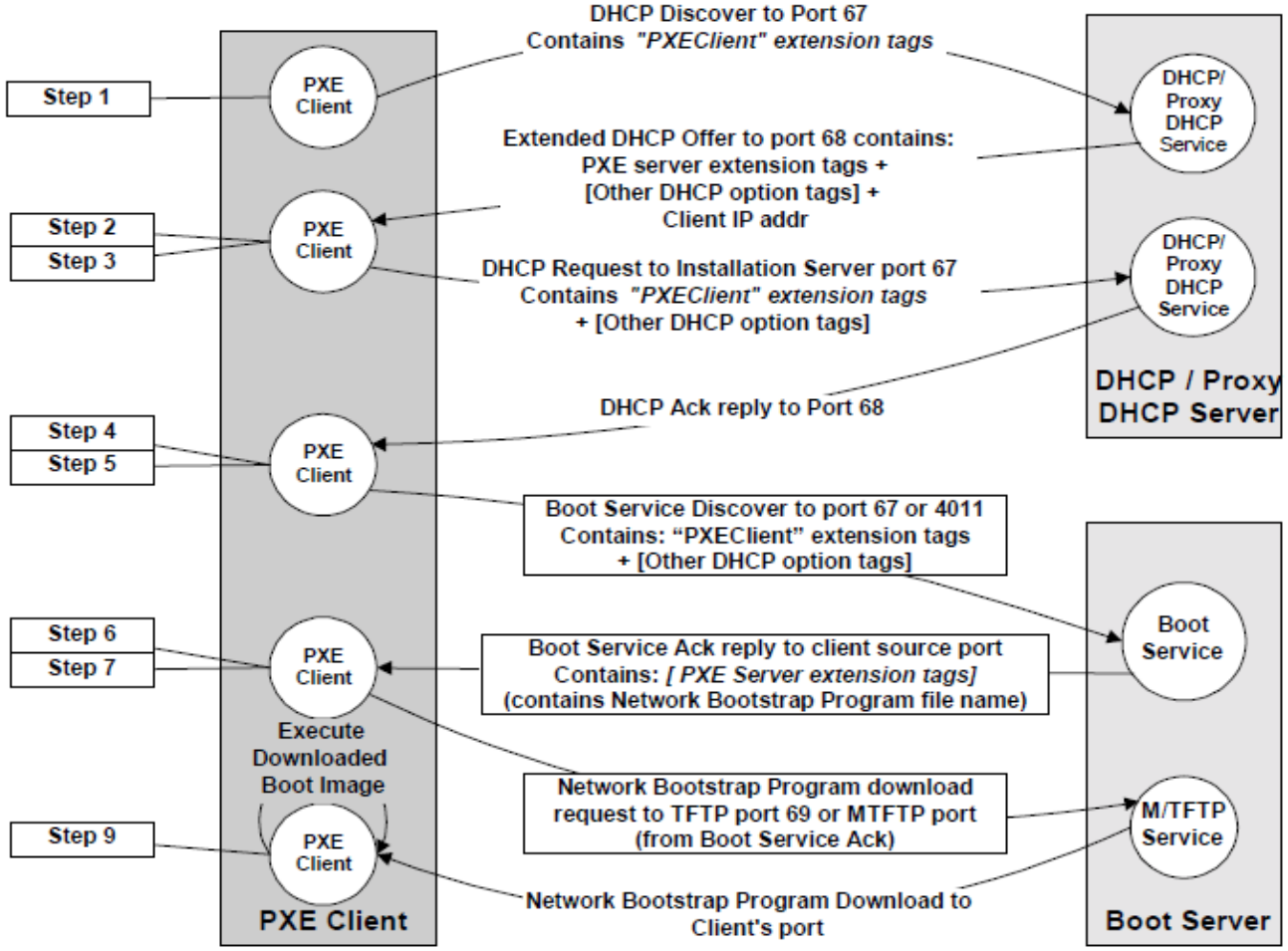
PXE Boot

Preboot eXecution Environment (PXE)

- Specification, which describes:
  - *client-server* environment that boots a software assembly, retrieved from a network, on PXE-enabled clients.
  - On the *client* side it requires only:
    - PXE-capable NIC, and
    - Network protocols such as DHCP and TFTP



PXE Boot – DHCP Messages

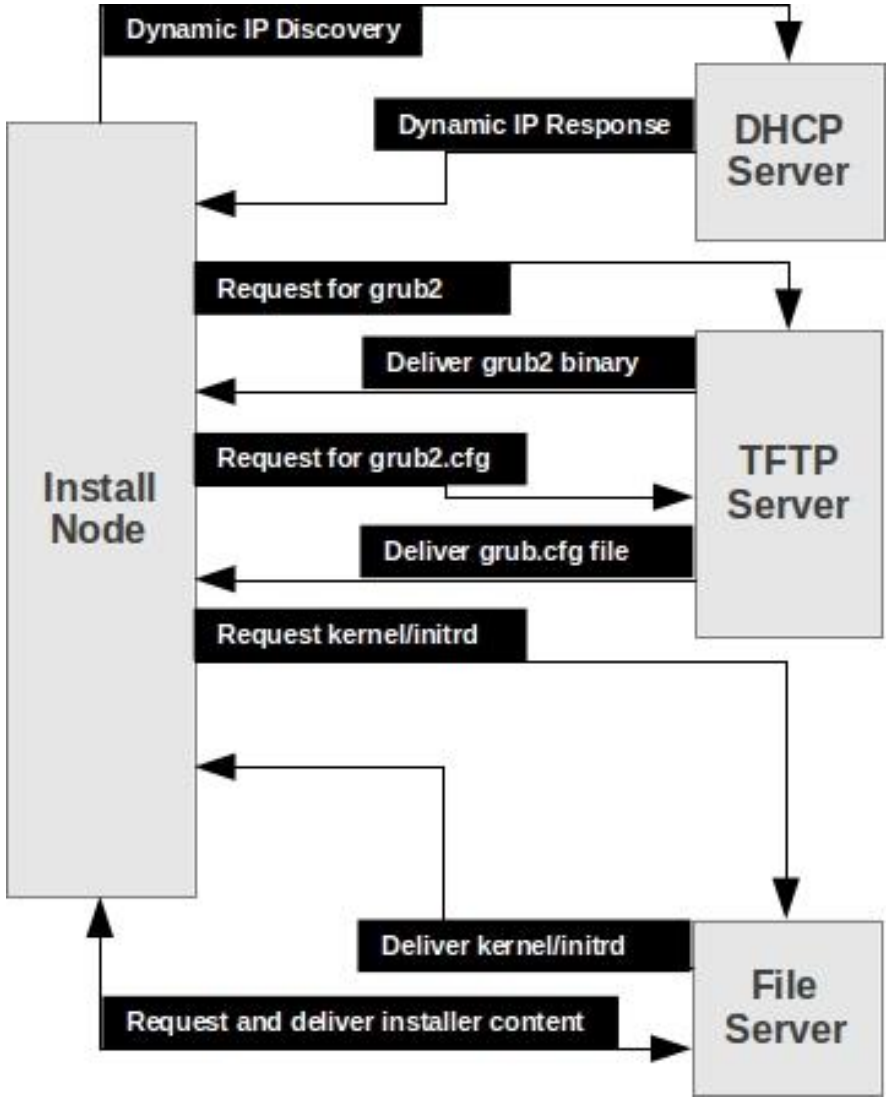


UEFI PXE Boot & GRUB2

- **GRUB2**
  - GRUB 2 is the default boot loader and manager for various OS's.
  - Allows *Dual-boot* (Windows + Linux) on a 64-bit UEFI based client.

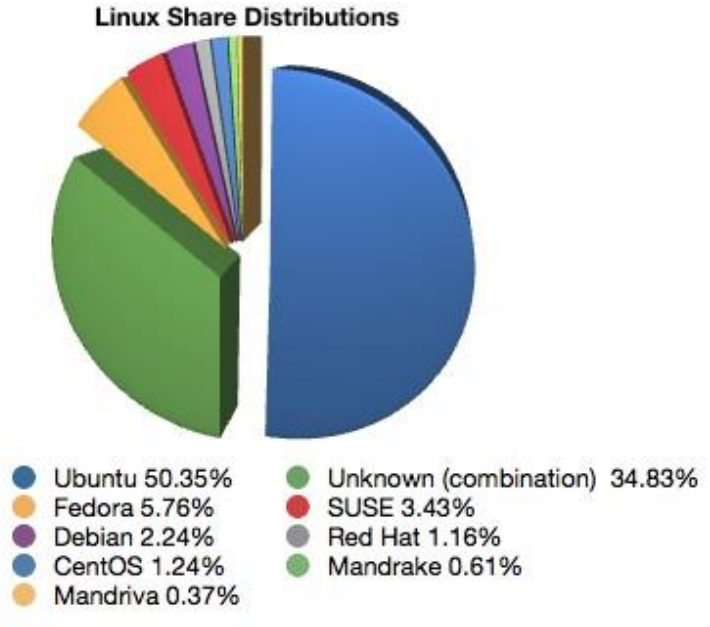
- **PXE + GRUB2**
  - When combined with PXE boot feature in UEFI, GRUB2 bootloader can be used to boot:
    - various OS (*Windows, Linux, MAC, Android)*, and
    - distributions (*CentOS, Ubuntu, Fedora, OpenSuse)*

    on a client.



UEFI + GRUB2 + CentOS 7 on QorIQ LS Series SoCs

- **CentOS 7 AARCH64 Distribution**
  - CentOS is rated amongst the top 10 distributions
  - Ranked as the ***Best Server OS*** along with Debian



- **UEFI + GRUB2 + CentOS 7 on QorIQ LS**
  - UEFI PXE boot can be used to chainload GRUB2 and CentOS 7 on QorIQ LS SoCs.

UEFI + GRUB2 + CentOS 7 on QorIQ LS Series SoCs

- **Setting up DHCP server for PXE boot**

Host Server: Ubuntu/Debian Linux

DHCP server: ISC DHCP Server

- Open ***/etc/dhcp/dhcpd.conf*** with write permissions on the server machine.
- Add a configuration block in the file, for PXE boot
- Place the *grub2.efi* file in the tftp server root directory
- Restart the DHCP server

```
host layerscape {

  hardware ethernet x:x:x:x:x:x; # MAC address*

  fixed-address x.x.x.x.; # IP address to be assigned

  option host-name "layerscape";

  next-server x.x.x.x; # IP address of the TFTP server

  filename "grubaa64.efi";

}
```

*Use the MAC address for which PXE boot entry was created.

```
$ sudo service isc-dhcp-server restart
```

UEFI + GRUB2 + CentOS 7 on QorIQ LS Series SoCs

- **Installing and Booting CentOS 7 on a LS2085A-RDB board**
  - Lets see a video.

DEMO VIDEO

QUESTIONS?

SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT