presented by

hp Linaro CAVIUM

Building ARM Servers With UEFI And ACPI

Dong Wei – HP

Roy Franz – Linaro/Cavium

LINUXCON NORTH AMERICA

August 22, 2014

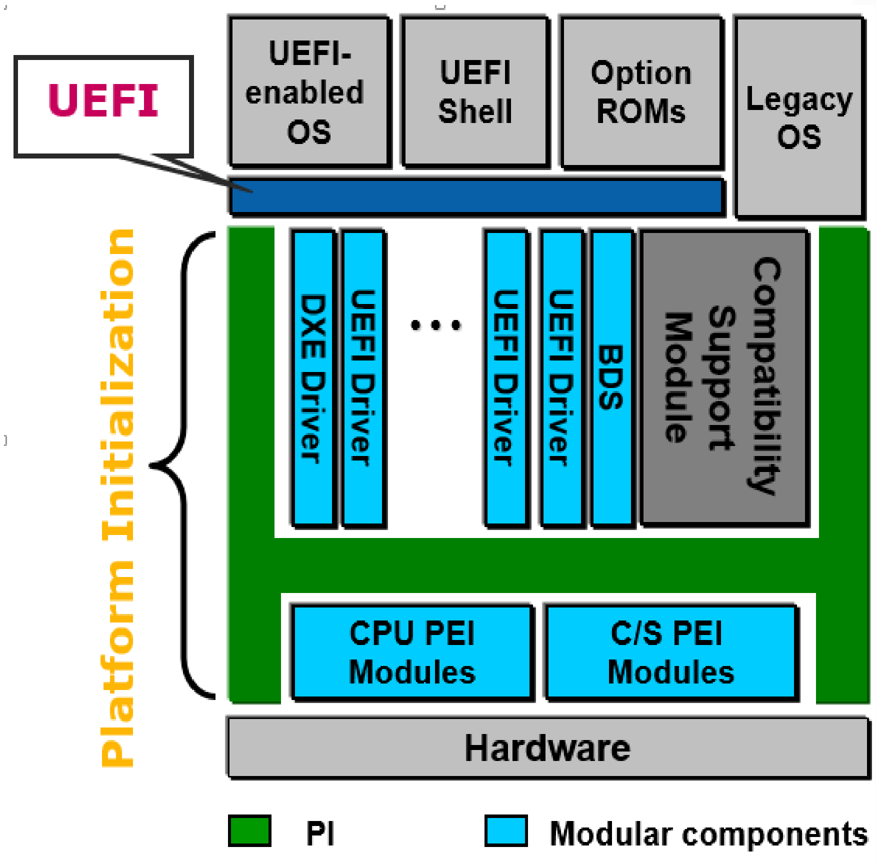
Agenda

- Quick Intro to UEFI/ACPI
- ARM Support in ACPI 5.1
- Linaro Enterprise Group
  - ARM server goals
  - Current status
  - Active work
  - Future work
- Questions?

UEFI Technology

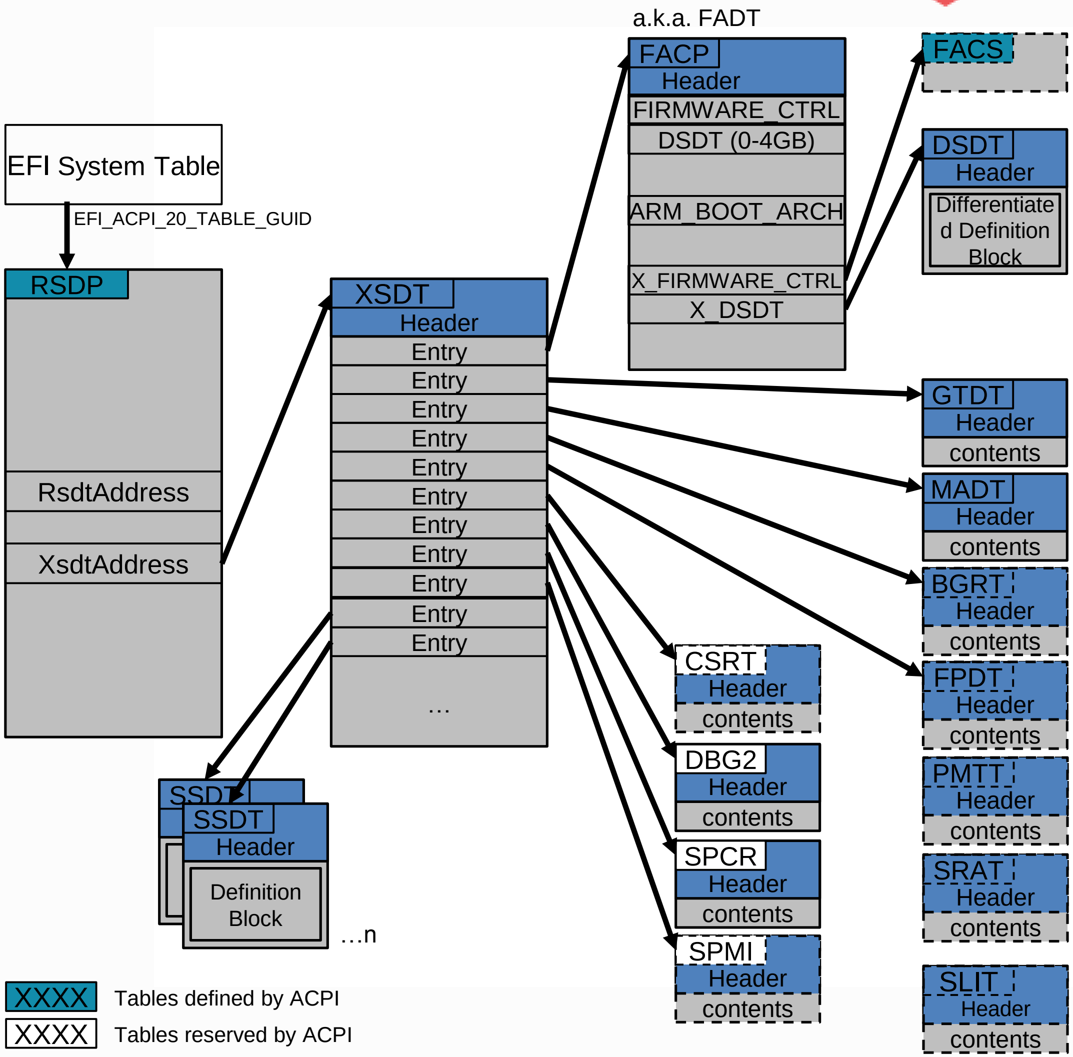
- Platform Initialization (PI)
  - Interfaces produced & consumed by firmware only; promote interoperability between firmware components

- UEFI
  - Pre-OS (and limited runtime program interfaces) between UEFI Applications (incl. OSes)/UEFI Drivers and system firmware



ACPI Technology

- Static tables and primary runtime interpreted control methods provided by system firmware to the OS for system configuration, power management and error handling
- Processor architecture agnostic
- Refer to SBBR for ARMv8 server ACPI requirements



UEFI & ACPI History

UEFI History

1995 HP/Intel needed a boot architecture for Itanium servers that overcame BIOS PC-AT limitations

1997 - 2000 Intel created EFI with HP and others in the industry, made it processor agnostic (x86, ia64)

2004 **tianocore.org**, open source EFI community launched

2005 **Unified EFI (UEFI)**
The UEFI Forum, with 11 promoters, was formed to standardize EFI, extended to x64

2009 UEFI extended to ARM AArch32

2012 Windows 8 and ubiquitous native UEFI adoption for client PCs (Boot Performance, Secure Boot focused)

2013 257 members and growing! Linux Distros extended support for UEFI Secure Boot. First Linux Foundation hosted UEFI Plugfest. UEFI v2.4 extended to ARM AArch64.

ACPI History

1996 Intel/Microsoft/Toshiba created ACPI 1.0 for 16 and 32 bit PC client devices

2000 Compaq/Intel/Microsoft/Phoenix/Toshiba publishes ACPI 2.0 for 64-bit support as well as support for multiprocessor workstations and servers

2004 HP/Intel/Microsoft/Phoenix/Toshiba published ACPI 3.0 further enhancing the spec to support both client and server systems

2009 ACPI 4.0 is published providing additional support for both client and server systems

2011 Hardware-reduced ACPI model was introduced into the published ACPI 5.0 spec to include the support for SoC devices. ARM specific descriptions are also introduced

2013 ACPI Asset transferred to the UEFI Forum. Ready for future ACPI.next development

2014 ACPI v5.1 for ARM AArch64 support
(e.g., ARM SBSA/SBBR servers)

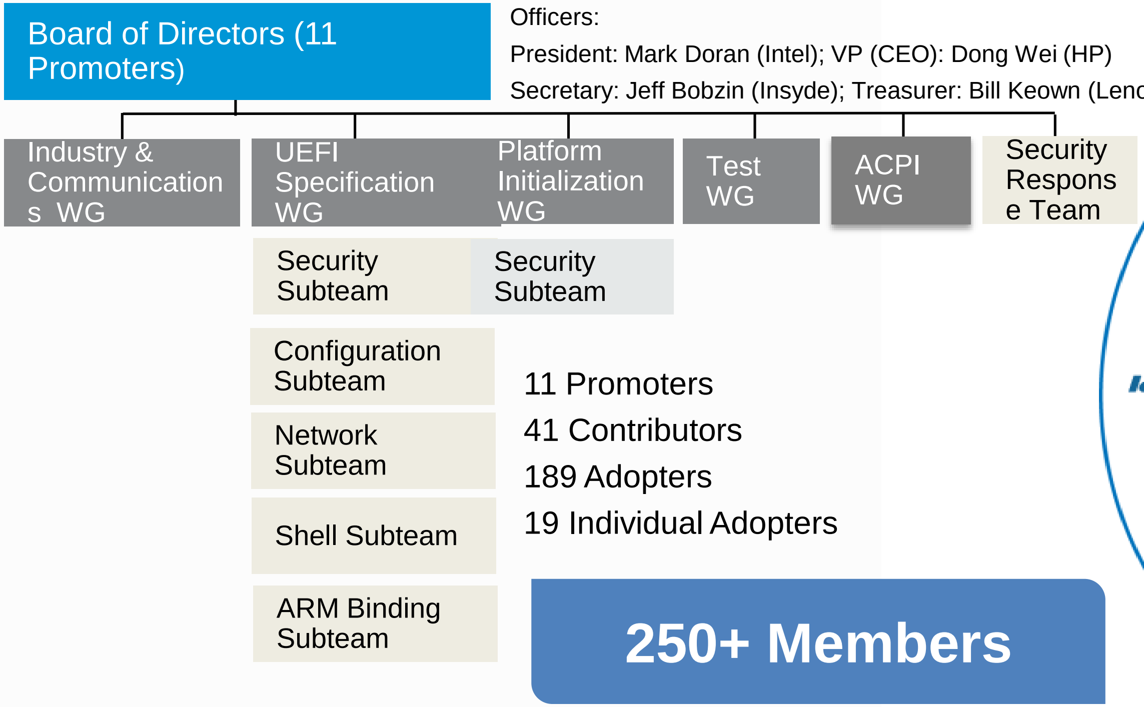
UEFI as the converged firmware infrastructure

The UEFI Forum Organization

Officers:

President: Mark Doran (Intel); VP (CEO): Dong Wei (HP)

Secretary: Jeff Bobzin (Insyde); Treasurer: Bill Keown (Lenovo)



Board of Directors (11 Promoters)

- Industry & Communications WG
- UEFI Specification WG
  - Security Subteam
  - Configuration Subteam
  - Network Subteam
  - Shell Subteam
  - ARM Binding Subteam
- Platform Initialization WG
  - Security Subteam
- Test WG
- ACPI WG
- Security Response Team

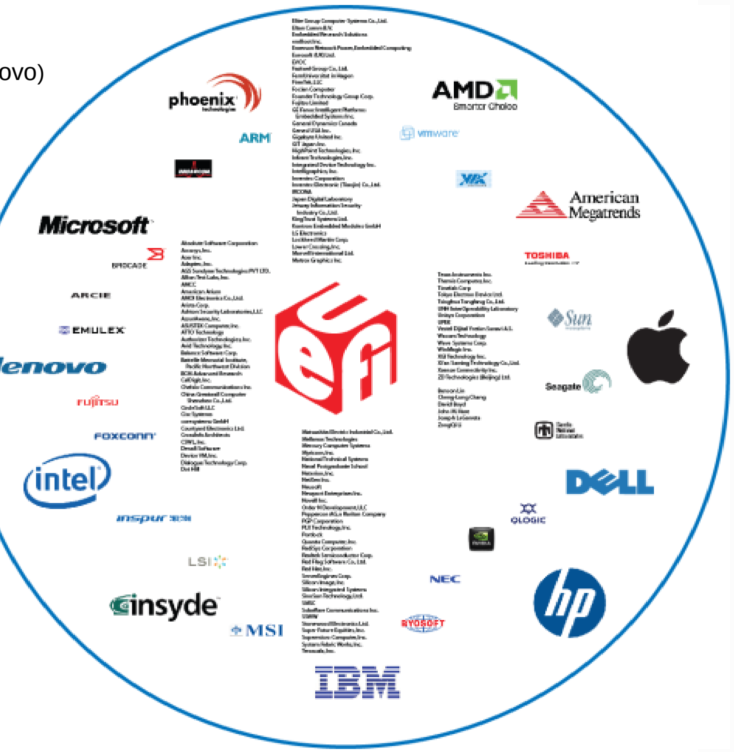
11 Promoters

41 Contributors

189 Adopters

19 Individual Adopters

250+ Members



AArch64 Binding In UEFI 2.4

- PE/COFF Machine Number
- UEFI Image File Name Convention
- Client System Architecture
- AArch64 Platform Conventions
- AArch64 Structure Definitions

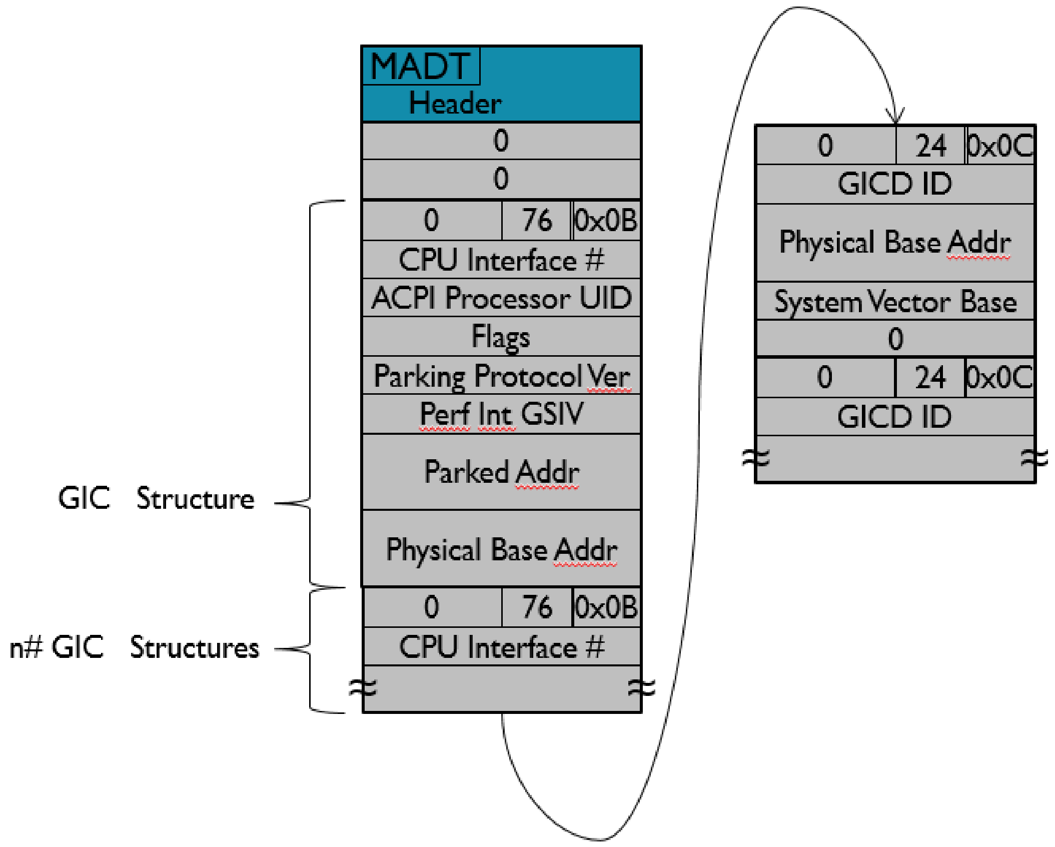
ACPI 5.1 Specification Status

- ACPI 5.1 adopted July 22; fixes many of the below problems
- ACPI 5.0 had a number of areas requiring improvements for ARM
  - ✓ ~~Missing virtualization, GICv2m, GICv3, GICv4 for alignment with SBSA~~
    - Improved GIC Architecture description and made compatible with SBSA Level I
  - ✓ ~~Missing PSCI support~~
  - ✓ ~~Missing platform/system memory mapped Generic Timer support for alignment with SBSA Level I~~
  - ✓ ~~Missing support for SBSA Level I Generic Watchdog Timer~~
  - ✓ ~~Missing strategy for clock management and other Device Specific Data features of an SoC~~
  - Missing support for SMMU or IO topology or GIC Interrupt Translation Service
  - Missing support for idle management and real support for core topology

ACPI 5.0: GIC

Missing support for alignment with SBSA

- Missing GIC Support for:
  - GICv2 virtualization
  - GICv2m (optionally required in SBSA Level 1)
  - GICv3
  - GICv4



MADT
Header
0
0

GIC Structure
0 | 76 | 0x0B
CPU Interface #
ACPI Processor UID
Flags
Parking Protocol Ver
Perf Int GSIV
Parked Addr
Physical Base Addr

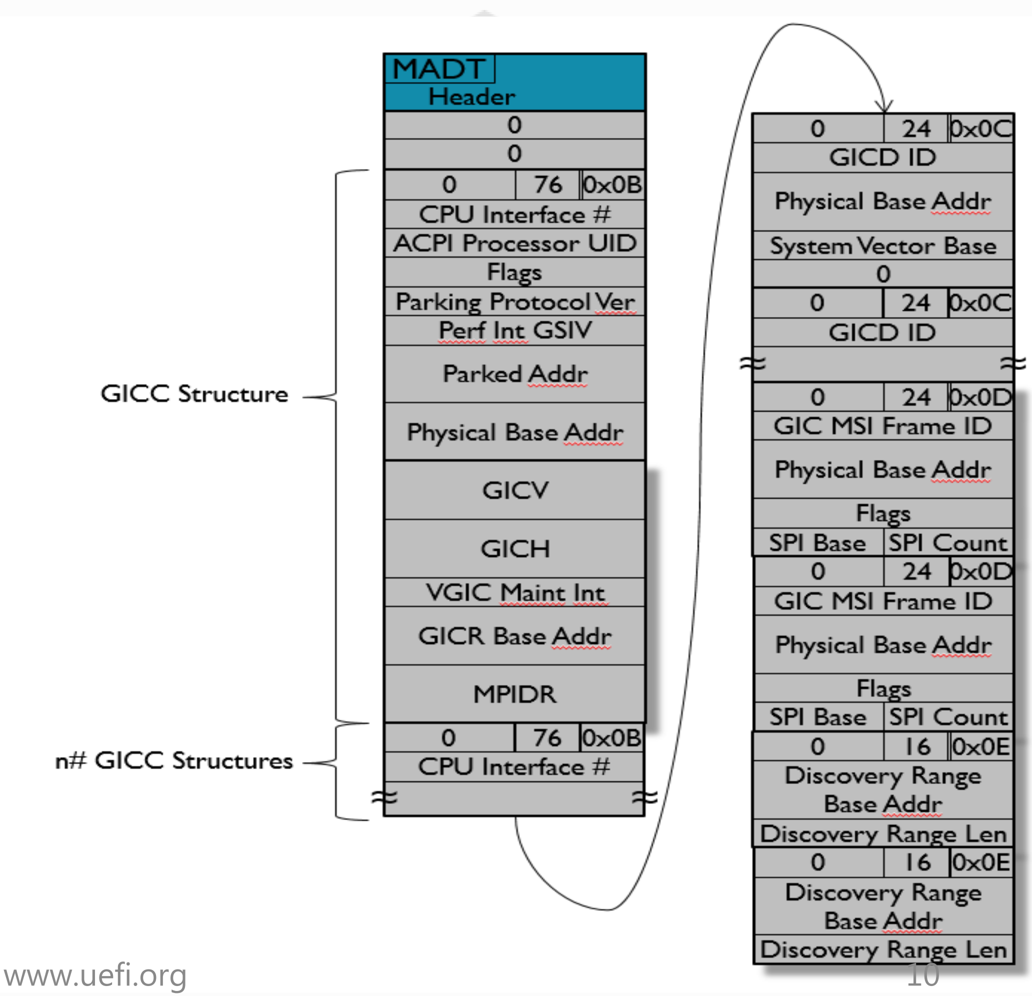
n# GIC Structures
0 | 76 | 0x0B
CPU Interface #

0 | 24 | 0x0C
GICD ID
Physical Base Addr
System Vector Base
0
0 | 24 | 0x0C
GICD ID

New Features In ACPI 5.1

Improved Generic Interrupt Controller Support

- GIC Support has been extended
  to cover:
  - GICv2 virtualization
  - GICv2m (optionally required in SBSA Level 1)
  - GICv3
    - Redistributors are supported
    - Interrupt Translation Service work in progress
  - Improved consistency with "ARM ARM" language
- Now called GICC and GICD structures of the MADT



MADT
Header
0
0
0 | 76 | 0x0B
CPU Interface #
ACPI Processor UID
Flags
Parking Protocol Ver
Perf Int GSIV
Parked Addr
Physical Base Addr
GICV
GICH
VGIC Maint Int
GICR Base Addr
MPIDR
0 | 76 | 0x0B
CPU Interface #
≈ ≈

GICC Structure

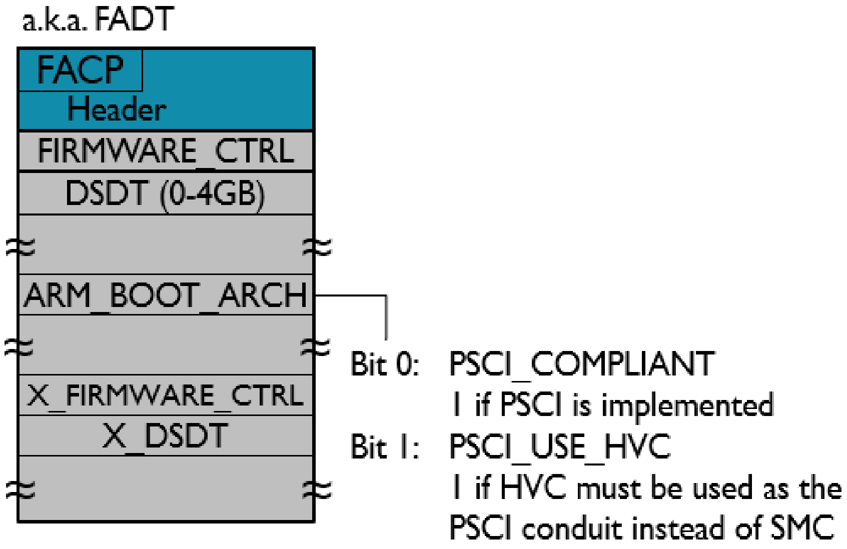
n# GICC Structures

0 | 24 | 0x0C
GICD ID
Physical Base Addr
System Vector Base
0
0 | 24 | 0x0C
GICD ID
≈ ≈
0 | 24 | 0x0D
GIC MSI Frame ID
Physical Base Addr
Flags
SPI Base | SPI Count
0 | 24 | 0x0D
GIC MSI Frame ID
Physical Base Addr
Flags
SPI Base | SPI Count
0 | 16 | 0x0E
Discovery Range Base Addr
Discovery Range Len
0 | 16 | 0x0E
Discovery Range Base Addr
Discovery Range Len

New Features In ACPI 5.1

PSCI Support

- PSCI discoverability is provided by a new ARM Boot Flags field in FADT
- MADT provides ways of identifying every core
  - Enables the use of PSCI for:
    - Secondary core boot
    - Dynamic addition/removal of cores (hotplug)
  - Creates a path for use in idle management



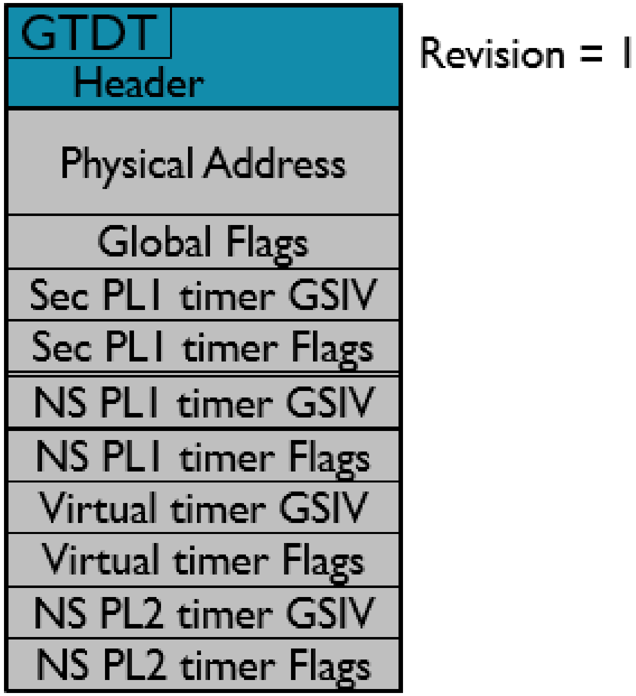
Bit 0: PSCI_COMPLIANT
1 if PSCI is implemented

Bit 1: PSCI_USE_HVC
1 if HVC must be used as the PSCI conduit instead of SMC

ACPI 5.0: Generic Timer

Limited Support for
Generic Timer Architecture

- GTDT narrowly described timers
  that were implemented at the
  time and cannot describe:
  - Always-on per processor timers
  - Memory-mapped platform timers
  - Platform watchdog timers



GTDT
Header

Physical Address

Global Flags

Sec PL1 timer GSIV

Sec PL1 timer Flags

NS PL1 timer GSIV

NS PL1 timer Flags

Virtual timer GSIV

Virtual timer Flags

NS PL2 timer GSIV

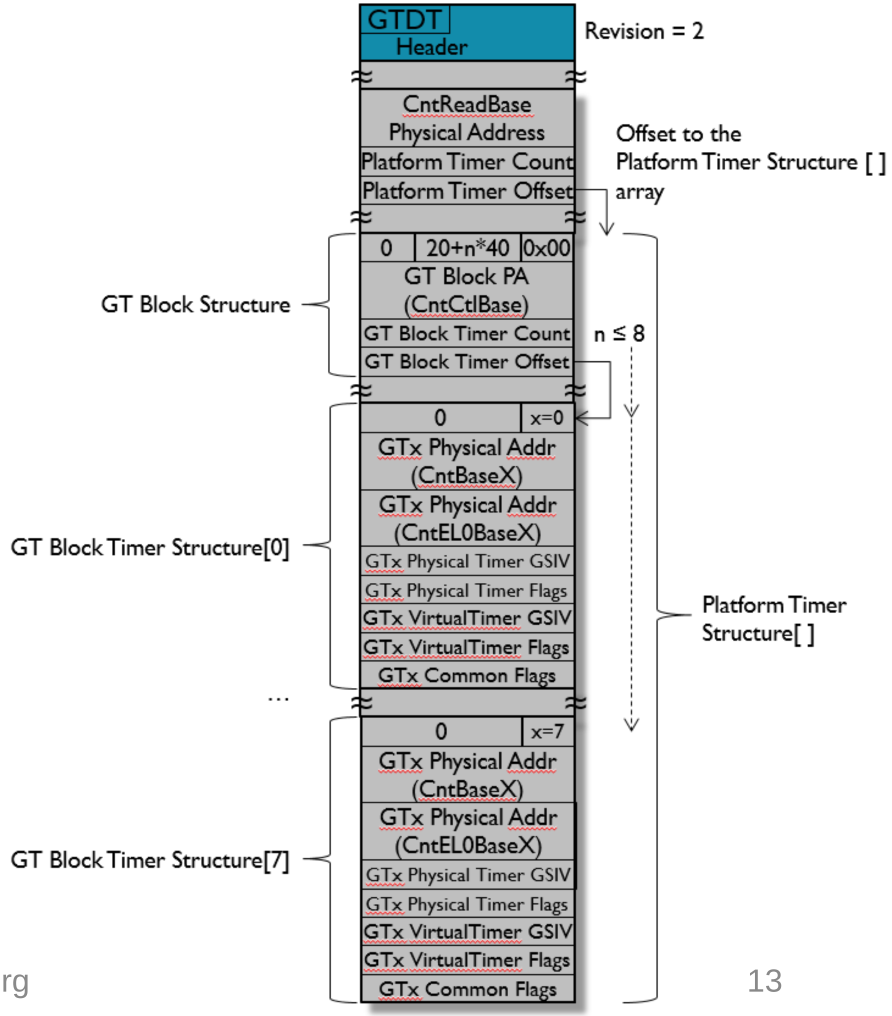
NS PL2 timer Flags

Revision = 1

New Features In ACPI 5.1

Extended Support For Generic Timer Architecture

- It is now possible to describe platform memory mapped timers that are compliant with the ARMv7 or ARMv8 Generic Timer Architecture
  - Covered by extension to the GTDT
    table in the Platform Timer Structure []
  - Secure or non-secure via GTx Common Flags
  - Always-on Capability via GTx Common Flags
- This is a requirement for SBSA Level I systems



GTDT Header

Revision = 2

CntReadBase Physical Address

Platform Timer Count

Platform Timer Offset

Offset to the Platform Timer Structure [ ] array

GT Block Structure

0 | 20+n*40 | 0x00

GT Block PA (CntCtlBase)

GT Block Timer Count

GT Block Timer Offset

n ≤ 8

GT Block Timer Structure[0]

0 | x=0

GTx Physical Addr (CntBaseX)

GTx Physical Addr (CntEL0BaseX)

GTx Physical Timer GSIV

GTx Physical Timer Flags

GTx VirtualTimer GSIV

GTx VirtualTimer Flags

GTx Common Flags

...

Platform Timer Structure[ ]

GT Block Timer Structure[7]

0 | x=7

GTx Physical Addr (CntBaseX)

GTx Physical Addr (CntEL0BaseX)

GTx Physical Timer GSIV

GTx Physical Timer Flags

GTx VirtualTimer GSIV

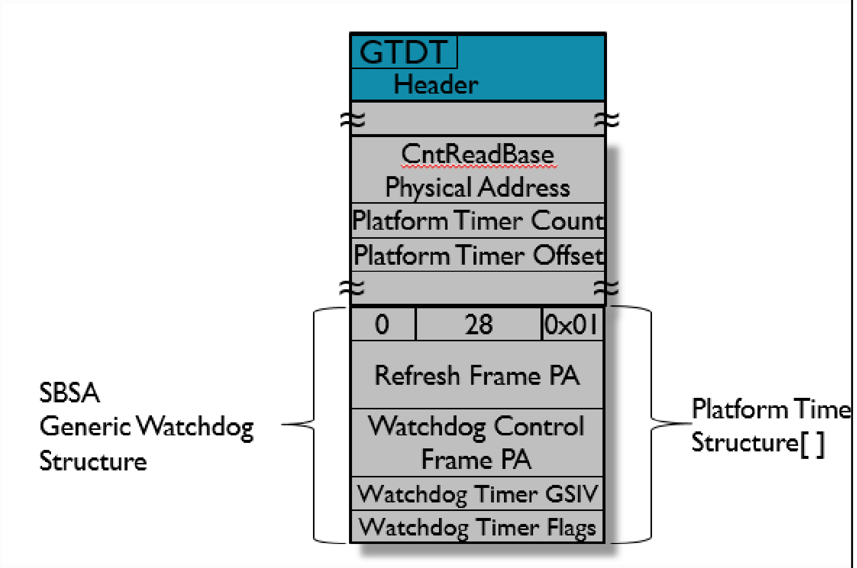
GTx VirtualTimer Flags

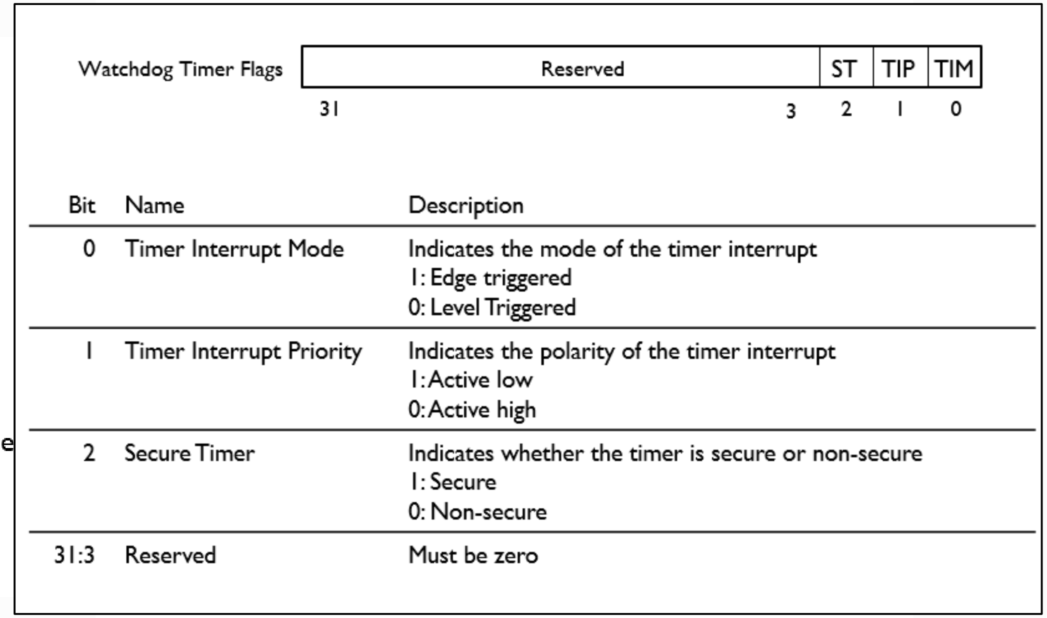
GTx Common Flags

New Features In ACPI 5.1

Support For SBSA Level I Generic Watchdog Timer

- It is now possible to describe platform SBSA Level I Generic Watchdog Timer
  - Covered by extension to the GTDT table in the Platform Timer Structure []





Watchdog Timer Flags	Reserved	ST	TIP	TIM
	31 3	2	1	0

Bit	Name	Description
0	Timer Interrupt Mode	Indicates the mode of the timer interrupt 1: Edge triggered 0: Level Triggered
1	Timer Interrupt Priority	Indicates the polarity of the timer interrupt 1: Active low 0: Active high
2	Secure Timer	Indicates whether the timer is secure or non-secure 1: Secure 0: Non-secure
31:3	Reserved	Must be zero

Optional Features In ACPI 5.1

Device Specific Data (_DSD)

- An optional object used to describe device properties to device drivers
- _DSD returns a variable-length package of Device Data Descriptor structures
  - UUID and Data Structure tuples
- UUIDs may be created by governing bodies (e.g. PCI SIG, UEFI Forum), OEMs or hardware vendors
- UUID and data structures pairs are published via http://www.uefi.org/acpi
- This method will help us provide more generic solutions in clock control or other bespoke features.

Optional Features In ACPI 5.1

Cache Coherency Attribute (_CCA)

- A device identification object specifies whether a device and its descendants support hardware managed cache coherency
- _CCA returns
  - 0: the device does not have hardware managed cache coherency
    - Software managed to ensure stale or invalid data is not accessed from the caches
  - 1: the device has hardware managed cache coherency
- Allows platform designers to provide hardware cache coherency support on an as-needed basis for cost and performance reasons, without requiring new drivers to have knowledge of the platform
- Provides flexibility in the firmware to indicate to the OS what support is provided in the platform

New Key Features In ACPI 5.1

- ACPI 5.1 draft started in mid-Jan. 2014
  - UEFI Forum official IP review period for final draft of the specification opened on June 13, 2014
  - Publication July 22, 2014
  - Press release August 12, 2014
- Covered new key and optional features for ARM
  - GIC
  - PSCI
  - Generic Timer Architecture & SBSA Generic Watchdog Timer
  - Device Specific Data
  - Cache Coherency Attribute
- Other new deltas mentioned in the ACPI 5.1 specification revision history…
- ACPI 5.1 specification is available for download here http://uefi.org/specifications

Quote

“UEFI and ACPI are the prevalent open standards enabling flexible device configuration and power-state management in server class systems. The ARMv8 server ecosystem is aligning around open industry standards with UEFI emerging as the boot model and ACPI as the runtime interfaces.”

Lakshmi Mandyam
Director of Server Systems and Ecosystems
ARM

Linaro Enterprise Group (LEG)

- Formed within Linaro in late 2012 to collaborate on and accelerate the development of foundational software for ARM server Linux
- UEFI and ACPI enablement is crucial
  - LEG is Working with many projects
- Other LEG server enablement work
  - OpenJDK, LAMP, Virtualization, +more
- ARM Ltd., many others also working on UEFI/ACPI
- Upstream is the goal
- Monthly binary/source releases of most projects
  - Includes in-progress features

ARM Server Goals

- Standards based servers to facilitate interoperability
  - Multiple Linux distros installable
  - Single kernel supports multiple platforms
  - New hardware supported by existing software
- Should work like existing servers

UEFI/ACPI As Key Enablers

- UEFI provides standard boot architecture
  - In widespread use by distros and users
  - Enables installers, network boot and more in standard and familiar way
- ACPI provides stable HW description
  - Stability over time
  - Defined process for changes
- Both new for ARM platforms
- ARM servers have no legacy
  - This freedom must be used wisely.

Completed UEFI Work

- EDK2 running on supported platforms
- Linux EFI loader/runtime services
  - 3.16 for arm64
  - 3.18 likely for arm32
- arm64 EFI GRUB support
- FDT for non-ACPI systems

Completed ACPI Work

- ACPICA support for 5.1 FADT, MADT, and GTDT released in version 20140727
- Linux support for ACPI 5.1 on arm64 likely in v3.18
- Test suites integrated into LAVA
  - ACPI ASL: ACPI tool tests
  - FWTS: Canonical’s Firmware Test Suite
  - ACPI API: Linux user->kernel API tests

Supported Platforms

- ARM Juno evaluation board
- ARM FVP/Foundation models
- Applied Micro Mustang (in progress)
- Non-ACPI platforms(32 bit):
  - Huawei D01 (16 core A15 server)
  - ARM VExpress evaluation boards
  - QEMU VExpress system model

Active Work

- XEN arm64 UEFI/ACPI support
- ASWG work for arm64 features not yet described by ACPI
- Ongoing Linux arm64 ACPI support
- GRUB EFI XEN multi-boot support
- UEFI networking support in FVP models and QEMU
- UEFI Secure Boot
- UEFI as virtual machine firmware

Next Steps

- Continue to produce ACPI Errata and Feature ECRs for ASWG
- Continued Linux kernel improvements
  - New platforms becoming available
  - New ACPI features/errata
- UEFI secure variable storage
- Continued UEFI test coverage improvement

Ongoing UEFI Enablement

- Linaro is encouraging collaboration on platforms and drivers
  - Mainline EDK2 only for arch code, reference platforms
  - Linaro EDK2 tree accepting platforms and drivers
- UEFI beyond servers
  - GSoC project for BeagleBone Black support
  - Android on UEFI
    - Fastboot protocol

ACPI Reference

- Linaro ACPI repository
  - https://git.linaro.org/leg/acpi/acpi.git
- Linaro ACPI mailing list
  - linaro-acpi@lists.linaro.org
- Wiki
  - https://wiki.linaro.org/LEG/Engineering/Kernel/ACPI

UEFI Reference

- Linaro EDK2 repository
  - git://git.linaro.org/uefi/linaro-edk2.git
- Linaro UEFI mailing list
  - linaro-uefi@lists.linaro.org
- Linaro releases
  - http://www.linaro.org/downloads/
- Wiki
  - https://wiki.linaro.org/LEG/Engineering/Kernel/UEFI

Linaro Reference

- Main Linaro page
  - http://www.linaro.org/
- LEG wiki main page:
  - https://wiki.linaro.org/LEG
- Downloads
  - http://www.linaro.org/downloads/
  - Monthly releases, source and binary for UEFI, Linux, and toolchains

For more information on the Unified EFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by