RISC-V UEFI BOOT PROTOCOL Specification

Sunil V L, RISC-V Platform HSC Group

Version 0.1, 1/10/2022: Draft

Table of Contents

Chapter 1. Introduction

Either Device Tree (DT) or Advanced Configuration and Power Interface (ACPI) firmware tables are used to convey the information about hardware to the Operating Systems. Some of the information are known only at boot time and needed very early before the Operating Systems/boot loaders can parse the firmware tables.

One example is the boot hartid on RISC-V systems. On non-UEFI systems, this is typically passed as an argument to the kernel (in a0). However, UEFI systems need to follow UEFI application calling conventions and hence it can not be passed in a0. While there can be a solution using /chosen node in DT based systems to pass this information, a simple and common interface across DT and ACPI platforms is desired on UEFI platforms to retrieve such information.

This specification introduces a new UEFI protocol for RISC-V systems which provides early information to the bootloaders or Operating Systems. Firmwares like EDK2/u-boot need to implement this protocol on RISC-V UEFI systems.

This protocol is typically used by the bootloaders before **ExitBootServices()** call and pass the information to the Operating Systems.

Chapter 2. EFI_RISCV_BOOT_PROTOCOL

EFI_RISCV_BOOT_PROTOCOL provides the interface for bootloaders / Operating Systems to get certain information prior to parsing the firmware tables like ACPI.

The version of EFI_RISCV_BOOT_PROTOCOL specified by this specification is 0x00010000. All future revisions must be backwards compatible. If a new version of the specification breaks backwards compatibility, a new GUID must be defined.

2.1. GUID

```
#define EFI_RISCV_BOOT_PROTOCOL_GUID \
    { 0xccd15fec, 0x6f73, 0x4eec, \
    { 0x83, 0x95, 0x3e, 0x69, 0xe4, 0xb9, 0x40, 0xbf } }
```

2.2. Protocol Interface Structure

```
typedef struct _EFI_RISCV_BOOT_PROTOCOL {
    EFI_GET_PROTOCOL_VERSION  GetProtocolVersion;
    EFI_GET_BOOT_HARTID       GetBootHartId;
} EFI_RISCV_BOOT_PROTOCOL;
```

2.2.1. EFI_RISCV_BOOT_PROTOCOL.GetProtocolVersion

It provides the version of the EFI_RISCV_BOOT_PROTOCOL implemented by the fimrware.

The version of the EFI_RISCV_BOOT_PROTOCOL. The version specified by this specification is 0x00010000. All future revisions must be backwards compatible. If a new version of the specification breaks backwards compatibility, a new GUID must be defined.

Prototype

```
typedef EFI_STATUS
(EFIAPI *EFI_GET_PROTOCOL_VERSION) (
    IN EFI_RISCV_BOOT_PROTOCOL *This,
    OUT UINT32                 *Version
    );
```

Parameters

Table 1. GetProtocolVersion Parameters

Parameter	Description
This	Pointer to the protocol

Parameter	Description
Version	Pointer to the variable receiving the version of the protocol implemented by the firmware.

Status Codes Returned

Table 2. GetProtocolVersion Return Value

Return Value	Description
EFI_SUCCESS	The protocol version could be returned.
EFI_INVALID_PARAMETER	**This** parameter is NULL or does not point to a valid EFI_RISCV_BOOT_PROTOCOL implementation.
EFI_INVALID_PARAMETER	**Version** parameter is NULL.

2.2.2. EFI_RISCV_BOOT_PROTOCOL.GetBootHartId

This interface provides the hartid of the boot cpu.

Prototype

```
typedef EFI_STATUS
(EFIAPI *EFI_GET_BOOT_HARTID) (
    IN EFI_RISCV_BOOT_PROTOCOL *This,
    OUT UINT32                 *BootHartId
    );
```

Parameters

Table 3. GetBootHartId Parameters

Parameter	Description
This	Pointer to the protocol
BootHartId	Pointer to the variable receiving the hartid of the boot cpu.

Status Codes Returned

Table 4. GetBootHartId Return Value

Return Value	Description
EFI_SUCCESS	The boot hart id could be returned.
EFI_INVALID_PARAMETER	**This** parameter is NULL or does not point to a valid EFI_RISCV_BOOT_PROTOCOL implementation.
EFI_INVALID_PARAMETER	**BootHartId** parameter is NULL.

References

- Discussion on the requirement