INTEL CORPORATION

Simple Firmware Interface

Draft 0.5

Len Brown

10/21/2008

Simple Firmware Interface

DRAFT 0.5
10/21/2008

Len Brown
len.brown@intel.com
Intel Software and Services Group – Open Source Technology Center

Introduction

This document defines the Simple Firmware Interface (SFI). Platform firmware prepares SFI tables upon system initialization for the benefit of the Operating System (OS). The OS consults the SFI tables to augment platform knowledge that it gleans from native hardware interfaces, such as CPUID and PCI.

The goal of SFI is to tell the OS what it needs to know using minimal firmware and OS overhead. SFI may be used in lieu of ACPI and UEFI on some modern platforms.

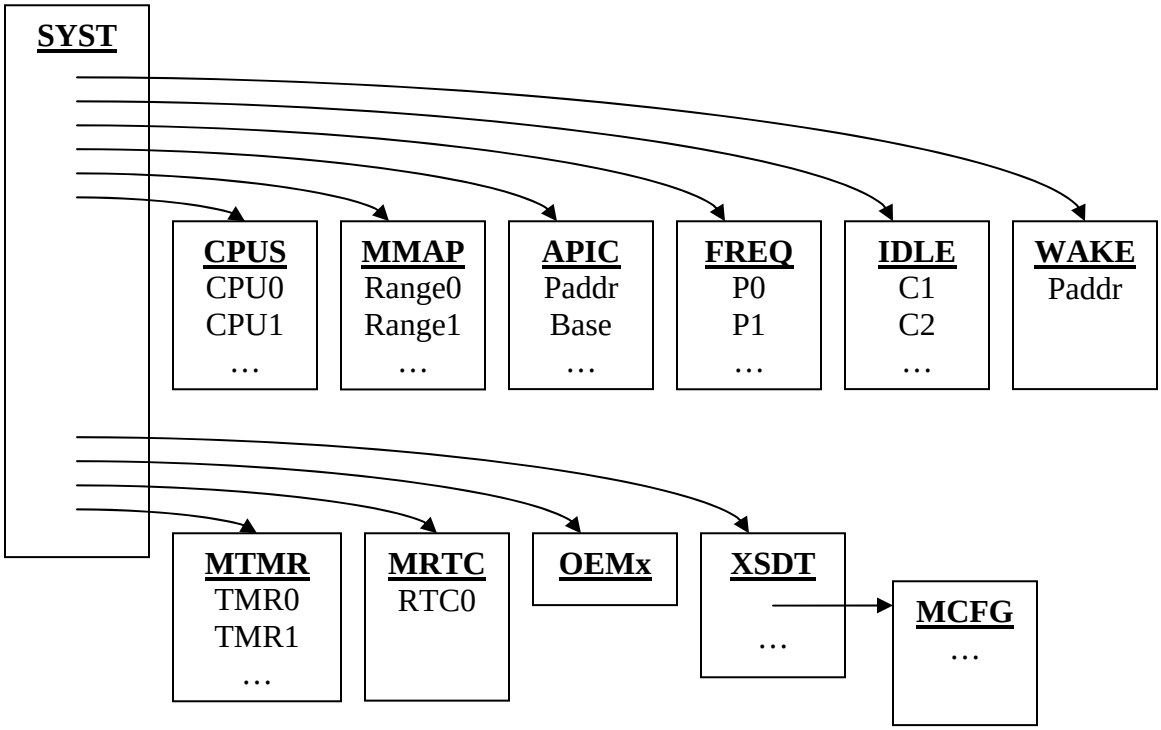

Figure 1 SFI Table Structure

The remainder of this document specifies the format of the SFI tables. At the end, there are also sections describing SFI's relationship to ACPI and UEFI.

SFI Common Table Header Format

All SFI tables have the common header (DESCRIPTION_HEADER), shown here. This format is a sub-set of the ACPI static table header format[1].

Field	Byte Length	Byte Offset	Description
Signature	4	0	The ASCII string representation of the table identifier. The OS ignores tables with signatures that it does not recognize even though the values in the Length and Checksum fields are correct.
Length	4	4	The length of the table, in bytes, including the header, starting from offset 0. This field records the size of the entire table.
Revision	1	8	The revision of the structure corresponding to the signature field for this table. Larger revision numbers are backward compatible to lower revision numbers with the same signature.
Checksum	1	9	The entire table, including the checksum field, must add to zero to be considered valid.
OEMID	6	10	An OEM-supplied string that identifies the OEM.
OEM Table ID	8	16	An OEM-supplied string that the OEM uses to identify the particular data table.

SFI System Table

The SFI System table consists of an SFI header followed by an array of entries. Each entry is simply a 64-bit physical address. Each address points to a table with an SFI common header.

Field	Byte Length	Byte Offset	Description
Signature	4	0	'SYST' table signature.
Length	4	4	Length, in bytes, of the entire table. The length implies the number of Entry fields (*n*) at the end of the table.
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID string
OEM Table ID	8	16	OEM Table ID string
Entry	8**n*	24	An array of 64-bit physical addresses that point to other DESCRIPTION_HEADERs. OSPM assumes at least the DESCRIPTION_HEADER is addressable, and then can further address the table based upon its Length field.

[1] SFI deleted 3 fields from the end of the ACPI common header – OEM Revision, Creator ID, and Creator Revision.

The OS finds the System Table by searching 16-byte boundaries between physical address 0x000E0000 and 0x000FFFFF. The OS shall search this region starting at the low address and shall stop searching when the 1st valid SFI System Table is found.

SFI CPU Table

The optional SFI CPU table enumerates the local-APIC IDs of all the processors present and enabled in the system.

Field	Byte Length	Byte Offset	Description
Signature	4	0	'CPUS' table signature.
Length	4	4	Length, in bytes, of the entire table. The length implies the number of Entry fields (*n*) at the end of the table.
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID string
OEM Table ID	8	16	OEM Table ID string
Entry for Processor 0	4	24	Processor0 local APIC ID.
			…
Entry for Processor n			ProcessorN local APIC ID.

SFI APIC Table

The optional SFI APIC table enumerates the IO-APICs present in the system.

Field	Byte Length	Byte Offset	Description
Signature	4	0	'APIC' table signature.
Length	4	4	Length, in bytes, of the entire table. The length implies the number of Entry fields (*n*) at the end of the table.
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID string
OEM Table ID	8	16	OEM Table ID string
Entry for IO-APIC 0	8	24	Physical Address IO-APIC 0.[2]
			…
Entry for IO-APIC n	8		Physical Address of IO-APIC n.

[2] The SFI IO-APIC entry specifies only the base address of the IO-APIC. It omits two unneeded fields that are included in the ACPI MADT IO-APIC entry, the APIC-id and the global system interrupt (GSI) base.

SFI Memory Map Table

The optional SFI Memory Map table describes the RAM present in the system. It contains memory descriptors as defined in GetMemoryMap() API of the Unified Extensible Firmware Interface v2.1 (UEFI). [3]

Field	Byte Length	Byte Offset	Description
Signature	4	0	'MMAP' table signature.
Length	4	4	Length, in bytes, of the entire table. The length implies the number of Entry fields (*n*) at the end of the table.
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID string
OEM Table ID	8	16	OEM Table ID string
Memory Descriptor	36*n	24	An array of EFI_MEMORY_DESCRIPTOR entries. The memory descriptor format is defined in the UEFI specification.

SFI Idle Table

The optional SFI Idle table describes the power saving CPU idle states (C-states) available to the OS. These states are accessed via native hardware support for MWAIT hint extensions. The IDLE table also enumerates the worst-case latency in microseconds to enter and exit each C-state. The C-states are enumerated in order of increasing latency and power savings. There are no latency restrictions.

The Idle table applies to every logical processor in the system. If there are topology dependencies between the logical processors that affect how the OS enters idle, the OS must discover those dependencies via native hardware interfaces.

Field	Byte Length	Byte Offset	Description
Signature	4	0	'IDLE' table signature.
Length	4	4	Length, in bytes, of the entire table. The length implies the number of Entry fields (*n*) at the end of the table.
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID string
OEM Table ID	8	16	OEM Table ID string

[3] http://www.uefi.org

Field	Byte Length	Byte Offset	Description
Entry for C1	4	20	C1 MWAIT hint
	4	24	C1 latency in microseconds
			…
Entry for Cn	4		Cn MWAIT hint
	4		Cn latency in microseconds

SFI Frequency Table

The optional SFI Frequency Table enumerates the processor performance states (P-states) available to the OS. PERF_CTL is the native MSR used to affect P-state transitions.

The FREQ table applies to every logical processor in the system. If there are topology dependencies between the logical processors that affect how the OS enters P-states, the OS must discover them via native hardware interfaces.

Field	Byte Length	Byte Offset	Description
Signature	4	0	'FREQ' table signature.
Length	4	4	Length, in bytes, of the entire table. The length implies the number of Entry fields (*n*) at the end of the table.
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID string
OEM Table ID	8	16	OEM Table ID string
Entry for P1	4	24	P0 Frequency in MHz
	4	28	P0 transition latency in microseconds
	4	32	P0 Control, value to write to PERF_CTL to enter P1
			…
Entry for Pn	4		Pn Frequency in MHz
	4		Pn transition latency in microseconds
	4		Pn Control, value to write to PERF_CTL to enter Pn

The SFI Frequency Table has no knowledge of Intel© Dynamic Acceleration Technology (IDA). As with ACPI systems, the firmware writer may choose to enable IDA only for P0 and not for P1 and deeper. Or the OS may know how to control IDA via native hardware interfaces.

SFI M-Timer Table

The optional SFI M-Timer Table enumerates these system timers if present. The OS programming model for the M-Timer is described in the [Moorestown OS Writer's Guide].

Field	Byte Length	Byte Offset	Description
Signature	4	0	'MTMR' table signature.
Length	4	4	Length, in bytes, of the entire table. The length implies the number of Entry fields (*n*) at the end of the table.
Revision	1	8	2
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID string
OEM Table ID	8	16	OEM Table ID string
Entry for M-Timer 0	8	24	Physical Address of M-Timer 0.
	4	32	Frequency of M-Timer 0 [Hz].
	4	36	IRQ of M-Timer 0
			…
Entry for M-Timer n	8		Physical Address of M-Timer n.
	4		Frequency of M-Timer n [Hz].
	4		IRQ of M-Timer n

SFI M-RTC Table

The optional SFI M-Real-Time-Clock Table enumerates these devices, if they are present. The OS programming model for the M-RTC is described in the [Moorestown OS Writer's Guide].

Field	Byte Length	Byte Offset	Description
Signature	4	0	'MRTC' table signature.
Length	4	4	Length, in bytes, of the entire table. The length implies the number of Entry fields (*n*) at the end of the table.
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID string
OEM Table ID	8	16	OEM Table ID string
Entry for M-RTC 0	8	24	Physical Address of M-RTC 0.
	4	32	IRQ of M-RTC 0

Field	Byte Length	Byte Offset	Description
			…
Entry for M-RTC n	8		Physical Address of M-RTC n.
	4		IRQ of M-RTC n.

SFI OEMx Table

The OEMx table id allows OEMs to define vendor-specific SFI tables while avoiding name-space collision with other platform vendors.

When the OS discovers an OEMx SFI table (OEM0, OEM1, etc.) it shall qualify that table signature with the 6-byte "OEM-id" and the 8-byte "OEM Table id". If the OS can not identify the table using these three fields, then it shall ignore the table

SFI Wake Vector Table

The optional SFI Wake Vector table contains a 64-bit entry where the OS writes its resume vector.

Field	Byte Length	Byte Offset	Description
Signature	4	0	'WAKE' table signature.
Length	4	4	32, length, in bytes, of the entire table.
Revision	1	8	1
Checksum	1	9	Entire table must sum to zero.
OEMID	6	10	OEM ID string
OEM Table ID	8	16	OEM Table ID string
Wake Vector	8	24	Physical Address of Wake Routine

Relationship with ACPI

An SFI-compliant platform will not likely implement ACPI. If the platform supports ACPI, there would be little benefit to implementing SFI, which is effectively a tiny sub-set of ACPI. However, for development purposes, it is possible for a platform to export both SFI and ACPI support.

An OS optimized to run on an SFI platform would not include ACPI support – for the ACPI support would go unused on the SFI platform. However, an OS can include both SFI and ACPI support to have the flexibility to run on a variety of platforms.

As shown in figure 1, the SFI System table may optionally point to an ACPI Extended System Description Table (XSDT). The format of the XSDT is defined in the ACPI specification[4], but as an SFI static header is a sub-set of an ACPI static header, the XSDT appears to the SFI-capable OS as just another SFI table. The XSDT can point to any of the tables with table signatures defined or reserved by the ACPI specification. In particular, SFI systems will likely access a PCI Memory Configuration Table (MCFG), as defined in the PCI Firmware Specification.

The OS simply ignores any tables with signatures it does not recognize, whether those tables are discovered directly via the SFI System Table, or indirectly via the ACPI XSDT.

Relationship with UEFI

An SFI compliant platform may or may not implement UEFI.

However, for those systems which choose not to implement UEFI, SFI supports the static MMAP table, providing the same information as UEFI's GetMemoryMap() API.

[4] http://www.acpi.info