**# Title**: ACPI MADT MPWakeup

**# Status**: Submitted to industry standard forum

**# Document**: ACPI Specification 6.5

**# License**: SPDX-License-Identifier: CC-BY-4.0

**# Submitter**: [TianoCore Community](<https://www.tianocore.org>)

Jiewen Yao <Jiewen.yao@intel.com>

Jon Lange <jlange@microsoft.com>

**# Summary of the change**

**[Background]**

Current ACPI specification supports a mailbox-based multiprocessor wakeup mechanism. (<https://uefi.org/specs/ACPI/6.5/05_ACPI_Software_Programming_Model.html#multiprocessor-wakeup-structure>). (See Figure 1)

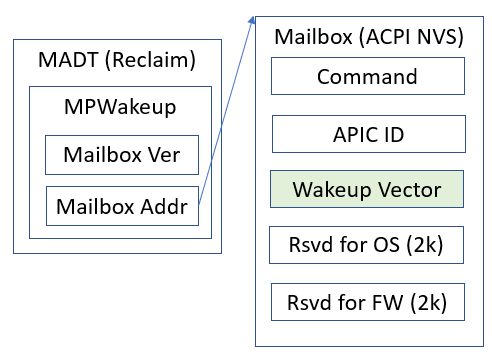


Figure 1

However, there is no restriction on wakeup. That means the OS may have freedom to:

1. Use different Wakeup Vector for different APs.
2. Fill arbitrary address to the wakeup vector, from address 0 to top of the memory.
3. Trigger wakeup action simultaneously, as long as AP confirms command is received but before jumps).

**[Problem Statement]**

The ACPI specification defined MADT wakeup mechanism is difficult to implement for firmware:

1. It requires maintaining memory mappings (static or on-demand) for the arbitrary wakeup vector after ExitBootServices, which adds complexity.
2. These wakeups can theoretically execute in parallel, which adds complexity.
3. It can consume a relatively large amount of RAM for page tables since each AP being woken up in parallel might need several page-table pages for each mapping.

Although it is not the original intention, the current specification allows the flexibility, which brings unnecessary complexity to the firmware implementation.

**[Existing Solution Analysis]**

1. Linux OS

Today, Linux chooses to use **wakeup vector below 1MB**, same address for AP, and wakeup AP one by one. (See figure 2)

*Reference:*

[*https://github.com/torvalds/linux/blob/master/arch/x86/kernel/acpi/boot.c#L347*](https://github.com/torvalds/linux/blob/master/arch/x86/kernel/acpi/boot.c#L347)

[*https://github.com/torvalds/linux/blob/master/arch/x86/kernel/smpboot.c#L1084*](https://github.com/torvalds/linux/blob/master/arch/x86/kernel/smpboot.c#L1084)

[*https://github.com/torvalds/linux/blob/master/arch/x86/realmode/rm/trampoline\_64.S#L196*](https://github.com/torvalds/linux/blob/master/arch/x86/realmode/rm/trampoline_64.S#L196)

1. Windows OS

Windows OS plans to use **mailbox ReservedForOS** **region** as wakeup vector. (See figure 2)

1. UEFI firmware

Today, UEFI firmware TDVF only supports the **wakeup vector** **below 4GB** because it uses static paging for below 4GB. It is a known issue <https://bugzilla.tianocore.org/show_bug.cgi?id=4185>.

*Reference:*

[*https://github.com/tianocore/edk2/blob/master/OvmfPkg/ResetVector/Ia32/PageTables64.asm#L46*](https://github.com/tianocore/edk2/blob/master/OvmfPkg/ResetVector/Ia32/PageTables64.asm)

[*https://github.com/tianocore/edk2/blob/master/OvmfPkg/TdxDxe/X64/ApRunLoop.nasm*](https://github.com/tianocore/edk2/blob/master/OvmfPkg/TdxDxe/X64/ApRunLoop.nasm)

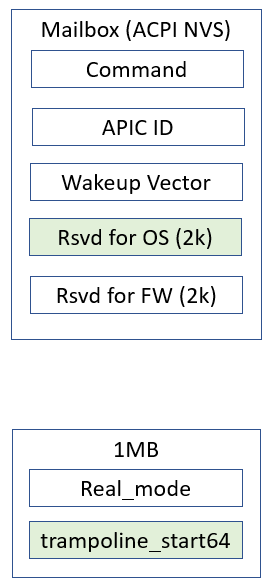


Figure 2

**[Proposal]**

This proposal adds restriction on the wakeup vector location: It must be in “**Mailbox ReservedForOS region**” or “**below 1MB memory**”.

1. OS shall choose either “below 1MB memory” or “Mailbox ReservedForOs region”.
2. Firmware shall support both “below 1MB memory” and “Mailbox ReservedForOS region”.

That means the firmware just needs to create a static page table to cover below 1MB memory and Mailbox ReservedForOs region, and all APs can share the same page table.

Other possible options

1. Limit the wakeup vector location to firmware accepted memory
   1. CONs: That may help the confidential computing virtual firmware use case. But it is not useful for the physical firmware, because the unaccepted memory concept does not exist in physical firmware.
2. Limit the wakeup vector location to be same for all APs
   1. CONs: The firmware may still need to construct page table dynamically to cover the wakeup vector location, if the firmware cannot allocate page table to cover full memory region. It is much less convenient than constructing the page tables ahead of time.
3. Limit the wakeup vector time to be one by one
   1. CONs: Same as b).

**# Benefits of the change**

With this proposal, wakeup vector being in “**Mailbox ReservedForOS region**” or “**below 1MB memory**”, the firmware implementation can be simplified. It just needs to create a static page table to cover below 1MB memory and Mailbox ReservedForOs region.

**# Impact of the change**

The proposal is compatible with existing implementation in Linux, Windows, and firmware.

No code change is required.

**# Detailed description of the change [normative updates]**

YELLOW means addition, RED means deletion.

#### 5.2.12.19. Multiprocessor Wakeup Structure

The OS section contains command, flags, APIC ID, and a wakeup address. After the OS detects the processor number from the MADT table, the OS may prepare the wakeup routine, fill the wakeup address field in the mailbox, indicate which processor need to be wakeup in the APID ID field, and send wakeup command. Once an application processor detects the wakeup command and its own APIC ID, the application processor will jump to the OS-provided wakeup address. There are two places where a Wakeup vector address can be located: 1) Mailbox ReservedForOs Region, or 2) below 1MB memory if the system has below 1MB memory reported in memory map. The wakeup vector address pointing to the ReservedForOs area is preferred. The application processor will ignore the command if the APIC ID does not match its own.

*Table 5.44****Multiprocessor Wakeup Mailbox Structure***

| **Field** | **Byte Length** | **Byte Offset** | **Description** |
| --- | --- | --- | --- |
| Command | 2 | 0 | 0: Noop - no operation.  1: Wakeup – jump to the wakeup vector.  2-0xFFFF: Reserved |
| *Reserved* | 2 | 2 | Must be 0. |
| ApicId | 4 | 4 | The processor’s local APIC ID. The application processor shall check if the ApicId field matches its own APIC ID. The application processor shall ignore the command in case of APIC ID mismatch. |
| WakeupVector | 8 | 8 | The wakeup address for application processor(s).  For Intel processors, the execution environment is:  Interrupts must be disabled.  RFLAGES.IF set to 0.  Long mode enabled.  Paging mode is enabled and physical memory for waking vector is identity mapped (virtual address equals physical address)  Waking vector must be contained within one physical page. Waking vector can be in two places: Mailbox ReservedForOS region, or below 1MB memory.  Selectors are set to flat and otherwise not used. |
| ReservedForOs | 2032 | 16 | Reserved for OS use. |
| ReservedForFirmware | 2048 | 2048 | Reserved for firmware use. |

**# Special Instructions**

NO