intel_idle CPU Idle Time Management Driver

Copyright: © 2020 Intel Corporation
Author: Rafael J. Wysocki <rafael.j.wysocki@intel.com>

General Information

intel_idle is a part of the :doc:`CPU idle time management subsystem <cpuidle>` in the Linux kernel (CPUIdle). It is the default CPU idle time management driver for the Nehalem and later generations of Intel processors, but the level of support for a particular processor model in it depends on whether or not it recognizes that processor model and may also depend on information coming from the platform firmware. [To understand intel_idle it is necessary to know how CPUIdle works in general, so this is the time to get familiar with Documentation/admin-guide/pm/cpuidle.rst if you have not done that yet.]

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\pm\(linux-master)(Documentation)(admin-guide)(pm)intel_idle.rst, line 16); *backlink***
>
> Unknown interpreted text role "doc".

intel_idle uses the MWAIT instruction to inform the processor that the logical CPU executing it is idle and so it may be possible to put some of the processor's functional blocks into low-power states. That instruction takes two arguments (passed in the EAX and ECX registers of the target CPU), the first of which, referred to as a *hint*, can be used by the processor to determine what can be done (for details refer to Intel Software Developerâ€™s Manual [1]). Accordingly, intel_idle refuses to work with processors in which the support for the MWAIT instruction has been disabled (for example, via the platform firmware configuration menu) or which do not support that instruction at all.

intel_idle is not modular, so it cannot be unloaded, which means that the only way to pass early-configuration-time parameters to it is via the kernel command line.

Enumeration of Idle States

Each MWAIT hint value is interpreted by the processor as a license to reconfigure itself in a certain way in order to save energy. The processor configurations (with reduced power draw) resulting from that are referred to as C-states (in the ACPI terminology) or idle states. The list of meaningful MWAIT hint values and idle states (i.e. low-power configurations of the processor) corresponding to them depends on the processor model and it may also depend on the configuration of the platform.

In order to create a list of available idle states required by the CPUIdle subsystem (see :ref:`idle-states-representation` in Documentation/admin-guide/pm/cpuidle.rst), intel_idle can use two sources of information: static tables of idle states for different processor models included in the driver itself and the ACPI tables of the system. The former are always used if the processor model at hand is recognized by intel_idle and the latter are used if that is required for the given processor model (which is the case for all server processor models recognized by intel_idle) or if the processor model is not recognized. [There is a module parameter that can be used to make the driver use the ACPI tables with any processor model recognized by it; see below.]

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\pm\(linux-master)(Documentation)(admin-guide)(pm)intel_idle.rst, line 55); *backlink***
>
> Unknown interpreted text role "ref".

If the ACPI tables are going to be used for building the list of available idle states, intel_idle first looks for a _CST object under one of the ACPI objects corresponding to the CPUs in the system (refer to the ACPI specification [2] for the description of _CST and its output package). Because the CPUIdle subsystem expects that the list of idle states supplied by the driver will be suitable for all of the CPUs handled by it and intel_idle is registered as the CPUIdle driver for all of the CPUs in the system, the driver looks for the first _CST object returning at least one valid idle state description and such that all of the idle states included in its return package are of the FFH (Functional Fixed Hardware) type, which means that the MWAIT instruction is expected to be used to tell the processor that it can enter one of them. The return package of that _CST is then assumed to be applicable to all of the other CPUs in the system and the idle state descriptions extracted from it are stored in a preliminary list of idle states coming from the ACPI tables. [This step is skipped if intel_idle is configured to ignore the ACPI tables; see below.]

Next, the first (index 0) entry in the list of available idle states is initialized to represent a "polling idle state" (a pseudo-idle state in which the target CPU continuously fetches and executes instructions), and the subsequent (real) idle state entries are populated as follows.

If the processor model at hand is recognized by intel_idle, there is a (static) table of idle state descriptions for it in the driver. In that case, the "internal" table is the primary source of information on idle states and the information from it is copied to the final list of

available idle states. If using the ACPI tables for the enumeration of idle states is not required (depending on the processor model), all of the listed idle state are enabled by default (so all of them will be taken into consideration by `CPUIdle` governors during CPU idle state selection). Otherwise, some of the listed idle states may not be enabled by default if there are no matching entries in the preliminary list of idle states coming from the ACPI tables. In that case user space still can enable them later (on a per-CPU basis) with the help of the `disable` idle state attribute in `sysfs` (see :ref:`idle-states-representation` in Documentation/admin-guide/pm/cpuidle.rst). This basically means that the idle states "known" to the driver may not be enabled by default if they have not been exposed by the platform firmware (through the ACPI tables).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\pm\(linux-master)(Documentation)(admin-guide)(pm)intel_idle.rst, line 90); *backlink*

Unknown interpreted text role "ref".

If the given processor model is not recognized by `intel_idle`, but it supports `MWAIT`, the preliminary list of idle states coming from the ACPI tables is used for building the final list that will be supplied to the `CPUIdle` core during driver registration. For each idle state in that list, the description, `MWAIT` hint and exit latency are copied to the corresponding entry in the final list of idle states. The name of the idle state represented by it (to be returned by the `name` idle state attribute in `sysfs`) is "CX_ACPI", where X is the index of that idle state in the final list (note that the minimum value of X is 1, because 0 is reserved for the "polling" state), and its target residency is based on the exit latency value. Specifically, for C1-type idle states the exit latency value is also used as the target residency (for compatibility with the majority of the "internal" tables of idle states for various processor models recognized by `intel_idle`) and for the other idle state types (C2 and C3) the target residency value is 3 times the exit latency (again, that is because it reflects the target residency to exit latency ratio in the majority of cases for the processor models recognized by `intel_idle`). All of the idle states in the final list are enabled by default in this case.

Initialization

The initialization of `intel_idle` starts with checking if the kernel command line options forbid the use of the `MWAIT` instruction. If that is the case, an error code is returned right away.

The next step is to check whether or not the processor model is known to the driver, which determines the idle states enumeration method (see above), and whether or not the processor supports `MWAIT` (the initialization fails if that is not the case). Then, the `MWAIT` support in the processor is enumerated through `CPUID` and the driver initialization fails if the level of support is not as expected (for example, if the total number of `MWAIT` substates returned is 0).

Next, if the driver is not configured to ignore the ACPI tables (see below), the idle states information provided by the platform firmware is extracted from them.

Then, `CPUIdle` device objects are allocated for all CPUs and the list of available idle states is created as explained above.

Finally, `intel_idle` is registered with the help of cpuidle_register_driver() as the `CPUIdle` driver for all CPUs in the system and a CPU online callback for configuring individual CPUs is registered via cpuhp_setup_state(), which (among other things) causes the callback routine to be invoked for all of the CPUs present in the system at that time (each CPU executes its own instance of the callback routine). That routine registers a `CPUIdle` device for the CPU running it (which enables the `CPUIdle` subsystem to operate that CPU) and optionally performs some CPU-specific initialization actions that may be required for the given processor model.

Kernel Command Line Options and Module Parameters

The *x86* architecture support code recognizes three kernel command line options related to CPU idle time management: `idle=poll`, `idle=halt`, and `idle=nomwait`. If any of them is present in the kernel command line, the `MWAIT` instruction is not allowed to be used, so the initialization of `intel_idle` will fail.

Apart from that there are four module parameters recognized by `intel_idle` itself that can be set via the kernel command line (they cannot be updated via sysfs, so that is the only way to change their values).

The `max_cstate` parameter value is the maximum idle state index in the list of idle states supplied to the `CPUIdle` core during the registration of the driver. It is also the maximum number of regular (non-polling) idle states that can be used by `intel_idle`, so the enumeration of idle states is terminated after finding that number of usable idle states (the other idle states that potentially might have been used if `max_cstate` had been greater are not taken into consideration at all). Setting `max_cstate` can prevent `intel_idle` from exposing idle states that are regarded as "too deep" for some reason to the `CPUIdle` core, but it does so by making them effectively invisible until the system is shut down and started again which may not always be desirable. In practice, it is only really necessary to do that if the idle states in question cannot be enabled during system startup, because in the working state of the system the CPU power management quality of service (PM QoS) feature can be used to prevent `CPUIdle` from touching those idle states even if they have been enumerated (see :ref:`cpu-pm-qos` in Documentation/admin-guide/pm/cpuidle.rst). Setting `max_cstate` to 0 causes the `intel_idle` initialization to fail.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\pm\(linux-master)(Documentation)(admin-guide)

(pm)intel_idle.rst, line 177); *backlink*

Unknown interpreted text role "ref".

The no_acpi and use_acpi module parameters (recognized by intel_idle if the kernel has been configured with ACPI support) can be set to make the driver ignore the system's ACPI tables entirely or use them for all of the recognized processor models, respectively (they both are unset by default and use_acpi has no effect if no_acpi is set).

The value of the states_off module parameter (0 by default) represents a list of idle states to be disabled by default in the form of a bitmask.

Namely, the positions of the bits that are set in the states_off value are the indices of idle states to be disabled by default (as reflected by the names of the corresponding idle state directories in sysfs, :file:`state0`, :file:`state1` ... :file:`state<i>` ..., where <i> is the index of the given idle state; see :ref:`idle-states-representation` in Documentation/admin-guide/pm/cpuidle.rst).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\pm\(linux-master) (Documentation) (admin-guide) (pm)intel_idle.rst, line 204); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\pm\(linux-master) (Documentation) (admin-guide) (pm)intel_idle.rst, line 204); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\pm\(linux-master) (Documentation) (admin-guide) (pm)intel_idle.rst, line 204); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\pm\(linux-master) (Documentation) (admin-guide) (pm)intel_idle.rst, line 204); *backlink*

Unknown interpreted text role "ref".

For example, if states_off is equal to 3, the driver will disable idle states 0 and 1 by default, and if it is equal to 8, idle state 3 will be disabled by default and so on (bit positions beyond the maximum idle state index are ignored).

The idle states disabled this way can be enabled (on a per-CPU basis) from user space via sysfs.

Core and Package Levels of Idle States

Typically, in a processor supporting the MWAIT instruction there are (at least) two levels of idle states (or C-states). One level, referred to as "core C-states", covers individual cores in the processor, whereas the other level, referred to as "package C-states", covers the entire processor package and it may also involve other components of the system (GPUs, memory controllers, I/O hubs etc.).

Some of the MWAIT hint values allow the processor to use core C-states only (most importantly, that is the case for the MWAIT hint value corresponding to the C1 idle state), but the majority of them give it a license to put the target core (i.e. the core containing the logical CPU executing MWAIT with the given hint value) into a specific core C-state and then (if possible) to enter a specific package C-state at the deeper level. For example, the MWAIT hint value representing the C3 idle state allows the processor to put the target core into the low-power state referred to as "core C3" (or CC3), which happens if all of the logical CPUs (SMT siblings) in that core have executed MWAIT with the C3 hint value (or with a hint value representing a deeper idle state), and in addition to that (in the majority of cases) it gives the processor a license to put the entire package (possibly including some non-CPU components such as a GPU or a memory controller) into the low-power state referred to as "package C3" (or PC3), which happens if all of the cores have gone into the CC3 state and (possibly) some additional conditions are satisfied (for instance, if the GPU is covered by PC3, it may be required to be in a certain GPU-specific low-power state for PC3 to be reachable).

As a rule, there is no simple way to make the processor use core C-states only if the conditions for entering the corresponding package C-states are met, so the logical CPU executing MWAIT with a hint value that is not core-level only (like for C1) must always assume that this may cause the processor to enter a package C-state. [That is why the exit latency and target residency values corresponding to the majority of MWAIT hint values in the "internal" tables of idle states in intel_idle reflect the properties of package C-states.] If using package C-states is not desirable at all, either :ref:`PM QoS <cpu-pm-qos>` or the max_cstate module parameter of intel_idle described above must be used to restrict the range of permissible idle states to the ones with core-level

only `MWAIT` hint values (like `C1`).

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\admin-guide\pm\(linux-master)(Documentation)(admin-guide)(pm)intel_idle.rst, line 251); *backlink*

Unknown interpreted text role "ref".

References

[1] *IntelÂ® 64 and IA-32 Architectures Software Developerâ€™s Manual Volume 2B*, https://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-2b-manual.html

[2] *Advanced Configuration and Power Interface (ACPI) Specification*, https://uefi.org/specifications