A Minimum Complete Tutorial of CPU Power Management, C-states and P-states
20 minute read

Introduction

How does CPU save power ? You will learn all the basics about the CPU power management in this post.

The processor/CPU is designed to operate forever under a specific load. As almost none of us are doing a `24x7`calculation employing all the resources of the CPU continuously, it is most of the time not operating in its designed maximum. So what is the point of keeping the whole CPU powered on at its full capacity ? This is the point of CPU power management. The power management topic covers much more than this, including the RAM, GPU etc., but I am going to tell you only about the CPU side of the story in this post.

If you know about C-states and P-states and how CPU enters to and exits from these states, probably there is nothing new in this post for you. If not, continue reading.

I was planning to include a few real-world examples from Linux, but the post was getting longer and taking even more time to finish, so I will do that in another post.

The general information in this post is applicable to all modern CPUs but the details can be very specific to the individual CPU (esp. to the processor family) and I am using an Intel® Xeon® E3-1245 v5 @ 3.50GHz, this is a Xeon E3–1200 v5 series (formerly Skylake) processor (model number 0x5e).

The main references used in this post are:

- Intel(R) Xeon Processor E3–1200 v5 Product Family Datasheets
- Intel(R) Xeon Processor E3–1245 v5 Product Specification
- Sofware Impact to Platform Energy-Efficiency (Intel White Paper)
- Intel(R) 64 and IA-32 Architectures Software Developer's Manual
- ACPI Specification v6.2
- ACPI page on Wikipedia
- Linux Kernel Sources version 4.13.0

Unless otherwise noted, all original figures and tables are from the datasheet.

Changes after publication

- 2019/03/01: Added CoreFreq output to last section.

CPU features

On its official product page, my CPU has the following features listed:

- Idle States
- Enhanced Intel SpeedStep® Technology

You can see on the page that "Idle States (C-states) are used to save power when the processor is idle." and "Intel SpeedStep® Technology switches both voltage and frequency in tandem between high and low levels in response to processor load."

You will learn what both of these descriptions mean throughout this post.

How to decrease CPU power consumption while it is operating ?

On a production CPU (so not considering the things that can be done while designing the CPU), to save power, you can do two things. You can:

- eliminate the power consumption of a subsystem (a core or other resources like clock or cache) by completely powering it down (so cutting down the voltage, reducing it to zero)

or

- decrease the power consumption by decreasing the voltage and/or the frequency of the subsystem and/or the whole processor

The first one is simple to understand, if you shut down the power, you do not consume any power.

The second requires a bit more explanation. The power consumption of an integrated circuit (such as a processor) is proportional linearly to frequency and quadratically to voltage.

$P \sim f V^2$

> Note for those of you knowing more about digital electronics:
>
> Pcpu is actually = Pdynamic + Psc + Pleak
>
> On an active CPU, Pdynamic is the most important part, it is the part I mention here, being proportional to the frequency and quadratically to the voltage.
>
> Psc=Pshort-circuit is proportional to the frequency.
>
> Pleak is proportional to the voltage.

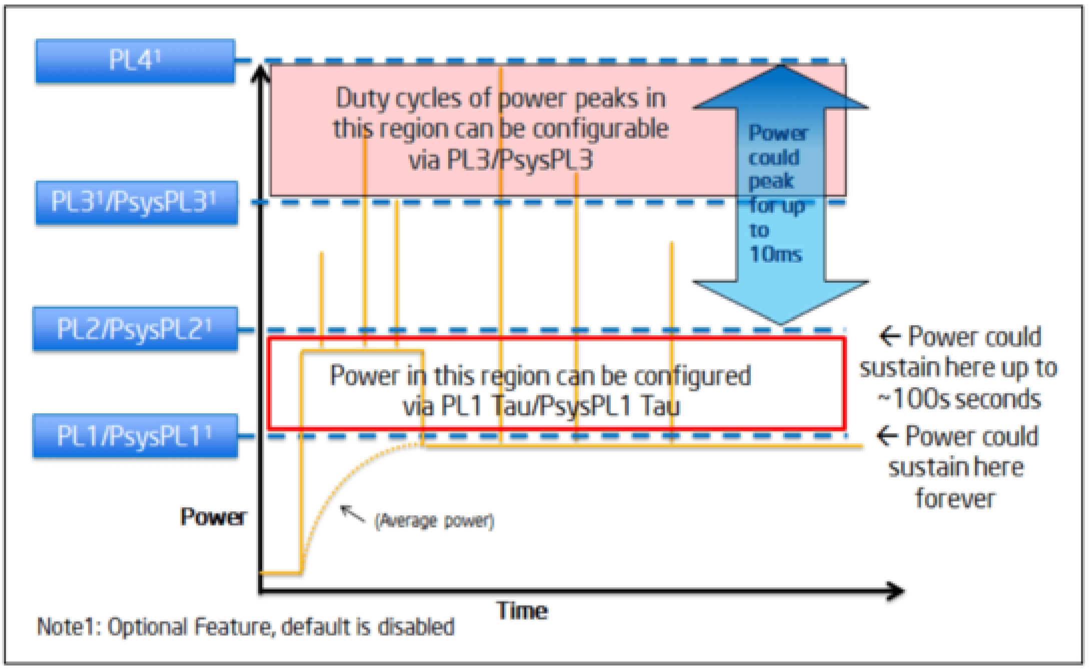
Also, the voltage and frequency is not independent, and seems to have a linear relationship. (http://async.org.uk/tech-reports/NCL-EEE-MICRO-TR-2015-197.pdf)

Higher (serial) performance requires both higher frequency and higher voltage, thus having even a larger effect on the power consumption.

What is the maximum power a CPU consume ?

This depends a lot on the processor but E3–1245 v5 @ 3.50 Ghz has Thermal Design Power (TDP) of 80 Watts. This is an average value the CPU can sustain forever (Power Limit/PL1 in the figure below) and it is usually the value that the cooling solutions should be designed at for reliability. Actual power CPU consumes can temporarily go higher (as shown as PL2, PL3 and PL4 in the figure below). TDP is measured under a high-complexity (~worst case) load with all cores active at the base frequency (of 3.50 Ghz).

Figure 5-1. Package Power Control


PL4¹
PL3¹/PsysPL3¹
PL2/PsysPL2¹
PL1/PsysPL1¹
Duty cycles of power peaks in this region can be configurable via PL3/PsysPL3
Power could peak for up to 10ms
← Power could sustain here up to ~100s seconds
Power in this region can be configured via PL1 Tau/PsysPL1 Tau
← Power could sustain here forever
Power
(Average power)
Time
Note1: Optional Feature, default is disabled

CPU Package Power Control

You can see in the figure above that the CPU can consume more power than TDP at PL2 for up to 100 seconds, that is actually quite a long duration.

Processor power states (C-states) vs. performance states (P-states)

The two ways to decrease the power consumption of a processor:

- powering down subsystems
- voltage/frequency reduction

is accomplished by using:

- C-states
- P-states

respectively.

C-states describe the first case, so they are the idle (power saving) states. In order to power down a subsystem, that subsystem should not be running anything, so it should be at idle, doing nothing, executing nothing. So C-state x, Cx, means one or more subsystems of the CPU is at idle, powered down.

On the other hand, P-states describe the second case, so they are the executing (power saving) states. The subsystem is actually running but it does not require full performance so the voltage and/or frequency it operates is decreased. So P-state x, Px, means the subsystem it refers to (e.g. a CPU core) is operating at a specific (frequency, voltage) pair.

Because most modern CPUs have multiple cores in a single package, C-states are further divided into core C-states (CC-states) and package C-states (PC-states). The reason for PC-states is there are other (shared) components in the processor that can also be powered down after all cores using them are powered down (e.g.

the shared cache). However, as a user or programmer, we cannot control these, since we do not interact with the package directly, but we interact with the individual cores. So we can only affect the CC-states directly, PC-states are indirectly affected based on the CC-states of the cores.

The states are numbered starting from zero like C0, C1… and P0, P1… The higher the number is the more power is saved. C0 means no power saving by shutting down something, so everything is powered on. P0 means maximum performance, thus maximum frequency, voltage and power used.

C-states

The basic C-states (defined by ACPI) are:

- C0: Active, CPU/Core is executing instructions. P-states are relevant here, CPU/Core may be operating at its maximum performance (thus at P0) or at a lower performance/power (thus at anything other than P0).
- C1: Halt, nothing is being executed, but it can return to C0 instantenously. Since it is not working (but halted), P-states are not relevant for C1 or any Cx other than C0.
- C2: Stop-Clock, similar to C1 but it takes longer time to go back to C0.
- C3: Sleep. It can go back to C0, but it will take considerably longer time.

Many modern CPUs have plenty more C-states and according to its datasheet Intel® Xeon® E3-1200 v5 Family has C0, C1, C1E (C1 Enhanced), C2, C3, C6, C7 and C8. C1 and C1E are CC-states only, and C2 is only a PC-state. All others are both a CC-state and a PC-state.

> Note: There are also thread level C-states because of Intel Hyper-Threading. However, the individual threads can only request C-states but power saving action only takes place when the core enters to that C-state. I will in general omit talking about Thread C-states and Hyper-Threading in this post.

The descriptions of these states from the datasheet are:

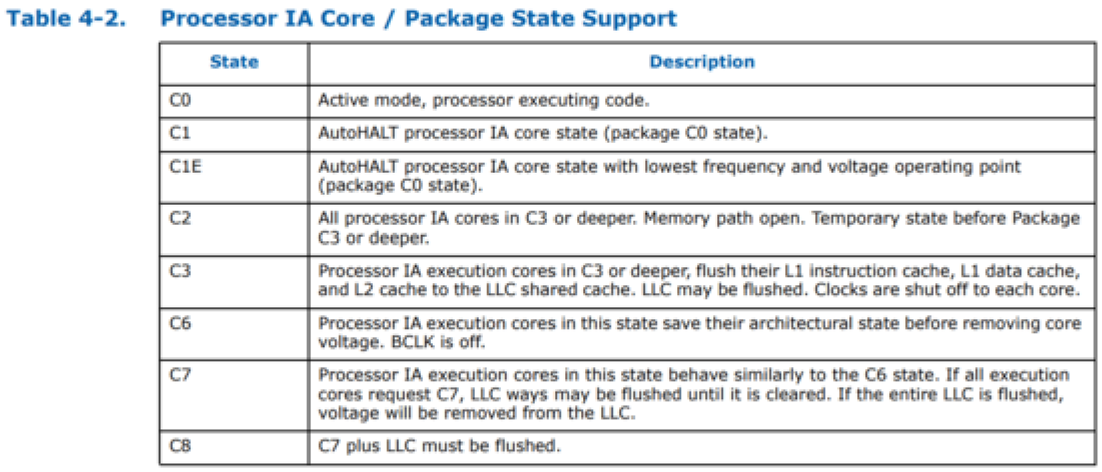


Table 4-2. Processor IA Core / Package State Support

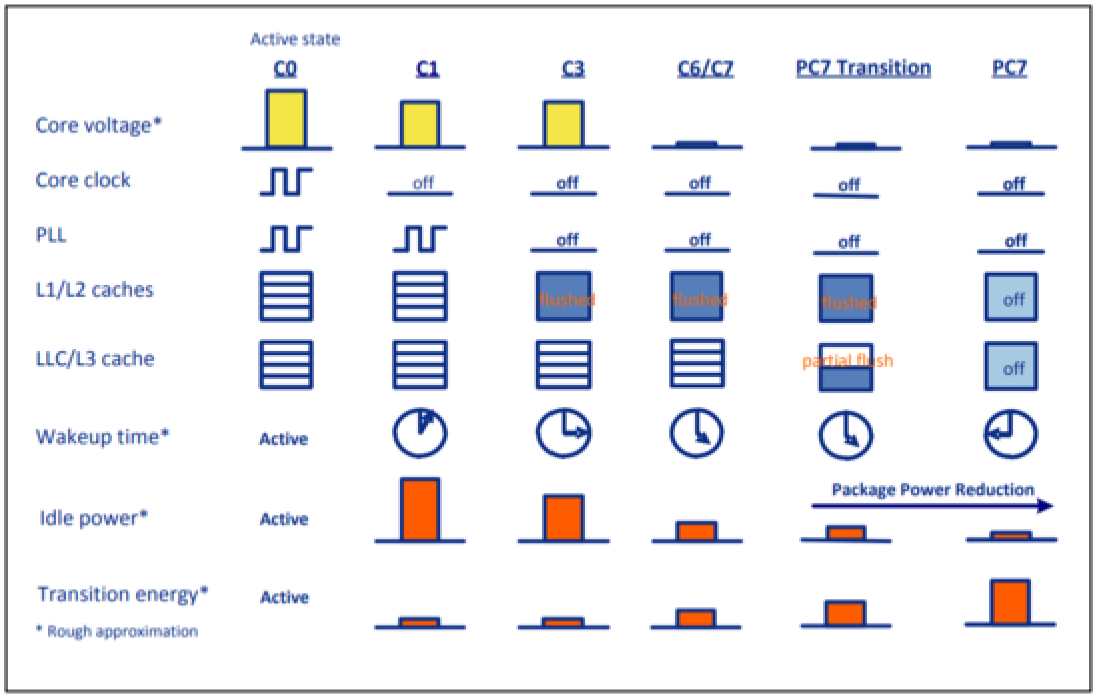
State	Description
C0	Active mode, processor executing code.
C1	AutoHALT processor IA core state (package C0 state).
C1E	AutoHALT processor IA core state with lowest frequency and voltage operating point (package C0 state).
C2	All processor IA cores in C3 or deeper. Memory path open. Temporary state before Package C3 or deeper.
C3	Processor IA execution cores in C3 or deeper, flush their L1 instruction cache, L1 data cache, and L2 cache to the LLC shared cache. LLC may be flushed. Clocks are shut off to each core.
C6	Processor IA execution cores in this state save their architectural state before removing core voltage. BCLK is off.
C7	Processor IA execution cores in this state behave similarly to the C6 state. If all execution cores request C7, LLC ways may be flushed until it is cleared. If the entire LLC is flushed, voltage will be removed from the LLC.
C8	C7 plus LLC must be flushed.

Processor IA Core / Package State Support Table

> Note: LLC above refers to Last Level Cache, meaning the shared L3 cache in the processor.

and here there is a visual description:

Figure 4: Flexible C-states to select Idle Power Level vs. Responsiveness



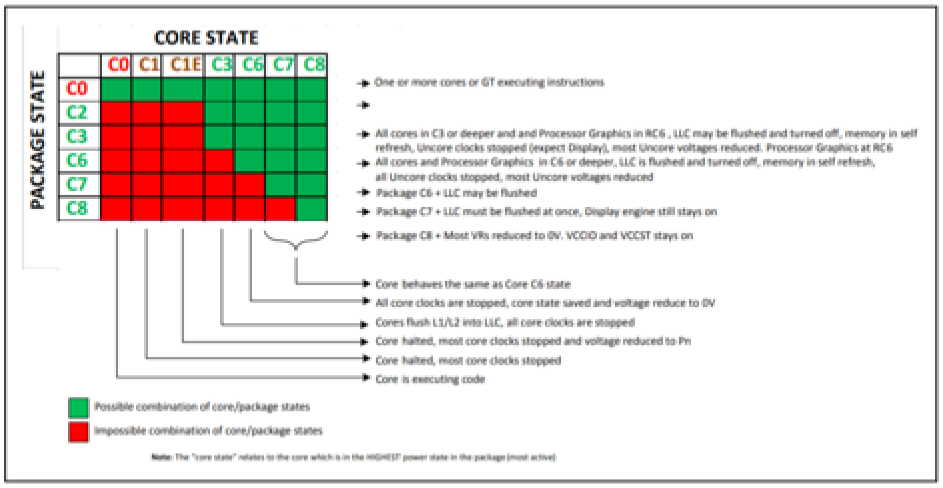
Flexible C-states to select Idle Power Level vs. Responsiveness (from Software Impact to Platform Energy-Efficiency White Paper)

A very basic timeline of power saving using C-states is:

- The normal operation is at C0.
- First, the clocks of idle core is stopped. (C1)
- Then, the local caches (L1/L2) of the core is flushed and the core is powered down. (C3)
- Then, when all the cores are powered down, the shared cache (L3/LLC) of the package is flushed and at the end the package/whole CPU can be (almost) powered down. I said almost because I guess there has to be something powered on to return back to C0.

As you might guess, CC-states and PC-states are not independent, so some combinations are impossible. Figure below summarizes this:

Figure 4-2. Processor Package and IA Core C-States



	C0	C1	C1E	C3	C6	C7	C8
C0	🟩	🟩	🟩	🟩	🟩	🟩	🟩
C2	🟥	🟥	🟥	🟩	🟩	🟩	🟩
C3	🟥	🟥	🟥	🟩	🟩	🟩	🟩
C6	🟥	🟥	🟥	🟥	🟩	🟩	🟩
C7	🟥	🟥	🟥	🟥	🟥	🟩	🟩
C8	🟥	🟥	🟥	🟥	🟥	🟥	🟩

CORE STATE
PACKAGE STATE
One or more cores or GT executing instructions
All cores in C3 or deeper and and Processor Graphics in RC6 , LLC may be flushed and turned off, memory in self refresh, Uncore clocks stopped (expect Display), most Uncore voltages reduced. Processor Graphics at RC6
All cores and Processor Graphics in C6 or deeper, LLC is flushed and turned off, memory in self refresh, all Uncore clocks stopped, most Uncore voltages reduced
Package C6 + LLC may be flushed
Package C7 + LLC must be flushed at once, Display engine still stays on
Package C8 + Most VRs reduced to 0V. VCCIO and VCCST stays on
Core behaves the same as Core C6 state
All core clocks are stopped, core state saved and voltage reduce to 0V
Cores flush L1/L2 into LLC, all core clocks are stopped
Core halted, most core clocks stopped and voltage reduced to Pn
Core halted, most core clocks stopped
Core is executing code

🟩 Possible combination of core/package states
🟥 Impossible combination of core/package states

Note: The "core state" relates to the core which is in the HIGHEST power state in the package (most active)

Processor Package and IA Core C-States of Intel Xeon E3–1200 v5 Product Family

Obviously, if a core is running (C0), the package cannot be in a state other than C0. On the other hand, if a core is completely powered down (C8), the package can still be at C0 if any other core is still running.

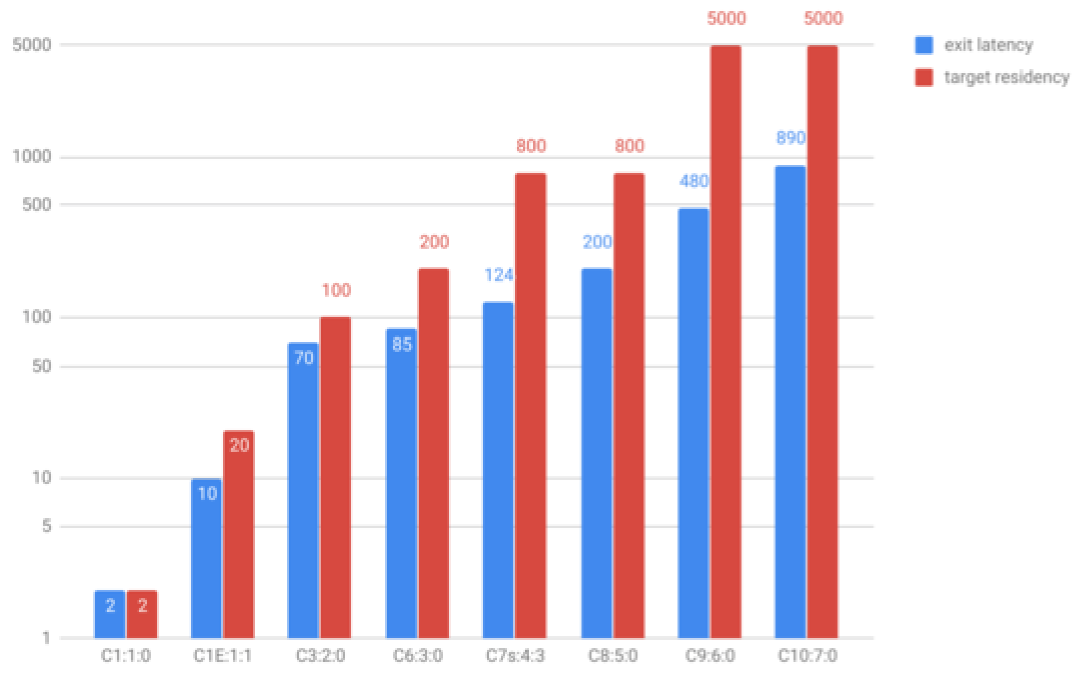
Note: Intel Software Developer’s Manual mentions sub C-states, or substates, meaning a C-state (type) actually contains one or more sub C-states. After checking the intel_idle driver code in the Linux kernel, I understand for example C1 and C1E has the same state type C1, but they are subtypes 0 and 1.

The number of subtypes of the eight C-state types (0..7) for C1..C8 is discovered by CPUID instruction. Looking at my CPU, cpuid program output returns:

```
MONITOR/MWAIT (5):
      smallest monitor-line size (bytes)       = 0x40 (64)
      largest monitor-line size (bytes)        = 0x40 (64)
      enum of Monitor-MWAIT exts supported     = true
      supports intrs as break-event for MWAIT  = true
      number of C0 sub C-states using MWAIT    = 0x0 (0)
      number of C1 sub C-states using MWAIT    = 0x2 (2)
      number of C2 sub C-states using MWAIT    = 0x1 (1)
      number of C3 sub C-states using MWAIT    = 0x2 (2)
      number of C4 sub C-states using MWAIT    = 0x4 (4)
      number of C5 sub C-states using MWAIT    = 0x1 (1)
      number of C6 sub C-states using MWAIT    = 0x0 (0)
      number of C7 sub C-states using MWAIT    = 0x0 (0)
```

Also note from the Intel manual: “The definition of C0 through C7 states for MWAIT extension are processor-specific C-states, not ACPI Cstates”. So do not confuse these with the ACPI C-states I mentioned before, they are obviously related and there is a mapping between but they are not exactly same.

I have created the chart below from the `intel_idle` driver source code for my cpu (model 0x5e). The horizontal axis labels indicate the C-state name:processor specific state:processor specific substate, and the vertical axis indicates exit latency and target residency values from the driver source code. exit_latency is used as a guidance to assess the real time latency impact of this state (e.g. how much it takes to return back from this state to C0), and target residency means the minimum duration the core has to stay in this state to overcome the enter/exit power costs. Pay attention to logarithmic scale on the vertical axis, the latency and break even power impact of deeper states are exponentially higher.



C-state exit latency and target residency constants in intel_idle driver source code

Note: Although there is C9 and C10 in the table, since they have 0 substates, they are not used in my cpu. Other CPUs in the same family may support these states as well.

ACPI power states

Before P-states, it is better to mention a bit about the ACPI Power States. These are normally, as a user, what we know while using the computer. The so called global system states (Gx states) are listed below:

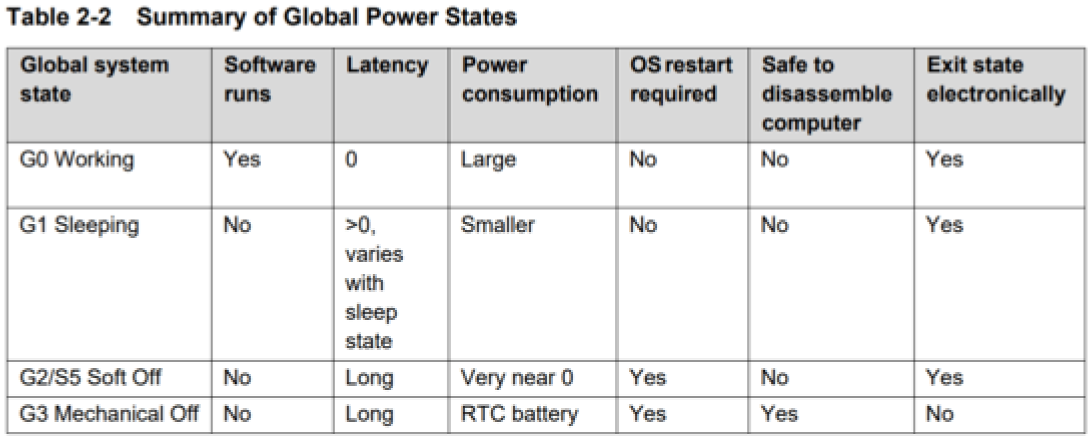


Table 2-2 Summary of Global Power States

Global system state	Software runs	Latency	Power consumption	OS restart required	Safe to disassemble computer	Exit state electronically
G0 Working	Yes	0	Large	No	No	Yes
G1 Sleeping	No	>0, varies with sleep state	Smaller	No	No	Yes
G2/S5 Soft Off	No	Long	Very near 0	Yes	No	Yes
G3 Mechanical Off	No	Long	RTC battery	Yes	Yes	No

Summary of Global Power States (from ACPI Specification v6.2)

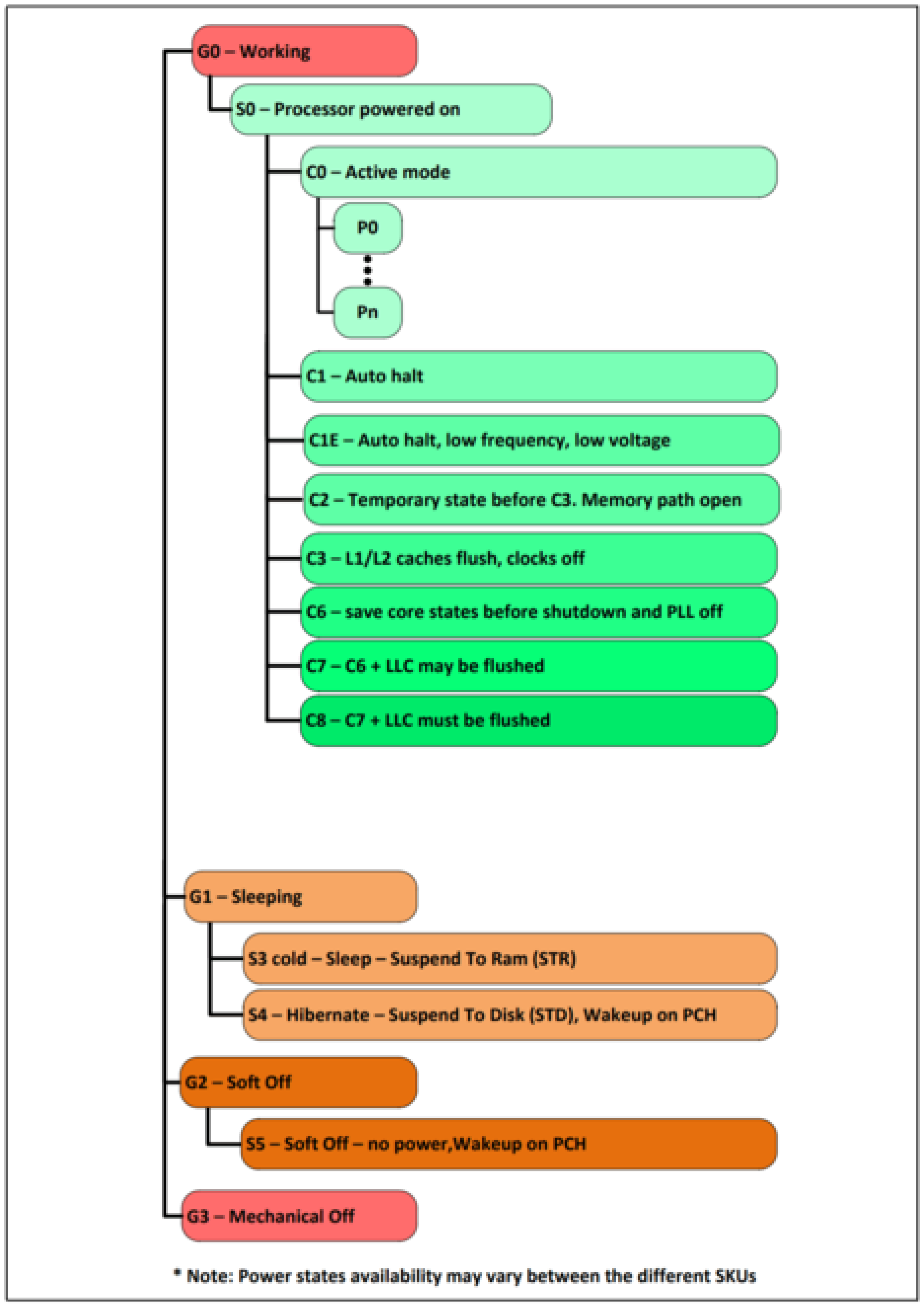
There is also a special global state G1/S4, Non-Volatile Sleep, where the system state is saved to non-volatile storage (such as disk) and then powered off. This makes it possible to consume minimal amount of power like Soft Off state, but returning from this state to G0 is possible without restart. It is what we know as Hibernation or Suspend to Disk.

Then there is the sleeping states (Sx states). Including S0, no sleep state, there are 6 sleeping states. S1-S4 are used with G1, and S5 is the Soft Off sleeping state used with G2. So a short summary:

- GO/S0: Computer is running, not sleeping.
- G1: Sleeping
  - G1/S1: Power on Suspend. The system state is preserved, so CPU and CPU Caches are still powered.
  - G1/S2: CPU is powered off. So CPU and CPU Caches are lost.
  - G1/S3: Standby, Sleep or Suspend to RAM (STR). System RAM remains powered.
  - G1/S4: Hibernation or Suspend to Disk. Everthing is saved to non-volatile memory (like disk), so RAM and possibly all the system is powered off.
- G2/S5: Soft Off. Like mechanical off but things that can wake up the computer from power save is minimally powered. No state is saved, so a reboot is needed to go back to G0.
- G3: Mechanical Off. The Power Supply Unit is disconnected (e.g. via a power switch). Only the things like real-time clock (RTC) is running because they have their own small batteries. Obviously no state is saved, so a reboot is needed to go back to G0.

For my CPU, see the Figure below, all of the C-states mentioned above is in ACPI G0/S0. Basically what it says is, once you go to any sleeping mode (G1), the CPU (package) is powered down.

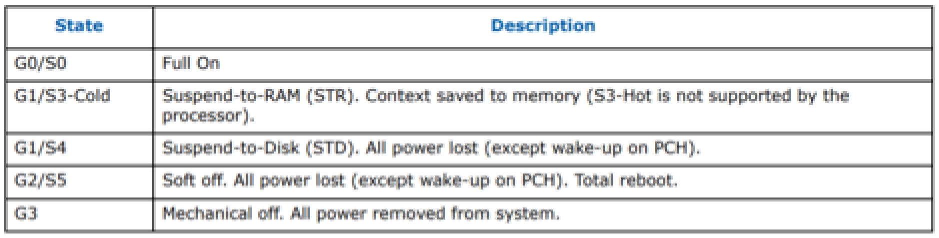
Figure 4-1. Processor Power States


G0 – Working
S0 – Processor powered on
C0 – Active mode
P0
Pn
C1 – Auto halt
C1E – Auto halt, low frequency, low voltage
C2 – Temporary state before C3. Memory path open
C3 – L1/L2 caches flush, clocks off
C6 – save core states before shutdown and PLL off
C7 – C6 + LLC may be flushed
C8 – C7 + LLC must be flushed
G1 – Sleeping
S3 cold – Sleep – Suspend To Ram (STR)
S4 – Hibernate – Suspend To Disk (STD), Wakeup on PCH
G2 – Soft Off
S5 – Soft Off – no power,Wakeup on PCH
G3 – Mechanical Off
* Note: Power states availability may vary between the different SKUs

Processor Power States of Intel Xeon Processor E3-1200 v5 Product Family

So the supported ACPI states are:

Table 4-1. System States



State	Description
G0/S0	Full On
G1/S3-Cold	Suspend-to-RAM (STR). Context saved to memory (S3-Hot is not supported by the processor).
G1/S4	Suspend-to-Disk (STD). All power lost (except wake-up on PCH).
G2/S5	Soft off. All power lost (except wake-up on PCH). Total reboot.
G3	Mechanical off. All power removed from system.

ACPI System States supported by Intel Xeon Processor E3–1200 v5 Product Family

ACPI G/S state and CPU C-state combinations

It is nice to see all these combinations in a table:

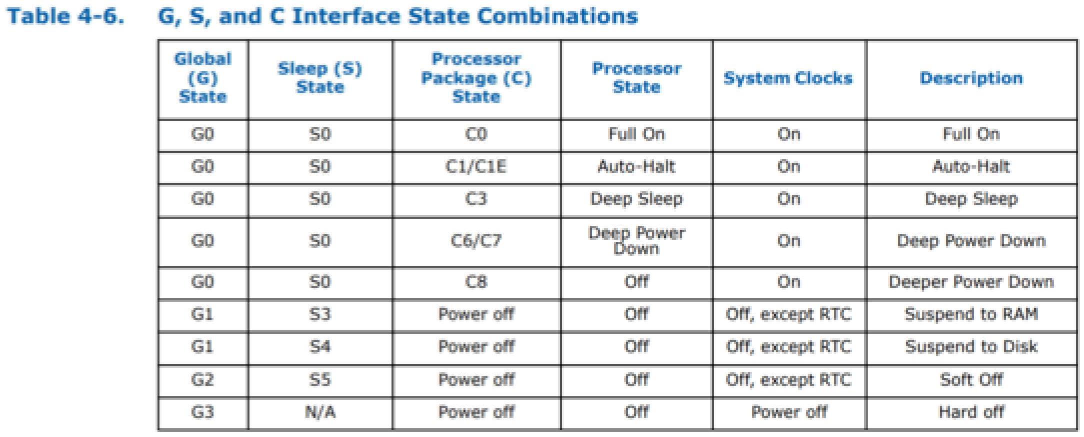


Table 4-6. G, S, and C Interface State Combinations

Global (G) State	Sleep (S) State	Processor Package (C) State	Processor State	System Clocks	Description
G0	S0	C0	Full On	On	Full On
G0	S0	C1/C1E	Auto-Halt	On	Auto-Halt
G0	S0	C3	Deep Sleep	On	Deep Sleep
G0	S0	C6/C7	Deep Power Down	On	Deep Power Down
G0	S0	C8	Off	On	Deeper Power Down
G1	S3	Power off	Off	Off, except RTC	Suspend to RAM
G1	S4	Power off	Off	Off, except RTC	Suspend to Disk
G2	S5	Power off	Off	Off, except RTC	Soft Off
G3	N/A	Power off	Off	Power off	Hard off

G, S, and C Interface State Combinations of Intel Xeon Processor E3–1200 v5 Product Family

In G0/S0/C8, only the processor package is powered but all of the cores are powered down.

In G1 (S3 or S4), there are no (valid) C-state (no CC no PC), because the CPU is completely powered down.

For G3, there are no S-state. The system is not sleeping, it is mechnically powered down, it cannot be waken up, it has to be powered on first.

How to programmatically request a lower-power C-state ?

The modern (but not only) way to request a low power state (thus state change from C0 to others) is to use MWAIT or HLT instructions. These are priviliged instructions, they cannot be executed by user programs.

MWAIT (Monitor Wait) instructs the processor to enter an optimized state (C-state) while waiting for a write/write to a specified address range (set up by another instruction, MONITOR). For power management, MWAIT is used with EAX, and EAX[7:4] bits indicate the target C-state and EAX[3:0] indicate the sub C-state.

> Note: I believe at the moment only AMD has this, but there are also MONITORX/MWAITX instructions which, in addition to monitoring the write on an address-range, also checks the expiration of a timer. This is also called Timed MWAIT.

HLT (Halt) instruction stops execution and the core goes to HALT state until an interrupt occurs. This means the core switches to C1 or C1E state.

What triggers a CPU core to enter Cx-state ?

A simple answer is this:

- C0 is entered at boot, when an interrupt happens or a write event occurs on the address monitored by an MWAIT instruction.

- C1/C1E is entered with HLT or MWAIT instructions.
- C3 is entered with MWAIT instruction, then L1 and L2 caches are flushed to LLC and all core clocks are stopped. However, the core maintains its state since it is still powered.
- C6 is entered with MWAIT instruction, then the processor state is saved into a dedicated SRAM and the voltage to core is reduced to zero. At this state, core has no power. When exiting C6, the processor state is restored back from the SRAM.
- C7 and C8 are same as C6 for the core.

  Just to remind again, I omitted the hyper-threading in this answer.

As I showed in the chart before, the transitions to/from the deep C-states have higher latencies and energy costs. Thus such transitions should be done carefully especially on battery powered devices.

Is it possible to disable C-states (so always C0 is used) ?

It is possible but not recommended. In the datasheet (section 4.2.2 page 64) there is this note: "Long term reliability cannot be assured unless all the Low-Power Idle States are enabled". So you should really not disable C-states.

How does an interrupt affect the processor/core in a sleeping state ?

When an interrupt is triggered, corresponding core has to be woken up and put into C0 state. However, for example, Intel Xeon E3–1200 v5 has a feature called "Power Aware Interrupt Routing (PAIR)" which has two benefits:

- For power saving, the interrupt can be routed to an active core to not wake up an idle core
- For performance, the interrupt can be routed to an idle (C1) core rather than an already (heavily) working core

P-states

P-states means the CPU core is also in C0 state because it has to be powered to execute a code. P-states basically allow to change the voltage and frequency (in other words operating point) of the CPU core to decrease the power consumption. There are a set of P-states which corresponds to different operating points (voltage-frequency pairs), and a P-state refers one such operating point. The highest (frequency and voltage) operating point is the maximum performance state which is P0.

It is possible to use P-states in Intel Xeon E3–1200 v5 in either OS/Operating System controlled (Intel SpeedStep® Technology) or HW/Hardware controlled (Intel® Speed Shift Technology) mode. The information below about P-states is specific to Intel Xeon E3–1200 v5 Family, however, I guess it is same or similar for other modern processors.

OS-controlled P-states

In this method, OS is aware of the P-states and a specific P-state is requested by the OS. This basically means selecting an operating frequency, whereas the voltage is automatically selected by the processor depending on the frequency and other factors. After a P-state is requested by writing it to a model specific register (meaning a 16-bit value is written to IA32_PERF_CTL), the voltage is transitioned to an automatically calculated value and the clock generator (PLL) locks to the frequency requested. All cores share the same P-state, so it is not possible to set this individually for a core. The current P-state / operating point can be read from another model specific register (IA32_PERF_STATUS).

The P-state transition is rapid, so many transitions per second can be performed. This is very different than C-state transitions which have higher latency and power costs.

HW-controlled P-states

In this method, OS is aware of the hardware support for the control P-states and it makes a request specifying only workload requirements, no specific P-state or frequency is requested. Based on the hints given by OS, and many other factors and constraints, a P-state is selected by the hardware.

I am going to say more about this in another post, but just to give you an idea, my current Linux desktop operates in this mode, which I understand by checking IA32_PM_ENABLE, and the maximum (non-guaranteed) performance level is 39, and the lowest performance level is 1. This roughly means there are 39 different P-states. At the moment, 39 is requested by the OS both as minimum and maximum performance and the preference is on performance, these are all because I have disabled the dynamic CPU frequency change in the kernel.

A note about Intel® Turbo Boost

Since the TDP (thermal design power) is the maximum power CPU can sustain, when the power consumption is below this value and under certain other conditions, CPU frequency can be increased beyond base frequency (from 3.50 GHz to 3.90 GHz in this CPU) since this condition does not increase power consumption beyond TDP. Turbo Boost can also temporarily increase power consumption up to PL2/Power Limit 2 for a short period of time. The behaviour of Turbo Boost can be modified by giving hints to hardware.

Does the information about C-states and P-states above applies to mobile/laptop processors or embedded/phone processors ?

As an example, a recent MacBook Air processor i5–5350U has mainly all the features I described above (I am not sure about HW-controlled P-states). I also looked at ARM Cortex-A documentation, and although the terms are different, it looks like it has very similar power control mechanisms.

How does all these actually work, for example in Linux ?

That is going to be answered in another post, stay tuned.

How can I monitor my CPU ?

There are not many applications you can see these information, but you can, for example, use CoreFreq.

Here displaying the system information. (below is a part of all the output)

```
$ ./corefreq-cli -s
Processor                            [Intel(R) Xeon(R) CPU E3-1245 v5 @ 3.50GHz]
|- Architecture                                                      [Skylake/S]
|- Vendor ID                                                      [GenuineIntel]
|- Microcode                                                        [        198]
|- Signature                                                            [ 06_5E]
|- Stepping                                                             [     3]
|- Online CPU                                                           [  4/4 ]
|- Base Clock                                                           [100.12]
|- Frequency            (MHz)                     Ratio
                 Min     800.94                  [   8 ]
                 Max    3504.10                  [  35 ]
|- Factory                                                              [100.00]
                        3500                     [  35 ]
|- Turbo Boost                                                          [UNLOCK]
                  1C    3904.57                  <  39 >
                  2C    3804.45                  <  38 >
                  3C    3704.33                  <  37 >
                  4C    3604.22                  <  36 >
|- Uncore                                                               [UNLOCK]
                 Min     800.94                  <   8 >
                 Max    3904.57                  <  39 >

...

Technologies:
|- System Management Mode                                 SMM-Dual       [ ON]
|- Hyper-Threading                                             HTT       [OFF]
|- SpeedStep                                                  EIST       < ON>
|- Dynamic Acceleration                                        IDA       [ ON]
|- Turbo Boost                                               TURBO       < ON>
|- Virtualization                                              VMX       [ ON]
   |- I/O MMU                                                VT-d       [OFF]
   |- Hypervisor                                                         [OFF]

Performance Monitoring:
|- Version                                                      PM       [  4]
|- Counters:          General                   Fixed
|                     8 x 48 bits             3 x 48 bits
|- Enhanced Halt State                                         C1E       <OFF>
|- C1 Auto Demotion                                            C1A       < ON>
|- C3 Auto Demotion                                            C3A       < ON>
|- C1 UnDemotion                                               C1U       < ON>
|- C3 UnDemotion                                               C3U       < ON>
|- Frequency ID control                                        FID       [OFF]
|- Voltage ID control                                          VID       [OFF]
|- P-State Hardware Coordination Feedback              MPERF/APERF       [ ON]
|- Hardware-Controlled Performance States                      HWP       [ ON]
|- Hardware Duty Cycling                                       HDC       [ ON]
|- Package C-State
   |- Configuration Control                                 CONFIG   [   LOCK]
   |- Lowest C-State                                         LIMIT   [      0]
   |- I/O MWAIT Redirection                                IOMWAIT   [Disable]
   |- Max C-State Inclusion                                  RANGE   [      0]
|- MWAIT States:    C0     C1     C2     C3     C4     C5     C6     C7
|                    0      2      1      2      4      1      0      0
```

```
|- Core Cycles                                                    [Present]
|- Instructions Retired                                           [Present]
|- Reference Cycles                                               [Present]
|- Last Level Cache References                                    [Present]
|- Last Level Cache Misses                                        [Present]
|- Branch Instructions Retired                                    [Present]
|- Branch Mispredicts Retired                                     [Present]

Power & Thermal Monitoring:
|- Clock Modulation                                        ODCM   <Disable>
   |- DutyCycle                                                   <  6.25%>
|- Power Management                                    PWR MGMT   [   LOCK]
   |- Energy Policy                                   Bias Hint   [      0]
|- Junction Temperature                                   TjMax   [  0:100]
|- Digital Thermal Sensor                                   DTS   [Present]
|- Power Limit Notification                                 PLN   [Present]
|- Package Thermal Management                               PTM   [Present]
|- Thermal Monitor 1                                    TM1|TTP   [ Enable]
|- Thermal Monitor 2                                    TM2|HTC   [Present]
|- Units
   |- Power                                         watt   [  0.125000000]
   |- Energy                                       joule   [  0.000061035]
   |- Window                                      second   [  0.000976562]
```

Here displaying the information about the kernel, including the idle driver: (below is a part of all the output)

```
$ ./corefreq-cli -k
Linux:
|- Release                                            [4.15.0-45-generic]
|- Version                          [#48-Ubuntu SMP Tue Jan 29 16:28:13 UTC 2019]
|- Machine                                                       [x86_64]
...
Idle driver                                                 [@intel_idle]
   |- State:          POLL    C1      C1E     C3      C6      C7s     C8
   |- Power:          -1      0       0       0       0       0       0
   |- Latency:        0       2       10      70      85      124     200
   |- Residency:      0       2       20      100     200     800     800
```

Here monitoring the package:

```
$ ./corefreq-cli -g
                Cycles          State(%)
PC02            1121802850        32.49
PC03            1298328500        37.83
PC06                     0         0.00
PC07                     0         0.00
PC08                     0         0.00
PC09                     0         0.00
PC10                     0         0.00
PTSC            3503877892
UNCORE              150231
```

Here monitoring the counters for (core) C-states:

```
$ ./corefreq-cli -c
CPU Freq(MHz) Ratio  Turbo  C0(%)  C1(%)  C3(%)  C6(%)  C7(%)  Min TMP:TS  Max
#00  355.67 ( 3.55)  10.15  10.28  26.43   0.04  11.49  51.77  41 / 45:55 / 56
#01  355.64 ( 3.55)  10.15  10.38  19.21   0.68  15.44  54.28  42 / 45:55 / 55
#02  389.95 ( 3.89)  11.13  11.35  15.67   0.16  18.17  54.65  40 / 43:57 / 54
#03  365.38 ( 3.65)  10.43  10.61  19.77   0.18  13.93  55.51  40 / 43:57 / 54

    Averages:        Turbo  C0(%)  C1(%)  C3(%)  C6(%)  C7(%)    TjMax:    Pkg:
                     10.46  10.66  20.27   0.27  14.76  54.05     100 C    46 C
```

Here monitoring the power and voltage:

```
$ ./corefreq-cli -V
CPU Freq(MHz) VID  Vcore
#00  130.70     0  0.0000
#01  120.08     0  0.0000
#02  124.18     0  0.0000
#03  103.46  9784  1.1943

              Package        Cores          Uncore         Memory
Energy(J):   13.415222168    2.248596191    0.000000000    0.951416016
Power(W) :   26.830444336    4.497192383    0.000000000    1.902832031
```