

Activa tu SpeedStep nativo en OSX usando el DSDT.

Creado por EvOSX86 Team

Muchos son los Fixes que hemos ido mostrando a través de la edición del DSDT de nuestras máquinas, en estas líneas descubriréis como hacer que la función SpeedStep (EIST) funcione nativamente y regulada a través del archiconocido y antaño tan peligroso AppleIntelCPUPowmanagement.kext.

Como todos sabemos existen distintos ensambladores de Bios, los mas comunes Award-Phoenix y AMI difieren en la forma de implementar el standard ACPI en nuestros sistemas, debido a esto, el código a insertar variará un poco aunque el resultado final es practicamente el mismo, .

En las siguientes líneas os mostraré un par de ejemplos de código que insertados en el dsdt hará que podais activar speedstep en vuestros sistemas, si bien en algunos casos requerirá de algo más de trabajo por vuestra parte ya que será necesario que averigúeis los distintos pasos o power-steps (PSS) y para ello tendreis que utilizar un LiveCD linux\* para volcar unas determinadas tablas que están "hardcoded" en direcciones de memoria y que en OSX no es tan sencillo/posible hacerlo. Obviamente todas las funciones SpeedStep (EIST, C1E....) deberán estar activadas en la bios.

En el siguiente ejemplo os muestro código a insertar para una placa GA P35/P45 con bios Award y CPU Q6600

#### [code]

```
Scope (_PR)
{
    Processor (CPU0, 0x00, 0x00000410, 0x06)
    {
        Name (_CST, Package (0x04) // Método C-Stados de la CPU
        {
            0x02,
            Package (0x04)
            {
                ResourceTemplate ()
                {
                    Register (FFixedHW,
                        0x00, // Bit Width
                        0x00, // Bit Offset
                        0x0000000000000000, // Address
                    )
                },
                One,
                0x14,
                0x03E8
            },
            Package (0x04)
            {
                ResourceTemplate ()
                {
                    Register (SystemIO,
```

```

        0x08,          // Bit Width
        0x00,          // Bit Offset
        0x000000000000414, // Address
    )
},

0x02,
0x28,
0x02EE
},

Package (0x04)
{
    ResourceTemplate ()
    {
        Register (SystemIO,
            0x08,          // Bit Width
            0x00,          // Bit Offset
            0x000000000000415, // Address
        )
    },

    0x03,
    0x3C,
    0x01F4
}
})
Name (_PCT, Package (0x02))
{
    ResourceTemplate ()
    {
        Register (FFixedHW,
            0x00,          // Bit Width
            0x00,          // Bit Offset
            0x000000000000000, // Address
        )
    },

    ResourceTemplate ()
    {
        Register (FFixedHW,
            0x00,          // Bit Width
            0x00,          // Bit Offset
            0x000000000000000, // Address
        )
    }
})
Name (_PSS, Package (0x05) // Los distintos P-States)
{
    Package (0x06)
    {
        0x0A6B, // velocidad
        0x00015BA8, // Voltaje
        0xA0, // Transicion (latencia)
        0x0A, // Busmaster (latencia)
        0x0A27, // Control
        0x0A27 // Status
    },

    Package (0x06)
    {
        0x0960,
        0x00013498,
        0xA0,
        0x0A,
        0x0924,
        0x0924
    },

    Package (0x06)
    {
        0x0855,
        0x00011558,
        0xA0,
        0x0A,
        0x0821,
        0x0821
    },

    Package (0x06)
    {
        0x074B,
        0xF618,
        0xA0,
        0x0A,
        0x071E,
        0x071E
    },

    Package (0x06)
    {
        0x0640,
        0xD6D8,
        0xA0,
        0x0A,
        0x061B,

```

```

        0x061B
    }
})
Method (_PPC, 0, NotSerialized)
{
    Return (Zero)
}
}
(.....)

```

Esto se repetirá 3 veces más debido a que tenemos 4 Cores

```

}
[/code]

```

Os pongo Tb un ejemplo de los P-States de un C2D E8400

[code]

```

    Name (_PSS, Package (0x04) // esto indica que usaremos 4 P-states
    {
        Package (0x06)
        {
            0x0BB5,
            0xFDE8,
            0x0A,
            0x0A,
            0x0920,
            0x0922
        },
        Package (0x06)
        {
            0x0A68,
            0xD6D8,
            0x0A,
            0x0A,
            0x081E,
            0x0820
        },
        Package (0x06)
        {
            0x091B,
            0xAFC8,
            0x0A,
            0x0A,
            0x071A,
            0x071C
        },
        Package (0x06)
        {
            0x07CE,
            0x8CA0,
            0x0A,
            0x0A,
            0x0616,
            0x0618
        }
    })
Method (_PPC, 0, NotSerialized)
{
    Return (Zero)
}
}

```

[/code]

Seguidamente os muestro código para una placa MSI P45 con Bios AMI , esto es aplicable tb a casi todas las Asus tb...

[code]

```

External (PDC0)
External (CFGD)

Scope (_PR)
{
    Processor (P001, 0x01, 0x00000810, 0x06)
    {
        Method (_CST, 0, NotSerialized)
    }
}

```

```

{
  If (LAnd (And (CFGD, 0x01000000), LNot (And (PDC0, 0x10
  )))
  {
    Return (Package (0x02)
    {
      0x01,
      Package (0x04)
      {
        ResourceTemplate ()
        {
          Register (FFixedHW,
            0x00, // Bit Width
            0x00, // Bit Offset
            0x0000000000000000, // Address
            .)
        },
        0x01,
        0x9D,
        0x03E8
      }
    })
  }

  If (And (PDC0, 0x0300))
  {
    If (And (CFGD, 0x20))
    {
      Return (Package (0x03)
      {
        0x02,
        Package (0x04)
        {
          ResourceTemplate ()
          {
            Register (FFixedHW,
              0x01, // Bit Width
              0x02, // Bit Offset
              0x0000000000000000, // Address
              .)
          },
          0x01,
          0x01,
          0x03E8
        },
        Package (0x04)
        {
          ResourceTemplate ()
          {
            Register (FFixedHW,
              0x01, // Bit Width
              0x02, // Bit Offset
              0x0000000000000010, // Address
              .)
          },
          0x02,
          0x01,
          0x01F4
        }
      })
    }
  }

  If (And (CFGD, 0x20))
  {
    Return (Package (0x03)
    {
      0x02,
      Package (0x04)
      {
        ResourceTemplate ()
        {
          Register (FFixedHW,
            0x01, // Bit Width
            0x02, // Bit Offset
            0x0000000000000000, // Address
            .)
        },
        0x01,
        0x01,
        0x03E8
      },
      Package (0x04)
      {
        ResourceTemplate ()
        {
          Register (SystemIO,
            0x08, // Bit Width
            0x00, // Bit Offset
            0x000000000000000C, // Address

```



```

[ 0.000000] ACPI: IRQ2 used by override.
[ 0.000000] ACPI: IRQ9 used by override.
[ 0.000000] ACPI: HPET id: 0x8086a301 base: 0xfed00000
[ 0.000000] Using ACPI (MADT) for SMP configuration information
[ 0.017526] ACPI: Core revision 20080926
[ 0.019609] ACPI: Checking initramfs for custom DSDT
[ 0.520197] ACPI: bus type pci registered
[ 0.520708] ACPI: EC: Look up EC in DSDT
[ 0.530793] ACPI: Interpreter enabled
[ 0.530801] ACPI: (supports S0 S1 S3 S4 S5)
[ 0.530813] ACPI: Using IOAPIC for interrupt routing
[ 0.532613] PCI: MCFG area at e0000000 reserved in ACPI motherboard resources
[ 0.538253] ACPI: No dock devices found.
[ 0.538293] ACPI: PCI Root Bridge [PCI0] (0000:00)
[ 0.539006] pci 0000:00:1f.0: quirk: region 0800-087f claimed by ICH6 ACPI/GPIO/TCO
[ 0.539753] ACPI: PCI Interrupt Routing Table [_SB_.PCI0._PRT]
[ 0.539887] ACPI: PCI Interrupt Routing Table [_SB_.PCI0.P0P2._PRT]
[ 0.539950] ACPI: PCI Interrupt Routing Table [_SB_.PCI0.P0P1._PRT]
[ 0.540070] ACPI: PCI Interrupt Routing Table [_SB_.PCI0.P0P4._PRT]
[ 0.540135] ACPI: PCI Interrupt Routing Table [_SB_.PCI0.P0P8._PRT]
[ 0.540201] ACPI: PCI Interrupt Routing Table [_SB_.PCI0.P0P9._PRT]
[ 0.550384] ACPI: PCI Interrupt Link [LNKA] (IRQs 3 4 5 6 7 10 *11 12 14 15)
[ 0.550465] ACPI: PCI Interrupt Link [LNKB] (IRQs 3 4 5 6 7 *10 11 12 14 15)
[ 0.550546] ACPI: PCI Interrupt Link [LNKC] (IRQs 3 4 *5 6 7 10 11 12 14 15)
[ 0.550626] ACPI: PCI Interrupt Link [LNKD] (IRQs 3 4 5 6 7 10 11 12 *14 15)
[ 0.550706] ACPI: PCI Interrupt Link [LNKE] (IRQs 3 4 5 6 7 10 11 12 14 15) *0, disabled.
[ 0.550787] ACPI: PCI Interrupt Link [LNKF] (IRQs *3 4 5 6 7 10 11 12 14 15)
[ 0.550867] ACPI: PCI Interrupt Link [LNKG] (IRQs 3 4 5 6 7 10 11 12 14 *15)
[ 0.550947] ACPI: PCI Interrupt Link [LNKH] (IRQs 3 4 5 6 *7 10 11 12 14 15)
[ 0.551031] ACPI Warning (tbutils-0217): Incorrect checksum in table [OEMB] - 77, should be 76 [20080926]
[ 0.551063] ACPI: WMI: Mapper loaded
[ 0.551090] PCI: Using ACPI for IRQ routing
[ 0.572006] pnp: PnP ACPI init
[ 0.572012] ACPI: bus type pnp registered
[ 0.574023] pnp: PnP ACPI: found 14 devices
[ 0.574024] ACPI: ACPI bus type pnp unregistered
[ 0.574026] PnPBIOS: Disabled by ACPI PNP
[ 1.001181] ACPI: Power Button (FF) [PWRF]
[ 1.001211] ACPI: Power Button (CM) [PWRB]
[ 1.001528] ACPI: SSDT CFF8E0D0, 01D2 (r1 AMI CPU1PM 1 INTL 20060113)
[ 1.001859] processor ACPI_CPU:00: registered as cooling_device0
[ 1.002028] ACPI: SSDT CFF8E2B0, 0143 (r1 AMI CPU2PM 1 INTL 20060113)
[ 1.002346] processor ACPI_CPU:01: registered as cooling_device1
[ 1.002513] ACPI: SSDT CFF8E400, 0143 (r1 AMI CPU3PM 1 INTL 20060113)
[ 1.002828] processor ACPI_CPU:02: registered as cooling_device2
[ 1.002991] ACPI: SSDT CFF8E550, 0143 (r1 AMI CPU4PM 1 INTL 20060113)
[ 1.003305] processor ACPI_CPU:03: registered as cooling_device3

```

[/code]

Lo que nos tenemos que fijar es en estas ultimas tablas y direcciones

[code]

```

[ 1.001528] ACPI: SSDT CFF8E0D0, 01D2 (r1 AMI CPU1PM 1 INTL 20060113)
[ 1.001859] processor ACPI_CPU:00: registered as cooling_device0
[ 1.002028] ACPI: SSDT CFF8E2B0, 0143 (r1 AMI CPU2PM 1 INTL 20060113)
[ 1.002346] processor ACPI_CPU:01: registered as cooling_device1
[ 1.002513] ACPI: SSDT CFF8E400, 0143 (r1 AMI CPU3PM 1 INTL 20060113)
[ 1.002828] processor ACPI_CPU:02: registered as cooling_device2
[ 1.002991] ACPI: SSDT CFF8E550, 0143 (r1 AMI CPU4PM 1 INTL 20060113)

```

[/code]

ahora procederemos a volcar esas tablas usando acpidump, desde el terminal tecleamos

[code]

```

acpidump -a 0xCFF8E0D0 -l 0x01D2 > cpu0ist // uno de cada vez
acpidump -a 0xCFF8E2B0 -l 0x0143 > cpu1ist
acpidump -a 0xCFF8E400 -l 0x0143 > cpu2ist
acpidump -a 0xCFF8E550 -l 0x0143 > cpu3ist

```

[/code]

solo nos queda descompilar estos binarios usando nuestro IASL, para ello teclearemos

[code]iasl -d cpu0ist [/code]

con esto conseguiremos obtendremos un .dsl en el que encontraremos entre otros datos los PSS.

Ahora pensareis que ya hemos acabado, bueno si y no, nos queda comentar un dato importante, OSX usa unos perfiles de energía para cada tipo de modelo y cpu, estos perfiles se encuentran almacenados/ regulados por un plugin dentro del IOPPlatformPluginFamily.kext , para ser exactos ACPI\_SMC\_PlatformPlugin.kext , si editamos el info.plist de este plugin observaremos los distintos perfiles para los distintos modelos/sistemas mac. Tendremos que adaptar nuestra inyección Smbios para que encaje dentro de un perfil/modelo mac que se adapte a nuestra máquina o utilizar unos de los perfiles "unused" y adaptarlo a nosotros, en mi caso yo he elegido esta última opción y he modelado un legacy que inyecta los parámetros adecuados a mi sistema y que permanece intacto independientemente de los updates....