



Ahora ya podemos editar el DSDT.dsl usando un editor de texto de nuestra elección (se sugiere textmate)

Para este ejemplo utilizaremos un dsdt de un equipo portátil.

Nota: El código marcado en rojo no es utilizado.

El color azul reseña la parte del código que debemos adaptar a nuestra tarjeta de video.

#### Device (PEGP)

```
{
  Name (.ADR, 0x00010000)
  Device (GFX0)
  {
    Name (.ADR, Zero)
    Name (.SUN, One)
    Method (_DSM, 4, NotSerialized)
    {
      Store (Package (0x1A))
      {
        "@0,compatible",
        Buffer (0x0B)
        {
          "NVDA,NVMac"
        },
        "@0,device_type",
        Buffer (0x08)
        {
          "display"
        },
        "@0,name",
        Buffer (0x0F)
        {
          "NVDA,Display-A"
        },
        "@1,compatible",
        Buffer (0x0B)
        {
          "NVDA,NVMac"
        },
        "@1,device_type",
        Buffer (0x08)
        {
          "display"
        },
        "@1,name",
        Buffer (0x0F)
        {
          "NVDA,Display-B"
        },
        "NVCAP", ←----- Nuestro NVCAP para salidas múltiples extraído de IOREG.
        Buffer (0x18)
        {
          /* 0000 */ 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00,
          /* 0008 */ 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
          /* 0010 */ 0x00, 0x00, 0x00, 0x00
        },
        "NVPM", ←----- Nuestro NVPM (gestión de energía), normalmente no hay que cambiarlo, pero es bueno compararlo en IOREG.
        Buffer (0x20)
        {
          /* 0000 */ 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
          /* 0008 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
          /* 0010 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
          /* 0018 */ 0x00, 0x00, 0x00, 0x00
        },
        "VRAM,totalsize",
        Buffer (0x04)
        {
          0x00, 0x00, 0x00, 0x20
        },
        "device_type",
        Buffer (0x0D)
        {
          "NVDA,GeForce"
        },
        "model",
        Buffer (0x1F) ←----- Dependerá de la longitud del nombre + los espacios + 1.
        {
          "Nvidia GeForce 8600M GS 256 MB" ←----- El nombre de nuestra tarjeta y su memoria (a vuestra elección).
        },
        "rom-revision",
        Buffer (0x20)
        {
          "DSDT ROM v.1a #irc.osx86.es (c)"
        },
        "REG", ←----- Nuestro REG extraído de IOREG.
        Buffer (0x78)
        {
          /* 0000 */ 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
          /* 0008 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
          /* 0010 */ 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x01, 0x02,
          /* 0018 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
          /* 0020 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
          /* 0028 */ 0x14, 0x00, 0x01, 0x42, 0x00, 0x00, 0x00, 0x00,
          /* 0030 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
          /* 0038 */ 0x00, 0x00, 0x00, 0x10, 0x1C, 0x00, 0x01, 0x02,
          /* 0040 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
          /* 0048 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02,
          /* 0050 */ 0x24, 0x00, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00,
          /* 0058 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
          /* 0060 */ 0x80, 0x00, 0x00, 0x00, 0x30, 0x00, 0x01, 0x02,
          /* 0068 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
          /* 0070 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00
        },
      }, Local0)
      DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
      Return (Local0)
    }
  }
}
```

## Renombrado de nuestra tarjeta en el código foraneo.

Si os fijáis en las letras en azul que se sugiere, veréis que están marcadas el Buffer y el nombre de la tarjeta, ¿porque?, pues es simple, deberéis de cambiar el nombre de la tarjeta por el vuestro, y el buffer lo que indica es el número de caracteres (incluyendo los espacios) que hay en el nombre entrecomillado+1, si cambiáis el nombre, pero no el buffer, la compilación fallará.

Como averiguar el buffer (que además está en hexadecimal).  
Es bien sencillo, necesitamos un conversor decimal a hexadecimal y al contrario.

Una web que os puede venir muy conveniente es esta:

<http://www.parkenet.com/apl/HexDecConverter.html>  
o esta:  
<http://www.paulschou.com/tools/xlate/>

Bien, empecemos:

**Ejemplo de buffer para la línea "Nvidia GeForce 8600M GS 256 MB"**, el número de letras y espacios contenidos aquí es **30 (decimal)**, añadimos **1 al resultado (realmente lo que pasa es que la cuenta numérica empieza desde cero)**, lo que nos da **31**, usando las webs arriba indicadas, ponemos **31** en decimal y nos da como resultado **1F** en hexadecimal, es decir, el buffer empleado.

**Si cambiamos el nombre al de nuestra tarjeta, deberemos de usar este mismo planteamiento para calcular el tamaño del buffer.**

## 2a. El código cargador.

Daremos un repaso al concepto de código cargador del que ya hemos hablado en la anterior guía.

Nuestro código no es ACPI, es una cadena de información que añadiremos al código ACPI, y si la ponemos tal cual, nuestro registro OSX no la cargará al no ser esta compatible ACPI, para que esta sea cargada, tendremos que utilizar un "cargador", que no es más que un poco de código que definirá nuestra entrada de código y permitirá su carga por parte del sistema.

Donde insertar el "cargador"?

Esto es sencillo, debéis de buscar con el editor de texto una zona del código que se llama `_WAK`, y cuando termina la definición de la misma, encajar el código cargador.

### Código cargador:

```
Method (DTGP, 5, NotSerialized)
{
  If (LEqual (Arg0, Buffer (0x10))
  {
    /* 0000 */ 0xC6, 0xB7, 0xB5, 0xA0, 0x18, 0x13, 0x1C, 0x44,
    /* 0008 */ 0xB0, 0xC9, 0xFE, 0x69, 0x5E, 0xAF, 0x94, 0x9B
  })
  {
    If (LEqual (Arg1, One))
    {
      If (LEqual (Arg2, Zero))
      {
        Store (Buffer (One)
        {
          0x03
        }, Arg4)
        Return (One)
      }
      If (LEqual (Arg2, One))
      {
        Return (One)
      }
    }
  }
  Store (Buffer (One)
  {
    0x00
  }, Arg4)
  Return (Zero)
}
```

### Zona de carga:

El código de esta zona dependerá de la máquina, y básicamente tenéis que buscar el final del código `_WAK`, que terminará así:

### Return (Package (0x02))

```
{
  Zero,
  Zero
}
```

He incluir debajo el código cargador.

### EJEMPLO:

### Zona de carga más cargador:

```
Method (_WAK, 1, NotSerialized)
{
  PBXH (One, 0xAB)
  If (LOr (LEqual (Arg0, 0x03), LEqual (Arg0, 0x04)))
  {
    If (And (CFGD, 0x10000000))
    {
      If (LAnd (And (CFGD, 0xF0), LEqual (OSYS, 0x07D1)))
      {
        TRAP (0x3D)
      }
    }
  }
  If (LEqual (RP2D, Zero))
  {
    Notify (\_SB.PCI0.RP02, Zero)
  }
  If (LEqual (Arg0, 0x03)) {}
  If (LEqual (Arg0, 0x04))
  {
    \_SB.PCI0.LPCB.EC0.SELE ()
  }
}
```

```

    }
    PBXH (Zero, 0xCD)
    Return (Package (0x02)
    {
        Zero,
        Zero
    })
}

Method (DTGP, 5, NotSerialized)
{
    If (LEqual (Arg0, Buffer (0x10)
    {
        /* 0000 */ 0xC6, 0xB7, 0xB5, 0xA0, 0x18, 0x13, 0x1C, 0x44,
        /* 0008 */ 0xB0, 0xC9, 0xFE, 0x69, 0x5E, 0xAF, 0x94, 0x9B
    )))
    {
        If (LEqual (Arg1, One))
        {
            If (LEqual (Arg2, Zero))
            {
                Store (Buffer (One)
                {
                    0x03
                }, Arg4)
                Return (One)
            }

            If (LEqual (Arg2, One))
            {
                Return (One)
            }
        }
    }

    Store (Buffer (One)
    {
        0x00
    }, Arg4)
    Return (Zero)
}

```

Salváis el archivo y lo compilamos para ver si se han producido errores en el código.

Abrimos el terminal, nos logueamos como root y accedemos a la carpeta del dsdt (cd "ruta a la carpeta") seguidamente tecleamos

```
iasl -f dsdt.dsl
```

Si se producen errores, es que habéis puesto mal el cargador, si ocurren warnings, es que es posible que vengan de antes, no les hagáis mucho caso ahora.

Si no se producen errores, borrad el archivo DSDT.aml que se ha generado y proseguimos el trabajo dentro de DSDT.dsl

### 3. Comparación de nuestro loreg con el código añadido, Nvcaps, reg y renombrado.

Ahora procederemos a buscar los datos necesarios en nuestro IOREG para insertarlos en cada parte correspondiente en el código.

Buscaremos primero las NVCAPS o configuración de las salidas de nuestra tarjeta de video y el NVPM o gestión de energía.

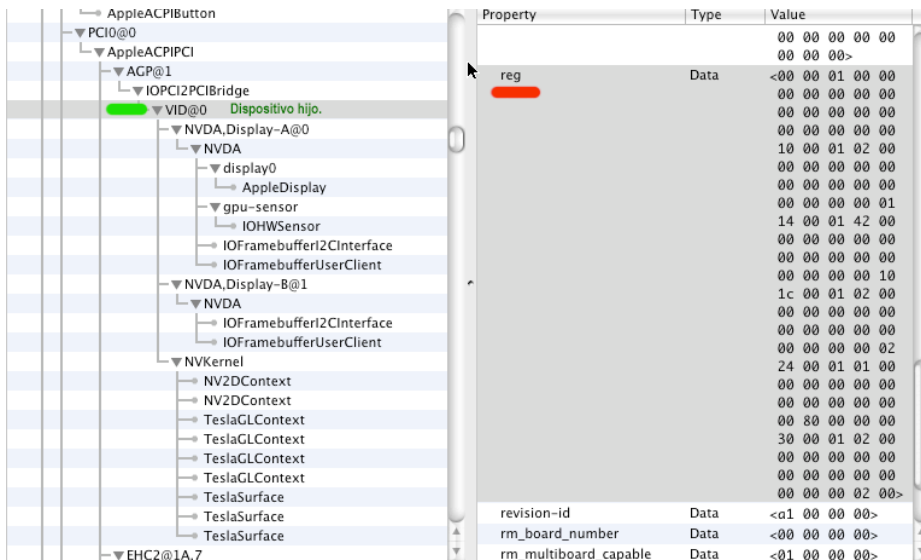
Abrimos el IOREG y buscamos nuestra tarjeta de video, señalando el dispositivo hijo y buscaremos la entrada NVCAP y la NVPM:

The screenshot shows the IORegistryExplorer interface. On the left, a tree view shows the hierarchy: AppleACPIButton -> PCI0@0 -> AppleACPIPCI -> AGP@1 -> IOPCI2PCIBridge -> VID@0 (highlighted as 'Dispositivo hijo'). Under VID@0, there are two NVDA devices: NVDA,Display-A@0 and NVDA,Display-B@1. The NVDA,Display-A@0 device is selected, and its properties are shown in the right pane. The properties include:

Property	Type	Value
IOPowerManagement	Dictionary	2 values
model	Data	<"NVIDIA GeForce 8600M GT">
name	Data	<"NVDA,Parent">
NVCAP	Data	<04 00 00 00 00 00 03 00 0c 00 00 00 00 00 00 07 00 00 00 00>
NVDA,current-arch	Number	0x84
NVDA,gart-width	Number	0x40
NVKernelLoaded	Data	<01 00 00 00>
NVPM	Data	<01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00>
reg	Data	<00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00>

Una vez tengamos los valores **NVCAP** y **NVPM**, los rectificaremos en nuestro código.

Ahora miremos el valor de la entrada **REG**:



Modificaremos el valor de **REG** en el código gráfico, para que tenga el valor real que tiene en nuestro sistema.

## 4. Estructura básica de dispositivos de video en el DSDT y añadido de nuestro código.

### 4a. Ejemplo utilizando un portátil.

Abramos el DSDT.dsl y busquemos mediante la palabra clave "device" hasta que lleguemos a:

**Device (AGP)**

Encontraremos el árbol de dispositivos de esta manera (diferentes colores para cada dispositivo):

#### Resumidamente sin nuestro código:

```

Device (AGP) (Dispositivo madre)
{
    Nombre y dirección.
    DEVICE VID0 (Dispositivo hijo)
}
{
    Nombre y dirección de VID0
    Inicio del método
    {
        Método
    }
    Device (TV)
    Device (CRT)
    Device (LCD)
    Device (DVI)
}

```

#### Resumidamente CON nuestro código:

```

Device (AGP) (Dispositivo madre)
{
    Nombre y dirección.
    DEVICE VID0 (Dispositivo hijo)
}
{
    Nombre y dirección de VID0
    Inicio del método
    {
        Método
    }
    Device (TV)
    Device (CRT)
    Device (LCD)
}

```

Device (DVI)

Device (GFX) <----- Nuestro codigo.

}

## Extendido:

Device (AGP)

```
{
  Name (_ADR, 0x00010000)
  Device (VID)
  {
    Name (_ADR, Zero)
    Method (_DOS, 1, NotSerialized)
    {
      Store (Arg0, MISA)
      SMI (0x0E, MISA)
    }

    Method (_DGD, 0, NotSerialized)
    {
      Store (SMI (0x6D, Zero), Local0)
      If (LEqual (Local0, One))
      {
        Return (Package (0x04)
          {
            0x00010100,
            0x00010200,
            0x00010110,
            0x00010210
          })
      }
      Else
      {
        Return (Package (0x04)
          {
            0x00010100,
            0x00010200,
            0x00010118,
            0x00010120
          })
      }
    }
  }
}
```

Device (TV)

```
{
  Method (_ADR, 0, NotSerialized)
  {
    Return (0x0200)
  }

  Method (_DCS, 0, NotSerialized)
  {
    Store (SMI (0x8E, 0x04), Local0)
    Return (Local0)
  }

  Method (_DGS, 0, NotSerialized)
  {
    Store (SMI (0x99, 0x04), Local0)
    Return (Local0)
  }

  Method (_DSS, 1, NotSerialized)
  {
    DSS (0x04, Arg0)
  }
}
```

Device (CRT)

```
{
  Method (_ADR, 0, NotSerialized)
  {
    Return (0x0100)
  }

  Method (_DCS, 0, NotSerialized)
  {
    Store (SMI (0x8E, 0x02), Local0)
    Return (Local0)
  }

  Method (_DGS, 0, NotSerialized)
  {
    Store (SMI (0x99, 0x02), Local0)
    Return (Local0)
  }

  Method (_DSS, 1, NotSerialized)
  {
    DSS (0x02, Arg0)
  }
}
```

Device (LCD)

```
{
  Method (_ADR, 0, NotSerialized)
  {
    Store (SMI (0x6D, Zero), Local0)
    If (LEqual (Local0, One))
    {
      Return (0x0110)
    }
    Else
    {
      Return (0x0118)
    }
  }

  Method (_DCS, 0, NotSerialized)
  {
    Store (SMI (0x8E, One), Local0)
    Return (Local0)
  }

  Method (_DGS, 0, NotSerialized)
  {
    Store (SMI (0x99, One), Local0)
    Return (Local0)
  }

  Method (_DSS, 1, NotSerialized)
  {
    DSS (One, Arg0)
  }

  Name (BTVL, 0x64)
  Name (DBCL_Package (0x04) 0)
  Method (_BCL, 0, NotSerialized)
  {
    SX10 ()
    SX30 (0x19)
    SX30 (Zero)
    SX11 ()
    Store (SX40 (), Index (DBCL, Zero))
    Store (SX40 (), Index (DBCL, One))
    Store (SX40 (), Index (DBCL, 0x02))
    Store (SX40 (), Index (DBCL, 0x03))
    Store (SX40 (), Index (DBCL, 0x04))
    Store (SX40 (), Index (DBCL, 0x05))
  }
}
```

```

Store (SX40 (), Index (DBCL, 0x06))
Store (SX40 (), Index (DBCL, 0x07))
Store (SX40 (), Index (DBCL, 0x08))
Store (SX40 (), Index (DBCL, 0x09))
SX12 ()
Return (DBCL)
}
}

Method (_BCM, 1, NotSerialized)
{
    SX10 ()
    SX30 (0x19)
    SX30 (One)
    SX30 (Arg0)
    Store (Arg0, BTVL)
    SX11 ()
    SX12 ()
}

Method (_BQC, 0, NotSerialized)
{
    SX10 ()
    SX30 (0x19)
    SX30 (0x02)
    SX11 ()
    Store (SX40 (), Local0)
    Store (Local0, BTVL)
    SX12 ()
    Return (Local0)
}
}

Device (DVI)
{
    Method (_ADR, 0, NotSerialized)
    {
        Store (SMI (0x0D, Zero), Local0)
        If (LEqual (Local0, One))
        {
            Return (0x0210)
        }
        Else
        {
            Return (0x0120)
        }
    }
}

Method (_DCS, 0, NotSerialized)
{
    Store (SMI (0x8E, 0x08), Local0)
    Return (Local0)
}

Method (_DGS, 0, NotSerialized)
{
    Store (SMI (0x90, 0x08), Local0)
    Return (Local0)
}

Method (_DSS, 1, NotSerialized)
{
    DSS (0x08, Arg0)
}
}

Device (GFX0) <----- Comienzo de nuestro codigo
{
    Name (_ADR, Zero)
    Name (_SUN, One)
    Method (_DSM, 4, NotSerialized)
    {
        Store (Package (0x1A)
        {
            "@0,compatible",
            Buffer (0x0B)
            {
                "NVDA,NVMac"
            },
            "@0,device_type",
            Buffer (0x08)
            {
                "display"
            },
            "@0,name",
            Buffer (0x0F)
            {
                "NVDA,Display-A"
            },
            "@1,compatible",
            Buffer (0x0B)
            {
                "NVDA,NVMac"
            },
            "@1,device_type",
            Buffer (0x08)
            {
                "display"
            },
            "@1,name",
            Buffer (0x0F)
            {
                "NVDA,Display-B"
            },
            "NVCAPI",
            Buffer (0x18)
            {
                /* 0000 */ 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00,
                /* 0008 */ 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                /* 0010 */ 0x00, 0x00, 0x00, 0x00
            },
            "NVPM",
            Buffer (0x20)
            {
                /* 0000 */ 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                /* 0008 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                /* 0010 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                /* 0018 */ 0x00, 0x00, 0x00, 0x00
            },
            "VRAM,totalsize",
            Buffer (0x04)
            {
                0x00, 0x00, 0x00, 0x20
            },
            "device_type",
            Buffer (0x0D)
            {
                "NVDA,GeForce"
            },
            "model",
            Buffer (0x1F)
            {
                "Nvidia GeForce 8600M GS 256 MB"
            },
            "rom-revision",
            Buffer (0x20)

```



```

"@0,compatible",
Buffer (0x0B)
{
  "NVDA,NVMac"
},

"@0,device_type",
Buffer (0x08)
{
  "display"
},

"@0,name",
Buffer (0x0F)
{
  "NVDA,Display-A"
},

"@1,compatible",
Buffer (0x0B)
{
  "NVDA,NVMac"
},

"@1,device_type",
Buffer (0x08)
{
  "display"
},

"@1,name",
Buffer (0x0F)
{
  "NVDA,Display-B"
},

"NVCAP",
Buffer (0x18)
{
  /* 0000 */ 0x04, 0x00, 0x01, 0x00, 0x00, 0x00, 0x03, 0x00,
  /* 0008 */ 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07,
  /* 0010 */ 0x00, 0x00, 0x00, 0x00
},

"NVPM",
Buffer (0x20)
{
  /* 0000 */ 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  /* 0008 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  /* 0010 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  /* 0018 */ 0x00, 0x00, 0x00, 0x00
},

"VRAM,totalsize",
Buffer (0x04)
{
  0x00, 0x00, 0x00, 0x20
},

"device_type",
Buffer (0x0D)
{
  "NVDA,GeForce"
},

"model",
Buffer (0x22)
{
  "POV Nvidia GeForce 8800 GT 512 MB"
},

"rom-revision",
Buffer (0x23)
{
  "DSDT ROM v.1a by #irc.osx86.es (c)"
},

"reg",
Buffer (0x78)
{
  /* 0000 */ 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,
  /* 0008 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  /* 0010 */ 0x00, 0x00, 0x00, 0x00, 0x10, 0x00, 0x01, 0x02,
  /* 0018 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  /* 0020 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
  /* 0028 */ 0x14, 0x00, 0x01, 0x42, 0x00, 0x00, 0x00, 0x00,
  /* 0030 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  /* 0038 */ 0x00, 0x00, 0x00, 0x10, 0x1C, 0x00, 0x01, 0x02,
  /* 0040 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  /* 0048 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02,
  /* 0050 */ 0x24, 0x00, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00,
  /* 0058 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
  /* 0060 */ 0x80, 0x00, 0x00, 0x00, 0x30, 0x00, 0x01, 0x02,
  /* 0068 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

```

```

        /* 0070 */ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00
    }
    }, Local0)
    DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
    Return (Local0)
}
}
}

Method (_S3D, 0, NotSerialized)
{
    If (LEqual (OSFL, 0x02))
    {
        Return (0x02)
    }
    Else
    {
        Return (0x03)
    }
}

Method (_STA, 0, NotSerialized)
{
    Return (0x0F)
}

```

## 5. Compilación y prueba.

Ahora solo nos quedará salvar el archivo y compilarlo para ver si tiene errores de sintaxis o no.

Abrimos el terminal, nos logueamos como root y accedemos a la carpeta en donde tenemos nuestro dsdt (cd "ruta a la carpeta") tecleamos

```
iasl dsdt.dsl
```

Si se producen errores, es que habéis puesto mal el cargador, si ocurren warnings, es que es posible que vengan de antes, no les hagáis mucho caso ahora.

Si los tiene, aseguraos de que no hayan quedado corchetes huérfanos o borrados.

Si no los tiene, renombrad el archivo DSDT.aml que tenéis en el root del sistema a "DSDTold.aml" y copiad el nuevo (que se abra creado al compilar) en la raíz del disco.

Borramos el inyector que estemos utilizando (Nvkush, Nvinject ó NVdarwin) o la cadena EFI si es que usamos éste método y procedemos a reiniciar. Si nuestro emplazamiento ha sido el correcto y la codificación adecuada nuestra gráfica debería de estar perfectamente reconocida y con todo activado.

Si fallase el nuevo DSDT.aml, (por ejemplo se congela el sistema), podéis arrancar el sistema utilizando el comando DSDT=DSDTold.aml (introducido cuando veáis el arranque chameleon presionando F8 y seleccionando la instalación de Leopard).

Esperamos que esta tercera guía sobre las posibilidades del sistema DSDT haya sido de vuestro agrado. Estaremos actualizando las guías a medida que tengamos nuevos datos....

## ANEXO

En este anexo, veremos como optimizar un ordenador portátil, quitándole el código gráfico que no es usado por OSX, consiguiendo con esto una mayor fluidez en el video ya que sólo estará en el registro el código que es funcional y nuestro código añadido.

En la vista resumida podremos observar mejor el árbol de los devices, por lo que podremos determinar que debe quedarse y que debe de borrarse al no ser código útil y prepararlo para la inclusión de nuestro código de video.

En resumen, nuestro código debería de lucir así después de borrar los devices innecesarios:

### ANTES:

```

Device (AGP)
{
    Nombre y dirección.
    Cabezera de VIDEO
}
{
    Nombre y dirección de VIDEO
    Inicio del método
    {
        Método
    }
}
Device (TV)
Device (CRT)
Device (LCD)

```

```

Device (DVI)
}
}

```

## DESPUES:

```

Device (AGP)
{
  Nombre y dirección.
  Cabezera de VIDEO
}
Nombre y dirección de VIDEO
Inicio del método
{
  Método
}
}
}

```

## Extendido ANTES:

```

Device (AGP)
{
  Name (_ADR, 0x00010000)
  Device (VID)
  {
    Name (_ADR, Zero)
    Method (_DOS, 1, NotSerialized)
    {
      Store (Arg0, MISA)
      SMI (0x0E, MISA)
    }

    Method (_DOD, 0, NotSerialized)
    {
      Store (SMI (0x6D, Zero), Local0)
      If (LEqual (Local0, One))
      {
        Return (Package (0x04)
        {
          0x00010100,
          0x00010200,
          0x00010110,
          0x00010210
        })
      }
      Else
      {
        Return (Package (0x04)
        {
          0x00010100,
          0x00010200,
          0x00010118,
          0x00010120
        })
      }
    }
  }
}

Device (TV)
{
  Method (_ADR, 0, NotSerialized)
  {
    Return (0x0200)
  }

  Method (_DCS, 0, NotSerialized)
  {
    Store (SMI (0x8E, 0x04), Local0)
    Return (Local0)
  }

  Method (_DGS, 0, NotSerialized)
  {
    Store (SMI (0x90, 0x04), Local0)
    Return (Local0)
  }

  Method (_DSS, 1, NotSerialized)
  {
    DSS (0x04, Arg0)
  }
}

Device (CRT)
{
  Method (_ADR, 0, NotSerialized)
  {
    Return (0x0100)
  }

  Method (_DCS, 0, NotSerialized)
  {
    Store (SMI (0x8E, 0x02), Local0)
    Return (Local0)
  }

  Method (_DGS, 0, NotSerialized)
  {
    Store (SMI (0x90, 0x02), Local0)
    Return (Local0)
  }

  Method (_DSS, 1, NotSerialized)
  {
    DSS (0x02, Arg0)
  }
}

Device (LCD)
{
  Method (_ADR, 0, NotSerialized)
  {
    Store (SMI (0x6D, Zero), Local0)
    If (LEqual (Local0, One))
    {
      Return (0x0110)
    }
    Else
    {
      Return (0x0118)
    }
  }
}

```



}

Obviamente, si damos a compilar después de borrar todos esos devices, nos reportará un buen número de errores ya que habrán desaparecido muchas referencias, así que deberemos de buscar las líneas marcadas como errores y eliminarlos de la siguiente manera:

**Errores generados al compilar cuando eliminamos el device "LCD":**

```
/Users/xxx/Desktop/pruebas dsdt/Tools/DSDToriginal.dsl 721: Notify (\_SB.PCI0.AGP.VID.LCD, 0x86)
Error 4063 - Object does not exist ^ (\_SB.PCI0.AGP.VID.LCD)
```

```
/Users/xxx/Desktop/pruebas dsdt/Tools/DSDToriginal.dsl 726: Notify (\_SB.PCI0.AGP.VID.LCD, 0x87)
Error 4063 - Object does not exist ^ (\_SB.PCI0.AGP.VID.LCD)
```

Nos movemos a la línea 721 y 726 como nos indica y encontramos:

```
If (LGreaterEqual (OSID (), 0x20))
{
    If (And (Local0, 0x04))
    {
        Notify (\_SB.PCI0.AGP.VID.LCD, 0x86)
        Notify (\_SB.PCI0.VID.LCD, 0x86)
    }

    If (And (Local0, 0x02))
    {
        Notify (\_SB.PCI0.AGP.VID.LCD, 0x87)
        Notify (\_SB.PCI0.VID.LCD, 0x87)
    }
}
```

Si eliminamos esas dos entradas, y compilamos de nuevo, los errores habrán desaparecido.

```
If (LGreaterEqual (OSID (), 0x20))
{
    If (And (Local0, 0x04))
    {
        Notify (\_SB.PCI0.VID.LCD, 0x86)
    }

    If (And (Local0, 0x02))
    {
        Notify (\_SB.PCI0.VID.LCD, 0x87)
    }
}
```

Se recomienda compilar cada vez que eliminamos una entrada, para así poder corregir poco a poco nuestro código.

**Guía by:** Roisoft y Pere