```
/*
 * Intel ACPI Component Architecture
 * AML/ASL+ Disassembler version 20200925 (64-bit version)
 * Copyright (c) 2000 - 2020 Intel Corporation
 *
 * Disassembling to symbolic ASL+ operators
 *
 * Disassembly of iASLrht679.aml, Sun May 14 14:02:16 2023
 *
 * Original Table Header:
 *     Signature        "DSDT"
 *     Length           0x000089FE (35326)
 *     Revision         0x03
 *     Checksum         0x13
 *     OEM ID           "APPLE "
 *     OEM Table ID     "MacBookP"
 *     OEM Revision     0x00010000 (65536)
 *     Compiler ID      "INTL"
 *     Compiler Version 0x20140424 (538182692)
 */
DefinitionBlock ("", "DSDT", 3, "APPLE ", "MacBookP", 0x00010000)
{
    /*
     * iASL Warning: There were 8 external control methods found during
     * disassembly, but only 6 were resolved (2 unresolved). Additional
     * ACPI tables may be required to properly disassemble the code. This
     * resulting disassembler output file may not compile because the
     * disassembler did not know how many arguments to assign to the
     * unresolved methods. Note: SSDTs can be dynamically loaded at
     * runtime and may or may not be available via the host OS.
     *
     * In addition, the -fe option can be used to specify a file containing
     * control method external declarations with the associated method
     * argument counts. Each line of the file must be of the form:
     *     External (<method pathname>, MethodObj, <argument count>)
     * Invocation:
     *     iasl -fe refs.txt -d dsdt.aml
     *
     * The following methods were unresolved and many not compile properly
     * because the disassembler had to guess at the number of arguments
     * required for each:
     */
    External (_SB_.PCI0.AMPE, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.CMPE, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.CNHI, MethodObj)    // 0 Arguments
    External (_SB_.PCI0.PEG0.GFX0, UnknownObj)
    External (_SB_.PCI0.TGPE, MethodObj)    // 0 Arguments
```

```
External (_SB_.PCI0.UPCK, MethodObj)    // 0 Arguments
External (_SB_.PCI0.WTLT, MethodObj)    // 0 Arguments
External (_SB_.PCI0.XHC1, UnknownObj)
External (D318, FieldUnitObj)
External (D319, FieldUnitObj)
External (D31A, FieldUnitObj)
External (D31C, FieldUnitObj)
External (D320, FieldUnitObj)
External (D324, FieldUnitObj)
External (D418, FieldUnitObj)
External (D419, FieldUnitObj)
External (D41A, FieldUnitObj)
External (D41C, FieldUnitObj)
External (D420, FieldUnitObj)
External (D424, FieldUnitObj)
External (D518, FieldUnitObj)
External (D519, FieldUnitObj)
External (D51A, FieldUnitObj)
External (D51C, FieldUnitObj)
External (D520, FieldUnitObj)
External (D524, FieldUnitObj)
External (D618, FieldUnitObj)
External (D619, FieldUnitObj)
External (D61A, FieldUnitObj)
External (D61C, FieldUnitObj)
External (D620, FieldUnitObj)
External (D624, FieldUnitObj)
External (DP18, FieldUnitObj)
External (DP19, FieldUnitObj)
External (DP1A, FieldUnitObj)
External (DP1C, FieldUnitObj)
External (DP20, FieldUnitObj)
External (DP24, FieldUnitObj)
External (HDOS, MethodObj)    // Warning: Unknown method, guessing 0 argume
External (HNOT, MethodObj)    // Warning: Unknown method, guessing 1 argumer
External (NH10, FieldUnitObj)
External (NH14, FieldUnitObj)
External (PDC0, IntObj)
External (PDC1, IntObj)
External (PDC2, IntObj)
External (PDC3, IntObj)
External (PDC4, IntObj)
External (PDC5, IntObj)
External (PDC6, IntObj)
External (PDC7, IntObj)
External (UP18, FieldUnitObj)
External (UP19, FieldUnitObj)
```

External (UP1A, FieldUnitObj)
External (UP1C, FieldUnitObj)
External (UP20, FieldUnitObj)
External (UP24, FieldUnitObj)

Name (SP2O, 0x4E)
Name (SP1O, 0x164E)
Name (IO1B, 0x0600)
Name (IO1L, 0x70)
Name (IO2B, 0x0680)
Name (IO2L, 0x20)
Name (IO3B, 0x0290)
Name (IO3L, 0x10)
Name (SP3O, 0x2E)
Name (IO4B, 0x0A20)
Name (IO4L, 0x20)
Name (MCHB, 0xFED10000)
Name (MCHL, 0x4000)
Name (EGPB, 0xFED19000)
Name (EGPL, 0x1000)
Name (DMIB, 0xFED18000)
Name (DMIL, 0x1000)
Name (IFPB, 0xFED14000)
Name (IFPL, 0x1000)
Name (PEBS, 0xE0000000)
Name (PELN, 0x04000000)
Name (TTTB, 0xFED20000)
Name (TTTL, 0x00020000)
Name (SMBS, 0xEFA0)
Name (SMBL, 0x10)
Name (PBLK, 0x1810)
Name (PMLN, 0x80)
Name (LVL2, 0x1814)
Name (LVL3, 0x1815)
Name (LVL4, 0x1816)
Name (SMIP, 0xB2)
Name (GPBS, 0x0800)
Name (GPLN, 0x80)
Name (APCB, 0xFEC00000)
Name (APCL, 0x1000)
Name (PM30, 0x1830)
Name (SRCB, 0xFED1C000)
Name (SRCL, 0x4000)
Name (SUSW, 0xFF)
Name (HPTB, 0xFED00000)
Name (HPTC, 0xFED1C404)
Name (ACPH, 0xDE)

Name (ASSB, 0x00)
Name (AOTB, 0x00)
Name (AAXB, 0x00)
Name (PEHP, 0x01)
Name (SHPC, 0x01)
Name (PEPM, 0x01)
Name (PEER, 0x01)
Name (PECS, 0x01)
Name (ITKE, 0x00)
Name (DSSP, 0x00)
Name (FHPP, 0x01)
Name (FMBL, 0x01)
Name (FDTP, 0x02)
Name (BSH, 0x00)
Name (BEL, 0x01)
Name (BEH, 0x02)
Name (BRH, 0x03)
Name (BTF, 0x04)
Name (BHC, 0x05)
Name (BYB, 0x06)
Name (BPH, 0x07)
Name (BSHS, 0x08)
Name (BELS, 0x09)
Name (BRHS, 0x0A)
Name (BTFS, 0x0B)
Name (BEHS, 0x0C)
Name (BPHS, 0x0D)
Name (BTL, 0x10)
Name (BOF, 0x20)
Name (BEF, 0x21)
Name (BLLE, 0x22)
Name (BLLC, 0x23)
Name (BLCA, 0x24)
Name (TCGM, 0x01)
Name (TRTP, 0x01)
Name (TRTD, 0x02)
Name (TRTI, 0x03)
Name (GCDD, 0x01)
Name (DSTA, 0x0A)
Name (DSLO, 0x0C)
Name (DSLC, 0x0E)
Name (PITS, 0x10)
Name (SBCS, 0x12)
Name (SALS, 0x13)
Name (LSSS, 0x2A)
Name (SOOT, 0x35)
Name (PDBR, 0x4D)

```
Name (WOWE, 0x00)
Name (TAPD, 0x00)
OperationRegion (GNVS, SystemMemory, 0x8AD28A90, 0x0282)
Field (GNVS, AnyAcc, NoLock, Preserve)
{
    OSYS,   16,
    SMIF,   8,
    PRM0,   8,
    PRM1,   8,
    SCIF,   8,
    PRM2,   8,
    PRM3,   8,
    LCKF,   8,
    PRM4,   8,
    PRM5,   8,
    P80D,   32,
    LIDS,   8,
    PWRS,   8,
    DBGS,   8,
    THOF,   8,
    ACT1,   8,
    ACTT,   8,
    PSVT,   8,
    TC1V,   8,
    TC2V,   8,
    TSPV,   8,
    CRTT,   8,
    DTSE,   8,
    DTS1,   8,
    DTS2,   8,
    DTSF,   8,
    Offset (0x25),
    REVN,   8,
    Offset (0x28),
    APIC,   8,
    TCNT,   8,
    PCP0,   8,
    PCP1,   8,
    PPCM,   8,
    PPMF,   32,
    C67L,   8,
    NATP,   8,
    CMAP,   8,
    CMBP,   8,
    LPTP,   8,
    FDCP,   8,
    CMCP,   8,
```

CIRP, 8,
SMSC, 8,
W381, 8,
SMC1, 8,
EMAE, 8,
EMAP, 16,
EMAL, 16,
Offset (0x42),
MEFE, 8,
DSTS, 8,
Offset (0x46),
TPMP, 8,
TPME, 8,
MORD, 8,
TCGP, 8,
PPRP, 32,
PPRQ, 8,
LPPR, 8,
GTF0, 56,
GTF2, 56,
IDEM, 8,
GTF1, 56,
BDID, 8,
Offset (0x78),
OSCC, 8,
NEXP, 8,
SDGV, 8,
SDDV, 8,
Offset (0x81),
DSEN, 8,
ECON, 8,
GPIC, 8,
CTYP, 8,
L01C, 8,
VFN0, 8,
VFN1, 8,
ATMC, 8,
PTMC, 8,
ATRA, 8,
PTRA, 8,
PNHM, 32,
TBAB, 32,
TBAH, 32,
RTIP, 8,
TSOD, 8,
ATPC, 8,
PTPC, 8,

PFLV, 8,
BREV, 8,
DPBM, 8,
DPCM, 8,
DPDM, 8,
SDID, 8,
BLCP, 8,
BLCC, 8,
Offset (0xA5),
BLCT, 32,
BLCB, 32,
BICM, 8,
Offset (0xB2),
NHIB, 32,
T2PB, 32,
GVNV, 32,
MM64, 8,
DBGD, 8,
TBUP, 32,
TBDB, 32,
TBNH, 32,
TBD3, 32,
TBD4, 32,
TBD5, 32,
TBD6, 32,
NVME, 8,
Offset (0x12C),
SARV, 32,
ASLB, 32,
IMON, 8,
IGDS, 8,
CADL, 8,
PADL, 8,
CSTE, 16,
NSTE, 16,
DID9, 32,
DIDA, 32,
DIDB, 32,
IBTT, 8,
IPAT, 8,
IPSC, 8,
IBLC, 8,
IBIA, 8,
ISSC, 8,
IPCF, 8,
IDMS, 8,
IF1E, 8,

HVCO, 8,
NXD1, 32,
NXD2, 32,
NXD3, 32,
NXD4, 32,
NXD5, 32,
NXD6, 32,
NXD7, 32,
NXD8, 32,
GSMI, 8,
PAVP, 8,
GLID, 8,
KSV0, 32,
KSV1, 8,
BBAR, 32,
BLCS, 8,
BRTL, 8,
ALSE, 8,
ALAF, 8,
LLOW, 8,
LHIH, 8,
ALFP, 8,
AUDA, 32,
AUDB, 32,
AUDC, 32,
DIDC, 32,
DIDD, 32,
DIDE, 32,
DIDF, 32,
CADR, 32,
CCNT, 8,
Offset (0x1F4),
SGMD, 8,
SGFL, 8,
PWOK, 8,
HLRS, 8,
PWEN, 8,
PRST, 8,
CPSP, 32,
EECP, 8,
EVCP, 16,
XBAS, 32,
GBAS, 16,
SGGP, 8,
NVGA, 32,
NVHA, 32,
AMDA, 32,

```
    NDID,   8,
    DID1,   32,
    DID2,   32,
    DID3,   32,
    DID4,   32,
    DID5,   32,
    DID6,   32,
    DID7,   32,
    DID8,   32,
    OBS1,   32,
    OBS2,   32,
    OBS3,   32,
    OBS4,   32,
    OBS5,   32,
    OBS6,   32,
    OBS7,   32,
    OBS8,   32,
    LTRA,   8,
    OBFA,   8,
    LTRB,   8,
    OBFB,   8,
    LTRC,   8,
    OBFC,   8,
    SMSL,   16,
    SNSL,   16,
    P0UB,   8,
    P1UB,   8,
    P2UB,   8,
    EDPV,   8,
    NXDX,   32,
    DIDX,   32,
    PCSL,   8,
    SC7A,   8
}

Scope (\_SB)
{
    Name (PR00, Package (0x22)
    {
        Package (0x04)
        {
            0x001FFFFF,
            0x00,
            LNKF,
            0x00
        },
```

```
Package (0x04)
{
    0x001FFFFF,
    0x01,
    LNKD,
    0x00
},

Package (0x04)
{
    0x001FFFFF,
    0x02,
    LNKC,
    0x00
},

Package (0x04)
{
    0x001FFFFF,
    0x03,
    LNKA,
    0x00
},

Package (0x04)
{
    0x0003FFFF,
    0x00,
    LNKA,
    0x00
},

Package (0x04)
{
    0x0015FFFF,
    0x00,
    LNKE,
    0x00
},

Package (0x04)
{
    0x0015FFFF,
    0x01,
    LNKE,
    0x00
},
```

```
Package (0x04)
{
    0x0015FFFF,
    0x02,
    LNKF,
    0x00
},

Package (0x04)
{
    0x0015FFFF,
    0x03,
    LNKF,
    0x00
},

Package (0x04)
{
    0x0014FFFF,
    0x00,
    LNKA,
    0x00
},

Package (0x04)
{
    0x001DFFFF,
    0x00,
    LNKG,
    0x00
},

Package (0x04)
{
    0x001DFFFF,
    0x01,
    LNKD,
    0x00
},

Package (0x04)
{
    0x001DFFFF,
    0x02,
    LNKA,
    0x00
```

```
},

Package (0x04)
{
    0x001DFFFF,
    0x03,
    LNKC,
    0x00
},

Package (0x04)
{
    0x001AFFFF,
    0x00,
    LNKH,
    0x00
},

Package (0x04)
{
    0x001AFFFF,
    0x01,
    LNKF,
    0x00
},

Package (0x04)
{
    0x001AFFFF,
    0x02,
    LNKC,
    0x00
},

Package (0x04)
{
    0x001AFFFF,
    0x03,
    LNKD,
    0x00
},

Package (0x04)
{
    0x001BFFFF,
    0x00,
    LNKG,
```

```
    0x00
},

Package (0x04)
{
    0x0018FFFF,
    0x00,
    LNKE,
    0x00
},

Package (0x04)
{
    0x0019FFFF,
    0x00,
    LNKE,
    0x00
},

Package (0x04)
{
    0x0016FFFF,
    0x00,
    LNKA,
    0x00
},

Package (0x04)
{
    0x0016FFFF,
    0x01,
    LNKD,
    0x00
},

Package (0x04)
{
    0x0016FFFF,
    0x02,
    LNKC,
    0x00
},

Package (0x04)
{
    0x0016FFFF,
    0x03,
```

```
    LNKB,
    0x00
},

Package (0x04)
{
    0x001CFFFF,
    0x00,
    LNKA,
    0x00
},

Package (0x04)
{
    0x001CFFFF,
    0x01,
    LNKB,
    0x00
},

Package (0x04)
{
    0x001CFFFF,
    0x02,
    LNKC,
    0x00
},

Package (0x04)
{
    0x001CFFFF,
    0x03,
    LNKD,
    0x00
},

Package (0x04)
{
    0x0001FFFF,
    0x00,
    LNKA,
    0x00
},

Package (0x04)
{
    0x0001FFFF,
```

```
            0x01,
            LNKB,
            0x00
        },

        Package (0x04)
        {
            0x0001FFFF,
            0x02,
            LNKC,
            0x00
        },

        Package (0x04)
        {
            0x0001FFFF,
            0x03,
            LNKD,
            0x00
        },

        Package (0x04)
        {
            0x0002FFFF,
            0x00,
            LNKA,
            0x00
        }
    })
    Name (AR00, Package (0x22)
    {
        Package (0x04)
        {
            0x001FFFFF,
            0x00,
            0x00,
            0x15
        },

        Package (0x04)
        {
            0x001FFFFF,
            0x01,
            0x00,
            0x13
        },
```

```
Package (0x04)
{
    0x001FFFFF,
    0x02,
    0x00,
    0x12
},

Package (0x04)
{
    0x001FFFFF,
    0x03,
    0x00,
    0x10
},

Package (0x04)
{
    0x0003FFFF,
    0x00,
    0x00,
    0x10
},

Package (0x04)
{
    0x0014FFFF,
    0x00,
    0x00,
    0x10
},

Package (0x04)
{
    0x0015FFFF,
    0x00,
    0x00,
    0x14
},

Package (0x04)
{
    0x0015FFFF,
    0x01,
    0x00,
    0x14
},
```

```
Package (0x04)
{
    0x0015FFFF,
    0x02,
    0x00,
    0x15
},

Package (0x04)
{
    0x0015FFFF,
    0x03,
    0x00,
    0x15
},

Package (0x04)
{
    0x001DFFFF,
    0x00,
    0x00,
    0x16
},

Package (0x04)
{
    0x001DFFFF,
    0x01,
    0x00,
    0x13
},

Package (0x04)
{
    0x001DFFFF,
    0x02,
    0x00,
    0x10
},

Package (0x04)
{
    0x001DFFFF,
    0x03,
    0x00,
    0x12
```

```
},

Package (0x04)
{
    0x001AFFFF,
    0x00,
    0x00,
    0x17
},

Package (0x04)
{
    0x001AFFFF,
    0x01,
    0x00,
    0x15
},

Package (0x04)
{
    0x001AFFFF,
    0x02,
    0x00,
    0x12
},

Package (0x04)
{
    0x001AFFFF,
    0x03,
    0x00,
    0x13
},

Package (0x04)
{
    0x001BFFFF,
    0x00,
    0x00,
    0x16
},

Package (0x04)
{
    0x0018FFFF,
    0x00,
    0x00,
```

```
    0x14
},

Package (0x04)
{
    0x0019FFFF,
    0x00,
    0x00,
    0x14
},

Package (0x04)
{
    0x0016FFFF,
    0x00,
    0x00,
    0x10
},

Package (0x04)
{
    0x0016FFFF,
    0x01,
    0x00,
    0x13
},

Package (0x04)
{
    0x0016FFFF,
    0x02,
    0x00,
    0x12
},

Package (0x04)
{
    0x0016FFFF,
    0x03,
    0x00,
    0x11
},

Package (0x04)
{
    0x001CFFFF,
    0x00,
```

```
    0x00,
    0x10
},

Package (0x04)
{
    0x001CFFFF,
    0x01,
    0x00,
    0x11
},

Package (0x04)
{
    0x001CFFFF,
    0x02,
    0x00,
    0x12
},

Package (0x04)
{
    0x001CFFFF,
    0x03,
    0x00,
    0x13
},

Package (0x04)
{
    0x0001FFFF,
    0x00,
    0x00,
    0x10
},

Package (0x04)
{
    0x0001FFFF,
    0x01,
    0x00,
    0x11
},

Package (0x04)
{
    0x0001FFFF,
```

```
            0x02,
            0x00,
            0x12
        },

        Package (0x04)
        {
            0x0001FFFF,
            0x03,
            0x00,
            0x13
        },

        Package (0x04)
        {
            0x0002FFFF,
            0x00,
            0x00,
            0x10
        }
    })
    Name (PR04, Package (0x04)
    {
        Package (0x04)
        {
            0xFFFF,
            0x00,
            LNKA,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x01,
            LNKB,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x02,
            LNKC,
            0x00
        },
```

```
    Package (0x04)
    {
        0xFFFF,
        0x03,
        LNKD,
        0x00
    }
})
Name (AR04, Package (0x04)
{
    Package (0x04)
    {
        0xFFFF,
        0x00,
        0x00,
        0x10
    },

    Package (0x04)
    {
        0xFFFF,
        0x01,
        0x00,
        0x11
    },

    Package (0x04)
    {
        0xFFFF,
        0x02,
        0x00,
        0x12
    },

    Package (0x04)
    {
        0xFFFF,
        0x03,
        0x00,
        0x13
    }
})
Name (PR05, Package (0x04)
{
    Package (0x04)
    {
        0xFFFF,
```

```
            0x00,
            LNKB,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x01,
            LNKC,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x02,
            LNKD,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x03,
            LNKA,
            0x00
        }
    })
    Name (AR05, Package (0x04)
    {
        Package (0x04)
        {
            0xFFFF,
            0x00,
            0x00,
            0x11
        },

        Package (0x04)
        {
            0xFFFF,
            0x01,
            0x00,
            0x12
        },
```

```
    Package (0x04)
    {
        0xFFFF,
        0x02,
        0x00,
        0x13
    },

    Package (0x04)
    {
        0xFFFF,
        0x03,
        0x00,
        0x10
    }
})
Name (PR06, Package (0x04)
{
    Package (0x04)
    {
        0xFFFF,
        0x00,
        LNKC,
        0x00
    },

    Package (0x04)
    {
        0xFFFF,
        0x01,
        LNKD,
        0x00
    },

    Package (0x04)
    {
        0xFFFF,
        0x02,
        LNKA,
        0x00
    },

    Package (0x04)
    {
        0xFFFF,
        0x03,
        LNKB,
```

```
        0x00
    }
})
Name (AR06, Package (0x04)
{
    Package (0x04)
    {
        0xFFFF,
        0x00,
        0x00,
        0x12
    },

    Package (0x04)
    {
        0xFFFF,
        0x01,
        0x00,
        0x13
    },

    Package (0x04)
    {
        0xFFFF,
        0x02,
        0x00,
        0x10
    },

    Package (0x04)
    {
        0xFFFF,
        0x03,
        0x00,
        0x11
    }
})
Name (PR07, Package (0x04)
{
    Package (0x04)
    {
        0xFFFF,
        0x00,
        LNKD,
        0x00
    },
```

```
    Package (0x04)
    {
        0xFFFF,
        0x01,
        LNKA,
        0x00
    },

    Package (0x04)
    {
        0xFFFF,
        0x02,
        LNKB,
        0x00
    },

    Package (0x04)
    {
        0xFFFF,
        0x03,
        LNKC,
        0x00
    }
})
Name (AR07, Package (0x04)
{
    Package (0x04)
    {
        0xFFFF,
        0x00,
        0x00,
        0x13
    },

    Package (0x04)
    {
        0xFFFF,
        0x01,
        0x00,
        0x10
    },

    Package (0x04)
    {
        0xFFFF,
        0x02,
        0x00,
```

```
            0x11
        },

        Package (0x04)
        {
            0xFFFF,
            0x03,
            0x00,
            0x12
        }
    })
    Name (PR08, Package (0x04)
    {
        Package (0x04)
        {
            0xFFFF,
            0x00,
            LNKA,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x01,
            LNKB,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x02,
            LNKC,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x03,
            LNKD,
            0x00
        }
    })
    Name (AR08, Package (0x04)
    {
```

```
    Package (0x04)
    {
        0xFFFF,
        0x00,
        0x00,
        0x10
    },

    Package (0x04)
    {
        0xFFFF,
        0x01,
        0x00,
        0x11
    },

    Package (0x04)
    {
        0xFFFF,
        0x02,
        0x00,
        0x12
    },

    Package (0x04)
    {
        0xFFFF,
        0x03,
        0x00,
        0x13
    }
})
Name (PR09, Package (0x04)
{
    Package (0x04)
    {
        0xFFFF,
        0x00,
        LNKA,
        0x00
    },

    Package (0x04)
    {
        0xFFFF,
        0x01,
        LNKB,
```

```
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x02,
            LNKC,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x03,
            LNKD,
            0x00
        }
    })
    Name (AR09, Package (0x04)
    {
        Package (0x04)
        {
            0xFFFF,
            0x00,
            0x00,
            0x10
        },

        Package (0x04)
        {
            0xFFFF,
            0x01,
            0x00,
            0x11
        },

        Package (0x04)
        {
            0xFFFF,
            0x02,
            0x00,
            0x12
        },

        Package (0x04)
        {
```

```
            0xFFFF,
            0x03,
            0x00,
            0x13
        }
    })
    Name (PR0E, Package (0x04)
    {
        Package (0x04)
        {
            0xFFFF,
            0x00,
            LNKC,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x01,
            LNKD,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x02,
            LNKA,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x03,
            LNKB,
            0x00
        }
    })
    Name (AR0E, Package (0x04)
    {
        Package (0x04)
        {
            0xFFFF,
            0x00,
            0x00,
```

```
            0x12
        },

        Package (0x04)
        {
            0xFFFF,
            0x01,
            0x00,
            0x13
        },

        Package (0x04)
        {
            0xFFFF,
            0x02,
            0x00,
            0x10
        },

        Package (0x04)
        {
            0xFFFF,
            0x03,
            0x00,
            0x11
        }
    })
    Name (PR0F, Package (0x04)
    {
        Package (0x04)
        {
            0xFFFF,
            0x00,
            LNKD,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x01,
            LNKA,
            0x00
        },

        Package (0x04)
        {
```

```
            0xFFFF,
            0x02,
            LNKB,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x03,
            LNKC,
            0x00
        }
    })
    Name (AR0F, Package (0x04)
    {
        Package (0x04)
        {
            0xFFFF,
            0x00,
            0x00,
            0x13
        },

        Package (0x04)
        {
            0xFFFF,
            0x01,
            0x00,
            0x10
        },

        Package (0x04)
        {
            0xFFFF,
            0x02,
            0x00,
            0x11
        },

        Package (0x04)
        {
            0xFFFF,
            0x03,
            0x00,
            0x12
        }
```

```
})
Name (PR02, Package (0x04)
{
    Package (0x04)
    {
        0xFFFF,
        0x00,
        LNKA,
        0x00
    },

    Package (0x04)
    {
        0xFFFF,
        0x01,
        LNKB,
        0x00
    },

    Package (0x04)
    {
        0xFFFF,
        0x02,
        LNKC,
        0x00
    },

    Package (0x04)
    {
        0xFFFF,
        0x03,
        LNKD,
        0x00
    }
})
Name (AR02, Package (0x04)
{
    Package (0x04)
    {
        0xFFFF,
        0x00,
        0x00,
        0x10
    },

    Package (0x04)
    {
```

```
            0xFFFF,
            0x01,
            0x00,
            0x11
        },

        Package (0x04)
        {
            0xFFFF,
            0x02,
            0x00,
            0x12
        },

        Package (0x04)
        {
            0xFFFF,
            0x03,
            0x00,
            0x13
        }
    })
    Name (PR0A, Package (0x04)
    {
        Package (0x04)
        {
            0xFFFF,
            0x00,
            LNKB,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x01,
            LNKC,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x02,
            LNKD,
            0x00
        },
```

```
    Package (0x04)
    {
        0xFFFF,
        0x03,
        LNKA,
        0x00
    }
})
Name (AR0A, Package (0x04)
{
    Package (0x04)
    {
        0xFFFF,
        0x00,
        0x00,
        0x11
    },

    Package (0x04)
    {
        0xFFFF,
        0x01,
        0x00,
        0x12
    },

    Package (0x04)
    {
        0xFFFF,
        0x02,
        0x00,
        0x13
    },

    Package (0x04)
    {
        0xFFFF,
        0x03,
        0x00,
        0x10
    }
})
Name (PR0B, Package (0x04)
{
    Package (0x04)
    {
```

```
            0xFFFF,
            0x00,
            LNKC,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x01,
            LNKD,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x02,
            LNKA,
            0x00
        },

        Package (0x04)
        {
            0xFFFF,
            0x03,
            LNKB,
            0x00
        }
    })
    Name (AR0B, Package (0x04)
    {
        Package (0x04)
        {
            0xFFFF,
            0x00,
            0x00,
            0x12
        },

        Package (0x04)
        {
            0xFFFF,
            0x01,
            0x00,
            0x13
        },
```

```
    Package (0x04)
    {
        0xFFFF,
        0x02,
        0x00,
        0x10
    },

    Package (0x04)
    {
        0xFFFF,
        0x03,
        0x00,
        0x11
    }
})
Name (PR0C, Package (0x04)
{
    Package (0x04)
    {
        0xFFFF,
        0x00,
        LNKD,
        0x00
    },

    Package (0x04)
    {
        0xFFFF,
        0x01,
        LNKA,
        0x00
    },

    Package (0x04)
    {
        0xFFFF,
        0x02,
        LNKB,
        0x00
    },

    Package (0x04)
    {
        0xFFFF,
        0x03,
```

```
        LNKC,
        0x00
    }
})
Name (AR0C, Package (0x04)
{
    Package (0x04)
    {
        0xFFFF,
        0x00,
        0x00,
        0x13
    },

    Package (0x04)
    {
        0xFFFF,
        0x01,
        0x00,
        0x10
    },

    Package (0x04)
    {
        0xFFFF,
        0x02,
        0x00,
        0x11
    },

    Package (0x04)
    {
        0xFFFF,
        0x03,
        0x00,
        0x12
    }
})
Name (PR01, Package (0x0C)
{
    Package (0x04)
    {
        0xFFFF,
        0x00,
        LNKF,
        0x00
    },
```

```
Package (0x04)
{
    0xFFFF,
    0x01,
    LNKG,
    0x00
},

Package (0x04)
{
    0xFFFF,
    0x02,
    LNKH,
    0x00
},

Package (0x04)
{
    0xFFFF,
    0x03,
    LNKE,
    0x00
},

Package (0x04)
{
    0x0001FFFF,
    0x00,
    LNKG,
    0x00
},

Package (0x04)
{
    0x0001FFFF,
    0x01,
    LNKF,
    0x00
},

Package (0x04)
{
    0x0001FFFF,
    0x02,
    LNKE,
    0x00
```

```
    },

    Package (0x04)
    {
        0x0001FFFF,
        0x03,
        LNKH,
        0x00
    },

    Package (0x04)
    {
        0x0005FFFF,
        0x00,
        LNKC,
        0x00
    },

    Package (0x04)
    {
        0x0005FFFF,
        0x01,
        LNKE,
        0x00
    },

    Package (0x04)
    {
        0x0005FFFF,
        0x02,
        LNKG,
        0x00
    },

    Package (0x04)
    {
        0x0005FFFF,
        0x03,
        LNKF,
        0x00
    }
})
Name (AR01, Package (0x0C)
{
    Package (0x04)
    {
        0xFFFF,
```

```
    0x00,
    0x00,
    0x15
},

Package (0x04)
{
    0xFFFF,
    0x01,
    0x00,
    0x16
},

Package (0x04)
{
    0xFFFF,
    0x02,
    0x00,
    0x17
},

Package (0x04)
{
    0xFFFF,
    0x03,
    0x00,
    0x14
},

Package (0x04)
{
    0x0001FFFF,
    0x00,
    0x00,
    0x16
},

Package (0x04)
{
    0x0001FFFF,
    0x01,
    0x00,
    0x15
},

Package (0x04)
{
```

```
            0x0001FFFF,
            0x02,
            0x00,
            0x14
        },

        Package (0x04)
        {
            0x0001FFFF,
            0x03,
            0x00,
            0x17
        },

        Package (0x04)
        {
            0x0005FFFF,
            0x00,
            0x00,
            0x12
        },

        Package (0x04)
        {
            0x0005FFFF,
            0x01,
            0x00,
            0x14
        },

        Package (0x04)
        {
            0x0005FFFF,
            0x02,
            0x00,
            0x16
        },

        Package (0x04)
        {
            0x0005FFFF,
            0x03,
            0x00,
            0x15
        }
    })
    Name (PRSA, ResourceTemplate ()
```

```
{
    IRQ (Level, ActiveLow, Shared, )
        {3,4,5,6,7,10,11,12,14,15}
})
Alias (PRSA, PRSB)
Alias (PRSA, PRSC)
Alias (PRSA, PRSD)
Alias (PRSA, PRSE)
Alias (PRSA, PRSF)
Alias (PRSA, PRSG)
Alias (PRSA, PRSH)
Device (PCI0)
{
    Name (_HID, EisaId ("PNP0A08") /* PCI Express Bus */)  // _HID: Hardware
    Name (_CID, EisaId ("PNP0A03") /* PCI Bus */)  // _CID: Compatible ID
    Name (_ADR, 0x00)  // _ADR: Address
    Method (^BN00, 0, NotSerialized)
    {
        Return (0x00)
    }

    Method (_BBN, 0, NotSerialized)  // _BBN: BIOS Bus Number
    {
        Return (BN00 ())
    }

    Name (_UID, 0x00)  // _UID: Unique ID
    Method (_PRT, 0, NotSerialized)  // _PRT: PCI Routing Table
    {
        If (PICM)
        {
            Return (AR00 ())
        }

        Return (PR00 ())
    }

    OperationRegion (HBUS, PCI_Config, 0x00, 0x0100)
    Field (HBUS, DWordAcc, NoLock, Preserve)
    {
        Offset (0x40),
        EPEN,   1,
            ,   11,
        EPBR,   20,
        Offset (0x48),
        MHEN,   1,
            ,   14,
```

```
MHBR,   17,
Offset (0x50),
GCLK,   1,
Offset (0x54),
D0EN,   1,
Offset (0x60),
PXEN,   1,
PXSZ,   2,
    ,   23,
PXBR,   6,
Offset (0x68),
DIEN,   1,
    ,   11,
DIBR,   20,
Offset (0x70),
    ,   20,
MEBR,   12,
Offset (0x80),
    ,   4,
PM0H,   2,
Offset (0x81),
PM1L,   2,
    ,   2,
PM1H,   2,
Offset (0x82),
PM2L,   2,
    ,   2,
PM2H,   2,
Offset (0x83),
PM3L,   2,
    ,   2,
PM3H,   2,
Offset (0x84),
PM4L,   2,
    ,   2,
PM4H,   2,
Offset (0x85),
PM5L,   2,
    ,   2,
PM5H,   2,
Offset (0x86),
PM6L,   2,
    ,   2,
PM6H,   2,
Offset (0x87),
Offset (0xA8),
    ,   20,
```

```
    TUUD,   19,
    Offset (0xBC),
        ,   20,
    TLUD,   12,
    Offset (0xC8),
        ,   7,
    HTSE,   1
}

OperationRegion (MCHT, SystemMemory, 0xFED10000, 0x6000)
Field (MCHT, ByteAcc, NoLock, Preserve)
{
    Offset (0x5994),
    RPSL,   8,
    Offset (0x5998),
    RP0C,   8,
    RP1C,   8,
    RPNC,   8
}

Name (BUF0, ResourceTemplate ()
{
    WordBusNumber (ResourceProducer, MinFixed, MaxFixed, PosDecode,
        0x0000,             // Granularity
        0x0000,             // Range Minimum
        0x00FF,             // Range Maximum
        0x0000,             // Translation Offset
        0x0100,             // Length
        ,, )
    DWordIO (ResourceProducer, MinFixed, MaxFixed, PosDecode, EntireRa
        0x00000000,         // Granularity
        0x00000000,         // Range Minimum
        0x00000CF7,         // Range Maximum
        0x00000000,         // Translation Offset
        0x00000CF8,         // Length
        ,, , TypeStatic, DenseTranslation)
    IO (Decode16,
        0x0CF8,             // Range Minimum
        0x0CF8,             // Range Maximum
        0x01,               // Alignment
        0x08,               // Length
        )
    DWordIO (ResourceProducer, MinFixed, MaxFixed, PosDecode, EntireRa
        0x00000000,         // Granularity
        0x00000D00,         // Range Minimum
        0x0000FFFF,         // Range Maximum
        0x00000000,         // Translation Offset
```

```
            0x0000F300,         // Length
            ,, , TypeStatic, DenseTranslation)
        DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
            0x00000000,         // Granularity
            0x000A0000,         // Range Minimum
            0x000BFFFF,         // Range Maximum
            0x00000000,         // Translation Offset
            0x00020000,         // Length
            ,, , AddressRangeMemory, TypeStatic)
        DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
            0x00000000,         // Granularity
            0x000C0000,         // Range Minimum
            0x000C3FFF,         // Range Maximum
            0x00000000,         // Translation Offset
            0x00004000,         // Length
            ,, _Y00, AddressRangeMemory, TypeStatic)
        DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
            0x00000000,         // Granularity
            0x000C4000,         // Range Minimum
            0x000C7FFF,         // Range Maximum
            0x00000000,         // Translation Offset
            0x00004000,         // Length
            ,, _Y01, AddressRangeMemory, TypeStatic)
        DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
            0x00000000,         // Granularity
            0x000C8000,         // Range Minimum
            0x000CBFFF,         // Range Maximum
            0x00000000,         // Translation Offset
            0x00004000,         // Length
            ,, _Y02, AddressRangeMemory, TypeStatic)
        DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
            0x00000000,         // Granularity
            0x000CC000,         // Range Minimum
            0x000CFFFF,         // Range Maximum
            0x00000000,         // Translation Offset
            0x00004000,         // Length
            ,, _Y03, AddressRangeMemory, TypeStatic)
        DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
            0x00000000,         // Granularity
            0x000D0000,         // Range Minimum
            0x000D3FFF,         // Range Maximum
```

```
                0x00000000,         // Translation Offset
                0x00004000,         // Length
                ,, _Y04, AddressRangeMemory, TypeStatic)
            DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
                0x00000000,         // Granularity
                0x000D4000,         // Range Minimum
                0x000D7FFF,         // Range Maximum
                0x00000000,         // Translation Offset
                0x00004000,         // Length
                ,, _Y05, AddressRangeMemory, TypeStatic)
            DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
                0x00000000,         // Granularity
                0x000D8000,         // Range Minimum
                0x000DBFFF,         // Range Maximum
                0x00000000,         // Translation Offset
                0x00004000,         // Length
                ,, _Y06, AddressRangeMemory, TypeStatic)
            DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
                0x00000000,         // Granularity
                0x000DC000,         // Range Minimum
                0x000DFFFF,         // Range Maximum
                0x00000000,         // Translation Offset
                0x00004000,         // Length
                ,, _Y07, AddressRangeMemory, TypeStatic)
            DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
                0x00000000,         // Granularity
                0x000E0000,         // Range Minimum
                0x000E3FFF,         // Range Maximum
                0x00000000,         // Translation Offset
                0x00004000,         // Length
                ,, _Y08, AddressRangeMemory, TypeStatic)
            DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
                0x00000000,         // Granularity
                0x000E4000,         // Range Minimum
                0x000E7FFF,         // Range Maximum
                0x00000000,         // Translation Offset
                0x00004000,         // Length
                ,, _Y09, AddressRangeMemory, TypeStatic)
            DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
                0x00000000,         // Granularity
                0x000E8000,         // Range Minimum
```

```
                0x000EBFFF,         // Range Maximum
                0x00000000,         // Translation Offset
                0x00004000,         // Length
                ,, _Y0A, AddressRangeMemory, TypeStatic)
            DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
                0x00000000,         // Granularity
                0x000EC000,         // Range Minimum
                0x000EFFFF,         // Range Maximum
                0x00000000,         // Translation Offset
                0x00004000,         // Length
                ,, _Y0B, AddressRangeMemory, TypeStatic)
            DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
                0x00000000,         // Granularity
                0x000F0000,         // Range Minimum
                0x000FFFFF,         // Range Maximum
                0x00000000,         // Translation Offset
                0x00010000,         // Length
                ,, _Y0C, AddressRangeMemory, TypeStatic)
            DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
                0x00000000,         // Granularity
                0x00000000,         // Range Minimum
                0xFEAFFFFF,         // Range Maximum
                0x00000000,         // Translation Offset
                0xFEB00000,         // Length
                ,, _Y0D, AddressRangeMemory, TypeStatic)
            DWordMemory (ResourceProducer, PosDecode, MinFixed, MaxFixed, Ca
ReadWrite,
                0x00000000,         // Granularity
                0xFED40000,         // Range Minimum
                0xFED44FFF,         // Range Maximum
                0x00000000,         // Translation Offset
                0x00005000,         // Length
                ,, , AddressRangeMemory, TypeStatic)
        })
        Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
        {
            If (PM1L)
            {
                CreateDWordField (BUF0, \_SB.PCI0._Y00._LEN, C0LN)  // _LEN: Len
                C0LN = Zero
            }

            If ((PM1L == 0x01))
            {
```

```
    CreateBitField (BUF0, \_SB.PCI0._Y00._RW, C0RW)  // _RW_: Read-W
    C0RW = Zero
}

If (PM1H)
{
    CreateDWordField (BUF0, \_SB.PCI0._Y01._LEN, C4LN)  // _LEN: Len
    C4LN = Zero
}

If ((PM1H == 0x01))
{
    CreateBitField (BUF0, \_SB.PCI0._Y01._RW, C4RW)  // _RW_: Read-W
    C4RW = Zero
}

If (PM2L)
{
    CreateDWordField (BUF0, \_SB.PCI0._Y02._LEN, C8LN)  // _LEN: Len
    C8LN = Zero
}

If ((PM2L == 0x01))
{
    CreateBitField (BUF0, \_SB.PCI0._Y02._RW, C8RW)  // _RW_: Read-W
    C8RW = Zero
}

If (PM2H)
{
    CreateDWordField (BUF0, \_SB.PCI0._Y03._LEN, CCLN)  // _LEN: Len
    CCLN = Zero
}

If ((PM2H == 0x01))
{
    CreateBitField (BUF0, \_SB.PCI0._Y03._RW, CCRW)  // _RW_: Read-W
    CCRW = Zero
}

If (PM3L)
{
    CreateDWordField (BUF0, \_SB.PCI0._Y04._LEN, D0LN)  // _LEN: Len
    D0LN = Zero
}

If ((PM3L == 0x01))
```

```
{
    CreateBitField (BUF0, \_SB.PCI0._Y04._RW, D0RW)  // _RW_: Read-W
    D0RW = Zero
}

If (PM3H)
{
    CreateDWordField (BUF0, \_SB.PCI0._Y05._LEN, D4LN)  // _LEN: Len
    D4LN = Zero
}

If ((PM3H == 0x01))
{
    CreateBitField (BUF0, \_SB.PCI0._Y05._RW, D4RW)  // _RW_: Read-W
    D4RW = Zero
}

If (PM4L)
{
    CreateDWordField (BUF0, \_SB.PCI0._Y06._LEN, D8LN)  // _LEN: Len
    D8LN = Zero
}

If ((PM4L == 0x01))
{
    CreateBitField (BUF0, \_SB.PCI0._Y06._RW, D8RW)  // _RW_: Read-W
    D8RW = Zero
}

If (PM4H)
{
    CreateDWordField (BUF0, \_SB.PCI0._Y07._LEN, DCLN)  // _LEN: Len
    DCLN = Zero
}

If ((PM4H == 0x01))
{
    CreateBitField (BUF0, \_SB.PCI0._Y07._RW, DCRW)  // _RW_: Read-W
    DCRW = Zero
}

If (PM5L)
{
    CreateDWordField (BUF0, \_SB.PCI0._Y08._LEN, E0LN)  // _LEN: Leng
    E0LN = Zero
}
```

```
If ((PM5L == 0x01))
{
    CreateBitField (BUF0, \_SB.PCI0._Y08._RW, E0RW)  // _RW_: Read-W
    E0RW = Zero
}

If (PM5H)
{
    CreateDWordField (BUF0, \_SB.PCI0._Y09._LEN, E4LN)  // _LEN: Len
    E4LN = Zero
}

If ((PM5H == 0x01))
{
    CreateBitField (BUF0, \_SB.PCI0._Y09._RW, E4RW)  // _RW_: Read-W
    E4RW = Zero
}

If (PM6L)
{
    CreateDWordField (BUF0, \_SB.PCI0._Y0A._LEN, E8LN)  // _LEN: Len
    E8LN = Zero
}

If ((PM6L == 0x01))
{
    CreateBitField (BUF0, \_SB.PCI0._Y0A._RW, E8RW)  // _RW_: Read-W
    E8RW = Zero
}

If (PM6H)
{
    CreateDWordField (BUF0, \_SB.PCI0._Y0B._LEN, ECLN)  // _LEN: Ler
    ECLN = Zero
}

If ((PM6H == 0x01))
{
    CreateBitField (BUF0, \_SB.PCI0._Y0B._RW, ECRW)  // _RW_: Read-W
    ECRW = Zero
}

If (PM0H)
{
    CreateDWordField (BUF0, \_SB.PCI0._Y0C._LEN, F0LN)  // _LEN: Len
    F0LN = Zero
}
```

```
            If ((PM0H == 0x01))
            {
                CreateBitField (BUF0, \_SB.PCI0._Y0C._RW, F0RW)  // _RW_: Read-W
                F0RW = Zero
            }

            CreateDWordField (BUF0, \_SB.PCI0._Y0D._MIN, M1MN)  // _MIN: Minim
            CreateDWordField (BUF0, \_SB.PCI0._Y0D._MAX, M1MX)  // _MAX: Max
Address
            CreateDWordField (BUF0, \_SB.PCI0._Y0D._LEN, M1LN)  // _LEN: Lengt
            M1MN = (TLUD << 0x14)
            M1LN = ((M1MX - M1MN) + 0x01)
            Return (BUF0) /* \_SB_.PCI0.BUF0 */
        }

        Name (GUID, ToUUID ("33db4d5b-1ff7-401c-9657-7441c03dd766") /* PCI H
*/)
        Name (SUPP, 0x00)
        Name (CTRL, 0x00)
        Method (_OSC, 4, Serialized)  // _OSC: Operating System Capabilities
        {
            Local0 = Arg3
            CreateDWordField (Local0, 0x00, CDW1)
            CreateDWordField (Local0, 0x04, CDW2)
            CreateDWordField (Local0, 0x08, CDW3)
            SUPP = CDW2 /* \_SB_.PCI0._OSC.CDW2 */
            CTRL = CDW3 /* \_SB_.PCI0._OSC.CDW3 */
            If ((0x01 == OSDW ()))
            {
                If (((Arg0 == GUID) && NEXP))
                {
                    If (~(CDW1 & 0x01))
                    {
                        If ((CTRL & 0x02))
                        {
                            NHPG ()
                        }

                        If ((CTRL & 0x04))
                        {
                            NPME ()
                        }
                    }

                    If ((Arg1 != One))
                    {
```

```
            CDW1 |= 0x08
        }

        If ((CDW3 != CTRL))
        {
            CDW1 |= 0x10
        }

        CDW3 = CTRL /* \_SB_.PCI0.CTRL */
        OSCC = CTRL /* \_SB_.PCI0.CTRL */
        Return (Local0)
    }
    Else
    {
        CDW1 |= 0x04
        Return (Local0)
    }
  }
  Else
  {
    Return (Local0)
  }
}

Scope (\_SB.PCI0)
{
  Method (AR00, 0, NotSerialized)
  {
    Return (\_SB.AR00)
  }

  Method (PR00, 0, NotSerialized)
  {
    Return (\_SB.PR00)
  }

  Method (AR01, 0, NotSerialized)
  {
    Return (\_SB.AR01)
  }

  Method (PR01, 0, NotSerialized)
  {
    Return (\_SB.PR01)
  }

  Method (AR02, 0, NotSerialized)
```

```
{
    Return (\_SB.AR02)
}

Method (PR02, 0, NotSerialized)
{
    Return (\_SB.PR02)
}

Method (AR04, 0, NotSerialized)
{
    Return (\_SB.AR04)
}

Method (PR04, 0, NotSerialized)
{
    Return (\_SB.PR04)
}

Method (AR05, 0, NotSerialized)
{
    Return (\_SB.AR05)
}

Method (PR05, 0, NotSerialized)
{
    Return (\_SB.PR05)
}

Method (AR06, 0, NotSerialized)
{
    Return (\_SB.AR06)
}

Method (PR06, 0, NotSerialized)
{
    Return (\_SB.PR06)
}

Method (AR07, 0, NotSerialized)
{
    Return (\_SB.AR07)
}

Method (PR07, 0, NotSerialized)
{
    Return (\_SB.PR07)
```

```
    }

    Method (AR08, 0, NotSerialized)
    {
        Return (\_SB.AR08)
    }

    Method (PR08, 0, NotSerialized)
    {
        Return (\_SB.PR08)
    }

    Method (AR09, 0, NotSerialized)
    {
        Return (\_SB.AR09)
    }

    Method (PR09, 0, NotSerialized)
    {
        Return (\_SB.PR09)
    }

    Method (AR0A, 0, NotSerialized)
    {
        Return (\_SB.AR0A)
    }

    Method (PR0A, 0, NotSerialized)
    {
        Return (\_SB.PR0A)
    }

    Method (AR0B, 0, NotSerialized)
    {
        Return (\_SB.AR0B)
    }

    Method (PR0B, 0, NotSerialized)
    {
        Return (\_SB.PR0B)
    }
}

Device (MCHC)
{
    Name (_ADR, 0x00)  // _ADR: Address
}
```

```
Device (PEG0)
{
    Name (_ADR, 0x00010000)  // _ADR: Address
    Method (_PRW, 0, NotSerialized)  // _PRW: Power Resources for Wake
    {
        If (OSDW ())
        {
            Return (Package (0x02)
            {
                0x69,
                0x03
            })
        }
        Else
        {
            Return (Package (0x02)
            {
                0x69,
                0x03
            })
        }
    }

    Method (_PRT, 0, NotSerialized)  // _PRT: PCI Routing Table
    {
        If (PICM)
        {
            Return (AR02 ())
        }

        Return (PR02 ())
    }
}

Device (IGPU)
{
    Name (_ADR, 0x00020000)  // _ADR: Address
    OperationRegion (GFXH, PCI_Config, 0x00, 0x40)
    Field (GFXH, ByteAcc, NoLock, Preserve)
    {
        VID0,   16,
        DID0,   16
    }

    Method (_DSM, 4, NotSerialized)  // _DSM: Device-Specific Method
    {
```

```
    If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b") /* Unk
    {
        If (((VID0 & 0xFFFF) != 0xFFFF))
        {
            Local0 = Package (0x02)
                {
                    "hda-gfx",
                    Buffer (0x0A)
                    {
                        "onboard-1"
                    }
                }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }
    }

    Return (0x00)
}

Method (HDFR, 0, Serialized)
{
    Return (GP92) /* \GP92 */
}

Method (HDFW, 1, Serialized)
{
    If ((Arg0 <= 0x00))
    {
        GD92 = 0x01
    }
    ElseIf ((Arg0 <= 0x01))
    {
        GD92 = 0x00
        GP92 = 0x00
    }
}

Method (HDLR, 0, Serialized)
{
    Return (GP52) /* \GP52 */
}

Method (HDLW, 1, Serialized)
{
    If ((Arg0 <= 0x00))
    {
```

```
        GD52 = 0x01
    }
    ElseIf ((Arg0 <= 0x01))
    {
        GD52 = 0x00
        GP52 = 0x00
    }
}

Method (PCPC, 0, NotSerialized)
{
}

Method (PAPR, 0, NotSerialized)
{
    Return (0x00)
}

Method (_DOS, 1, NotSerialized)  // _DOS: Disable Output Switching
{
    DSEN = (Arg0 & 0x07)
    If (((Arg0 & 0x03) == 0x00))
    {
        If (CondRefOf (HDOS))
        {
            HDOS ()
        }
    }
}

Method (_DOD, 0, Serialized)  // _DOD: Display Output Devices
{
    If (CondRefOf (IDAB)){}
    Else
    {
        NDID = 0x00
        If ((DIDL != Zero))
        {
            DID1 = SDDL (DIDL)
        }

        If ((DDL2 != Zero))
        {
            DID2 = SDDL (DDL2)
        }

        If ((DDL3 != Zero))
```

```
    {
        DID3 = SDDL (DDL3)
    }

    If ((DDL4 != Zero))
    {
        DID4 = SDDL (DDL4)
    }

    If ((DDL5 != Zero))
    {
        DID5 = SDDL (DDL5)
    }

    If ((DDL6 != Zero))
    {
        DID6 = SDDL (DDL6)
    }

    If ((DDL7 != Zero))
    {
        DID7 = SDDL (DDL7)
    }

    If ((DDL8 != Zero))
    {
        DID8 = SDDL (DDL8)
    }
}

If ((NDID == 0x01))
{
    Name (TMP1, Package (0x01)
    {
        0xFFFFFFFF
    })
    TMP1 [0x00] = (0x00010000 | DID1)
    Return (TMP1) /* \_SB_.PCI0.IGPU._DOD.TMP1 */
}

If ((NDID == 0x02))
{
    Name (TMP2, Package (0x02)
    {
        0xFFFFFFFF,
        0xFFFFFFFF
    })
```

```
    TMP2 [0x00] = (0x00010000 | DID1)
    TMP2 [0x01] = (0x00010000 | DID2)
    Return (TMP2) /* \_SB_.PCI0.IGPU._DOD.TMP2 */
}

If ((NDID == 0x03))
{
    Name (TMP3, Package (0x03)
    {
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF
    })
    TMP3 [0x00] = (0x00010000 | DID1)
    TMP3 [0x01] = (0x00010000 | DID2)
    TMP3 [0x02] = (0x00010000 | DID3)
    Return (TMP3) /* \_SB_.PCI0.IGPU._DOD.TMP3 */
}

If ((NDID == 0x04))
{
    Name (TMP4, Package (0x04)
    {
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF
    })
    TMP4 [0x00] = (0x00010000 | DID1)
    TMP4 [0x01] = (0x00010000 | DID2)
    TMP4 [0x02] = (0x00010000 | DID3)
    TMP4 [0x03] = (0x00010000 | DID4)
    Return (TMP4) /* \_SB_.PCI0.IGPU._DOD.TMP4 */
}

If ((NDID == 0x05))
{
    Name (TMP5, Package (0x05)
    {
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF
    })
    TMP5 [0x00] = (0x00010000 | DID1)
    TMP5 [0x01] = (0x00010000 | DID2)
```

```
    TMP5 [0x02] = (0x00010000 | DID3)
    TMP5 [0x03] = (0x00010000 | DID4)
    TMP5 [0x04] = (0x00010000 | DID5)
    Return (TMP5) /* \_SB_.PCI0.IGPU._DOD.TMP5 */
}

If ((NDID == 0x06))
{
    Name (TMP6, Package (0x06)
    {
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF
    })
    TMP6 [0x00] = (0x00010000 | DID1)
    TMP6 [0x01] = (0x00010000 | DID2)
    TMP6 [0x02] = (0x00010000 | DID3)
    TMP6 [0x03] = (0x00010000 | DID4)
    TMP6 [0x04] = (0x00010000 | DID5)
    TMP6 [0x05] = (0x00010000 | DID6)
    Return (TMP6) /* \_SB_.PCI0.IGPU._DOD.TMP6 */
}

If ((NDID == 0x07))
{
    Name (TMP7, Package (0x07)
    {
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF,
        0xFFFFFFFF
    })
    TMP7 [0x00] = (0x00010000 | DID1)
    TMP7 [0x01] = (0x00010000 | DID2)
    TMP7 [0x02] = (0x00010000 | DID3)
    TMP7 [0x03] = (0x00010000 | DID4)
    TMP7 [0x04] = (0x00010000 | DID5)
    TMP7 [0x05] = (0x00010000 | DID6)
    TMP7 [0x06] = (0x00010000 | DID7)
    Return (TMP7) /* \_SB_.PCI0.IGPU._DOD.TMP7 */
}
```

```
    If ((NDID == 0x08))
    {
        Name (TMP8, Package (0x08)
        {
            0xFFFFFFFF,
            0xFFFFFFFF,
            0xFFFFFFFF,
            0xFFFFFFFF,
            0xFFFFFFFF,
            0xFFFFFFFF,
            0xFFFFFFFF,
            0xFFFFFFFF
        })
        TMP8 [0x00] = (0x00010000 | DID1)
        TMP8 [0x01] = (0x00010000 | DID2)
        TMP8 [0x02] = (0x00010000 | DID3)
        TMP8 [0x03] = (0x00010000 | DID4)
        TMP8 [0x04] = (0x00010000 | DID5)
        TMP8 [0x05] = (0x00010000 | DID6)
        TMP8 [0x06] = (0x00010000 | DID7)
        TMP8 [0x07] = (0x00010000 | DID8)
        Return (TMP8) /* \_SB_.PCI0.IGPU._DOD.TMP8 */
    }

    Return (Package (0x01)
    {
        0x0400
    })
}

Name (EDPV, 0x00)
Name (DIDX, 0x00)
Name (NXDX, 0x00)
Name (BRTN, Package (0x12)
{
    0x50,
    0x2F,
    0x00,
    0x07,
    0x0D,
    0x14,
    0x1B,
    0x21,
    0x28,
    0x2F,
    0x35,
```

```
        0x3C,
        0x43,
        0x49,
        0x50,
        0x57,
        0x5D,
        0x64
    })
    Method (ABCL, 0, NotSerialized)
    {
        If ((OSYS < 0x07DC))
        {
            BRTN [0x00] = DerefOf (BRTN [0x0F])
            BRTN [0x01] = DerefOf (BRTN [0x0A])
            Return (BRTN) /* \_SB_.PCI0.IGPU.BRTN */
        }
        Else
        {
            Return (Package (0x67)
            {
                0x50,
                0x32,
                0x00,
                0x01,
                0x02,
                0x03,
                0x04,
                0x05,
                0x06,
                0x07,
                0x08,
                0x09,
                0x0A,
                0x0B,
                0x0C,
                0x0D,
                0x0E,
                0x0F,
                0x10,
                0x11,
                0x12,
                0x13,
                0x14,
                0x15,
                0x16,
                0x17,
                0x18,
```

0x19,
0x1A,
0x1B,
0x1C,
0x1D,
0x1E,
0x1F,
0x20,
0x21,
0x22,
0x23,
0x24,
0x25,
0x26,
0x27,
0x28,
0x29,
0x2A,
0x2B,
0x2C,
0x2D,
0x2E,
0x2F,
0x30,
0x31,
0x32,
0x33,
0x34,
0x35,
0x36,
0x37,
0x38,
0x39,
0x3A,
0x3B,
0x3C,
0x3D,
0x3E,
0x3F,
0x40,
0x41,
0x42,
0x43,
0x44,
0x45,
0x46,
0x47,

```
            0x48,
            0x49,
            0x4A,
            0x4B,
            0x4C,
            0x4D,
            0x4E,
            0x4F,
            0x50,
            0x51,
            0x52,
            0x53,
            0x54,
            0x55,
            0x56,
            0x57,
            0x58,
            0x59,
            0x5A,
            0x5B,
            0x5C,
            0x5D,
            0x5E,
            0x5F,
            0x60,
            0x61,
            0x62,
            0x63,
            0x64
        })
    }
}

Method (ABCM, 1, NotSerialized)
{
    If (((Arg0 >= 0x00) && (Arg0 <= 0x64)))
    {
        BRTL = Arg0
        \_SB.PCI0.IGPU.AINT (0x01, Arg0)
    }

    Return (Zero)
}

Method (ABQC, 0, NotSerialized)
{
    Return (BRTL) /* \BRTL */
```

```
}

Device (DD01)
{
   Method (_ADR, 0, Serialized)  // _ADR: Address
   {
      If (((0x0F00 & DID1) == 0x0302))
      {
         EDPV = 0x01
         NXDX = NXD1 /* \NXD1 */
         DIDX = DID1 /* \DID1 */
         Return (0x01)
      }

      If ((DID1 == 0x00))
      {
         Return (0x01)
      }
      Else
      {
         Return ((0xFFFF & DID1))
      }
   }

   Method (_DCS, 0, NotSerialized)  // _DCS: Display Current Status
   {
      Return (CDDS (DID1))
   }

   Method (_DGS, 0, NotSerialized)  // _DGS: Display Graphics State
   {
      If (CondRefOf (SNXD))
      {
         Return (NXD1) /* \NXD1 */
      }

      Return (NDDS (DID1))
   }

   Method (_DSS, 1, NotSerialized)  // _DSS: Device Set State
   {
      If (((Arg0 & 0xC0000000) == 0xC0000000))
      {
         CSTE = NSTE /* \NSTE */
      }
   }
```

```
    Method (_BCL, 0, NotSerialized)  // _BCL: Brightness Control Levels
    {
        Return (\_SB.PCI0.IGPU.ABCL ())
    }

    Method (_BCM, 1, NotSerialized)  // _BCM: Brightness Control Method
    {
        \_SB.PCI0.IGPU.ABCM (Arg0)
        Return (Zero)
    }

    Method (_BQC, 0, NotSerialized)  // _BQC: Brightness Query Current
    {
        Return (\_SB.PCI0.IGPU.ABQC ())
    }
}

Device (DD02)
{
    Method (_ADR, 0, Serialized)  // _ADR: Address
    {
        If (((0x0F00 & DID2) == 0x0302))
        {
            EDPV = 0x02
            NXDX = NXD2 /* \NXD2 */
            DIDX = DID2 /* \DID2 */
            Return (0x02)
        }

        If ((DID2 == 0x00))
        {
            Return (0x02)
        }
        Else
        {
            Return ((0xFFFF & DID2))
        }
    }

    Method (_DCS, 0, NotSerialized)  // _DCS: Display Current Status
    {
        If ((LIDS == 0x00))
        {
            Return (0x00)
        }

        Return (CDDS (DID2))
```

```
    }

    Method (_DGS, 0, NotSerialized)  // _DGS: Display Graphics State
    {
        If (CondRefOf (SNXD))
        {
            Return (NXD2) /* \NXD2 */
        }

        Return (NDDS (DID2))
    }

    Method (_DSS, 1, NotSerialized)  // _DSS: Device Set State
    {
        If (((Arg0 & 0xC0000000) == 0xC0000000))
        {
            CSTE = NSTE /* \NSTE */
        }
    }
}

Device (DD03)
{
    Method (_ADR, 0, Serialized)  // _ADR: Address
    {
        If (((0x0F00 & DID3) == 0x0302))
        {
            EDPV = 0x03
            NXDX = NXD3 /* \NXD3 */
            DIDX = DID3 /* \DID3 */
            Return (0x03)
        }

        If ((DID3 == 0x00))
        {
            Return (0x03)
        }
        Else
        {
            Return ((0xFFFF & DID3))
        }
    }

    Method (_DCS, 0, NotSerialized)  // _DCS: Display Current Status
    {
        If ((DID3 == 0x00))
        {
```

```
            Return (0x0B)
        }
        Else
        {
            Return (CDDS (DID3))
        }
    }

    Method (_DGS, 0, NotSerialized)  // _DGS: Display Graphics State
    {
        If (CondRefOf (SNXD))
        {
            Return (NXD3) /* \NXD3 */
        }

        Return (NDDS (DID3))
    }

    Method (_DSS, 1, NotSerialized)  // _DSS: Device Set State
    {
        If (((Arg0 & 0xC0000000) == 0xC0000000))
        {
            CSTE = NSTE /* \NSTE */
        }
    }
}

Device (DD04)
{
    Method (_ADR, 0, Serialized)  // _ADR: Address
    {
        If (((0x0F00 & DID4) == 0x0302))
        {
            EDPV = 0x04
            NXDX = NXD4 /* \NXD4 */
            DIDX = DID4 /* \DID4 */
            Return (0x04)
        }

        If ((DID4 == 0x00))
        {
            Return (0x04)
        }
        Else
        {
            Return ((0xFFFF & DID4))
        }
```

```
    }

    Method (_DCS, 0, NotSerialized)  // _DCS: Display Current Status
    {
        If ((DID4 == 0x00))
        {
            Return (0x0B)
        }
        Else
        {
            Return (CDDS (DID4))
        }
    }

    Method (_DGS, 0, NotSerialized)  // _DGS: Display Graphics State
    {
        If (CondRefOf (SNXD))
        {
            Return (NXD4) /* \NXD4 */
        }

        Return (NDDS (DID4))
    }

    Method (_DSS, 1, NotSerialized)  // _DSS: Device Set State
    {
        If (((Arg0 & 0xC0000000) == 0xC0000000))
        {
            CSTE = NSTE /* \NSTE */
        }
    }
}

Device (DD05)
{
    Method (_ADR, 0, Serialized)  // _ADR: Address
    {
        If (((0x0F00 & DID5) == 0x0302))
        {
            EDPV = 0x05
            NXDX = NXD5 /* \NXD5 */
            DIDX = DID5 /* \DID5 */
            Return (0x05)
        }

        If ((DID5 == 0x00))
        {
```

```
            Return (0x05)
        }
        Else
        {
            Return ((0xFFFF & DID5))
        }
    }

    Method (_DCS, 0, NotSerialized)  // _DCS: Display Current Status
    {
        If ((DID5 == 0x00))
        {
            Return (0x0B)
        }
        Else
        {
            Return (CDDS (DID5))
        }
    }

    Method (_DGS, 0, NotSerialized)  // _DGS: Display Graphics State
    {
        If (CondRefOf (SNXD))
        {
            Return (NXD5) /* \NXD5 */
        }

        Return (NDDS (DID5))
    }

    Method (_DSS, 1, NotSerialized)  // _DSS: Device Set State
    {
        If (((Arg0 & 0xC0000000) == 0xC0000000))
        {
            CSTE = NSTE /* \NSTE */
        }
    }
}

Device (DD06)
{
    Method (_ADR, 0, Serialized)  // _ADR: Address
    {
        If (((0x0F00 & DID6) == 0x0302))
        {
            EDPV = 0x06
            NXDX = NXD6 /* \NXD6 */
```

```
            DIDX = DID6 /* \DID6 */
            Return (0x06)
        }

        If ((DID6 == 0x00))
        {
            Return (0x06)
        }
        Else
        {
            Return ((0xFFFF & DID6))
        }
    }

    Method (_DCS, 0, NotSerialized)  // _DCS: Display Current Status
    {
        If ((DID6 == 0x00))
        {
            Return (0x0B)
        }
        Else
        {
            Return (CDDS (DID6))
        }
    }

    Method (_DGS, 0, NotSerialized)  // _DGS: Display Graphics State
    {
        If (CondRefOf (SNXD))
        {
            Return (NXD6) /* \NXD6 */
        }

        Return (NDDS (DID6))
    }

    Method (_DSS, 1, NotSerialized)  // _DSS: Device Set State
    {
        If (((Arg0 & 0xC0000000) == 0xC0000000))
        {
            CSTE = NSTE /* \NSTE */
        }
    }
}

Device (DD07)
{
```

```
Method (_ADR, 0, Serialized)  // _ADR: Address
{
   If (((0x0F00 & DID7) == 0x0302))
   {
      EDPV = 0x07
      NXDX = NXD7 /* \NXD7 */
      DIDX = DID7 /* \DID7 */
      Return (0x07)
   }

   If ((DID7 == 0x00))
   {
      Return (0x07)
   }
   Else
   {
      Return ((0xFFFF & DID7))
   }
}

Method (_DCS, 0, NotSerialized)  // _DCS: Display Current Status
{
   If ((DID7 == 0x00))
   {
      Return (0x0B)
   }
   Else
   {
      Return (CDDS (DID7))
   }
}

Method (_DGS, 0, NotSerialized)  // _DGS: Display Graphics State
{
   If (CondRefOf (SNXD))
   {
      Return (NXD7) /* \NXD7 */
   }

   Return (NDDS (DID7))
}

Method (_DSS, 1, NotSerialized)  // _DSS: Device Set State
{
   If (((Arg0 & 0xC0000000) == 0xC0000000))
   {
      CSTE = NSTE /* \NSTE */
```

```
        }
    }
}

Device (DD08)
{
    Method (_ADR, 0, Serialized)  // _ADR: Address
    {
        If (((0x0F00 & DID8) == 0x0302))
        {
            EDPV = 0x08
            NXDX = NXD8 /* \NXD8 */
            DIDX = DID8 /* \DID8 */
            Return (0x08)
        }

        If ((DID8 == 0x00))
        {
            Return (0x08)
        }
        Else
        {
            Return ((0xFFFF & DID8))
        }
    }

    Method (_DCS, 0, NotSerialized)  // _DCS: Display Current Status
    {
        If ((DID8 == 0x00))
        {
            Return (0x0B)
        }
        Else
        {
            Return (CDDS (DID8))
        }
    }

    Method (_DGS, 0, NotSerialized)  // _DGS: Display Graphics State
    {
        If (CondRefOf (SNXD))
        {
            Return (NXD8) /* \NXD8 */
        }

        Return (NDDS (DID8))
    }
```

```
    Method (_DSS, 1, NotSerialized)  // _DSS: Device Set State
    {
        If (((Arg0 & 0xC0000000) == 0xC0000000))
        {
            CSTE = NSTE /* \NSTE */
        }
    }
}

Device (DD1F)
{
    Method (_ADR, 0, Serialized)  // _ADR: Address
    {
        If ((EDPV == 0x00))
        {
            Return (0x1F)
        }
        Else
        {
            Return ((0xFFFF & DIDX))
        }
    }

    Method (_DCS, 0, NotSerialized)  // _DCS: Display Current Status
    {
        If ((EDPV == 0x00))
        {
            Return (0x00)
        }
        Else
        {
            Return (CDDS (DIDX))
        }
    }

    Method (_DGS, 0, NotSerialized)  // _DGS: Display Graphics State
    {
        If (CondRefOf (SNXD))
        {
            Return (NXDX) /* \_SB_.PCI0.IGPU.NXDX */
        }

        Return (NDDS (DIDX))
    }

    Method (_DSS, 1, NotSerialized)  // _DSS: Device Set State
```

```
    {
        If (((Arg0 & 0xC0000000) == 0xC0000000))
        {
            CSTE = NSTE /* \NSTE */
        }
    }
}

Method (SDDL, 1, NotSerialized)
{
    NDID++
    Local0 = (Arg0 & 0x0F0F)
    Local1 = (0x80000000 | Local0)
    If ((DIDL == Local0))
    {
        Return (Local1)
    }

    If ((DDL2 == Local0))
    {
        Return (Local1)
    }

    If ((DDL3 == Local0))
    {
        Return (Local1)
    }

    If ((DDL4 == Local0))
    {
        Return (Local1)
    }

    If ((DDL5 == Local0))
    {
        Return (Local1)
    }

    If ((DDL6 == Local0))
    {
        Return (Local1)
    }

    If ((DDL7 == Local0))
    {
        Return (Local1)
    }
```

```
    If ((DDL8 == Local0))
    {
        Return (Local1)
    }

    Return (0x00)
}

Method (CDDS, 1, NotSerialized)
{
    Local0 = (Arg0 & 0x0F0F)
    If ((0x00 == Local0))
    {
        Return (0x1D)
    }

    If ((CADL == Local0))
    {
        Return (0x1F)
    }

    If ((CAL2 == Local0))
    {
        Return (0x1F)
    }

    If ((CAL3 == Local0))
    {
        Return (0x1F)
    }

    If ((CAL4 == Local0))
    {
        Return (0x1F)
    }

    If ((CAL5 == Local0))
    {
        Return (0x1F)
    }

    If ((CAL6 == Local0))
    {
        Return (0x1F)
    }
```

```
    If ((CAL7 == Local0))
    {
        Return (0x1F)
    }

    If ((CAL8 == Local0))
    {
        Return (0x1F)
    }

    Return (0x1D)
}

Method (NDDS, 1, NotSerialized)
{
    Local0 = (Arg0 & 0x0F0F)
    If ((0x00 == Local0))
    {
        Return (0x00)
    }

    If ((NADL == Local0))
    {
        Return (0x01)
    }

    If ((NDL2 == Local0))
    {
        Return (0x01)
    }

    If ((NDL3 == Local0))
    {
        Return (0x01)
    }

    If ((NDL4 == Local0))
    {
        Return (0x01)
    }

    If ((NDL5 == Local0))
    {
        Return (0x01)
    }

    If ((NDL6 == Local0))
```

```
    {
        Return (0x01)
    }

    If ((NDL7 == Local0))
    {
        Return (0x01)
    }

    If ((NDL8 == Local0))
    {
        Return (0x01)
    }

    Return (0x00)
}

Scope (\_SB.PCI0)
{
    OperationRegion (MCHP, PCI_Config, 0x40, 0xC0)
    Field (MCHP, AnyAcc, NoLock, Preserve)
    {
        Offset (0x14),
        AUDE,   8,
        Offset (0x60),
        TASM,   10,
        Offset (0x62)
    }
}

OperationRegion (IGDP, PCI_Config, 0x40, 0xC0)
Field (IGDP, AnyAcc, NoLock, Preserve)
{
    Offset (0x12),
        ,   1,
    GIVD,   1,
        ,   2,
    GUMA,   3,
    Offset (0x14),
        ,   4,
    GMFN,   1,
    Offset (0x18),
    Offset (0xA4),
    ASLE,   8,
    Offset (0xA8),
    GSSE,   1,
    GSSB,   14,
```

```
    GSES,   1,
    Offset (0xB0),
        ,   12,
    CDVL,   1,
    Offset (0xB2),
    Offset (0xB5),
    LBPC,   8,
    Offset (0xBC),
    ASLS,   32
}

OperationRegion (IGDM, SystemMemory, ASLB, 0x2000)
Field (IGDM, AnyAcc, NoLock, Preserve)
{
    SIGN,   128,
    SIZE,   32,
    OVER,   32,
    SVER,   256,
    VVER,   128,
    GVER,   128,
    MBOX,   32,
    DMOD,   32,
    Offset (0x100),
    DRDY,   32,
    CSTS,   32,
    CEVT,   32,
    Offset (0x120),
    DIDL,   32,
    DDL2,   32,
    DDL3,   32,
    DDL4,   32,
    DDL5,   32,
    DDL6,   32,
    DDL7,   32,
    DDL8,   32,
    CPDL,   32,
    CPL2,   32,
    CPL3,   32,
    CPL4,   32,
    CPL5,   32,
    CPL6,   32,
    CPL7,   32,
    CPL8,   32,
    CADL,   32,
    CAL2,   32,
    CAL3,   32,
    CAL4,   32,
```

CAL5, 32,
CAL6, 32,
CAL7, 32,
CAL8, 32,
NADL, 32,
NDL2, 32,
NDL3, 32,
NDL4, 32,
NDL5, 32,
NDL6, 32,
NDL7, 32,
NDL8, 32,
ASLP, 32,
TIDX, 32,
CHPD, 32,
CLID, 32,
CDCK, 32,
SXSW, 32,
EVTS, 32,
CNOT, 32,
NRDY, 32,
Offset (0x200),
SCIE, 1,
GEFC, 4,
GXFC, 3,
GESF, 8,
Offset (0x204),
PARM, 32,
DSLP, 32,
Offset (0x300),
ARDY, 32,
ASLC, 32,
TCHE, 32,
ALSI, 32,
BCLP, 32,
PFIT, 32,
CBLV, 32,
BCLM, 320,
CPFM, 32,
EPFM, 32,
PLUT, 592,
PFMB, 32,
CCDV, 32,
PCFT, 32,
Offset (0x400),
GVD1, 49152,
PHED, 32,

```
    BDDC,   2048
}

Name (DBTB, Package (0x15)
{
    0x00,
    0x07,
    0x38,
    0x01C0,
    0x0E00,
    0x3F,
    0x01C7,
    0x0E07,
    0x01F8,
    0x0E38,
    0x0FC0,
    0x00,
    0x00,
    0x00,
    0x00,
    0x00,
    0x7000,
    0x7007,
    0x7038,
    0x71C0,
    0x7E00
})
Name (CDCT, Package (0x05)
{
    Package (0x02)
    {
        0xE4,
        0x0140
    },

    Package (0x02)
    {
        0xDE,
        0x014D
    },

    Package (0x02)
    {
        0xDE,
        0x014D
    },
```

```
    Package (0x02)
    {
        0x00,
        0x00
    },

    Package (0x02)
    {
        0xDE,
        0x014D
    }
})
Name (SUCC, 0x01)
Name (NVLD, 0x02)
Name (CRIT, 0x04)
Name (NCRT, 0x06)
Method (GBDA, 0, Serialized)
{
    If ((GESF == 0x00))
    {
        PARM = 0x00020000
        GESF = Zero
        Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
    }

    If ((GESF == 0x01))
    {
        PARM = 0x00200000
        GESF = Zero
        Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
    }

    If ((GESF == 0x04))
    {
        PARM &= 0xEFFF0000
        PARM &= (DerefOf (DBTB [IBTT]) << 0x10)
        PARM |= IBTT /* \_SB_.PCI0.IGPU.PARM */
        GESF = Zero
        Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
    }

    If ((GESF == 0x05))
    {
        PARM = IPSC /* \IPSC */
        PARM |= (IPAT << 0x08)
        PARM += 0x0100
        PARM |= (LIDS << 0x10)
```

```
            PARM += 0x00010000
            PARM |= (IBIA << 0x14)
            GESF = Zero
            Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
        }

        If ((GESF == 0x06))
        {
            GESF = Zero
            Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
        }

        If ((GESF == 0x07))
        {
            PARM = GIVD /* \_SB_.PCI0.IGPU.GIVD */
            PARM ^= 0x01
            PARM |= (GMFN << 0x01)
            PARM |= (0x03 << 0x0B)
            PARM |= (IDMS << 0x11)
            PARM |= (DerefOf (DerefOf (CDCT [HVCO]) [CDVL]) <<
                0x15) /* \_SB_.PCI0.IGPU.PARM */
            GESF = 0x01
            Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
        }

        If ((GESF == 0x0A))
        {
            PARM = 0x00
            If (ISSC)
            {
                PARM |= 0x03
            }

            GESF = 0x00
            Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
        }

        If ((GESF == 0x0B))
        {
            PARM = KSV0 /* \KSV0 */
            GESF = KSV1 /* \KSV1 */
            Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
        }

        GESF = Zero
        Return (CRIT) /* \_SB_.PCI0.IGPU.CRIT */
    }
```

```
Method (SBCB, 0, Serialized)
{
    If ((GESF == 0x00))
    {
        PARM = 0x00
        PARM = 0x00020000
        GESF = Zero
        Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
    }

    If ((GESF == 0x01))
    {
        PARM = 0x00
        GESF = Zero
        PARM = 0x00200000
        Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
    }

    If ((GESF == 0x03))
    {
        GESF = Zero
        PARM = Zero
        Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
    }

    If ((GESF == 0x04))
    {
        GESF = Zero
        PARM = Zero
        Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
    }

    If ((GESF == 0x05))
    {
        GESF = Zero
        PARM = Zero
        Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
    }

    If ((GESF == 0x06))
    {
        GESF = Zero
        PARM = Zero
        Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
    }
```

```
If ((GESF == 0x07))
{
    If ((PARM == 0x00))
    {
        Local0 = CLID /* \_SB_.PCI0.IGPU.CLID */
        If ((0x80000000 & Local0))
        {
            CLID &= 0x0F
            GLID (CLID)
        }
    }

    GESF = Zero
    PARM = Zero
    Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
}

If ((GESF == 0x08))
{
    GESF = Zero
    PARM = Zero
    Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
}

If ((GESF == 0x09))
{
    IBTT = (PARM & 0xFF)
    GESF = Zero
    PARM = Zero
    Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
}

If ((GESF == 0x0A))
{
    IPSC = (PARM & 0xFF)
    If (((PARM >> 0x08) & 0xFF))
    {
        IPAT = ((PARM >> 0x08) & 0xFF)
        IPAT--
    }

    IBIA = ((PARM >> 0x14) & 0x07)
    GESF = Zero
    PARM = Zero
    Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
}
```

```
If ((GESF == 0x0B))
{
    IF1E = ((PARM >> 0x01) & 0x01)
    If ((PARM & (0x0F << 0x0D)))
    {
        IDMS = ((PARM >> 0x0D) & 0x0F)
    }
    Else
    {
        IDMS = ((PARM >> 0x11) & 0x0F)
    }

    GESF = Zero
    PARM = Zero
    Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
}

If ((GESF == 0x10))
{
    GESF = Zero
    PARM = Zero
    Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
}

If ((GESF == 0x11))
{
    PARM = (LIDS << 0x08)
    PARM += 0x0100
    GESF = Zero
    Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
}

If ((GESF == 0x12))
{
    If ((PARM & 0x01))
    {
        If (((PARM >> 0x01) == 0x01))
        {
            ISSC = 0x01
        }
        Else
        {
            GESF = Zero
            Return (CRIT) /* \_SB_.PCI0.IGPU.CRIT */
        }
    }
    Else
```

```
    {
        ISSC = 0x00
    }

    GESF = Zero
    PARM = Zero
    Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
}

If ((GESF == 0x13))
{
    GESF = Zero
    PARM = Zero
    Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
}

If ((GESF == 0x14))
{
    PAVP = (PARM & 0x0F)
    GESF = Zero
    PARM = Zero
    Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
}

If ((GESF == 0x15))
{
    If ((PARM == 0x01))
    {
        \_SB.PCI0.AUDE |= 0x20
        \_SB.PCI0.HDAU.ASTR ()
        \_SB.PCI0.HDAU.AINI ()
        \_SB.PCI0.HDAU.CXDC ()
        Notify (\_SB.PCI0, 0x00) // Bus Check
    }

    If ((PARM == 0x00))
    {
        \_SB.PCI0.AUDE &= 0xDF
        Notify (\_SB.PCI0, 0x00) // Bus Check
    }

    GESF = Zero
    PARM = Zero
    Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
}

GESF = Zero
```

```
    Return (SUCC) /* \_SB_.PCI0.IGPU.SUCC */
}

Method (GSCI, 0, Serialized)
{
    If ((GEFC == 0x04))
    {
        GXFC = GBDA ()
    }

    If ((GEFC == 0x06))
    {
        GXFC = SBCB ()
    }

    GEFC = 0x00
    SCIS = 0x01
    GSSE = 0x00
    SCIE = 0x00
    Return (Zero)
}

Method (PDRD, 0, NotSerialized)
{
    If (!DRDY)
    {
        Sleep (ASLP)
    }

    Return (!DRDY)
}

Method (PSTS, 0, NotSerialized)
{
    If ((CSTS > 0x02))
    {
        Sleep (ASLP)
    }

    Return ((CSTS == 0x03))
}

Method (GNOT, 2, NotSerialized)
{
    If (PDRD ())
    {
        Return (0x01)
```

```
    }

    CEVT = Arg0
    CSTS = 0x03
    If (((CHPD == 0x00) && (Arg1 == 0x00)))
    {
        If (((OSYS > 0x07D0) || (OSYS < 0x07D6)))
        {
            Notify (\_SB.PCI0, Arg1)
        }
        Else
        {
            Notify (\_SB.PCI0.IGPU, Arg1)
        }
    }

    If (CondRefOf (HNOT))
    {
        HNOT (Arg0)
    }
    Else
    {
        Notify (\_SB.PCI0.IGPU, 0x80) // Status Change
    }

    Return (0x00)
}

Method (GHDS, 1, NotSerialized)
{
    TIDX = Arg0
    Return (GNOT (0x01, 0x00))
}

Method (GLID, 1, NotSerialized)
{
    If ((Arg0 == 0x01))
    {
        CLID = 0x03
    }
    Else
    {
        CLID = Arg0
    }

    Return (GNOT (0x02, 0x00))
}
```

```
Method (GDCK, 1, NotSerialized)
{
    CDCK = Arg0
    Return (GNOT (0x04, 0x00))
}

Method (PARD, 0, NotSerialized)
{
    If (!ARDY)
    {
        Sleep (ASLP)
    }

    Return (!ARDY)
}

Method (AINT, 2, NotSerialized)
{
    If (!(TCHE & (0x01 << Arg0)))
    {
        Return (0x01)
    }

    If (PARD ())
    {
        Return (0x01)
    }

    If ((Arg0 == 0x02))
    {
        If (CPFM)
        {
            Local0 = (CPFM & 0x0F)
            Local1 = (EPFM & 0x0F)
            If ((Local0 == 0x01))
            {
                If ((Local1 & 0x06))
                {
                    PFIT = 0x06
                }
                ElseIf ((Local1 & 0x08))
                {
                    PFIT = 0x08
                }
                Else
                {
```

```
                PFIT = 0x01
            }
        }

        If ((Local0 == 0x06))
        {
            If ((Local1 & 0x08))
            {
                PFIT = 0x08
            }
            ElseIf ((Local1 & 0x01))
            {
                PFIT = 0x01
            }
            Else
            {
                PFIT = 0x06
            }
        }

        If ((Local0 == 0x08))
        {
            If ((Local1 & 0x01))
            {
                PFIT = 0x01
            }
            ElseIf ((Local1 & 0x06))
            {
                PFIT = 0x06
            }
            Else
            {
                PFIT = 0x08
            }
        }
    }
    Else
    {
        PFIT ^= 0x07
    }

    PFIT |= 0x80000000
    ASLC = 0x04
}
ElseIf ((Arg0 == 0x01))
{
    BCLP = ((Arg1 * 0xFF) / 0x64)
```

```
            BCLP |= 0x80000000
            ASLC = 0x02
        }
        ElseIf ((Arg0 == 0x00))
        {
            ALSI = Arg1
            ASLC = 0x01
        }
        Else
        {
            Return (0x01)
        }

        ASLE = 0x01
        Return (0x00)
    }

    Method (SCIP, 0, NotSerialized)
    {
        If ((OVER != 0x00))
        {
            Return (!GSMI)
        }

        Return (0x00)
    }

    Device (\_SB.MEM2)
    {
        Name (_HID, EisaId ("PNP0C01") /* System Board */)  // _HID: Hardwar
        Name (_UID, 0x02)  // _UID: Unique ID
        Name (CRS, ResourceTemplate ()
        {
            Memory32Fixed (ReadWrite,
                0x20000000,         // Address Base
                0x00200000,         // Address Length
                )
            Memory32Fixed (ReadWrite,
                0x40000000,         // Address Base
                0x00200000,         // Address Length
                )
        })
        Method (_CRS, 0, NotSerialized)  // _CRS: Current Resource Settings
        {
            Return (CRS) /* \_SB_.MEM2.CRS_ */
        }
    }
```

```
}

Device (SBUS)
{
   Name (_ADR, 0x001F0003)  // _ADR: Address
   OperationRegion (SMBP, PCI_Config, 0x40, 0xC0)
   Field (SMBP, DWordAcc, NoLock, Preserve)
   {
         ,   2,
      I2CE,   1
   }

   OperationRegion (SMPB, PCI_Config, 0x20, 0x04)
   Field (SMPB, DWordAcc, NoLock, Preserve)
   {
         ,   5,
      SBAR,   11
   }

   OperationRegion (SMBI, SystemIO, (SBAR << 0x05), 0x10)
   Field (SMBI, ByteAcc, NoLock, Preserve)
   {
      HSTS,   8,
      Offset (0x02),
      HCON,   8,
      HCOM,   8,
      TXSA,   8,
      DAT0,   8,
      DAT1,   8,
      HBDR,   8,
      PECR,   8,
      RXSA,   8,
      SDAT,   16
   }

   Method (SSXB, 2, Serialized)
   {
      If (STRT ())
      {
         Return (0x00)
      }

      I2CE = 0x00
      HSTS = 0xBF
      TXSA = Arg0
      HCOM = Arg1
      HCON = 0x48
```

```
    If (COMP ())
    {
        HSTS |= 0xFF
        Return (0x01)
    }

    Return (0x00)
}

Method (SRXB, 1, Serialized)
{
    If (STRT ())
    {
        Return (0xFFFF)
    }

    I2CE = 0x00
    HSTS = 0xBF
    TXSA = (Arg0 | 0x01)
    HCON = 0x44
    If (COMP ())
    {
        HSTS |= 0xFF
        Return (DAT0) /* \_SB_.PCI0.SBUS.DAT0 */
    }

    Return (0xFFFF)
}

Method (SWRB, 3, Serialized)
{
    If (STRT ())
    {
        Return (0x00)
    }

    I2CE = 0x00
    HSTS = 0xBF
    TXSA = Arg0
    HCOM = Arg1
    DAT0 = Arg2
    HCON = 0x48
    If (COMP ())
    {
        HSTS |= 0xFF
        Return (0x01)
    }
```

```
    Return (0x00)
}

Method (SRDB, 2, Serialized)
{
    If (STRT ())
    {
        Return (0xFFFF)
    }

    I2CE = 0x00
    HSTS = 0xBF
    TXSA = (Arg0 | 0x01)
    HCOM = Arg1
    HCON = 0x48
    If (COMP ())
    {
        HSTS |= 0xFF
        Return (DAT0) /* \_SB_.PCI0.SBUS.DAT0 */
    }

    Return (0xFFFF)
}

Method (SWRW, 3, Serialized)
{
    If (STRT ())
    {
        Return (0x00)
    }

    I2CE = 0x00
    HSTS = 0xBF
    TXSA = Arg0
    HCOM = Arg1
    DAT1 = (Arg2 & 0xFF)
    DAT0 = ((Arg2 >> 0x08) & 0xFF)
    HCON = 0x4C
    If (COMP ())
    {
        HSTS |= 0xFF
        Return (0x01)
    }

    Return (0x00)
}
```

```
Method (SRDW, 2, Serialized)
{
    If (STRT ())
    {
        Return (0xFFFF)
    }

    I2CE = 0x00
    HSTS = 0xBF
    TXSA = (Arg0 | 0x01)
    HCOM = Arg1
    HCON = 0x4C
    If (COMP ())
    {
        HSTS |= 0xFF
        Return (((DAT0 << 0x08) | DAT1))
    }

    Return (0xFFFFFFFF)
}

Method (SBLW, 4, Serialized)
{
    If (STRT ())
    {
        Return (0x00)
    }

    I2CE = Arg3
    HSTS = 0xBF
    TXSA = Arg0
    HCOM = Arg1
    DAT0 = SizeOf (Arg2)
    Local1 = 0x00
    HBDR = DerefOf (Arg2 [0x00])
    HCON = 0x54
    While ((SizeOf (Arg2) > Local1))
    {
        Local0 = 0x4E20
        While ((!(HSTS & 0x80) && Local0))
        {
            Local0--
        }

        If (!Local0)
        {
```

```
            KILL ()
            Return (0x00)
        }

        Local1++
        If ((SizeOf (Arg2) > Local1))
        {
            HBDR = DerefOf (Arg2 [Local1])
            HSTS = 0x80
        }
    }

    HSTS = 0x80
    If (COMP ())
    {
        HSTS |= 0xFF
        Return (0x01)
    }

    Return (0x00)
}

Method (SBLR, 3, Serialized)
{
    Name (TBUF, Buffer (0x0100){})
    If (STRT ())
    {
        Return (0x00)
    }

    I2CE = Arg2
    HSTS = 0xBF
    TXSA = (Arg0 | 0x01)
    HCOM = Arg1
    HCON = 0x54
    Local0 = 0x0FA0
    While ((!(HSTS & 0x80) && Local0))
    {
        Local0--
        Stall (0x32)
    }

    If (!Local0)
    {
        KILL ()
        Return (0x00)
    }
```

```
    TBUF [0x00] = DAT0 /* \_SB_.PCI0.SBUS.DAT0 */
    HSTS = 0x80
    Local1 = 0x01
    While ((Local1 < DerefOf (TBUF [0x00])))
    {
        Local0 = 0x0FA0
        While ((!(HSTS & 0x80) && Local0))
        {
            Local0--
            Stall (0x32)
        }

        If (!Local0)
        {
            KILL ()
            Return (0x00)
        }

        TBUF [Local1] = HBDR /* \_SB_.PCI0.SBUS.HBDR */
        HSTS = 0x80
        Local1++
    }

    If (COMP ())
    {
        HSTS |= 0xFF
        Return (TBUF) /* \_SB_.PCI0.SBUS.SBLR.TBUF */
    }

    Return (0x00)
}

Method (STRT, 0, Serialized)
{
    Local0 = 0xC8
    While (Local0)
    {
        If ((HSTS & 0x40))
        {
            Local0--
            Sleep (0x01)
            If ((Local0 == 0x00))
            {
                Return (0x01)
            }
        }
```

```
        Else
        {
            Local0 = 0x00
        }
    }

    Local0 = 0x0FA0
    While (Local0)
    {
        If ((HSTS & 0x01))
        {
            Local0--
            Stall (0x32)
            If ((Local0 == 0x00))
            {
                KILL ()
            }
        }
        Else
        {
            Return (0x00)
        }
    }

    Return (0x01)
}

Method (COMP, 0, Serialized)
{
    Local0 = 0x0FA0
    While (Local0)
    {
        If ((HSTS & 0x02))
        {
            Return (0x01)
        }
        Else
        {
            Local0--
            Stall (0x32)
            If ((Local0 == 0x00))
            {
                KILL ()
            }
        }
    }
```

```
        Return (0x00)
    }

    Method (KILL, 0, Serialized)
    {
        HCON |= 0x02
        HSTS |= 0xFF
    }

    Device (BUS0)
    {
        Name (_CID, "smbus")  // _CID: Compatible ID
        Name (_ADR, 0x00)  // _ADR: Address
    }

    Device (BUS1)
    {
        Name (_CID, "smbus")  // _CID: Compatible ID
        Name (_ADR, 0x01)  // _ADR: Address
    }
}

Device (LPCB)
{
    Name (_ADR, 0x001F0000)  // _ADR: Address
    Scope (\_SB)
    {
        OperationRegion (\_SB.PCI0.LPCB.LPC1, PCI_Config, 0x40, 0xC0)
        Field (\_SB.PCI0.LPCB.LPC1, AnyAcc, NoLock, Preserve)
        {
            Offset (0x20),
            PARC,   8,
            PBRC,   8,
            PCRC,   8,
            PDRC,   8,
            Offset (0x28),
            PERC,   8,
            PFRC,   8,
            PGRC,   8,
            PHRC,   8
        }

        Device (LNKA)
        {
            Name (_HID, EisaId ("PNP0C0F") /* PCI Interrupt Link Device */)  // _
            Name (_UID, 0x01)  // _UID: Unique ID
            Method (_DIS, 0, Serialized)  // _DIS: Disable Device
```

```
    {
        PARC |= 0x80
    }

    Name (_PRS, ResourceTemplate ()  // _PRS: Possible Resource Set
    {
        IRQ (Level, ActiveLow, Shared, )
            {1,3,4,5,6,7,10,12,14,15}
    })
    Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
    {
        Name (RTLA, ResourceTemplate ()
        {
            IRQ (Level, ActiveLow, Shared, _Y0E)
                {}
        })
        CreateWordField (RTLA, \_SB.LNKA._CRS._Y0E._INT, IRQ0)  // _
        IRQ0 = Zero
        IRQ0 = (0x01 << (PARC & 0x0F))
        Return (RTLA) /* \_SB_.LNKA._CRS.RTLA */
    }

    Method (_SRS, 1, Serialized)  // _SRS: Set Resource Settings
    {
        CreateWordField (Arg0, 0x01, IRQ0)
        FindSetRightBit (IRQ0, Local0)
        Local0--
        PARC = Local0
    }

    Method (_STA, 0, Serialized)  // _STA: Status
    {
        If ((PARC & 0x80))
        {
            Return (0x09)
        }
        Else
        {
            Return (0x0B)
        }
    }
}

Device (LNKB)
{
    Name (_HID, EisaId ("PNP0C0F") /* PCI Interrupt Link Device */)  // _
    Name (_UID, 0x02)  // _UID: Unique ID
```

```
    Method (_DIS, 0, Serialized)  // _DIS: Disable Device
    {
        PBRC |= 0x80
    }

    Name (_PRS, ResourceTemplate ()  // _PRS: Possible Resource Set
    {
        IRQ (Level, ActiveLow, Shared, )
            {1,3,4,5,6,7,11,12,14,15}
    })
    Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
    {
        Name (RTLB, ResourceTemplate ()
        {
            IRQ (Level, ActiveLow, Shared, _Y0F)
                {}
        })
        CreateWordField (RTLB, \_SB.LNKB._CRS._Y0F._INT, IRQ0)  // _
        IRQ0 = Zero
        IRQ0 = (0x01 << (PBRC & 0x0F))
        Return (RTLB) /* \_SB_.LNKB._CRS.RTLB */
    }

    Method (_SRS, 1, Serialized)  // _SRS: Set Resource Settings
    {
        CreateWordField (Arg0, 0x01, IRQ0)
        FindSetRightBit (IRQ0, Local0)
        Local0--
        PBRC = Local0
    }

    Method (_STA, 0, Serialized)  // _STA: Status
    {
        If ((PBRC & 0x80))
        {
            Return (0x09)
        }
        Else
        {
            Return (0x0B)
        }
    }
}

Device (LNKC)
{
    Name (_HID, EisaId ("PNP0C0F") /* PCI Interrupt Link Device */)  // _
```

```
    Name (_UID, 0x03)  // _UID: Unique ID
    Method (_DIS, 0, Serialized)  // _DIS: Disable Device
    {
        PCRC |= 0x80
    }

    Name (_PRS, ResourceTemplate ()  // _PRS: Possible Resource Sett
    {
        IRQ (Level, ActiveLow, Shared, )
            {1,3,4,5,6,7,10,12,14,15}
    })
    Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
    {
        Name (RTLC, ResourceTemplate ()
        {
            IRQ (Level, ActiveLow, Shared, _Y10)
                {}
        })
        CreateWordField (RTLC, \_SB.LNKC._CRS._Y10._INT, IRQ0)  // _
        IRQ0 = Zero
        IRQ0 = (0x01 << (PCRC & 0x0F))
        Return (RTLC) /* \_SB_.LNKC._CRS.RTLC */
    }

    Method (_SRS, 1, Serialized)  // _SRS: Set Resource Settings
    {
        CreateWordField (Arg0, 0x01, IRQ0)
        FindSetRightBit (IRQ0, Local0)
        Local0--
        PCRC = Local0
    }

    Method (_STA, 0, Serialized)  // _STA: Status
    {
        If ((PCRC & 0x80))
        {
            Return (0x09)
        }
        Else
        {
            Return (0x0B)
        }
    }
}

Device (LNKD)
{
```

```
    Name (_HID, EisaId ("PNP0C0F") /* PCI Interrupt Link Device */)  // _
    Name (_UID, 0x04)  // _UID: Unique ID
    Method (_DIS, 0, Serialized)  // _DIS: Disable Device
    {
        PDRC |= 0x80
    }

    Name (_PRS, ResourceTemplate ()  // _PRS: Possible Resource Set
    {
        IRQ (Level, ActiveLow, Shared, )
            {1,3,4,5,6,7,11,12,14,15}
    })
    Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
    {
        Name (RTLD, ResourceTemplate ()
        {
            IRQ (Level, ActiveLow, Shared, _Y11)
                {}
        })
        CreateWordField (RTLD, \_SB.LNKD._CRS._Y11._INT, IRQ0)  // _
        IRQ0 = Zero
        IRQ0 = (0x01 << (PDRC & 0x0F))
        Return (RTLD) /* \_SB_.LNKD._CRS.RTLD */
    }

    Method (_SRS, 1, Serialized)  // _SRS: Set Resource Settings
    {
        CreateWordField (Arg0, 0x01, IRQ0)
        FindSetRightBit (IRQ0, Local0)
        Local0--
        PDRC = Local0
    }

    Method (_STA, 0, Serialized)  // _STA: Status
    {
        If ((PDRC & 0x80))
        {
            Return (0x09)
        }
        Else
        {
            Return (0x0B)
        }
    }
}

Device (LNKE)
```

```
{
    Name (_HID, EisaId ("PNP0C0F") /* PCI Interrupt Link Device */)  // _
    Name (_UID, 0x05)  // _UID: Unique ID
    Method (_DIS, 0, Serialized)  // _DIS: Disable Device
    {
        PERC |= 0x80
    }

    Name (_PRS, ResourceTemplate ()  // _PRS: Possible Resource Set
    {
        IRQ (Level, ActiveLow, Shared, )
            {1,3,4,5,6,7,10,12,14,15}
    })
    Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
    {
        Name (RTLE, ResourceTemplate ()
        {
            IRQ (Level, ActiveLow, Shared, _Y12)
                {}
        })
        CreateWordField (RTLE, \_SB.LNKE._CRS._Y12._INT, IRQ0)  // _
        IRQ0 = Zero
        IRQ0 = (0x01 << (PERC & 0x0F))
        Return (RTLE) /* \_SB_.LNKE._CRS.RTLE */
    }

    Method (_SRS, 1, Serialized)  // _SRS: Set Resource Settings
    {
        CreateWordField (Arg0, 0x01, IRQ0)
        FindSetRightBit (IRQ0, Local0)
        Local0--
        PERC = Local0
    }

    Method (_STA, 0, Serialized)  // _STA: Status
    {
        If ((PERC & 0x80))
        {
            Return (0x09)
        }
        Else
        {
            Return (0x0B)
        }
    }
}
```

```
Device (LNKF)
{
   Name (_HID, EisaId ("PNP0C0F") /* PCI Interrupt Link Device */)  // _
   Name (_UID, 0x06)  // _UID: Unique ID
   Method (_DIS, 0, Serialized)  // _DIS: Disable Device
   {
      PFRC |= 0x80
   }

   Name (_PRS, ResourceTemplate ()  // _PRS: Possible Resource Sett
   {
      IRQ (Level, ActiveLow, Shared, )
         {1,3,4,5,6,7,11,12,14,15}
   })
   Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
   {
      Name (RTLF, ResourceTemplate ()
      {
         IRQ (Level, ActiveLow, Shared, _Y13)
            {}
      })
      CreateWordField (RTLF, \_SB.LNKF._CRS._Y13._INT, IRQ0)  // _I
      IRQ0 = Zero
      IRQ0 = (0x01 << (PFRC & 0x0F))
      Return (RTLF) /* \_SB_.LNKF._CRS.RTLF */
   }

   Method (_SRS, 1, Serialized)  // _SRS: Set Resource Settings
   {
      CreateWordField (Arg0, 0x01, IRQ0)
      FindSetRightBit (IRQ0, Local0)
      Local0--
      PFRC = Local0
   }

   Method (_STA, 0, Serialized)  // _STA: Status
   {
      If ((PFRC & 0x80))
      {
         Return (0x09)
      }
      Else
      {
         Return (0x0B)
      }
   }
}
```

```
Device (LNKG)
{
   Name (_HID, EisaId ("PNP0C0F") /* PCI Interrupt Link Device */)  // _
   Name (_UID, 0x07)  // _UID: Unique ID
   Method (_DIS, 0, Serialized)  // _DIS: Disable Device
   {
      PGRC |= 0x80
   }

   Name (_PRS, ResourceTemplate ()  // _PRS: Possible Resource Set
   {
      IRQ (Level, ActiveLow, Shared, )
         {1,3,4,5,6,7,10,12,14,15}
   })
   Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
   {
      Name (RTLG, ResourceTemplate ()
      {
         IRQ (Level, ActiveLow, Shared, _Y14)
            {}
      })
      CreateWordField (RTLG, \_SB.LNKG._CRS._Y14._INT, IRQ0)  // _
      IRQ0 = Zero
      IRQ0 = (0x01 << (PGRC & 0x0F))
      Return (RTLG) /* \_SB_.LNKG._CRS.RTLG */
   }

   Method (_SRS, 1, Serialized)  // _SRS: Set Resource Settings
   {
      CreateWordField (Arg0, 0x01, IRQ0)
      FindSetRightBit (IRQ0, Local0)
      Local0--
      PGRC = Local0
   }

   Method (_STA, 0, Serialized)  // _STA: Status
   {
      If ((PGRC & 0x80))
      {
         Return (0x09)
      }
      Else
      {
         Return (0x0B)
      }
   }
```

```
}

Device (LNKH)
{
   Name (_HID, EisaId ("PNP0C0F") /* PCI Interrupt Link Device */)  // _
   Name (_UID, 0x08)  // _UID: Unique ID
   Method (_DIS, 0, Serialized)  // _DIS: Disable Device
   {
      PHRC |= 0x80
   }

   Name (_PRS, ResourceTemplate ()  // _PRS: Possible Resource Set
   {
      IRQ (Level, ActiveLow, Shared, )
         {1,3,4,5,6,7,11,12,14,15}
   })
   Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
   {
      Name (RTLH, ResourceTemplate ()
      {
         IRQ (Level, ActiveLow, Shared, _Y15)
            {}
      })
      CreateWordField (RTLH, \_SB.LNKH._CRS._Y15._INT, IRQ0)  // _
      IRQ0 = Zero
      IRQ0 = (0x01 << (PHRC & 0x0F))
      Return (RTLH) /* \_SB_.LNKH._CRS.RTLH */
   }

   Method (_SRS, 1, Serialized)  // _SRS: Set Resource Settings
   {
      CreateWordField (Arg0, 0x01, IRQ0)
      FindSetRightBit (IRQ0, Local0)
      Local0--
      PHRC = Local0
   }

   Method (_STA, 0, Serialized)  // _STA: Status
   {
      If ((PHRC & 0x80))
      {
         Return (0x09)
      }
      Else
      {
         Return (0x0B)
      }
```

```
      }
   }
}

OperationRegion (LPC0, PCI_Config, 0x40, 0xC0)
Field (LPC0, AnyAcc, NoLock, Preserve)
{
   Offset (0x40),
   IOD0,   8,
   IOD1,   8,
   Offset (0xB0),
   RAEN,   1,
      ,   13,
   RCBA,   18
}

Device (DMAC)
{
   Name (_HID, EisaId ("PNP0200") /* PC-class DMA Controller */)  // _HI
   Name (_CRS, ResourceTemplate ()  // _CRS: Current Resource Setting
   {
      IO (Decode16,
         0x0000,             // Range Minimum
         0x0000,             // Range Maximum
         0x01,               // Alignment
         0x20,               // Length
         )
      IO (Decode16,
         0x0081,             // Range Minimum
         0x0081,             // Range Maximum
         0x01,               // Alignment
         0x11,               // Length
         )
      IO (Decode16,
         0x0093,             // Range Minimum
         0x0093,             // Range Maximum
         0x01,               // Alignment
         0x0D,               // Length
         )
      IO (Decode16,
         0x00C0,             // Range Minimum
         0x00C0,             // Range Maximum
         0x01,               // Alignment
         0x20,               // Length
         )
      DMA (Compatibility, NotBusMaster, Transfer8_16, )
         {4}
```

```
            })
        }

        Device (FWHD)
        {
            Name (_HID, EisaId ("INT0800") /* Intel 82802 Firmware Hub Device */)
Hardware ID
            Name (_CRS, ResourceTemplate ()  // _CRS: Current Resource Setting
            {
                Memory32Fixed (ReadOnly,
                    0xFF000000,         // Address Base
                    0x01000000,         // Address Length
                    )
            })
        }

        Device (HPET)
        {
            Name (_HID, EisaId ("PNP0103") /* HPET System Timer */)  // _HID: Ha
            Name (_CID, EisaId ("PNP0C01") /* System Board */)  // _CID: Compati
            Name (BUF0, ResourceTemplate ()
            {
                IRQNoFlags ()
                    {0}
                IRQNoFlags ()
                    {8}
                Memory32Fixed (ReadWrite,
                    0xFED00000,         // Address Base
                    0x00004000,         // Address Length
                    _Y16)
            })
            Method (_STA, 0, NotSerialized)  // _STA: Status
            {
                If ((OSYS >= 0x07D1))
                {
                    If (HPAE)
                    {
                        Return (0x0F)
                    }
                }
                ElseIf (HPAE)
                {
                    Return (0x0B)
                }

                Return (0x00)
            }
```

```
            Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
            {
                If (HPAE)
                {
                    CreateDWordField (BUF0, \_SB.PCI0.LPCB.HPET._Y16._BAS, HP
Base Address
                    If ((HPAS == 0x01))
                    {
                        HPT0 = 0xFED01000
                    }

                    If ((HPAS == 0x02))
                    {
                        HPT0 = 0xFED02000
                    }

                    If ((HPAS == 0x03))
                    {
                        HPT0 = 0xFED03000
                    }
                }

                Return (BUF0) /* \_SB_.PCI0.LPCB.HPET.BUF0 */
            }
        }

        Device (IPIC)
        {
            Name (_HID, EisaId ("PNP0000") /* 8259-compatible Programmable Int
*/)  // _HID: Hardware ID
            Name (_CRS, ResourceTemplate ()  // _CRS: Current Resource Setting
            {
                IO (Decode16,
                    0x0020,             // Range Minimum
                    0x0020,             // Range Maximum
                    0x01,               // Alignment
                    0x02,               // Length
                    )
                IO (Decode16,
                    0x0024,             // Range Minimum
                    0x0024,             // Range Maximum
                    0x01,               // Alignment
                    0x02,               // Length
                    )
                IO (Decode16,
                    0x0028,             // Range Minimum
```

```
    0x0028,             // Range Maximum
    0x01,               // Alignment
    0x02,               // Length
    )
IO (Decode16,
    0x002C,             // Range Minimum
    0x002C,             // Range Maximum
    0x01,               // Alignment
    0x02,               // Length
    )
IO (Decode16,
    0x0030,             // Range Minimum
    0x0030,             // Range Maximum
    0x01,               // Alignment
    0x02,               // Length
    )
IO (Decode16,
    0x0034,             // Range Minimum
    0x0034,             // Range Maximum
    0x01,               // Alignment
    0x02,               // Length
    )
IO (Decode16,
    0x0038,             // Range Minimum
    0x0038,             // Range Maximum
    0x01,               // Alignment
    0x02,               // Length
    )
IO (Decode16,
    0x003C,             // Range Minimum
    0x003C,             // Range Maximum
    0x01,               // Alignment
    0x02,               // Length
    )
IO (Decode16,
    0x00A0,             // Range Minimum
    0x00A0,             // Range Maximum
    0x01,               // Alignment
    0x02,               // Length
    )
IO (Decode16,
    0x00A4,             // Range Minimum
    0x00A4,             // Range Maximum
    0x01,               // Alignment
    0x02,               // Length
    )
IO (Decode16,
```

```
            0x00A8,             // Range Minimum
            0x00A8,             // Range Maximum
            0x01,               // Alignment
            0x02,               // Length
            )
        IO (Decode16,
            0x00AC,             // Range Minimum
            0x00AC,             // Range Maximum
            0x01,               // Alignment
            0x02,               // Length
            )
        IO (Decode16,
            0x00B0,             // Range Minimum
            0x00B0,             // Range Maximum
            0x01,               // Alignment
            0x02,               // Length
            )
        IO (Decode16,
            0x00B4,             // Range Minimum
            0x00B4,             // Range Maximum
            0x01,               // Alignment
            0x02,               // Length
            )
        IO (Decode16,
            0x00B8,             // Range Minimum
            0x00B8,             // Range Maximum
            0x01,               // Alignment
            0x02,               // Length
            )
        IO (Decode16,
            0x00BC,             // Range Minimum
            0x00BC,             // Range Maximum
            0x01,               // Alignment
            0x02,               // Length
            )
        IO (Decode16,
            0x04D0,             // Range Minimum
            0x04D0,             // Range Maximum
            0x01,               // Alignment
            0x02,               // Length
            )
        IRQNoFlags ()
            {2}
    })
}

Device (MATH)
```

```
            {
                Name (_HID, EisaId ("PNP0C04") /* x87-compatible Floating Point Proc
_HID: Hardware ID
                Name (_CRS, ResourceTemplate ()  // _CRS: Current Resource Setting
                {
                    IO (Decode16,
                        0x00F0,             // Range Minimum
                        0x00F0,             // Range Maximum
                        0x01,               // Alignment
                        0x01,               // Length
                        )
                    IRQNoFlags ()
                        {13}
                })
            }

            Device (LDRC)
            {
                Name (_HID, EisaId ("PNP0C02") /* PNP Motherboard Resources */)  //
ID
                Name (_UID, 0x02)  // _UID: Unique ID
                Name (_CRS, ResourceTemplate ()  // _CRS: Current Resource Setting
                {
                    IO (Decode16,
                        0x002E,             // Range Minimum
                        0x002E,             // Range Maximum
                        0x01,               // Alignment
                        0x02,               // Length
                        )
                    IO (Decode16,
                        0x004E,             // Range Minimum
                        0x004E,             // Range Maximum
                        0x01,               // Alignment
                        0x02,               // Length
                        )
                    IO (Decode16,
                        0x0061,             // Range Minimum
                        0x0061,             // Range Maximum
                        0x01,               // Alignment
                        0x01,               // Length
                        )
                    IO (Decode16,
                        0x0063,             // Range Minimum
                        0x0063,             // Range Maximum
                        0x01,               // Alignment
                        0x01,               // Length
                        )
```

```
IO (Decode16,
   0x0065,          // Range Minimum
   0x0065,          // Range Maximum
   0x01,            // Alignment
   0x01,            // Length
   )
IO (Decode16,
   0x0067,          // Range Minimum
   0x0067,          // Range Maximum
   0x01,            // Alignment
   0x01,            // Length
   )
IO (Decode16,
   0x0080,          // Range Minimum
   0x0080,          // Range Maximum
   0x01,            // Alignment
   0x01,            // Length
   )
IO (Decode16,
   0x0092,          // Range Minimum
   0x0092,          // Range Maximum
   0x01,            // Alignment
   0x01,            // Length
   )
IO (Decode16,
   0x00B2,          // Range Minimum
   0x00B2,          // Range Maximum
   0x01,            // Alignment
   0x02,            // Length
   )
IO (Decode16,
   0xFFFF,          // Range Minimum
   0xFFFF,          // Range Maximum
   0x01,            // Alignment
   0x01,            // Length
   )
IO (Decode16,
   0x1800,          // Range Minimum
   0x1800,          // Range Maximum
   0x01,            // Alignment
   0x80,            // Length
   )
IO (Decode16,
   0x0800,          // Range Minimum
   0x0800,          // Range Maximum
   0x01,            // Alignment
   0x80,            // Length
```

```
            )
    })
}

Device (RTC)
{
    Name (_HID, EisaId ("PNP0B00") /* AT Real-Time Clock */)  // _HID: Ha
    Name (_CRS, ResourceTemplate ()  // _CRS: Current Resource Setting
    {
        IO (Decode16,
            0x0070,             // Range Minimum
            0x0070,             // Range Maximum
            0x01,               // Alignment
            0x08,               // Length
            )
    })
    OperationRegion (CMS0, SystemCMOS, 0x00, 0x40)
    Field (CMS0, ByteAcc, NoLock, Preserve)
    {
        Offset (0x38),
        ISTB,   1,
        Offset (0x39)
    }
}

Device (TIMR)
{
    Name (_HID, EisaId ("PNP0100") /* PC-class System Timer */)  // _HID:
    Name (_CRS, ResourceTemplate ()  // _CRS: Current Resource Setting
    {
        IO (Decode16,
            0x0040,             // Range Minimum
            0x0040,             // Range Maximum
            0x01,               // Alignment
            0x04,               // Length
            )
        IO (Decode16,
            0x0050,             // Range Minimum
            0x0050,             // Range Maximum
            0x10,               // Alignment
            0x04,               // Length
            )
    })
}

Device (SMC)
{
```

```
    Name (_HID, EisaId ("APP0001"))  // _HID: Hardware ID
    Name (_CID, "smc-huronriver")  // _CID: Compatible ID
    Name (_STA, 0x0B)  // _STA: Status
    Name (_CRS, ResourceTemplate ()  // _CRS: Current Resource Setting
    {
        IO (Decode16,
            0x0300,             // Range Minimum
            0x0300,             // Range Maximum
            0x01,               // Alignment
            0x20,               // Length
            )
        Memory32Fixed (ReadWrite,
            0xFEF00000,         // Address Base
            0x00010000,         // Address Length
            )
        IRQNoFlags ()
            {6}
    })
}

Device (ALS0)
{
    Name (_HID, "ACPI0008" /* Ambient Light Sensor Device */)  // _HID: H
    Name (_CID, "smc-als")  // _CID: Compatible ID
    Name (BUFF, Buffer (0x02){})
    CreateByteField (BUFF, 0x00, OB0)
    CreateByteField (BUFF, 0x01, OB1)
    CreateWordField (BUFF, 0x00, ALSI)
    Method (_STA, 0, NotSerialized)  // _STA: Status
    {
        If ((OSYS >= 0x07D9))
        {
            Return (0x0F)
        }
        Else
        {
            Return (0x00)
        }
    }

    Method (_ALI, 0, NotSerialized)  // _ALI: Ambient Light Illuminance
    {
        OB0 = \_SB.PCI0.LPCB.EC.ALB0
        OB1 = \_SB.PCI0.LPCB.EC.ALB1
        Local0 = ALSI /* \_SB_.PCI0.LPCB.ALS0.ALSI */
        Return (Local0)
    }
```

```
    Name (_ALR, Package (0x05)  // _ALR: Ambient Light Response
    {
        Package (0x02)
        {
            0x0A,
            0x00
        },

        Package (0x02)
        {
            0x14,
            0x0A
        },

        Package (0x02)
        {
            0x32,
            0x50
        },

        Package (0x02)
        {
            0x5A,
            0x012C
        },

        Package (0x02)
        {
            0x64,
            0x03E8
        }
    })
}

Device (EC)
{
    Name (_HID, EisaId ("PNP0C09") /* Embedded Controller Device */)  //
    Name (_UID, 0x00)  // _UID: Unique ID
    Name (_CRS, ResourceTemplate ()  // _CRS: Current Resource Setting
    {
        IO (Decode16,
            0x0062,             // Range Minimum
            0x0062,             // Range Maximum
            0x00,               // Alignment
            0x01,               // Length
            )
```

```
    IO (Decode16,
        0x0066,             // Range Minimum
        0x0066,             // Range Maximum
        0x00,               // Alignment
        0x01,               // Length
        )
})
Name (_GPE, 0x4E)  // _GPE: General Purpose Events
Method (_PRW, 0, NotSerialized)  // _PRW: Power Resources for Wake
{
    If (OSDW ())
    {
        Return (Package (0x02)
        {
            0x70,
            0x04
        })
    }
    Else
    {
        Return (Package (0x02)
        {
            0x70,
            0x03
        })
    }
}

Name (ECOK, 0x00)
OperationRegion (ECOR, EmbeddedControl, 0x00, 0xFF)
Field (ECOR, ByteAcc, NoLock, Preserve)
{
    ECVS,   8,
    Offset (0x02),
    Offset (0x03),
    G3HT,   1,
    Offset (0x04),
    WBCB,   1,
    DSLP,   1,
    Offset (0x05),
    Offset (0x06),
    WKRS,   8,
    Offset (0x10),
    ECSS,   8,
    PLIM,   8,
    ALB0,   8,
    ALB1,   8,
```

WTLB, 8,
WTMB, 8,
Offset (0x20),
SPTR, 8,
SSTS, 8,
SADR, 8,
SCMD, 8,
SBFR, 256,
SCNT, 8,
SAAD, 8,
SAD0, 8,
SAD1, 8,
SMUX, 8,
Offset (0x60),
ELSW, 1,
EACP, 1,
ECDI, 1,
ENMI, 1,
Offset (0x61),
EMHP, 1,
Offset (0x62),
Offset (0x63),
Offset (0x64),
SWLO, 1,
SWLC, 1,
SWAI, 1,
SWAR, 1,
SWCI, 1,
SWCE, 1,
SWMI, 1,
SWMR, 1,
SWPB, 1,
SWGP, 1,
SWPM, 1,
SWWT, 1,
SWLB, 1,
Offset (0x66),
Offset (0x67),
Offset (0x68),
EWLO, 1,
EWLC, 1,
EWAI, 1,
EWAR, 1,
EWCI, 1,
EWCE, 1,
EWMI, 1,
EWMR, 1,

```
    EWPB,   1,
    EWGP,   1,
    EWPM,   1,
    ENWT,   1,
    EWLB,   1,
    EWDK,   1,
    Offset (0x6A),
    Offset (0x6B),
    Offset (0x6C),
    LWLO,   1,
    LWLC,   1,
    LWAI,   1,
    LWAR,   1,
    LWCI,   1,
    LWCE,   1,
    LWMI,   1,
    LWMR,   1,
    LWPB,   1,
    LWGP,   1,
    LWPM,   1,
    LWWT,   1,
    LWLB,   1,
    Offset (0x6E),
    Offset (0x6F),
    Offset (0x70)
}

Field (ECOR, ByteAcc, NoLock, Preserve)
{
    Offset (0x03),
    G3AD,   1,
    BLOD,   1,
    S4WE,   1,
    APWC,   1,
    BTPC,   1,
    Offset (0x04),
    Offset (0x6C),
    LWE0,   8,
    LWE1,   8,
    LWE2,   8,
    LWE3,   8
}

Field (ECOR, ByteAcc, NoLock, Preserve)
{
    Offset (0x24),
    SBDW,   16,
```

```
    Offset (0x46),
    SADW,   16
}

Method (WAKE, 0, NotSerialized)
{
    If (ECOK)
    {
        Return (WKRS) /* \_SB_.PCI0.LPCB.EC__.WKRS */
    }
    Else
    {
        Return (0x00)
    }
}

Device (SMB0)
{
    Name (_HID, "ACPI0001" /* SMBus 1.0 Host Controller */)  // _HID: H
    Name (_EC, 0x2010)  // _EC_: Embedded Controller
    Mutex (SMTX, 0x00)
    Method (_STA, 0, NotSerialized)  // _STA: Status
    {
        If (OSDW ())
        {
            Return (0x0F)
        }
        Else
        {
            Return (0x00)
        }
    }

    Device (SBS0)
    {
        Name (_HID, "ACPI0002" /* Smart Battery Subsystem */)  // _HID:
        Name (_SBS, 0x01)  // _SBS: Smart Battery Subsystem
    }

    Method (SBPC, 1, NotSerialized)
    {
        Local0 = Arg0
        While (Local0)
        {
            If ((SPTR == 0x00))
            {
                Return ((SSTS & 0x1F))
```

```
        }

        Sleep (0x01)
        Local0--
    }

    Return (0x18)
}

Method (SBRW, 3, NotSerialized)
{
    Local0 = One
    If (!Acquire (\_SB.PCI0.LPCB.EC.SMB0.SMTX, 0xFFFF))
    {
        If ((SPTR == 0x00))
        {
            SADR = (Arg0 << 0x01)
            SCMD = Arg1
            SPTR = 0x09
            Local0 = SBPC (0x03E8)
            If (!Local0)
            {
                Arg2 = SBDW /* \_SB_.PCI0.LPCB.EC__.SBDW */
            }
        }

        Release (\_SB.PCI0.LPCB.EC.SMB0.SMTX)
    }

    Return (Local0)
}

Method (SBRB, 3, NotSerialized)
{
    Local0 = One
    Local1 = Buffer (0x01)
        {
            0x00                                    // .
        }
    If (!Acquire (\_SB.PCI0.LPCB.EC.SMB0.SMTX, 0xFFFF))
    {
        If ((SPTR == 0x00))
        {
            SADR = (Arg0 << 0x01)
            SCMD = Arg1
            SPTR = 0x0B
            Local0 = SBPC (0x03E8)
```

```
            If (!Local0)
            {
                Arg2 = SBFR /* \_SB_.PCI0.LPCB.EC__.SBFR */
            }
        }

        Release (\_SB.PCI0.LPCB.EC.SMB0.SMTX)
    }

    Return (Local0)
  }
}

Method (_Q10, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
{
  If (OSDW ())
  {
    Notify (\_SB.PCI0.LPCB.EC.SMB0, 0x80) // Status Change
  }
  ElseIf ((SSTS & 0x40))
  {
    If (!Acquire (\_SB.PCI0.LPCB.EC.SMB0.SMTX, 0xFFFF))
    {
      Local0 = (SAAD >> 0x01)
      If ((Local0 == 0x0A))
      {
        \_SB.BAT0.BNOT (SADW)
      }

      SSTS = 0x00
      Release (\_SB.PCI0.LPCB.EC.SMB0.SMTX)
    }
  }
}

Method (_Q20, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
{
  LIDS = ELSW /* \_SB_.PCI0.LPCB.EC__.ELSW */
  \_SB.PCI0.IGPU.CLID = ELSW /* \_SB_.PCI0.LPCB.EC__.ELSW */
  Notify (\_SB.LID0, 0x80) // Status Change
}

Method (_Q21, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
{
  If (EACP)
  {
    PWRS = 0x01
```

```
    }
    Else
    {
        PWRS = 0x00
    }

    Notify (\_SB.ADP1, 0x80) // Status Change
    PNOT ()
}

Method (_Q40, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
{
    Notify (\_SB.PCI0.LPCB.ALS0, 0x80) // Status Change
}

Method (_Q5A, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
{
    Notify (\_SB.SLPB, 0x80) // Status Change
}

Method (_Q80, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
{
    Notify (\_PR.CPU0, 0x80) // Performance Capability Change
    Notify (\_PR.CPU1, 0x80) // Performance Capability Change
    Notify (\_PR.CPU2, 0x80) // Performance Capability Change
    Notify (\_PR.CPU3, 0x80) // Performance Capability Change
    Notify (\_PR.CPU4, 0x80) // Performance Capability Change
    Notify (\_PR.CPU5, 0x80) // Performance Capability Change
    Notify (\_PR.CPU6, 0x80) // Performance Capability Change
    Notify (\_PR.CPU7, 0x80) // Performance Capability Change
    If ((\_SB.PCI0.IGPU.VID0 == 0x8086))
    {
        Local0 = IGPS /* \IGPS */
        Local0 = (RP0C - Local0)
        RPSL = Local0
    }
    Else
    {
        Notify (\_SB.PCI0.PEG0.GFX0, 0x81) // Information Change
    }
}

Method (_QCE, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
{
}

Method (_QCF, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
```

```
    {
        If (!OSDW ())
        {
            Notify (\_SB.SLPB, 0x80) // Status Change
        }
    }

    Method (_QD0, 0, NotSerialized)  // _Qxx: EC Query, xx=0x00-0xFF
    {
    }

    Method (_REG, 2, NotSerialized)  // _REG: Region Availability
    {
        If (((Arg0 == 0x03) || (OSYS >= 0x07D6)))
        {
            ECOK = Arg1
            If ((Arg1 == 0x01))
            {
                ECSS = 0x00
                LIDS = ELSW /* \_SB_.PCI0.LPCB.EC__.ELSW */
                \_SB.PCI0.IGPU.CLID = ELSW /* \_SB_.PCI0.LPCB.EC__.ELS
                PWRS = EACP /* \_SB_.PCI0.LPCB.EC__.EACP */
                Notify (\_SB.ADP1, 0x80) // Status Change
            }
        }
    }
}

Scope (\_SB)
{
    Device (BAT0)
    {
        Name (_HID, EisaId ("PNP0C0A") /* Control Method Battery */)  // _H
        Name (_UID, 0x00)  // _UID: Unique ID
        Name (_PCL, Package (0x01)  // _PCL: Power Consumer List
        {
            \_SB
        })
        Name (BSSW, 0xFFFF)
        Name (PBIF, Package (0x0D)
        {
            0x00,
            0xFFFFFFFF,
            0xFFFFFFFF,
            0x01,
            0xFFFFFFFF,
            0xFA,
```

```
    0x64,
    0x0A,
    0x0A,
    " ",
    " ",
    " ",
    " "
})
Name (PBST, Package (0x04)
{
    0x00,
    0xFFFFFFFF,
    0xFFFFFFFF,
    0xFFFFFFFF
})
Method (_STA, 0, NotSerialized)  // _STA: Status
{
    If (OSDW ())
    {
        Return (0x00)
    }

    If (\_SB.PCI0.LPCB.EC.ECOK)
    {
        UBSS ()
        If ((BSSW & 0x01))
        {
            Return (0x1F)
        }
        Else
        {
            Return (0x0F)
        }
    }
    Else
    {
        Return (0x0F)
    }
}

Method (_BST, 0, NotSerialized)  // _BST: Battery Status
{
    If ((BSSW & 0x01))
    {
        UBST ()
    }
    Else
```

```
    {
        PBST [0x00] = 0x00
        PBST [0x01] = 0xFFFFFFFF
        PBST [0x02] = 0xFFFFFFFF
    }

    Return (PBST) /* \_SB_.BAT0.PBST */
}

Method (_BIF, 0, NotSerialized)  // _BIF: Battery Information
{
    If ((BSSW & 0x01))
    {
        UBIF ()
    }

    Return (PBIF) /* \_SB_.BAT0.PBIF */
}

Method (BNOT, 1, NotSerialized)
{
    Local0 = BSSW /* \_SB_.BAT0.BSSW */
    BSSW = Arg0
    Notify (\_SB.BAT0, 0x80) // Status Change
    If (((Local0 ^ Arg0) & 0x01))
    {
        Notify (\_SB.BAT0, 0x81) // Information Change
    }
}

Method (UBSS, 0, NotSerialized)
{
    \_SB.PCI0.LPCB.EC.SMB0.SBRW (0x0A, 0x01, RefOf (BSSW))
}

Method (UBIF, 0, NotSerialized)
{
    \_SB.PCI0.LPCB.EC.SMB0.SBRW (0x0B, 0x18, RefOf (Local0))
    PBIF [0x01] = (Local0 * 0x0A)
    \_SB.PCI0.LPCB.EC.SMB0.SBRW (0x0B, 0x10, RefOf (Local0))
    PBIF [0x02] = (Local0 * 0x0A)
    \_SB.PCI0.LPCB.EC.SMB0.SBRW (0x0B, 0x19, RefOf (Local0))
    PBIF [0x04] = Local0
    \_SB.PCI0.LPCB.EC.SMB0.SBRB (0x0B, 0x21, RefOf (Local0))
    PBIF [0x09] = Local0
    PBIF [0x0A] = Buffer (0x01)
        {
```

```
                0x00                                    // .
            }
            \_SB.PCI0.LPCB.EC.SMB0.SBRB (0x0B, 0x22, RefOf (Local0))
            PBIF [0x0B] = Local0
            \_SB.PCI0.LPCB.EC.SMB0.SBRB (0x0B, 0x20, RefOf (Local0))
            PBIF [0x0C] = Local0
        }

        Method (UBST, 0, NotSerialized)
        {
            \_SB.PCI0.LPCB.EC.SMB0.SBRW (0x0B, 0x09, RefOf (Local2))
            PBST [0x03] = Local2
            \_SB.PCI0.LPCB.EC.SMB0.SBRW (0x0B, 0x0A, RefOf (Local0))
            If ((Local0 & 0x8000))
            {
                Local0 = ~Local0
                Local0 = (Local0++ & 0xFFFF)
            }

            Local0 *= Local2
            PBST [0x01] = (Local0 / 0x03E8)
            \_SB.PCI0.LPCB.EC.SMB0.SBRW (0x0B, 0x0F, RefOf (Local0))
            PBST [0x02] = (Local0 * 0x0A)
            Local1 = 0x00
            If (PWRS)
            {
                \_SB.PCI0.LPCB.EC.SMB0.SBRW (0x0B, 0x16, RefOf (Local0))
                If (!(Local0 & 0x40))
                {
                    Local1 = 0x02
                }
            }
            Else
            {
                Local1 = 0x01
            }

            PBST [0x00] = Local1
        }
    }
  }
}

Device (HDEF)
{
    Name (_ADR, 0x001B0000)  // _ADR: Address
    Method (_PRW, 0, NotSerialized)  // _PRW: Power Resources for Wake
```

```
    {
        If (OSDW ())
        {
            Return (Package (0x02)
            {
                0x69,
                0x03
            })
        }
        Else
        {
            Return (Package (0x02)
            {
                0x69,
                0x03
            })
        }
    }

    Method (_PS0, 0, Serialized)  // _PS0: Power State 0
    {
        GD51 = 0x01
        Sleep (0x0F)
    }

    Method (_PS3, 0, Serialized)  // _PS3: Power State 3
    {
        GP51 = 0x00
        GD51 = 0x00
        Sleep (0x14)
    }
}

Device (RP01)
{
    Name (_ADR, 0x001C0000)  // _ADR: Address
    Method (_PRW, 0, NotSerialized)  // _PRW: Power Resources for Wake
    {
        If (OSDW ())
        {
            Return (Package (0x02)
            {
                0x69,
                0x03
            })
        }
        Else
```

```
        {
            Return (Package (0x02)
            {
                0x69,
                0x03
            })
        }
    }

    Method (_PRT, 0, NotSerialized)  // _PRT: PCI Routing Table
    {
        If (PICM)
        {
            Return (AR04 ())
        }

        Return (PR04 ())
    }

    Method (_DSM, 4, NotSerialized)  // _DSM: Device-Specific Method
    {
        If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b") /* Unk
        {
            Local0 = Package (0x02)
                {
                    "reg-ltrovr",
                    Buffer (0x08)
                    {
                        0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // ........
                    }
                }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }

        Return (0x00)
    }
}

Device (RP02)
{
    Name (_ADR, 0x001C0001)  // _ADR: Address
    Method (_PRW, 0, NotSerialized)  // _PRW: Power Resources for Wake
    {
        If (OSDW ())
        {
            Return (Package (0x02)
```

```
            {
                0x69,
                0x03
            })
        }
        Else
        {
            Return (Package (0x02)
            {
                0x69,
                0x03
            })
        }
    }

    Method (_PRT, 0, NotSerialized)  // _PRT: PCI Routing Table
    {
        If (PICM)
        {
            Return (AR05 ())
        }

        Return (PR05 ())
    }

    Method (_DSM, 4, NotSerialized)  // _DSM: Device-Specific Method
    {
        If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b") /* Unk
        {
            Local0 = Package (0x02)
                {
                    "reg-ltrovr",
                    Buffer (0x08)
                    {
                        0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // ........
                    }
                }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }

        Return (0x00)
    }
}

Device (RP03)
{
```

```
Name (_ADR, 0x001C0002)  // _ADR: Address
Method (_PRW, 0, NotSerialized)  // _PRW: Power Resources for Wake
{
    If (OSDW ())
    {
        Return (Package (0x02)
        {
            0x69,
            0x03
        })
    }
    Else
    {
        Return (Package (0x02)
        {
            0x69,
            0x04
        })
    }
}

Method (_PRT, 0, NotSerialized)  // _PRT: PCI Routing Table
{
    If (PICM)
    {
        Return (AR06 ())
    }

    Return (PR06 ())
}

Method (_DSM, 4, NotSerialized)  // _DSM: Device-Specific Method
{
    If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b") /* Unk
    {
        Local0 = Package (0x02)
            {
                "reg-ltrovr",
                Buffer (0x08)
                {
                    0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // ........
                }
            }
        DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
        Return (Local0)
    }
```

```
        Return (0x00)
    }
}

Device (RP05)
{
    Name (_ADR, 0x001C0004)  // _ADR: Address
    Method (_PRW, 0, NotSerialized)  // _PRW: Power Resources for Wake
    {
        If (OSDW ())
        {
            Return (Package (0x02)
            {
                0x69,
                0x03
            })
        }
        Else
        {
            Return (Package (0x02)
            {
                0x69,
                0x03
            })
        }
    }

    Method (_PRT, 0, NotSerialized)  // _PRT: PCI Routing Table
    {
        If (PICM)
        {
            Return (AR08 ())
        }

        Return (PR08 ())
    }

    Method (_DSM, 4, NotSerialized)  // _DSM: Device-Specific Method
    {
        If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b") /* Unk
        {
            Local0 = Package (0x02)
                {
                    "reg-ltrovr",
                    Buffer (0x08)
                    {
                        0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // ........
```

```
                }
            }
        DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
        Return (Local0)
    }

    Return (0x00)
  }
}

Device (RP06)
{
  Name (_ADR, 0x001C0005)  // _ADR: Address
  Method (_PRW, 0, NotSerialized)  // _PRW: Power Resources for Wake
  {
    If (OSDW ())
    {
      Return (Package (0x02)
      {
        0x69,
        0x03
      })
    }
    Else
    {
      Return (Package (0x02)
      {
        0x69,
        0x03
      })
    }
  }

  Method (_PRT, 0, NotSerialized)  // _PRT: PCI Routing Table
  {
    If (PICM)
    {
      Return (AR09 ())
    }

    Return (PR09 ())
  }

  Method (_DSM, 4, NotSerialized)  // _DSM: Device-Specific Method
  {
    If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b") /* Unk
    {
```

```
            Local0 = Package (0x02)
               {
                  "reg-ltrovr",
                  Buffer (0x08)
                  {
                      0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // ........
                  }
               }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
         }

         Return (0x00)
      }
   }

   Scope (\_SB.PCI0.RP03)
   {
      OperationRegion (A1E0, PCI_Config, 0x00, 0x0380)
      Field (A1E0, ByteAcc, NoLock, Preserve)
      {
         Offset (0x04),
         BMIE,   3,
         Offset (0x19),
         SECB,   8,
         SBBN,   8,
         Offset (0x1E),
             ,   13,
         MABT,   1,
         Offset (0x4A),
             ,   5,
         TPEN,   1,
         Offset (0x50),
             ,   4,
         LDIS,   1,
             ,   24,
         LACT,   1,
         Offset (0xA4),
         PSTA,   2,
         Offset (0xE2),
             ,   2,
         L23E,   1,
         L23D,   1,
         Offset (0x324),
             ,   3,
         LEDM,   1
      }
```

```
OperationRegion (A1E1, PCI_Config, 0x18, 0x04)
Field (A1E1, DWordAcc, NoLock, Preserve)
{
   BNIR,   32
}

Method (_BBN, 0, NotSerialized)  // _BBN: BIOS Bus Number
{
   If (((BMIE == 0x00) && (SECB == 0xFF)))
   {
      Return (SNBS) /* \_SB_.PCI0.RP03.SNBS */
   }
   Else
   {
      Return (SECB) /* \_SB_.PCI0.RP03.SECB */
   }
}

Method (_STA, 0, NotSerialized)  // _STA: Status
{
   Return (0x0F)
}

Name (BMIS, 0x00)
Name (SNBS, 0x00)
Name (BNIS, 0x00)
Method (APPD, 0, Serialized)
{
   If (!OSDW ())
   {
      Return (Zero)
   }

   If (((WOWE == 0x01) && (SLTP != 0x00)))
   {
      Return (Zero)
   }

   If (((TAPD == 0x00) && (SLTP != 0x00)))
   {
      Return (Zero)
   }

   \_SB.PCI0.RP03.ARPT.PSTA = 0x03
   If ((SLTP == 0x00))
   {
```

```
    L23E = 0x01
    Local0 = 0x00
    While (L23E)
    {
        If ((Local0 > 0x04))
        {
            Break
        }

        Sleep (0x01)
        Local0++
    }

    LEDM = 0x01
    If (((TAPD == 0x00) || (WOWE == 0x01)))
    {
        GP94 = 0x00
        GD94 = 0x00
        Local0 = 0x00
        While ((PMFS == 0x01))
        {
            If ((Local0 > 0x04))
            {
                Break
            }

            Stall (0x05)
            Local0++
        }

        Local1 = (CKEN & ~(0x04 << 0x10))
        Local1 = ((0x04 << 0x18) | Local1)
        Local1 = (Local1 | 0x04)
        CKEN = Local1
        Local0 = 0x00
        While ((PMFS == 0x01))
        {
            If ((Local0 > 0x04))
            {
                Break
            }

            Stall (0x05)
            Local0++
        }
    }
}
```

```
    If ((WOWE == 0x01))
    {
        GP70 = 0x00
        GD70 = 0x00
        GP70 = 0x01
    }

    If ((((BMIE != 0x00) && (BMIE != BMIS)) && (
        ((SECB != 0x00) && (SECB != SNBS)) && ((BNIR !=
        0x00) && (BNIR != BNIS)))))
    {
        BMIS = BMIE /* \_SB_.PCI0.RP03.BMIE */
        SNBS = SECB /* \_SB_.PCI0.RP03.SECB */
        BNIS = BNIR /* \_SB_.PCI0.RP03.BNIR */
    }

    BMIE = 0x00
    BNIR = 0x00FEFF00
    Local0 = TPEN /* \_SB_.PCI0.RP03.TPEN */
    PSTA = 0x03
    Local0 = TPEN /* \_SB_.PCI0.RP03.TPEN */
    Local0 = (Timer + 0x00989680)
    While ((Timer <= Local0))
    {
        If ((LACT == 0x00))
        {
            Break
        }

        Sleep (0x0A)
    }

    If ((WOWE == 0x01))
    {
        Return (Zero)
    }

    If ((TAPD == 0x01))
    {
        \_SB.PCI0.LPCB.EC.APWC = 0x00
        Sleep (0x64)
    }

    Return (Zero)
}
```

```
Method (APPU, 0, Serialized)
{
    If (!OSDW ())
    {
        WOWE = 0x00
        Return (Zero)
    }

    If (((WOWE == 0x01) && (SLTP != 0x00)))
    {
        WOWE = 0x00
        Return (Zero)
    }

    If (((TAPD == 0x00) && (SLTP != 0x00)))
    {
        WOWE = 0x00
        Return (Zero)
    }

    PSTA = 0x00
    If ((SECB != 0xFF))
    {
        Return (Zero)
    }

    BNIR = BNIS /* \_SB_.PCI0.RP03.BNIS */
    If ((SLTP == 0x00))
    {
        If (((TAPD == 0x00) || (WOWE == 0x01)))
        {
            Local0 = 0x00
            While ((PMFS == 0x01))
            {
                If ((Local0 > 0x04))
                {
                    Break
                }

                Stall (0x05)
                Local0++
            }

            Local1 = ((0x04 << 0x18) | (0x04 << 0x10))
            Local1 = (CKEN | Local1)
            Local1 |= 0x04
            CKEN = Local1
```

```
    Local0 = 0x00
    While ((PMFS == 0x01))
    {
        If ((Local0 > 0x04))
        {
            Break
        }

        Stall (0x05)
        Local0++
    }

    GD94 = 0x00
    GP94 = 0x00
    Sleep (0x32)
    GP94 = 0x01
    GD94 = 0x01
    Sleep (0x64)
}

L23D = 0x01
Local0 = 0x00
While (L23D)
{
    If ((Local0 > 0x04))
    {
        Break
    }

    Sleep (0x01)
    Local0++
}

LEDM = 0x00
If (((TAPD == 0x00) || (WOWE == 0x01)))
{
    Local2 = (Timer + 0x00989680)
    While ((Timer <= Local2))
    {
        If (((LACT == 0x01) && (\_SB.PCI0.RP03.ARPT.AVND != 0xFFF
        {
            Break
        }

        Sleep (0x0A)
    }
```

```
        WOWE = 0x00
        Return (Zero)
    }
}

WOWE = 0x00
If ((\_SB.PCI0.LPCB.EC.APWC == 0x01))
{
    Local2 = (Timer + 0x00989680)
    While ((Timer <= Local2))
    {
        If ((LACT == 0x01))
        {
            Break
        }

        Sleep (0x0A)
    }

    Return (Zero)
}

Local0 = 0x00
While (0x01)
{
    \_SB.PCI0.LPCB.EC.APWC = 0x01
    Sleep (0xFA)
    Local1 = 0x00
    Local2 = (Timer + 0x00989680)
    While ((Timer <= Local2))
    {
        If (((LACT == 0x01) && (\_SB.PCI0.RP03.ARPT.AVND != 0xFFFF)))
        {
            Local1 = 0x01
            Break
        }

        Sleep (0x0A)
    }

    If ((Local1 == 0x01))
    {
        MABT = 0x01
        Break
    }

    If ((Local0 == 0x04))
```

```
            {
                Break
            }

            Local0++
            \_SB.PCI0.LPCB.EC.APWC = 0x00
            Sleep (0x64)
        }

        Return (Zero)
    }

    Method (ALPR, 1, NotSerialized)
    {
        If ((Arg0 == 0x01))
        {
            APPD ()
        }
        Else
        {
            APPU ()
        }
    }

    Method (_PS0, 0, Serialized)  // _PS0: Power State 0
    {
        If (OSDW ())
        {
            ALPR (0x00)
        }
    }

    Method (_PS3, 0, Serialized)  // _PS3: Power State 3
    {
        If (OSDW ())
        {
            ALPR (0x01)
        }
    }

    Device (ARPT)
    {
        Name (_ADR, 0x00)  // _ADR: Address
        Name (_GPE, 0x5B)  // _GPE: General Purpose Events
        OperationRegion (ARE2, PCI_Config, 0x00, 0x80)
        Field (ARE2, ByteAcc, NoLock, Preserve)
        {
```

```
    AVND,   16,
    ADID,   16,
    Offset (0x4C),
    PSTA,   2
}

Method (_STA, 0, NotSerialized)  // _STA: Status
{
    Return (0x0F)
}

Method (_PRW, 0, NotSerialized)  // _PRW: Power Resources for Wake
{
    If (OSDW ())
    {
        Return (Package (0x02)
        {
            0x69,
            0x04
        })
    }
    Else
    {
        Return (Package (0x02)
        {
            0x69,
            0x04
        })
    }
}

Method (PRW0, 0, NotSerialized)
{
    Return (Package (0x01)
    {
        0x5B
    })
}

Method (_RMV, 0, NotSerialized)  // _RMV: Removal Status
{
    Return (0x00)
}

Method (WWEN, 1, NotSerialized)
{
    WOWE = Arg0
```

```
        }

        Method (PDEN, 1, NotSerialized)
        {
            TAPD = Arg0
        }
    }
}

Scope (\_SB.PCI0.RP02)
{
    OperationRegion (A1E0, PCI_Config, 0x00, 0x0380)
    Field (A1E0, ByteAcc, NoLock, Preserve)
    {
        Offset (0x04),
        BMIE,   3,
        Offset (0x19),
        SECB,   8,
        SBBN,   8,
        Offset (0x1E),
            ,   13,
        MABT,   1,
        Offset (0x4A),
            ,   5,
        TPEN,   1,
        Offset (0x50),
        ASPM,   2,
            ,   2,
        LDIS,   1,
        Offset (0x52),
            ,   13,
        LACT,   1,
        Offset (0xA4),
        PSTA,   2,
        Offset (0xE2),
            ,   2,
        L23E,   1,
        L23D,   1,
        Offset (0x324),
            ,   3,
        LEDM,   1
    }

    Device (CMRA)
    {
        Name (_ADR, 0x00)  // _ADR: Address
        OperationRegion (ARE3, PCI_Config, 0x00, 0xFF)
```

```
Field (ARE3, ByteAcc, NoLock, Preserve)
{
   AVND,   16,
   ADID,   16,
   Offset (0x4C),
   DPST,   2
}

Name (S2PM, 0x02)
Method (CMPE, 1, Serialized)
{
   If ((Arg0 <= 0x01))
   {
      If ((Arg0 == 0x01))
      {
         GD45 = 0x01
         Sleep (0x64)
         \_SB.PCI0.RP02.ASPM = S2PM /* \_SB_.PCI0.RP02.CMRA.S2
         \_SB.PCI0.RP02.PSTA = 0x00
         While ((\_SB.PCI0.RP02.PSTA != 0x00))
         {
            Sleep (0x01)
         }

         Local0 = (Timer + 0x00989680)
         While ((Timer <= Local0))
         {
            If ((LACT == 0x01))
            {
               Break
            }

            Sleep (0x0A)
         }

         L23D = 0x01
         Sleep (0x01)
         Local0 = 0x00
         While (L23D)
         {
            If ((Local0 > 0x04))
            {
               Break
            }

            Sleep (0x01)
            Local0++
```

```
            }

            LEDM = 0x00
        }
        Else
        {
            \_SB.PCI0.RP02.CMRA.DPST = 0x03
            While ((\_SB.PCI0.RP02.CMRA.DPST != 0x03))
            {
                Sleep (0x01)
            }

            If ((SLTP == 0x00))
            {
                L23E = 0x01
                Sleep (0x01)
                Local0 = 0x00
                While (L23E)
                {
                    If ((Local0 > 0x04))
                    {
                        Break
                    }

                    Sleep (0x01)
                    Local0++
                }

                LEDM = 0x01
            }

            \_SB.PCI0.RP02.PSTA = 0x03
            While ((\_SB.PCI0.RP02.PSTA != 0x03))
            {
                Sleep (0x01)
            }

            S2PM = \_SB.PCI0.RP02.ASPM
            \_SB.PCI0.RP02.ASPM = 0x00
            GP45 = 0x00
            GD45 = 0x00
            Sleep (0x0A)
        }
    }

    Return (Zero)
}
```

```
    }

    Method (_BBN, 0, NotSerialized)  // _BBN: BIOS Bus Number
    {
        Return (SECB) /* \_SB_.PCI0.RP02.SECB */
    }

    Method (_STA, 0, NotSerialized)  // _STA: Status
    {
        Return (0x0F)
    }
}

Scope (\_SB.PCI0.RP06)
{
    OperationRegion (A1E0, PCI_Config, 0x00, 0x40)
    Field (A1E0, ByteAcc, NoLock, Preserve)
    {
        Offset (0x04),
        BMIE,   3,
        Offset (0x19),
        SECB,   8,
        SBBN,   8,
        Offset (0x1E),
            ,   13,
        MABT,   1
    }

    OperationRegion (A1E1, PCI_Config, 0x00, 0x0380)
    Field (A1E1, ByteAcc, NoLock, Preserve)
    {
        Offset (0x4A),
            ,   5,
        TPEN,   1,
        Offset (0x50),
        ASPM,   2,
            ,   2,
        LDIS,   1,
        LRTN,   1,
        Offset (0x52),
        LSPD,   4,
            ,   7,
        LTRN,   1,
            ,   1,
        LACT,   1,
        Offset (0x64),
            ,   11,
```

```
    LTRS,   1,
    Offset (0x68),
        ,   10,
    LTRE,   1,
    Offset (0xA4),
    PSTA,   2,
    Offset (0xE2),
        ,   2,
    L23E,   1,
    L23D,   1,
    Offset (0x324),
        ,   3,
    LEDM,   1
}

Device (SSD0)
{
    Name (_ADR, 0x00)  // _ADR: Address
    Method (_RMV, 0, NotSerialized)  // _RMV: Removal Status
    {
        If ((DBGD == 0x01))
        {
            Return (0x01)
        }
        Else
        {
            Return (0x00)
        }
    }

    OperationRegion (SSE1, PCI_Config, 0x00, 0x10)
    Field (SSE1, ByteAcc, NoLock, Preserve)
    {
        Offset (0x0B),
        CLAS,   8
    }

    Method (_DSM, 4, NotSerialized)  // _DSM: Device-Specific Method
    {
        If ((NVME == 0x01))
        {
            Local0 = Package (0x06)
                {
                    "deep-idle",
                    0x01,
                    "use-msi",
                    0x01,
```

```
                "nvme-self-refresh",
                0x01
            }
        }
        Else
        {
            Local0 = Package (0x04)
            {
                "use-msi",
                0x01,
                "sata-express-power-off",
                0x01
            }
        }

        DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
        Return (Local0)
    }

    Method (_PS3, 0, Serialized)  // _PS3: Power State 3
    {
        If ((OSDW () && NVME))
        {
            \_SB.PCI0.RP06.L23E = 0x01
            Sleep (0x01)
            Local0 = 0x00
            While (\_SB.PCI0.RP06.L23E)
            {
                If ((Local0 > 0x04))
                {
                    Break
                }

                Sleep (0x01)
                Local0++
            }

            \_SB.PCI0.RP06.LEDM = 0x01
            If ((SLTP == 0x00))
            {
                \_SB.PCI0.RP06.PSTA = 0x03
                Local0 = 0x00
                While ((\_SB.PCI0.RP06.PSTA != 0x03))
                {
                    If ((Local0 > 0x1388))
                    {
                        Break
```

```
            }

            Sleep (0x01)
            Local0++
        }
    }

    GD38 = 0x00
    GP38 = 0x00
    Sleep (0x01)
    GD33 = 0x00
    GP33 = 0x00
    Sleep (0x0A)
  }

  Return (0x00)
}

Method (_PS0, 0, Serialized)  // _PS0: Power State 0
{
  If ((OSDW () && NVME))
  {
    GD33 = 0x01
    Sleep (0x0A)
    GD38 = 0x01
    Sleep (0x01)
    \_SB.PCI0.RP06.PSTA = 0x00
    Local0 = 0x00
    While ((\_SB.PCI0.RP06.PSTA != 0x00))
    {
      If ((Local0 > 0x1388))
      {
        Break
      }

      Sleep (0x01)
      Local0++
    }

    \_SB.PCI0.RP06.L23D = 0x01
    Sleep (0x01)
    Local0 = 0x00
    While ((Local0 <= 0x04))
    {
      Local2 = \_SB.PCI0.RP06.L23D
      If ((Local2 == 0x00))
      {
```

```
            Break
        }

        Sleep (0x01)
        Local0++
    }

    \_SB.PCI0.RP06.LEDM = 0x00
    Local0 = (Timer + 0x01C9C380)
    Local1 = 0x01
    While ((Timer <= Local0))
    {
        Local2 = \_SB.PCI0.RP06.LACT
        If ((Local2 == 0x01))
        {
            Local1 = 0x00
            Break
        }

        Sleep (0x01)
    }

    If ((Local1 != 0x00))
    {
        Return (Local1)
    }

    Local0 = (Timer + 0x01C9C380)
    Local1 = 0x02
    While ((Timer <= Local0))
    {
        Local2 = \_SB.PCI0.RP06.SSD0.CLAS
        If ((Local2 == 0x01))
        {
            Local1 = 0x00
            Break
        }

        Sleep (0x0A)
    }

    \_SB.PCI0.RP06.LTRS = 0x01
    \_SB.PCI0.RP06.LTRE = 0x01
  }

  Return (Local1)
}
```

```
}

Method (_PS0, 0, Serialized)  // _PS0: Power State 0
{
    If (((OSDW () && NVME) == 0x00))
    {
        If (OSDW ())
        {
            If ((SLTP == 0x00))
            {
                Local0 = 0x00
                While ((PMFS == 0x01))
                {
                    If ((Local0 > 0x04))
                    {
                        Break
                    }

                    Stall (0x05)
                    Local0++
                }

                Local1 = ((0x20 << 0x10) | (0x20 << 0x18))
                Local1 = (CKEN | Local1)
                Local1 |= 0x04
                CKEN = Local1
                Local0 = 0x00
                While ((PMFS == 0x01))
                {
                    If ((Local0 > 0x04))
                    {
                        Break
                    }

                    Stall (0x05)
                    Local0++
                }
            }
        }

        GD56 = 0x01
        PSTA = 0x00
        Local0 = 0x00
        While ((\_SB.PCI0.RP06.PSTA != 0x00))
        {
            If ((Local0 > 0x1388))
            {
```

```
                Break
            }

            Sleep (0x01)
            Local0++
        }

        Sleep (0x46)
        L23D = 0x01
        Sleep (0x01)
        Local0 = 0x00
        While (L23D)
        {
            If ((Local0 > 0x04))
            {
                Break
            }

            Sleep (0x01)
            Local0++
        }

        LEDM = 0x00
        Local0 = (Timer + 0x00989680)
        While ((Timer <= Local0))
        {
            If (((LACT == 0x01) && (\_SB.PCI0.RP06.SSD0.CLAS == 0x01)))
            {
                Break
            }

            Sleep (0x0A)
        }
    }
}

Method (_PS3, 0, Serialized)  // _PS3: Power State 3
{
    If (((OSDW () && NVME) == 0x00))
    {
        If ((SLTP == 0x00))
        {
            L23E = 0x01
            Sleep (0x01)
            Local0 = 0x00
            While (L23E)
            {
```

```
            If ((Local0 > 0x04))
            {
                Break
            }

            Sleep (0x01)
            Local0++
        }

        LEDM = 0x01
    }

    PSTA = 0x03
    Local0 = 0x00
    While ((\_SB.PCI0.RP06.PSTA != 0x03))
    {
        If ((Local0 > 0x1388))
        {
            Break
        }

        Sleep (0x01)
        Local0++
    }

    GP56 = 0x00
    GD56 = 0x00
    Sleep (0x32)
    If (OSDW ())
    {
        If ((SLTP == 0x00))
        {
            Local0 = 0x00
            While ((PMFS == 0x01))
            {
                If ((Local0 > 0x04))
                {
                    Break
                }

                Stall (0x05)
                Local0++
            }

            Local1 = (CKEN & ~(0x20 << 0x10))
            Local1 = ((0x20 << 0x18) | Local1)
            Local1 = (Local1 | 0x04)
```

```
                    CKEN = Local1
                    Local0 = 0x00
                    While ((PMFS == 0x01))
                    {
                        If ((Local0 > 0x04))
                        {
                            Break
                        }

                        Stall (0x05)
                        Local0++
                    }
                }
            }
        }
    }

    Device (SDMA)
    {
        Name (_ADR, 0x00150000)  // _ADR: Address
        Name (_UID, 0x01)  // _UID: Unique ID
        Name (RBUF, ResourceTemplate ()
        {
            Interrupt (ResourceConsumer, Level, ActiveLow, Shared, ,, )
            {
                0x00000015,
            }
        })
        Method (_CRS, 0, NotSerialized)  // _CRS: Current Resource Settings
        {
            Return (RBUF) /* \_SB_.PCI0.SDMA.RBUF */
        }

        Method (_STA, 0, NotSerialized)  // _STA: Status
        {
            Return (0x0F)
        }
    }

    Device (SPI1)
    {
        Name (_ADR, 0x00150004)  // _ADR: Address
        Name (_CID, "INT33C1" /* Intel Serial I/O SPI Host Controller */)  // _CID:
        Name (_DDN, "Intel(R) Low Power Subsystem SPI Host Controller - 9CE6
Device Name
        Name (_UID, 0x02)  // _UID: Unique ID
```

```
Name (CSST, 0x28)
Name (CSHT, 0x0A)
Name (RBUF, ResourceTemplate ()
{
    Interrupt (ResourceConsumer, Level, ActiveLow, Shared, ,, )
    {
        0x00000015,
    }
})
Method (_STA, 0, NotSerialized)  // _STA: Status
{
    Return (0x0F)
}

Method (_DSM, 4, NotSerialized)  // _DSM: Device-Specific Method
{
    If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b") /* Unk
    {
        Local0 = Package (0x0C)
            {
                "gspi-channel-number",
                Buffer (0x08)
                {
                    0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // ........
                },

                "gspi-sysclk-period",
                Buffer (0x08)
                {
                    0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // ........
                },

                "gspi-pin-cs",
                Buffer (0x08)
                {
                    0x57, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // W.......
                },

                "gspi-pin-clk",
                Buffer (0x08)
                {
                    0x58, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // X.......
                },

                "gspi-pin-mosi",
                Buffer (0x08)
                {
```

```
                    0x59, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // Y.......
                },

                "gspi-pin-miso",
                Buffer (0x08)
                {
                    0x5A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // Z.......
                }
            }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }

        Return (0x00)
    }

    Name (WBUF, Buffer (0x02)
    {
         0x79, 0x00                                     // y.
    })
    Name (DBUF, Buffer (0x10)
    {
        /* 0000 */  0x55, 0x10, 0x00, 0x06, 0x00, 0x02, 0x55, 0x11,  // U.....U.
        /* 0008 */  0x00, 0x07, 0x00, 0x02, 0x79, 0x00               // ....y.
    })
    Method (_CRS, 0, NotSerialized)  // _CRS: Current Resource Settings
    {
        If (!OSDW ())
        {
            Return (WBUF) /* \_SB_.PCI0.SPI1.WBUF */
        }

        Return (ConcatenateResTemplate (RBUF, DBUF))
    }
}

Scope (\_SB.PCI0.SPI1)
{
    Device (SPIT)
    {
        Name (_HID, EisaId ("APP000D"))  // _HID: Hardware ID
        Name (_CID, "apple-spi-topcase")  // _CID: Compatible ID
        Name (_DDN, "apple-spi-topcase")  // _DDN: DOS Device Name
        Name (_GPE, 0x1C)  // _GPE: General Purpose Events
        Name (_UID, 0x02)  // _UID: Unique ID
        Name (_ADR, 0x00)  // _ADR: Address
        Method (_PRW, 0, NotSerialized)  // _PRW: Power Resources for Wake
```

```
{
    If (OSDW ())
    {
        Return (Package (0x02)
        {
            0x1C,
            0x03
        })
    }

    Return (Package (0x02)
    {
        0x1C,
        0x03
    })
}

Method (_STA, 0, NotSerialized)  // _STA: Status
{
    Return (0x0F)
}

Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
{
    Name (UBUF, ResourceTemplate ()
    {
        SpiSerialBusV2 (0x0000, PolarityLow, FourWireMode, 0x08,
            ControllerInitiated, 0x007A1200, ClockPolarityLow,
            ClockPhaseFirst, "\\_SB.PCI0.SPI1",
            0x00, ResourceConsumer, , Exclusive,
            )
        Interrupt (ResourceConsumer, Level, ActiveLow, Exclusive, ,, )
        {
            0x0000001E,
        }
    })
    Name (ABUF, Buffer (0x02)
    {
         0x79, 0x00                                 // y.
    })
    If (!OSDW ())
    {
        Return (UBUF) /* \_SB_.PCI0.SPI1.SPIT._CRS.UBUF */
    }

    Return (ABUF) /* \_SB_.PCI0.SPI1.SPIT._CRS.ABUF */
}
```

```
Scope (\_GPE)
{
    Method (_L1C, 0, NotSerialized)  // _Lxx: Level-Triggered GPE, xx=0x
    {
        Notify (\_SB.PCI0.SPI1.SPIT, 0x02) // Device Wake
    }
}

Method (_DSM, 4, NotSerialized)  // _DSM: Device-Specific Method
{
    If (OSDW ())
    {
        If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b") /
*/))
        {
            Local0 = Package (0x10)
                {
                    "spiSclkPeriod",
                    Buffer (0x08)
                    {
                        0x7D, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // }.....
                    },

                    "spiWordSize",
                    Buffer (0x08)
                    {
                        0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // .....
                    },

                    "spiBitOrder",
                    Buffer (0x08)
                    {
                        0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // .....
                    },

                    "spiSPO",
                    Buffer (0x08)
                    {
                        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // .....
                    },

                    "spiSPH",
                    Buffer (0x08)
                    {
                        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // .....
                    },
```

```
                "spiCSDelay",
                Buffer (0x08)
                {
                    0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // ......
                },

                "resetA2RUsec",
                Buffer (0x08)
                {
                    0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // ......
                },

                "resetRecUsec",
                Buffer (0x08)
                {
                    0x0A, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00   // ......
                }
            }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }
    }

    Return (0x00)
}

Method (UIEN, 1, Serialized)
{
    If ((Arg0 <= 0x01))
    {
        If ((Arg0 == 0x01))
        {
            SIEN (0x00)
            GD26 = 0x01
        }
        Else
        {
            GP26 = 0x00
            GD26 = 0x00
        }
    }
}

Method (UIST, 0, Serialized)
{
    Local0 = GD26 /* \GD26 */
```

```
            If ((Local0 == 0x01))
            {
                Return (GL26) /* \GL26 */
            }
            Else
            {
                Return (GP26) /* \GP26 */
            }
        }

        Method (SIEN, 1, Serialized)
        {
            If ((Arg0 <= 0x01))
            {
                If ((Arg0 == 0x01))
                {
                    UIEN (0x00)
                    GD13 = 0x01
                }
                Else
                {
                    GP13 = 0x00
                    GD13 = 0x00
                }
            }
        }

        Method (SIST, 0, Serialized)
        {
            Local0 = GD13 /* \GD13 */
            If ((Local0 == 0x01))
            {
                Return (GL13) /* \GL13 */
            }
            Else
            {
                Return (GP13) /* \GP13 */
            }
        }
    }

    Method (ISOL, 1, Serialized)
    {
        If ((Arg0 <= 0x01))
        {
            If ((Arg0 == 0x01))
            {
```

```
                GP87 = 0x01
                GP88 = 0x00
                GP89 = 0x00
                GP90 = 0x00
                GD87 = 0x00
                GD88 = 0x00
                GD89 = 0x01
                GD90 = 0x00
                GU87 = 0x01
                GU88 = 0x01
                GU89 = 0x01
                GU90 = 0x01
                Local0 = GP87 /* \GP87 */
            }
            Else
            {
                GU87 = 0x00
                GU88 = 0x00
                GU89 = 0x00
                GU90 = 0x00
                GP87 = 0x00
                GP88 = 0x00
                GP89 = 0x00
                GP90 = 0x00
                GD87 = 0x00
                GD88 = 0x00
                GD89 = 0x00
                GD90 = 0x00
                Local0 = GU87 /* \GU87 */
            }

            Return (0x00)
        }

        Return (0xFFFFFFFF)
    }
  }
}

Device (ADP1)
{
  Name (_HID, "ACPI0003" /* Power Source Device */)  // _HID: Hardware ID
  Method (_PRW, 0, NotSerialized)  // _PRW: Power Resources for Wake
  {
    If (OSDW ())
    {
      Return (Package (0x02)
```

```
        {
            0x70,
            0x04
        })
    }
    Else
    {
        Return (Package (0x02)
        {
            0x70,
            0x03
        })
    }
}

Name (WK00, 0x01)
Method (SWAK, 1, NotSerialized)
{
    WK00 = (Arg0 & 0x03)
    If (!WK00)
    {
        WK00 = 0x01
    }
}

Method (_PSR, 0, NotSerialized)  // _PSR: Power Source
{
    Return (PWRS) /* \PWRS */
}

Method (_PCL, 0, NotSerialized)  // _PCL: Power Consumer List
{
    Return (\_SB)
}

Method (_PSW, 1, NotSerialized)  // _PSW: Power State Wake
{
    If (OSDW ())
    {
        If (\_SB.PCI0.LPCB.EC.ECOK)
        {
            If (Arg0)
            {
                If ((WK00 & 0x01))
                {
                    \_SB.PCI0.LPCB.EC.EWAI = 0x01
                }
```

```
                If ((WK00 & 0x02))
                {
                    \_SB.PCI0.LPCB.EC.EWAR = 0x01
                }
            }
            Else
            {
                \_SB.PCI0.LPCB.EC.EWAI = 0x00
                \_SB.PCI0.LPCB.EC.EWAR = 0x00
            }
        }
    }
}

Device (LID0)
{
    Name (_HID, EisaId ("PNP0C0D") /* Lid Device */)  // _HID: Hardware ID
    Method (_PRW, 0, NotSerialized)  // _PRW: Power Resources for Wake
    {
        If (OSDW ())
        {
            Return (Package (0x02)
            {
                0x70,
                0x04
            })
        }
        Else
        {
            Return (Package (0x02)
            {
                0x70,
                0x03
            })
        }
    }

    Method (_LID, 0, NotSerialized)  // _LID: Lid Status
    {
        LIDS = \_SB.PCI0.LPCB.EC.ELSW
        \_SB.PCI0.IGPU.CLID = \_SB.PCI0.LPCB.EC.ELSW
        Return (LIDS) /* \LIDS */
    }

    Method (_PSW, 1, NotSerialized)  // _PSW: Power State Wake
```

```
      {
         If (\_SB.PCI0.LPCB.EC.ECOK)
         {
            If (Arg0)
            {
               \_SB.PCI0.LPCB.EC.EWLO = 0x01
            }
            Else
            {
               \_SB.PCI0.LPCB.EC.EWLO = 0x00
            }
         }
      }
   }

   Device (PWRB)
   {
      Name (_HID, EisaId ("PNP0C0C") /* Power Button Device */)  // _HID: Hardw
   }
}

Scope (\_PR)
{
   Processor (CPU0, 0x01, 0x00000410, 0x06){}
   Processor (CPU1, 0x02, 0x00000410, 0x06){}
   Processor (CPU2, 0x03, 0x00000410, 0x06){}
   Processor (CPU3, 0x04, 0x00000410, 0x06){}
   Processor (CPU4, 0x05, 0x00000410, 0x06){}
   Processor (CPU5, 0x06, 0x00000410, 0x06){}
   Processor (CPU6, 0x07, 0x00000410, 0x06){}
   Processor (CPU7, 0x08, 0x00000410, 0x06){}
}

Mutex (MUTX, 0x00)
Name (SLTP, 0x00)
Name (EICM, 0x00)
Name (S3S4, 0x00)
Name (R118, 0x00)
Name (R119, 0x00)
Name (R11A, 0x00)
Name (R11C, 0x00)
Name (R120, 0x00)
Name (R124, 0x00)
Name (R218, 0x00)
Name (R219, 0x00)
Name (R21A, 0x00)
Name (R21C, 0x00)
```

```
Name (R220, 0x00)
Name (R224, 0x00)
Name (R318, 0x00)
Name (R319, 0x00)
Name (R31A, 0x00)
Name (R31C, 0x00)
Name (R320, 0x00)
Name (R324, 0x00)
Name (R418, 0x00)
Name (R419, 0x00)
Name (R41A, 0x00)
Name (R41C, 0x00)
Name (R420, 0x00)
Name (R424, 0x00)
Name (R518, 0x00)
Name (R519, 0x00)
Name (R51A, 0x00)
Name (R51C, 0x00)
Name (R520, 0x00)
Name (R524, 0x00)
Name (R618, 0x00)
Name (R619, 0x00)
Name (R61A, 0x00)
Name (R61C, 0x00)
Name (R620, 0x00)
Name (R624, 0x00)
Name (RH10, 0x00)
Name (RH14, 0x00)
OperationRegion (PRT0, SystemIO, 0x80, 0x04)
Field (PRT0, DWordAcc, NoLock, Preserve)
{
   P80H,   32
}

OperationRegion (PLMT, SystemIO, 0x0310, 0x0A)
Field (PLMT, WordAcc, NoLock, Preserve)
{
   CPLT,   8,
   IGPS,   8,
   MPLT,   8,
   CFIL,   8,
   EGPS,   8
}

OperationRegion (T2PM, SystemMemory, T2PB, 0x08)
Field (T2PM, DWordAcc, NoLock, Preserve)
{
```

```
    T2PR,   32,
    P2TR,   32
}

OperationRegion (RSTR, SystemMemory, NHIB, 0x0100)
Field (RSTR, DWordAcc, NoLock, Preserve)
{
    CIOR,   32,
    Offset (0xF0),
    ICME,   32
}

OperationRegion (S0BA, SystemMemory, 0xE00A8084, 0x04)
Field (S0BA, DWordAcc, NoLock, Preserve)
{
    S0D3,   2
}

OperationRegion (S4BA, SystemMemory, 0xE00AC084, 0x04)
Field (S4BA, DWordAcc, NoLock, Preserve)
{
    S4D3,   2
}

OperationRegion (TCOI, SystemIO, 0x1860, 0x08)
Field (TCOI, WordAcc, NoLock, Preserve)
{
    Offset (0x04),
        ,   9,
    SCIS,   1,
    Offset (0x06)
}

Method (P8XH, 2, Serialized)
{
    If ((Arg0 == 0x00))
    {
        P80D = ((P80D & 0xFFFFFF00) | Arg1)
    }

    If ((Arg0 == 0x01))
    {
        P80D = ((P80D & 0xFFFF00FF) | (Arg1 << 0x08))
    }

    If ((Arg0 == 0x02))
    {
```

```
        P80D = ((P80D & 0xFF00FFFF) | (Arg1 << 0x10))
    }

    If ((Arg0 == 0x03))
    {
        P80D = ((P80D & 0x00FFFFFF) | (Arg1 << 0x18))
    }

    P80H = P80D /* \P80D */
}

OperationRegion (SPRT, SystemIO, 0xB2, 0x02)
Field (SPRT, ByteAcc, NoLock, Preserve)
{
    SSMP,   8,
    SSMY,   8
}

Method (\_PIC, 1, NotSerialized)  // _PIC: Interrupt Model
{
    GPIC = Arg0
    PICM = Arg0
}

Method (GETB, 3, Serialized)
{
    Local0 = (Arg0 * 0x08)
    Local1 = (Arg1 * 0x08)
    CreateField (Arg2, Local0, Local1, TBF3)
    Return (TBF3) /* \GETB.TBF3 */
}

Method (PNOT, 0, Serialized)
{
    If ((TCNT > 0x01))
    {
        If ((PDC0 & 0x08))
        {
            Notify (\_PR.CPU0, 0x80) // Performance Capability Change
            If ((PDC0 & 0x10))
            {
                Notify (\_PR.CPU0, 0x81) // C-State Change
            }
        }

        If ((PDC1 & 0x08))
        {
```

```
    Notify (\_PR.CPU1, 0x80) // Performance Capability Change
    If ((PDC1 & 0x10))
    {
        Notify (\_PR.CPU1, 0x81) // C-State Change
    }
}

If ((PDC2 & 0x08))
{
    Notify (\_PR.CPU2, 0x80) // Performance Capability Change
    If ((PDC2 & 0x10))
    {
        Notify (\_PR.CPU2, 0x81) // C-State Change
    }
}

If ((PDC3 & 0x08))
{
    Notify (\_PR.CPU3, 0x80) // Performance Capability Change
    If ((PDC3 & 0x10))
    {
        Notify (\_PR.CPU3, 0x81) // C-State Change
    }
}

If ((PDC4 & 0x08))
{
    Notify (\_PR.CPU4, 0x80) // Performance Capability Change
    If ((PDC4 & 0x10))
    {
        Notify (\_PR.CPU4, 0x81) // C-State Change
    }
}

If ((PDC5 & 0x08))
{
    Notify (\_PR.CPU5, 0x80) // Performance Capability Change
    If ((PDC5 & 0x10))
    {
        Notify (\_PR.CPU5, 0x81) // C-State Change
    }
}

If ((PDC6 & 0x08))
{
    Notify (\_PR.CPU6, 0x80) // Performance Capability Change
    If ((PDC6 & 0x10))
```

```
            {
                Notify (\_PR.CPU6, 0x81) // C-State Change
            }
        }

        If ((PDC7 & 0x08))
        {
            Notify (\_PR.CPU7, 0x80) // Performance Capability Change
            If ((PDC7 & 0x10))
            {
                Notify (\_PR.CPU7, 0x81) // C-State Change
            }
        }
    }
    Else
    {
        Notify (\_PR.CPU0, 0x80) // Performance Capability Change
        Notify (\_PR.CPU0, 0x81) // C-State Change
    }
}

Method (DSPI, 0, Serialized)
{
    If (!OSDW ())
    {
        S0D3 = 0x03
        Local0 = IOSR /* \IOSR */
        Local1 = (Local0 & 0x01)
        While ((Local1 != 0x00))
        {
            Local0 = IOSR /* \IOSR */
            Local1 = (Local0 & 0x01)
        }

        IOIR = 0xCE00AA07
        IOSR = 0x0600
        IONR = 0xF000
        Local0 = IOSR /* \IOSR */
        Local0 |= 0x01
        IOSR = Local0
        Local0 = IOSR /* \IOSR */
        Local1 = (Local0 & 0x01)
        While ((Local1 != 0x00))
        {
            Local0 = IOSR /* \IOSR */
            Local1 = (Local0 & 0x01)
        }
```

```
Local1 = (Local0 & 0x06)
If ((Local1 == 0x00))
{
    Local3 = IODR /* \IODR */
}

Local3 |= 0x0100
IOSR = 0x0700
IODR = Local3
IONR = 0xF000
Local0 = IOSR /* \IOSR */
Local0 |= 0x01
IOSR = Local0
Local0 = IOSR /* \IOSR */
Local1 = (Local0 & 0x01)
While ((Local1 != 0x00))
{
    Local0 = IOSR /* \IOSR */
    Local1 = (Local0 & 0x01)
}

Local1 = (Local0 & 0x06)
If ((Local1 == 0x00)){}
S4D3 = 0x03
Local0 = IOSR /* \IOSR */
Local1 = (Local0 & 0x01)
While ((Local1 != 0x00))
{
    Local0 = IOSR /* \IOSR */
    Local1 = (Local0 & 0x01)
}

IOIR = 0xCE00AB07
IOSR = 0x0600
IONR = 0xF000
Local0 = IOSR /* \IOSR */
Local0 |= 0x01
IOSR = Local0
Local0 = IOSR /* \IOSR */
Local1 = (Local0 & 0x01)
While ((Local1 != 0x00))
{
    Local0 = IOSR /* \IOSR */
    Local1 = (Local0 & 0x01)
}
```

```
        Local1 = (Local0 & 0x06)
        If ((Local1 == 0x00))
        {
            Local3 = IODR /* \IODR */
        }

        Local3 |= 0x0100
        IOSR = 0x0700
        IODR = Local3
        IONR = 0xF000
        Local0 = IOSR /* \IOSR */
        Local0 |= 0x01
        IOSR = Local0
        Local0 = IOSR /* \IOSR */
        Local1 = (Local0 & 0x01)
        While ((Local1 != 0x00))
        {
            Local0 = IOSR /* \IOSR */
            Local1 = (Local0 & 0x01)
        }

        Local1 = (Local0 & 0x06)
        If ((Local1 == 0x00)){}
    }
}

Method (TRAP, 2, Serialized)
{
    SMIF = Arg1
    If ((Arg0 == \TRTP))
    {
        TRP0 = 0x00
    }

    If ((Arg0 == \TRTD))
    {
        DTSF = Arg1
        TRPD = 0x00
        Return (DTSF) /* \DTSF */
    }

    If ((Arg0 == \TRTI))
    {
        TRPH = 0x00
    }

    Return (SMIF) /* \SMIF */
```

```
}

Scope (\_SB)
{
    Method (_INI, 0, NotSerialized)  // _INI: Initialize
    {
        PINI ()
        If (!OSDW ())
        {
            GN46 = 0x00
            GN28 = 0x01
            G46O = 0x01
            G46Q = 0x01
        }
        Else
        {
            GN46 = 0x01
        }
    }

    Method (LPS0, 0, NotSerialized)
    {
        Return (0x01)
    }

    Device (PNLF)
    {
        Name (_ADR, 0x00)  // _ADR: Address
        Name (_HID, EisaId ("APP0002"))  // _HID: Hardware ID
        Name (_CID, "backlight")  // _CID: Compatible ID
        Name (_UID, 0x0F)  // _UID: Unique ID
        Name (_STA, 0x0B)  // _STA: Status
    }

    Device (SLPB)
    {
        Name (_HID, EisaId ("PNP0C0E") /* Sleep Button Device */)  // _HID: Hardw
        Name (_STA, 0x0B)  // _STA: Status
    }
}

Scope (\_SB.PCI0)
{
    Method (_INI, 0, NotSerialized)  // _INI: Initialize
    {
        OSYS = 0x07DC
        If (CondRefOf (\_OSI, Local0))
```

```
{
    If (_OSI ("Darwin"))
    {
        OSYS = 0x2710
    }

    If (\_OSI ("Linux"))
    {
        OSYS = 0x03E8
    }

    If (\_OSI ("Windows 2009"))
    {
        OSYS = 0x07D9
    }

    If (\_OSI ("Windows 2012"))
    {
        OSYS = 0x07DC
    }
}

If (!OSDW ()){}
GI82 = 0x01
If (!OSDW ())
{
    R118 = UP18 /* External reference */
    R119 = UP19 /* External reference */
    R11A = UP1A /* External reference */
    R11C = UP1C /* External reference */
    R120 = UP20 /* External reference */
    R124 = UP24 /* External reference */
    R218 = DP18 /* External reference */
    R219 = DP19 /* External reference */
    R21A = DP1A /* External reference */
    R21C = DP1C /* External reference */
    R220 = DP20 /* External reference */
    R224 = DP24 /* External reference */
    R318 = D318 /* External reference */
    R319 = D319 /* External reference */
    R31A = D31A /* External reference */
    R31C = D31C /* External reference */
    R320 = D320 /* External reference */
    R324 = D324 /* External reference */
    R418 = D418 /* External reference */
    R419 = D419 /* External reference */
    R41A = D41A /* External reference */
```

```
            R41C = D41C /* External reference */
            R420 = D420 /* External reference */
            R424 = D424 /* External reference */
            R518 = D518 /* External reference */
            R519 = D519 /* External reference */
            R51A = D51A /* External reference */
            R51C = D51C /* External reference */
            R520 = D520 /* External reference */
            R524 = D524 /* External reference */
            R618 = D618 /* External reference */
            R619 = D619 /* External reference */
            R61A = D61A /* External reference */
            R61C = D61C /* External reference */
            R620 = D620 /* External reference */
            R624 = D624 /* External reference */
            RH10 = NH10 /* External reference */
            RH14 = NH14 /* External reference */
            If ((BICM == 0x01))
            {
                CIOR = 0x0400
                Sleep (0x64)
                \_SB.PCI0.CNHI ()
                ICME = 0x06
                CIOR = 0x0400
                Sleep (0x03E8)
            }

            GP47 = 0x00
            GD47 = 0x00
        }
    }

    Method (TBTC, 1, Serialized)
    {
        P2TR = Arg0
        If ((Arg0 == 0x05))
        {
            GP47 = 0x00
            GD47 = 0x00
        }

        Local0 = 0x0FFF
        Local1 = T2PR /* \T2PR */
        While (((Local2 = (Local1 & 0x01)) == 0x00))
        {
            Local0--
            If ((Local0 == 0x00))
```

```
            {
                Break
            }

            Local1 = T2PR /* \T2PR */
        }

        P2TR = 0x00
        Local0 = 0x0FFF
        Local1 = T2PR /* \T2PR */
        While (((Local2 = (Local1 & 0x01)) != 0x00))
        {
            Local0--
            If ((Local0 == 0x00))
            {
                Break
            }

            Local1 = T2PR /* \T2PR */
        }
    }

    Method (NHPG, 0, Serialized)
    {
    }

    Method (NPME, 0, Serialized)
    {
    }
}

Scope (\)
{
    Name (PICM, 0x00)
    Method (OSDW, 0, NotSerialized)
    {
        If ((OSYS == 0x2710))
        {
            Return (0x01)
        }
        Else
        {
            Return (0x00)
        }
    }

    Method (PINI, 0, NotSerialized)
```

```
    {
        OSYS = 0x07DC
        If (CondRefOf (_OSI, Local0))
        {
            If (_OSI ("Darwin"))
            {
                OSYS = 0x2710
            }
            ElseIf (_OSI ("Linux"))
            {
                OSYS = 0x03E8
            }
            ElseIf (_OSI ("Windows 2009"))
            {
                OSYS = 0x07D9
            }
            ElseIf (_OSI ("Windows 2012"))
            {
                OSYS = 0x07DC
            }
        }
        Else
        {
            OSYS = 0x07DC
        }
    }
}

Scope (\_SB.PCI0)
{
    Device (PDRC)
    {
        Name (_HID, EisaId ("PNP0C02") /* PNP Motherboard Resources */)  // _HID
        Name (_UID, 0x01)  // _UID: Unique ID
        Name (BUF0, ResourceTemplate ()
        {
            Memory32Fixed (ReadWrite,
                0x00000000,         // Address Base
                0x00004000,         // Address Length
                _Y17)
            Memory32Fixed (ReadWrite,
                0x00000000,         // Address Base
                0x00008000,         // Address Length
                _Y19)
            Memory32Fixed (ReadWrite,
                0x00000000,         // Address Base
                0x00001000,         // Address Length
```

```
        _Y1A)
    Memory32Fixed (ReadWrite,
        0x00000000,         // Address Base
        0x00001000,         // Address Length
        _Y1B)
    Memory32Fixed (ReadWrite,
        0x00000000,         // Address Base
        0x00000000,         // Address Length
        _Y1C)
    Memory32Fixed (ReadWrite,
        0xFED20000,         // Address Base
        0x00020000,         // Address Length
        )
    Memory32Fixed (ReadOnly,
        0xFED90000,         // Address Base
        0x00004000,         // Address Length
        )
    Memory32Fixed (ReadWrite,
        0xFED45000,         // Address Base
        0x0004B000,         // Address Length
        )
    Memory32Fixed (ReadOnly,
        0xFF000000,         // Address Base
        0x01000000,         // Address Length
        )
    Memory32Fixed (ReadOnly,
        0xFEE00000,         // Address Base
        0x00100000,         // Address Length
        )
    Memory32Fixed (ReadWrite,
        0x00000000,         // Address Base
        0x00001000,         // Address Length
        _Y18)
})
Method (_CRS, 0, Serialized)  // _CRS: Current Resource Settings
{
    CreateDWordField (BUF0, \_SB.PCI0.PDRC._Y17._BAS, RBR0)  // _BAS
    RBR0 = (\_SB.PCI0.LPCB.RCBA << 0x0E)
    CreateDWordField (BUF0, \_SB.PCI0.PDRC._Y18._BAS, TBR0)  // _BAS
    TBR0 = TBAB /* \TBAB */
    CreateDWordField (BUF0, \_SB.PCI0.PDRC._Y18._LEN, TBLN)  // _LEN:
    If ((TBAB == 0x00))
    {
        TBLN = 0x00
    }

    CreateDWordField (BUF0, \_SB.PCI0.PDRC._Y19._BAS, MBR0)  // _BAS
```

```
            MBR0 = (\_SB.PCI0.MHBR << 0x0F)
            CreateDWordField (BUF0, \_SB.PCI0.PDRC._Y1A._BAS, DBR0)  // _BAS
            DBR0 = (\_SB.PCI0.DIBR << 0x0C)
            CreateDWordField (BUF0, \_SB.PCI0.PDRC._Y1B._BAS, EBR0)  // _BAS
            EBR0 = (\_SB.PCI0.EPBR << 0x0C)
            CreateDWordField (BUF0, \_SB.PCI0.PDRC._Y1C._BAS, XBR0)  // _BAS
            XBR0 = (\_SB.PCI0.PXBR << 0x1A)
            CreateDWordField (BUF0, \_SB.PCI0.PDRC._Y1C._LEN, XSZ0)  // _LEN
            XSZ0 = (0x10000000 >> \_SB.PCI0.PXSZ)
            Return (BUF0) /* \_SB_.PCI0.PDRC.BUF0 */
        }
    }
}

Scope (\)
{
    Name (PCHS, 0xFFFFFFFF)
    OperationRegion (IO_T, SystemIO, 0x0800, 0x10)
    Field (IO_T, ByteAcc, NoLock, Preserve)
    {
        TRPI,   16,
        Offset (0x04),
        Offset (0x06),
        Offset (0x08),
        TRP0,   8,
        Offset (0x0A),
        Offset (0x0B),
        Offset (0x0C),
        Offset (0x0D),
        Offset (0x0E),
        Offset (0x0F),
        Offset (0x10)
    }

    OperationRegion (IO_D, SystemIO, 0x0810, 0x04)
    Field (IO_D, ByteAcc, NoLock, Preserve)
    {
        TRPD,   8
    }

    OperationRegion (IO_H, SystemIO, 0x1000, 0x04)
    Field (IO_H, ByteAcc, NoLock, Preserve)
    {
        TRPH,   8
    }

    OperationRegion (RCRB, SystemMemory, \SRCB, 0x4000)
```

```
    Field (RCRB, DWordAcc, NoLock, Preserve)
    {
        Offset (0x1000),
        Offset (0x2330),
        IOIR,   32,
        IODR,   32,
        IOSR,   16,
        IONR,   16,
        Offset (0x3000),
        Offset (0x331C),
        Offset (0x331F),
        PMFS,   1,
        Offset (0x3320),
        CKEN,   32,
        Offset (0x3404),
        HPAS,   2,
            ,   5,
        HPAE,   1,
        Offset (0x3418),
            ,   1,
            ,   1,
        SATD,   1,
        SMBD,   1,
        HDAD,   1,
            ,   2,
        UH6D,   1,
        UH1D,   1,
        UH2D,   1,
        UH3D,   1,
        UH4D,   1,
        UH5D,   1,
        Offset (0x341A),
        RP1D,   1,
        RP2D,   1,
        RP3D,   1,
        RP4D,   1,
        RP5D,   1,
        RP6D,   1,
        RP7D,   1,
        RP8D,   1,
            ,   4,
        UH7D,   1
    }
}

Scope (\_GPE)
{
```

```
Method (_L67, 0, NotSerialized)  // _Lxx: Level-Triggered GPE, xx=0x00-0xFF
{
    \_SB.PCI0.SBUS.HSTS = 0x20
}

Method (_L66, 0, NotSerialized)  // _Lxx: Level-Triggered GPE, xx=0x00-0xFF
{
    If ((\_SB.PCI0.IGPU.GSSE && !GSMI))
    {
        \_SB.PCI0.IGPU.GSCI ()
    }
    Else
    {
        \_SB.PCI0.IGPU.GEFC = 0x00
        SCIS = 0x01
        \_SB.PCI0.IGPU.GSSE = 0x00
        \_SB.PCI0.IGPU.SCIE = 0x00
    }
}

Method (_L69, 0, NotSerialized)  // _Lxx: Level-Triggered GPE, xx=0x00-0xFF
{
    Notify (\_SB.PCI0.PEG0, 0x02) // Device Wake
    Notify (\_SB.PCI0.RP01, 0x02) // Device Wake
    Notify (\_SB.PCI0.RP02, 0x02) // Device Wake
    Notify (\_SB.PCI0.RP03, 0x02) // Device Wake
    Notify (\_SB.PCI0.RP05, 0x02) // Device Wake
    Notify (\_SB.PCI0.RP06, 0x02) // Device Wake
    \_SB.PCI0.TGPE ()
    Notify (\_SB.PCI0.RP03.ARPT, 0x02) // Device Wake
}

Method (_L6D, 0, NotSerialized)  // _Lxx: Level-Triggered GPE, xx=0x00-0xFF
{
    Notify (\_SB.PWRB, 0x02) // Device Wake
    Notify (\_SB.PCI0.XHC1, 0x02) // Device Wake
    If (OSDW ())
    {
        Notify (\_SB.PCI0.HDEF, 0x02) // Device Wake
    }
}

Name (ICMM, 0x00)
Method (_L52, 0, NotSerialized)  // _Lxx: Level-Triggered GPE, xx=0x00-0xFF
{
    If (!OSDW ())
    {
```

```
        If ((OSYS >= 0x07DC))
        {
            Sleep (0x02)
            While ((ICMM == 0x01))
            {
                Sleep (0x01)
            }

            \_SB.PCI0.CMPE ()
        }
    }
    ElseIf ((GI82 == 0x01))
    {
        GI82 = 0x00
    }
    Else
    {
        GI82 = 0x01
    }
}

Method (_L4D, 0, NotSerialized)  // _Lxx: Level-Triggered GPE, xx=0x00-0xFF
{
    ICMM = 0x01
    If ((GI77 == 0x01))
    {
        GI77 = 0x00
    }
    Else
    {
        GI77 = 0x01
    }

    If (!OSDW ())
    {
        If ((OSYS >= 0x07DC))
        {
            Sleep (0x01)
            If ((S3S4 == 0x01))
            {
                GD47 = 0x01
                GP47 = 0x01
                S3S4 = 0x00
                Local0 = 0x0F
                While ((Local0 > 0x00))
                {
                    If ((\_SB.PCI0.UPCK () == 0x01))
```

```
                {
                    Break
                }

                \_SB.PCI0.TBTC (0x0D)
                Local0--
            }
        }

        If ((EICM == 0x01))
        {
            If ((BICM == 0x01))
            {
                If ((\_SB.PCI0.UPCK () == 0x00))
                {
                    GD47 = 0x01
                    GP47 = 0x01
                }

                If ((\_SB.PCI0.WTLT () == 0x01))
                {
                    \_SB.PCI0.CNHI ()
                    EICM = 0x00
                    ICME = 0x06
                    CIOR = 0x0400
                    GP47 = 0x00
                    GD47 = 0x00
                    Sleep (0x03E8)
                }
            }
        }

        If ((\_SB.PCI0.UPCK () == 0x00))
        {
            EICM = 0x01
            \_SB.PCI0.CMPE ()
        }
    }
}
Else
{
    \_SB.PCI0.AMPE ()
}

ICMM = 0x00
Return (Zero)
}
```

```
}

Method (DTGP, 5, NotSerialized)
{
    If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b") /* Unknown U
    {
        If ((Arg1 == One))
        {
            If ((Arg2 == Zero))
            {
                Arg4 = Buffer (0x01)
                    {
                        0x03                                    // .
                    }
                Return (One)
            }

            If ((Arg2 == One))
            {
                Return (One)
            }
        }
    }

    Arg4 = Buffer (0x01)
        {
            0x00                                    // .
        }
    Return (Zero)
}

Name (_S0, Package (0x03)  // _S0_: S0 System State
{
    0x00,
    0x00,
    0x00
})
Name (_S4, Package (0x03)  // _S4_: S4 System State
{
    0x06,
    0x06,
    0x00
})
Name (_S5, Package (0x03)  // _S5_: S5 System State
{
    0x07,
    0x07,
```

```
    0x00
})
Method (_TTS, 1, NotSerialized)  // _TTS: Transition To State
{
    SLTP = Arg0
}

Method (_PTS, 1, NotSerialized)  // _PTS: Prepare To Sleep
{
    P80D = 0x00
    P8XH (0x00, Arg0)
    GN28 = 0x00
    GN46 = 0x01
    Sleep (0x64)
    \_SB.PCI0.LPCB.EC.ECSS = Arg0
    If (!OSDW ())
    {
        If ((Arg0 == 0x03))
        {
            \_SB.PCI0.LPCB.EC.EWDK = 0x01
        }

        If ((Arg0 == 0x03))
        {
            Local0 = 0x00
            While (((\_SB.PCI0.RP02.CMRA.DPST != 0x03) && (Local0 < 0x1388)))
            {
                Sleep (0x01)
                Local0++
            }

            \_SB.PCI0.RP02.CMRA.CMPE (0x00)
        }
    }

    If ((!OSDW () && (Arg0 >= 0x04)))
    {
        \_SB.PCI0.LPCB.EC.EWLO = 0x00
    }
}

Method (_WAK, 1, NotSerialized)  // _WAK: Wake
{
    S3S4 = 0x01
    EICM = 0x01
    P8XH (0x00, 0x00)
    \_SB.PCI0.LPCB.EC.ECSS = 0x00
```

```
    If (OSDW ()){}
    ElseIf ((Arg0 == 0x03))
    {
        \_SB.PCI0.RP02.CMRA.CMPE (0x01)
    }

    LIDS = \_SB.PCI0.LPCB.EC.ELSW
    \_SB.PCI0.IGPU.CLID = \_SB.PCI0.LPCB.EC.ELSW
    If (!OSDW ())
    {
        GN46 = 0x00
        GN28 = 0x01
        G46O = 0x01
        G46Q = 0x01
    }
    Else
    {
        GN28 = 0x00
        GN46 = 0x01
    }

    PWRS = \_SB.PCI0.LPCB.EC.EACP
    If (!OSDW ())
    {
        Notify (\_SB.PWRB, 0x02) // Device Wake
        \_SB.PCI0.LPCB.EC.LWE0 = 0x00
        \_SB.PCI0.LPCB.EC.LWE1 = 0x00
        \_SB.PCI0.LPCB.EC.LWE2 = 0x00
        \_SB.PCI0.LPCB.EC.LWE3 = 0x00
        GI82 = 0x01
        Sleep (0x64)
    }

    PNOT ()
    Return (Package (0x02)
    {
        0x00,
        0x00
    })
}

Scope (\)
{
    OperationRegion (GPIO, SystemIO, \GPBS, 0x0400)
    Field (GPIO, ByteAcc, NoLock, Preserve)
    {
            ,   28,
```

```
G28O,   1,
    ,   17,
G46O,   1,
Offset (0x10),
    ,   6,
G46Q,   1,
Offset (0x18),
GB00,   8,
GB01,   8,
GB02,   8,
GB03,   8,
Offset (0x100),
GU00,   1,
    ,   1,
GD00,   1,
GI00,   1,
    ,   27,
GP00,   1,
Offset (0x108),
GU01,   1,
    ,   1,
GD01,   1,
GI01,   1,
    ,   27,
GP01,   1,
Offset (0x110),
GU02,   1,
    ,   1,
GD02,   1,
GI02,   1,
    ,   27,
GP02,   1,
Offset (0x118),
GU03,   1,
    ,   1,
GD03,   1,
GI03,   1,
    ,   27,
GP03,   1,
Offset (0x120),
GU04,   1,
    ,   1,
GD04,   1,
GI04,   1,
    ,   27,
GP04,   1,
Offset (0x128),
```

GU05, 1,
, 1,
GD05, 1,
GI05, 1,
, 27,
GP05, 1,
Offset (0x130),
GU06, 1,
, 1,
GD06, 1,
GI06, 1,
, 27,
GP06, 1,
Offset (0x138),
GU07, 1,
, 1,
GD07, 1,
GI07, 1,
, 27,
GP07, 1,
Offset (0x140),
GU08, 1,
, 1,
GD08, 1,
GI08, 1,
, 27,
GP08, 1,
Offset (0x148),
GU09, 1,
, 1,
GD09, 1,
GI09, 1,
, 27,
GP09, 1,
Offset (0x150),
GU10, 1,
, 1,
GD10, 1,
GI10, 1,
, 27,
GP10, 1,
Offset (0x158),
GU11, 1,
, 1,
GD11, 1,
GI11, 1,
, 27,

```
GP11,   1,
Offset (0x160),
GU12,   1,
    ,   1,
GD12,   1,
GI12,   1,
    ,   26,
GL12,   1,
GP12,   1,
Offset (0x168),
GU13,   1,
    ,   1,
GD13,   1,
GI13,   1,
    ,   26,
GL13,   1,
GP13,   1,
Offset (0x170),
GU14,   1,
    ,   1,
GD14,   1,
GI14,   1,
    ,   27,
GP14,   1,
Offset (0x178),
GU15,   1,
    ,   1,
GD15,   1,
GI15,   1,
    ,   27,
GP15,   1,
Offset (0x180),
GU16,   1,
    ,   1,
GD16,   1,
GI16,   1,
    ,   27,
GP16,   1,
Offset (0x188),
GU17,   1,
    ,   1,
GD17,   1,
GI17,   1,
    ,   26,
GL17,   1,
GP17,   1,
Offset (0x190),
```

GU18, 1,
, 1,
GD18, 1,
GI18, 1,
, 27,
GP18, 1,
Offset (0x198),
GU19, 1,
, 1,
GD19, 1,
GI19, 1,
, 27,
GP19, 1,
Offset (0x1A0),
GU20, 1,
, 1,
GD20, 1,
GI20, 1,
, 27,
GP20, 1,
Offset (0x1A8),
GU21, 1,
, 1,
GD21, 1,
GI21, 1,
, 27,
GP21, 1,
Offset (0x1B0),
GU22, 1,
, 1,
GD22, 1,
GI22, 1,
, 27,
GP22, 1,
Offset (0x1B8),
GU23, 1,
, 1,
GD23, 1,
GI23, 1,
, 27,
GP23, 1,
Offset (0x1C0),
GU24, 1,
, 1,
GD24, 1,
GI24, 1,
, 27,

```
GP24,   1,
Offset (0x1C8),
GU25,   1,
    ,   1,
GD25,   1,
GI25,   1,
    ,   27,
GP25,   1,
Offset (0x1D0),
GU26,   1,
    ,   1,
GD26,   1,
GI26,   1,
    ,   26,
GL26,   1,
GP26,   1,
Offset (0x1D8),
GU27,   1,
    ,   1,
GD27,   1,
GI27,   1,
    ,   27,
GP27,   1,
Offset (0x1E0),
GU28,   1,
    ,   1,
GD28,   1,
GI28,   1,
    ,   27,
GP28,   1,
    ,   2,
GN28,   1,
Offset (0x1E8),
GU29,   1,
    ,   1,
GD29,   1,
GI29,   1,
    ,   27,
GP29,   1,
Offset (0x1F0),
GU30,   1,
    ,   1,
GD30,   1,
GI30,   1,
    ,   27,
GP30,   1,
Offset (0x1F8),
```

GU31,   1,
    ,   1,
GD31,   1,
GI31,   1,
    ,   27,
GP31,   1,
Offset (0x200),
GU32,   1,
    ,   1,
GD32,   1,
GI32,   1,
    ,   27,
GP32,   1,
Offset (0x208),
GU33,   1,
    ,   1,
GD33,   1,
GI33,   1,
    ,   27,
GP33,   1,
Offset (0x210),
GU34,   1,
    ,   1,
GD34,   1,
GI34,   1,
    ,   27,
GP34,   1,
Offset (0x218),
GU35,   1,
    ,   1,
GD35,   1,
GI35,   1,
    ,   27,
GP35,   1,
Offset (0x220),
GU36,   1,
    ,   1,
GD36,   1,
GI36,   1,
    ,   27,
GP36,   1,
Offset (0x228),
GU37,   1,
    ,   1,
GD37,   1,
GI37,   1,
    ,   27,

GP37, 1,
Offset (0x230),
GU38, 1,
, 1,
GD38, 1,
GI38, 1,
, 27,
GP38, 1,
Offset (0x238),
GU39, 1,
, 1,
GD39, 1,
GI39, 1,
, 27,
GP39, 1,
Offset (0x240),
GU40, 1,
, 1,
GD40, 1,
GI40, 1,
, 27,
GP40, 1,
Offset (0x248),
GU41, 1,
, 1,
GD41, 1,
GI41, 1,
, 27,
GP41, 1,
Offset (0x250),
GU42, 1,
, 1,
GD42, 1,
GI42, 1,
, 27,
GP42, 1,
Offset (0x258),
GU43, 1,
, 1,
GD43, 1,
GI43, 1,
, 27,
GP43, 1,
Offset (0x260),
GU44, 1,
, 1,
GD44, 1,

GI44,   1,
    ,   27,
GP44,   1,
Offset (0x268),
GU45,   1,
    ,   1,
GD45,   1,
GI45,   1,
    ,   27,
GP45,   1,
Offset (0x270),
GU46,   1,
    ,   1,
GD46,   1,
GI46,   1,
    ,   27,
GP46,   1,
    ,   2,
GN46,   1,
Offset (0x278),
GU47,   1,
    ,   1,
GD47,   1,
GI47,   1,
    ,   27,
GP47,   1,
Offset (0x280),
GU48,   1,
    ,   1,
GD48,   1,
GI48,   1,
    ,   27,
GP48,   1,
Offset (0x288),
GU49,   1,
    ,   1,
GD49,   1,
GI49,   1,
    ,   27,
GP49,   1,
Offset (0x290),
GU50,   1,
    ,   1,
GD50,   1,
GI50,   1,
    ,   27,
GP50,   1,

Offset (0x298),
GU51, 1,
, 1,
GD51, 1,
GI51, 1,
, 26,
GL51, 1,
GP51, 1,
Offset (0x2A0),
GU52, 1,
, 1,
GD52, 1,
GI52, 1,
, 27,
GP52, 1,
Offset (0x2A8),
GU53, 1,
, 1,
GD53, 1,
GI53, 1,
, 27,
GP53, 1,
Offset (0x2B0),
GU54, 1,
, 1,
GD54, 1,
GI54, 1,
, 27,
GP54, 1,
Offset (0x2B8),
GU55, 1,
, 1,
GD55, 1,
GI55, 1,
, 27,
GP55, 1,
Offset (0x2C0),
GU56, 1,
, 1,
GD56, 1,
GI56, 1,
, 27,
GP56, 1,
Offset (0x2C8),
GU57, 1,
, 1,
GD57, 1,

```
GI57,   1,
    ,   27,
GP57,   1,
Offset (0x2D0),
GU58,   1,
    ,   1,
GD58,   1,
GI58,   1,
    ,   27,
GP58,   1,
Offset (0x2D8),
GU59,   1,
    ,   1,
GD59,   1,
GI59,   1,
    ,   27,
GP59,   1,
Offset (0x2E0),
GU60,   1,
    ,   1,
GD60,   1,
GI60,   1,
    ,   27,
GP60,   1,
Offset (0x2E8),
GU61,   1,
    ,   1,
GD61,   1,
GI61,   1,
    ,   27,
GP61,   1,
Offset (0x2F0),
GU62,   1,
    ,   1,
GD62,   1,
GI62,   1,
    ,   27,
GP62,   1,
Offset (0x2F8),
GU63,   1,
    ,   1,
GD63,   1,
GI63,   1,
    ,   27,
GP63,   1,
Offset (0x300),
GU64,   1,
```

,   1,
GD64,   1,
GI64,   1,
,   27,
GP64,   1,
Offset (0x308),
GU65,   1,
,   1,
GD65,   1,
GI65,   1,
,   27,
GP65,   1,
Offset (0x310),
GU66,   1,
,   1,
GD66,   1,
GI66,   1,
,   27,
GP66,   1,
Offset (0x318),
GU67,   1,
,   1,
GD67,   1,
GI67,   1,
,   27,
GP67,   1,
Offset (0x320),
GU68,   1,
,   1,
GD68,   1,
GI68,   1,
,   27,
GP68,   1,
Offset (0x328),
GU69,   1,
,   1,
GD69,   1,
GI69,   1,
,   27,
GP69,   1,
Offset (0x330),
GU70,   1,
,   1,
GD70,   1,
GI70,   1,
,   26,
GL70,   1,

```
GP70,   1,
Offset (0x338),
GU71,   1,
    ,   1,
GD71,   1,
GI71,   1,
    ,   27,
GP71,   1,
Offset (0x340),
GU72,   1,
    ,   1,
GD72,   1,
GI72,   1,
    ,   27,
GP72,   1,
Offset (0x348),
GU73,   1,
    ,   1,
GD73,   1,
GI73,   1,
    ,   27,
GP73,   1,
Offset (0x350),
GU74,   1,
    ,   1,
GD74,   1,
GI74,   1,
    ,   27,
GP74,   1,
Offset (0x358),
GU75,   1,
    ,   1,
GD75,   1,
GI75,   1,
    ,   27,
GP75,   1,
Offset (0x360),
GU76,   1,
    ,   1,
GD76,   1,
GI76,   1,
    ,   27,
GP76,   1,
Offset (0x368),
GU77,   1,
    ,   1,
GD77,   1,
```

```
GI77,   1,
    ,   26,
GL77,   1,
GP77,   1,
Offset (0x370),
GU78,   1,
    ,   1,
GD78,   1,
GI78,   1,
    ,   27,
GP78,   1,
Offset (0x378),
GU79,   1,
    ,   1,
GD79,   1,
GI79,   1,
    ,   27,
GP79,   1,
Offset (0x380),
GU80,   1,
    ,   1,
GD80,   1,
GI80,   1,
    ,   27,
GP80,   1,
Offset (0x388),
GU81,   1,
    ,   1,
GD81,   1,
GI81,   1,
    ,   27,
GP81,   1,
Offset (0x390),
GU82,   1,
    ,   1,
GD82,   1,
GI82,   1,
    ,   27,
GP82,   1,
Offset (0x398),
GU83,   1,
    ,   1,
GD83,   1,
GI83,   1,
    ,   27,
GP83,   1,
Offset (0x3A0),
```

GU84,   1,
    ,   1,
GD84,   1,
GI84,   1,
    ,   27,
GP84,   1,
Offset (0x3A8),
GU85,   1,
    ,   1,
GD85,   1,
GI85,   1,
    ,   27,
GP85,   1,
Offset (0x3B0),
GU86,   1,
    ,   1,
GD86,   1,
GI86,   1,
    ,   27,
GP86,   1,
Offset (0x3B8),
GU87,   1,
    ,   1,
GD87,   1,
GI87,   1,
    ,   27,
GP87,   1,
Offset (0x3C0),
GU88,   1,
    ,   1,
GD88,   1,
GI88,   1,
    ,   27,
GP88,   1,
Offset (0x3C8),
GU89,   1,
    ,   1,
GD89,   1,
GI89,   1,
    ,   27,
GP89,   1,
Offset (0x3D0),
GU90,   1,
    ,   1,
GD90,   1,
GI90,   1,
    ,   27,

```
        GP90,   1,
        Offset (0x3D8),
        GU91,   1,
           ,   1,
        GD91,   1,
        GI91,   1,
           ,   27,
        GP91,   1,
        Offset (0x3E0),
        GU92,   1,
           ,   1,
        GD92,   1,
        GI92,   1,
           ,   27,
        GP92,   1,
        Offset (0x3E8),
        GU93,   1,
           ,   1,
        GD93,   1,
        GI93,   1,
           ,   27,
        GP93,   1,
        Offset (0x3F0),
        GU94,   1,
           ,   1,
        GD94,   1,
        GI94,   1,
           ,   27,
        GP94,   1,
        Offset (0x3F8)
    }
}

Scope (\_SB.PCI0)
{
    Device (HDAU)
    {
        Name (_ADR, 0x00030000)  // _ADR: Address
        OperationRegion (HDAH, PCI_Config, 0x00, 0x40)
        Field (HDAH, ByteAcc, NoLock, Preserve)
        {
            VID0,   16,
            DID0,   16,
            Offset (0x10),
            ABAR,   32
        }
```

```
Method (_STA, 0, NotSerialized)  // _STA: Status
{
    If ((VID0 != 0xFFFF))
    {
        Return (0x0F)
    }

    Return (0x00)
}

Method (_DSM, 4, NotSerialized)  // _DSM: Device-Specific Method
{
    If ((Arg0 == ToUUID ("a0b5b7c6-1318-441c-b0c9-fe695eaf949b") /* Unknc
    {
        If (((VID0 & 0xFFFF) != 0xFFFF))
        {
            Local0 = Package (0x02)
                {
                    "hda-gfx",
                    Buffer (0x0A)
                    {
                        "onboard-1"
                    }
                }
            DTGP (Arg0, Arg1, Arg2, Arg3, RefOf (Local0))
            Return (Local0)
        }
    }

    Return (0x80000002)
}

Method (ASTR, 0, Serialized)
{
    If (((ABAR != 0xFFFFFFFF) && ((ABAR & 0xFFFFC000) !=
        0x00)))
    {
        BBAR = (ABAR & 0xFFFFFFF0)
        BBAR += 0x1000
        OperationRegion (RPCY, SystemMemory, BBAR, 0x25)
        Field (RPCY, DWordAcc, NoLock, Preserve)
        {
            Offset (0x0C),
            EM4W,   32,
            EMWA,   32,
            Offset (0x1C),
            ADWA,   32
```

```
        }

        EMWA = AUDA /* \AUDA */
        ADWA = AUDB /* \AUDB */
        EM4W = AUDC /* \AUDC */
    }
}

Method (VSTR, 1, Serialized)
{
    Name (CONT, 0x03E8)
    Name (ADDR, 0x80000000)
    ADDR = Arg0
    OperationRegion (CCDC, SystemMemory, ADDR, 0x04)
    Field (CCDC, ByteAcc, NoLock, Preserve)
    {
        CDEC,   32
    }

    If (((ABAR != 0xFFFFFFFF) && ((ABAR & 0xFFFFC000) !=
        0x00)))
    {
        If ((CDEC != 0x00))
        {
            BBAR = (ABAR & 0xFFFFFFF0)
            OperationRegion (IPCV, SystemMemory, BBAR, 0x70)
            Field (IPCV, DWordAcc, NoLock, Preserve)
            {
                Offset (0x60),
                AVIC,   32,
                Offset (0x68),
                AIRS,   16
            }

            CONT = 0x03E8
            While ((((AIRS & 0x01) == 0x01) && (CONT != 0x00)))
            {
                Stall (0x01)
                CONT--
            }

            AIRS |= 0x02
            AVIC = CDEC /* \_SB_.PCI0.HDAU.VSTR.CDEC */
            AIRS |= 0x01
            CONT = 0x03E8
            While ((((AIRS & 0x01) == 0x01) && (CONT != 0x00)))
            {
```

```
            Stall (0x01)
            CONT--
         }
      }
   }
}

Method (CXDC, 0, Serialized)
{
   Name (IDDX, 0x80000000)
   If (((CADR != 0x00) && (CCNT != 0x00)))
   {
      IDDX = CADR /* \CADR */
      While ((IDDX < (CADR + (CCNT * 0x04))))
      {
         VSTR (IDDX)
         IDDX += 0x04
      }
   }
}

Method (AINI, 0, Serialized)
{
   Name (CONT, 0x03E8)
   If (((ABAR != 0xFFFFFFFF) && ((ABAR & 0xFFFFC000) !=
      0x00)))
   {
      BBAR = (ABAR & 0xFFFFFFF0)
      OperationRegion (IPCV, SystemMemory, BBAR, 0x70)
      Field (IPCV, DWordAcc, NoLock, Preserve)
      {
         GCAP,   16,
         Offset (0x08),
         GCTL,   32,
         Offset (0x0E),
         SSTS,   8,
         Offset (0x60),
         AVIC,   32,
         Offset (0x68),
         AIRS,   16
      }

      GCTL |= 0x01
      CONT = 0x03E8
      While ((((GCTL & 0x01) == 0x01) && (CONT != 0x00)))
      {
         Stall (0x01)
```

```
                CONT--
            }

            GCAP &= 0xFFFF
            SSTS |= 0x0F
            GCTL &= 0xFFFFFFFE
            CONT = 0x03E8
            While ((((GCTL & 0x01) == 0x01) && (CONT != 0x00)))
            {
                Stall (0x01)
                CONT--
            }

            GCTL |= 0x01
            CONT = 0x03E8
            While ((((GCTL & 0x01) == 0x01) && (CONT != 0x00)))
            {
                Stall (0x01)
                CONT--
            }
        }
    }
  }
}
```