Implementing, Testing And Debugging ACPI On Windows Platforms

Hanumant Yadav
Software Design Engineer
Windows Kernel Platform Group

Session Outline

- ACPI table overloading
- ACPI to WMI Mapper
- ACPI Verifier
- Debugging ASL / ACPI
- Common ASL Errors and Solutions
- Some upcoming changes in the ACPI driver
- Q&A

ACPI Table Overloading

ACPI Table Overloading

- What is it?
  - ACPI table overloading allows overloading / replacing a ACPI table that is in ROM, by adding a modified table of the same name to a specific location in the registry
- Why would you want to do that?
  - This feature allows platform developers to quickly test changes to the BIOS code without flashing a new image to the ROM
- What do you need to overload tables?
  - ASL.exe (version 2.02 or greater) http://www.microsoft.com/whdc/hwdev/tech/onnow/default.mspx
  - Have a checked ACPI.sys loaded on the target system

ACPI Table Overloading

- How to overload existing ACPI BIOS tables
  - Build the new BIOS ASL code (e.g. foo.aml)
  - Use the 2.0 ASL compiler's table load option (version 2.02 or greater) to load the new .AML file into the registry
    - ASL.exe /acpiload foo.aml
  - The new BIOS loads when the system reboots
- How to get back to the original ACPI tables?
  - Going back to the ROM tables can be achieved in one of two ways:
  - Use the "Last known Good configuration" feature of Windows to go back to a working registry
  - Use the Delete Registry Table feature of ASL.exe
    - ASL.exe /acpiload /d foo.aml

ACPI To WMI Mapper

ACPI To WMI Mapping Driver (wmiacpi.sys)

- What is it?
  - Enables an interface between ACPI and Windows Management Instrumentation (WMI)
  - WMI objects and methods can be accessed through user-mode COM applications and scripts
- What does it do?
  - Allows access to ACPI methods and namespace objects from user-mode without requiring a special driver
- What are some if the things it can be used for?
  - Testing
  - Initiate specific method execution
  - Query ACPI named objects
  - Expose CMOS and BIOS configuration settings
- More information
  http://www.microsoft.com/whdc/hwdev/driver/WMI/wmi-acpi.mspx

ACPI Verifier

ACPI Verifier

What is it?

- ACPI Verifier (ACPIVer) is a real time ACPI namespace verifier
  - Determines if system BIOS is compliant with ACPI

Where is it used?

- The ACPI Verifier is available in all of the WHQL HCT System kits
  - Mobile
  - Desktop
  - Server
- All kits run the same ACPI Verifier tests

ACPI Verifier

Why should I care?

- Uncovers subtle hard to detect BIOS errors
- Helps in verifying system's ACPI specification compliance
  - Up to Revision 2
- Required for Designed for Windows Logo systems

ACPI Verifier

How do I use it?

- Download the System HCT kit you are interested in getting a logo in
  - http://www.microsoft.com/whdc/hwdev
- Install and launch the kit
  - The kit displays two panes in its window
  - Left pane displays a list of tests
  - Choose ACPI Verifier from this list
  - Run test
  - This will cause ACPI Verifier to install and reboot the system
  - After it has rebooted ACPI Stress is run to cycle the system in and out of sleep states
  - Once it has finished a log is produced
  - The log will contain any failures that ACPI Verifier found
- To determine what objects are causing these problems you need to hook up a kernel debugger to the system and re-run the test. More verbose log information will be provided in the debugger and this will allow you to begin debugging any failures

ACPI Verifier

How does it work?

- The ACPI Verifier is a passive real time namespace verifier
  - It monitors the AML Interpreter
  - Performs verifications

Debugging ASL / ACPI

Debugging ASL / ACPI

!amli help

kd> !amli ?

- Help - ? [<Cmd>]
- Clear Breakpoints - bc <bp list> | *
- Disable Breakpoints - bd <bp list> | *
- Enable Breakpoints - be <bp list> | *
- List Breakpoints - bl
- Set Breakpoints - bp <MethodName> | <CodeAddr> ...
- Request entering debugger - debugger
- Dump Name Space Object - dns [[/s] [<NameStr> | <Addr>]]
- Find Namespace Object - find <NameSeg>
- List All Contexts - lc
- Display Nearest Method - ln [<MethodName> | <CodeAddr>]
- Step Over AML Code - p
- Display Context Info. - r
- Set Debugger Options - set [traceon | traceoff] [nesttraceon | nesttraceoff] [spewon | spewoff] [lbrkon | lbrkoff] [errbrkon | errbrkoff] [verboseon| verboseoff] [logon | logoff] [logmuton | logmutoff]
- Unassemble AML code - u [<MethodName> | <CodeAddr>]

Debugging ASL / ACPI

- For more information on AML debugger commands look in the debugger help file or type !amli ? <command>

Example:

kd> !amli ? Set

Set Debugger Options:

Usage: set [traceon | traceoff] [nesttraceon | nesttraceoff]
[spewon | spewoff] [lbrkon | lbrkoff] [errbrkon | errbrkoff]
[verboseon | verboseoff] [logon | logoff] [logmuton | logmutoff]

Debugging ASL / ACPI

Two Ways to Debug AML

1) Using the extension
   - Break in to kd and use:

     !amli <command>

2) Debugging at the AMLI prompt

     (AMLI(? for help)->)

   - All the same commands from the extension can be used at this prompt. Use the commands without prefixing them with !amli
   - There is an additional set of commands available that can only be used at this prompt. Type “?” at this prompt to get the full list

Debugging ASL / ACPI

Why Are There Two Ways To Do The Same Thing?

- The extension is a good tool to get general information such as dumping ACPI name space, un-assembling methods, setting breakpoints, etc.
- Some things can only be done from within the context of the interpreter. Example: Stepping through AML code as it executes. To force the interpreter to break in at the "AMLI(? for help)->" prompt, break into kd and use "!amli debugger", or set a breakpoint on a particular AML method you are interested in debugging

Debugging ASL / ACPI

Some useful commands and what they do

- Debugger
  - Causes the interpreter to break in at a “AMLI(? for help)->” prompt when ever the interpreter is activated to evaluate anything. Use this command at the regular kd prompt and then hit go (g). The Interpreter will break in then next time it is activated
- Set (spewon, verboseon, traceon, errbrkon)
  - Spewon: Enables debug output from the interpreter
  - Verboseon: Lists the names of methods as they are evaluated
  - Traceon: Similar to “verboseon” but much more verbose. This is very helpful in tracking SMI related hard hangs
  - errbrkon: Will cause the interpreter to break in at the “AMLI(? for help)->” prompt when it encounters any errors while evaluating AML

Debugging ASL / ACPI

- Find
  - searches for methods, field units etc in the acpi name space and lists them with their full path

```
AMLI(? for help)-> find _srs
\_SB.LNKA._SRS
\_SB.LNKB._SRS
\_SB.LNKC._SRS
\_SB.LNKD._SRS
```

- Dns
  - The Dump Name Space command is useful in determining what a particular name space object is (method, fieldunit, device etc). This command can be used to dump the entire name space, a sub tree or a particular object

```
AMLI(? for help)-> dns \bios

ACPI Name Space: \BIOS (80E5F378)
OpRegion(BIOS:RegionSpace=SystemMemory,
Offset=0xfcb07500,Len=2816)
```

Debugging ASL / ACPI

- u

  Unassembles a given full path (e.g.: _SB.LNKB._SRS) or a given address

  Example:

```
kd> !amli u \_SB.LNKA._CRS
```

  Or

```
kd> !amli u ffffffff80e5d701

ffffffff80e5d701 : CreateWordField(CRES, 0x1, IRQW)
ffffffff80e5d70c : And(\_SB_.PCI0.LPC_.PIRA, 0xf, Local0)
ffffffff80e5d723 : Store(One, Local1)
ffffffff80e5d726 : ShiftLeft(Local1, Local0, IRQW)
ffffffff80e5d72d : Return(CRES)
```

Debugging ASL / ACPI

- r

  Dumps the current context of the interpreter. This is a very useful command to use when the interpreter breaks in at the **"AMLI(? for help)->".** Example:

```
kd> !amli r
MethodObject=\_WAK
ffffffff80e0ff5c: Local0=Unknown()
ffffffff80e0ff70: Local1=Unknown()
ffffffff80e0ff84: Local2=Unknown()
ffffffff80e0ff98: Local3=Unknown()
ffffffff80e0ffac: Local4=Unknown()
ffffffff80e0ffc0: Local5=Unknown()
ffffffff80e0ffd4: Local6=Unknown()
ffffffff80e0ffe8: Local7=Unknown()
ffffffff80e0e040: RetObj=Unknown()

Next AML Pointer: ffffffff80e630df:[\_WAK+16]

ffffffff80e630df : If(S4BW
ffffffff80e630e5 : {
ffffffff80e630e5 : | Store(Zero, S4BW)
ffffffff80e630eb : }
```

Debugging ASL / ACPI

Breakpoint commands (bp, bc, be, bd, bl)

- Used to set clear, enable and disable breakpoints within AML methods

```
kd> !amli bl
 0: <e> ffffffff80e5e2f1:[\_SB.LNKD._SRS]
 1: <e> ffffffff80e5d969:[\_SB.LNKB._STA]
 2: <d> ffffffff80e630c9:[\_WAK]
 3: <e> ffffffff80e612c9:[\_SB.MBRD._CRS]
```

Debugging ASL / ACPI

LC - List current contexts:

```
AMLI(? for help)-> lc
 Ctxt=80e3f000, ThID=00000000, Flgs=A--C-----, pbOp=00000000,
     Obj=\_SB.LNKA._STA
 Ctxt=80e41000, ThID=00000000, Flgs=A--C-----, pbOp=00000000,
     Obj=\_SB.LNKB._STA
 Ctxt=80e9a000, ThID=00000000, Flgs=A--C-----, pbOp=00000000,
     Obj=\_SB.LNKC._STA
 Ctxt=80ea8000, ThID=00000000, Flgs=A--C-----, pbOp=00000000,
     Obj=\_SB.LNKD._STA
*Ctxt=80e12000, ThID=80e6eda8, Flgs=---CR----, pbOp=80e5d5ac,
     Obj=\_SB.LNKA._STA
```

Flags:

A – Asynchronous evaluation
Q – In the ready queue
R – Running
T – Timeout
P – Timer pending
N – Nested Evaluation
C – Needs a callback
W - Ready
D – Timer dispatch

Debugging ASL / ACPI

Other useful extensions

!acpiinf – ACPI information

```
ACPIInformation (823eaf40)
  RSDT                      - f8b0c548
  FADT                      - f8b0eb2e
  FACS                      - f8b0ffc0
  DSDT                      - f89c2574
  GlobalLock                - f8b0ffd0
  GlobalLockQueue           - F - 823eaf58 B -
       823eaf58
  GlobalLockQueueLock       - 00000000
  GlobalLockOwnerContext    - 00000000
  GlobalLockOwnerDepth      - 00000000
  ACPIOnly                  - FALSE
  PM1a_BLK                  - 00008000 (0400) (0321)
     400 - RTC_STS
     1 - TMR_EN  20 - GBL_EN  100 - PWRBTN_EN  200
       - SLPBTN_EN
  PM1b_BLK                  - 00000000 (N/A)
  PM1a_CTRL_BLK             - 00008004 (0005)
     1 - SCI_EN  4 - GBL_RLS
  PM1b_CTRL_BLK             - 00000000 (N/A)
  PM2_CTRL_BLK              - 00000000 (N/A)
  PM_TMR                    - 00008008 (0031e15b)
  GP0_BLK                   - 00008020 (00) (00)
  GP0_ENABLE                - 00008022 (00) (00)
  GP0_LEN                   - 00000004
  GP0_SIZE                  - 00000002
  GP1_BLK                   - 00000000 (N/A)
  GP1_ENABLE                - 00000000 (N/A)
  GP1_LEN                   - 0
  GP1_SIZE                  - 0
  GP1_BASE_INDEX            - ffff
  GPE_SIZE                  - 2
  PM1_EN_BITS               - 0321
     1 - TMR_EN  20 - GBL_EN  100 - PWRBTN_EN  200 -
       SLPBTN_EN
  PM1_WAKE_MASK             - 0000
  C2_LATENCY                - 0
  C3_LATENCY                - 0
  ACPI_FLAGS                - 0
  ACPI_CAPABILITIES         - 0
```

Debugging ASL / ACPI

- Other useful extensions
  - !FADT – Dumps the FADT
  - !RSDT – Dumps the RSDT
  - !FACS – Dumps the FACS
  - !MAPIC – Dumps the APIC

Common ASL Errors And Solutions

Common ASL Errors And Solutions

PCI IRQ Routing (PIC vs. APIC mode)

- If the _PIC control method is present, the _PRT method should return interrupt routing information based on the value _PIC was called with

```
Method(_PIC,1)
{
    Store(Arg0, PICS)
}
Method(_PRT)
{
    if (PICS)
    {
        …
    }
    else
    {
        …
    }
}
_PIC(0)  => PIC Mode
_PIC(1)  => APIC Mode
```

Common ASL Errors And Solutions

FADT Version

- Revision 2 FADT was introduced for Legacy Free systems. It was also modified to support ACPI 2.0 Processor Performance States and C-State support
- For complete field definition of Revision 2 FADT table, see Microsoft's Legacy Free document and Windows XP Native Processor control document
  - http://www.microsoft.com/whdc/hwdev/tech/onnow/procperfctrl.mspx
- If Revision 3 is used in the FADT table then the proper length must be set

Common ASL Errors And Solutions

Divide By Zero

- The following code can potentially cause a divide by zero error:

```
Store(SMSC, Local0)
Store(SMLN, Local1)
Divide(Local0, Local1, Local2, Local3)
```

- It is better to check the value of Local1 before calling Divide:

```
Store(SMSC, Local0)
Store(SMLN, Local1)
If(Local1)
{
  Divide(Local0, Local1, Local2, Local3)
}
```

Common ASL Errors And Solutions

Avoid Possible Infinite Loop

- The following code can potentially be stuck in an infinite loop if the hardware has a error:

```
While(SMST)
{
    Sleep(0x2)
}
```

- It may be better to check the return x number of times, then break out of the While loop:

```
Store(20, Local2)
While(Land(SMST, Local2))
{
    Decrement(Local2)
    Sleep(0x2)
}
```

Common ASL Errors And Solutions

Use _REG to Check EC Status

```
Device(EC0)
{
        Name(REGC,Ones) // REGC is a Named object initialized with ones
        Method(_REG,2)
        {// _REG control method
            If(Lequal(Arg0, 3))
            {
                Store(Arg1, REGC)
            }
        } // end of _REG
        Method(ECAV,0)
        {  // ECAV control method
            If(Lequal(REGC,Ones))
            {
                If(LgreaterEqual(_REV,2))
                {
                        Return(One)
                }
                Else
                {
                        Return(Zero)
                }
            }
            Return(REGC)
        }
}
```

Common ASL Errors And Solutions

ASL code can check the availability of the EC Operation Region as follows:

```
If (\_SB.Pci0.Ec0.ECAV())
{
   ...Regions are available...
}
Else
{
   ...Regions are not available...
}
```

Common ASL Errors And Solutions

_OSI

- Windows XP implements a new Method called _OSI. A white paper with details on _OSI is available from: http://www.microsoft.com/whdc/hwdev/tech/onnow/_osi-method.mspx
- _OSI accepts a string as its only parameter. This string tells the OS which OS versions are supported by the BIOS. This way the OS can maintain compatibility with older BIOS and exposes new features to newer BIOS
- Currently _OSI accepts the following strings:
  - “Windows 2000”
  - “Windows 2001”
  - “Windows 2001 SP1”
  - “Windows 2001.1”
  - “Windows 2001 SP2”
  - “Windows 2001.1 SP1”
- The following new string is being added for Longhorn:
  - “Windows 2006”

Common ASL Errors And Solutions

- Windows XP checks for certain operations that are deemed unsafe, as they can cause system instability. If these operations are detected, Windows XP logs an event log entry and in some cases prevents the operation
- The following is a list of some of the operations considered unsafe:
  - Accessing CMOS registers(0x70-0x71) and PCI Config Space (0xCF8 – 0xCFC) via an SystemIO Operation Region. A complete list of IO ports considered unsafe to access via SystemIO Operation Regions can be found in a white paper titled “I/O Ports Blocked from BIOS AML on Windows XP”
  - Creating a SystemMemory Operation Region in address ranges that were reported by INT 15 function E820 as reserved for the OS (AddressRangeMemory). With the exception of the 1st page of physical memory

Common ASL Errors And Solutions

S4 RTC Wake

- In order for RTC wake to work when the machine is in S4, the RTC_S4 flag must be set within the FACP table (if set HIGH, then the platform supports RTC wakeup in the S4 state)

Common ASL Errors And Solutions

Order of Operations

- The Operating System does not guarantee any specific calling order for ASL methods, unless otherwise specified in the ACPI specification. Therefore, the BIOS writer should not expect any specific sequence in which control methods would get called

Common ASL Errors And Solutions

Thermal Zones Should Be Real

- All systems that have thermal zones must have real hardware (thermal sensors) to support the methods provided. I.e., if a BIOS has a _TMP method then the OS expects that it will return valid data

Common ASL Errors And Solutions

Do not issue Notify() to power button to cause monitor to turn on

- When a machine wakes due to an PME# event or a remote event, BIOS ASL code should not do Notification on the fixed feature power button in order to wake the monitor

Common ASL Errors And Solutions

_PRW may not be empty

- Must evaluate to a package with at least 2 elements
  - Bit index in GPEx_EN of enable bit armed for wake
  - Lowest system power sleep state that allows wake

Common ASL Errors And Solutions

Potential problems with PAE (Physical Address Extension) mode

- Systems may run in PAE mode more often with less than 4GB of RAM as PAE mode is a requirement of the NX (No Execute) feature
  - Execution protection (NX, or no execute) is an operating system feature that relies on processor hardware to mark memory with an attribute indicating that code should not be executed from that memory. Execution protection functions on a per-virtual memory page basis, most often leveraging a bit in the page table entry (PTE) to mark the memory page

Common ASL Errors And Solutions

Potential problems with PAE (Physical Address Extension) mode

- Microsoft has seen BIOS and PAE incompatibilities
  - Engagement with some OEMs has led to concerns about SMI handling routines
  - Some SMI handling routines attempt to decode the last instruction executed by the OS to determine attempted action (IO Port Write)
  - BIOS has a 32-bit virtual address and attempts to walk PTE's to discover 32-bit Physical Address and actual instruction opcode
  - PTE's are 64-bits long in PAE mode, leading to incompatibility
  - Most common observed behavior is hard lock or spontaneous reboot

Common ASL Errors And Solutions

Potential problems with PAE (Physical Address Extension) mode

- Corrective action:
  - Use processor manufacturer prescribed steps to determine IO address. (Apparently most processors keep this information around in a structure accessible while in SMM)
    - AMD BIOS writer's guide: “BIOS and Kernel Developer’s GUID for AMD Athlon™ 64 and AMD Operton™ Processors” located at http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/26049.PDF
    - Contact Intel for documentation

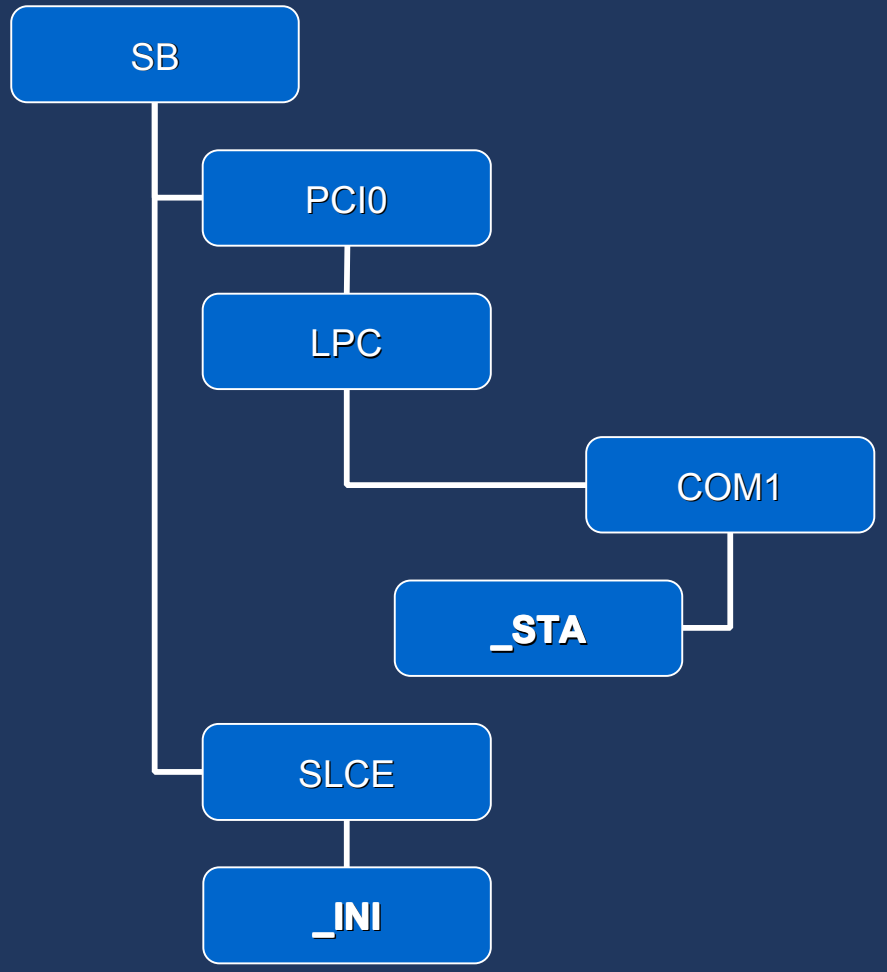
Common ASL Errors And Solutions

Hardware layout does not match the ACPI namespace layout

- We have seen a few cases related to Hot Plug controllers and Slices, where an ACPI device or its methods have a dependency to devices that are not its parent or in its parents hierarchy
- This is sort of representation can lead to serious problems because the Operating system does not understand or respect these dependencies

Common ASL Errors And Solutions

- In this name space representation we see that the value returned by the _STA method under the COM1 device changes after the _INI method under the SLCE device is evaluated
- This can cause problems and such dependencies should not be created
- One possible workaround to resolve this issue may be to create a method under SLCE, Lets' call it DPND, that can be called from the COM1 _STA
- DPND could do the necessary initialization so that COM1._STA can return the correct data every time



Some Upcoming Changes In The ACPI Driver

Some Upcoming Changes In The ACPI Driver

- For Windows XP SP2, a mechanism has been added that improves performance in the ACPI interrupt handling code
  - Windows XP had a problem whereby if other devices were sharing the ACPI interrupt, performance degradation could occur
    - This happened due to a delay by the ACPI driver in passing the interrupt to the proper handler
  - This feature is not "on" by default in Windows XP SP2 because of backward compatibility concerns
  - This feature will be "on" by default in Longhorn
  - To enable this feature on Windows XP SP2:
    - Open HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\ACPI\Parameters
    - Edit or add a DWORD value named "Attributes"
    - Bitwise OR 0x00000100 into the "Attributes" value
    - Restart the system

Some Upcoming Changes In The ACPI Driver

- ECDT support in Longhorn
  - For Windows XP an ACPI BIOS has to wait for the OS to Call the _REG method before it can start accessing EC operation regions via ASL. This limitation will no longer exist in Longhorn
  - Platform vendors can simplify BIOS code by providing the ECDT table
  - This table will cause longhorn to load Embedded Controller operation region support as soon as the AML interpreter is ready to execute code

Additional Resources

- Email
  - aslhelp @ microsoft.com – Writing ASL, debugging, general ACPI questions
  - onnow @ microsoft.com – Windows Power Management Questions
- Web Resources:
  - ACPI and Windows Power Management http://www.microsoft.com/whdc/hwdev/tech/onnow/default.mspx
  - ACPI Specification: http://www.acpi.info

Question & Answer