Development Exploration 102 – ACPI Exploration 01 (UEFI Configu ble)

bing4365 Posted on 2022-01-08 15:21:29 Read 6.8k Collection 33 Likes 4

tegory Column: UEFI Development Article Tags: uefi UEFI Programming Practice acpi EDK2 Low-level application development

UEFI Development This column includes this content 503 Subscribe 104 articles Subs our

ep it-> Author: Luo Bing https://blog.csdn.net/luobing4365)

ploration 01 (UEFI Configuration Table)

I had a task to modify the ACPI table. The problem itself was not complicated, but because the code needed to be ported to O ncountered many strange phenomena. Therefore, I spent a lot of time studying ACPI.

was completed, and my curiosity was aroused again. I was ready to use the belief of "studying things to gain knowledge" to sor ge I thought about various aspects of ACPI.

eral planning

'I exploration blog will probably have quite a few chapters. I plan to learn about ACPI from three perspectives, including the UEF ve, the operating system perspective, and the ACPI specification perspective.

ngement of the content will not be so standardized, I will write whatever comes to my mind, the general plan is as follows:

n UEFI configuration table;
uction to ACPI specification;
UEFI Protocol to analyze AML Code;
ew in ShellPkg
nentation of ACPI in EDK2
:A independent of operating system
ACPI analysis tools under Windows/Linux
ACPI related topics

a set of low-level specifications independent of the operating system, plays a huge role in modern operating systems. Currentl tions have been handed over to the UEFI official website for maintenance, and various versions of the specification documents ded from UEFI.org.

le first borrows the method from my friend lab-z's blog to find various ACPI-related tables through the UEFI configuration table. s as follows: http://www.lab-z.com/studsdt/

Configuration Table

ter of UEF Configuration Table is included in System Table. As the basic architecture of UEFI, System Table is used throughout velopment stage of UEFI Application and UEFI driver. Each basic UEFI module includes System Table in its entry parameters.

```
ypedef
FI_STATUS
EFIAPI *EFI_IMAGE_ENTRY_POINT) (
 IN EFI_HANDLE ImageHandle,
 IN EFI_SYSTEM_TABLE *SystemTable
;
```

e structure of EFI_SYSTEM_TABLE : (refer to MdePkg\Include\Uefi\UefiSpec.h)

```
ypedef struct {
 EFI_TABLE_HEADER Hdr; /// The table header for the EFI System Table.

 CHAR16 *FirmwareVendor; /// A pointer to a null terminated string that identifies the vendor
                  /// that produces the system firmware for the platform.
 UINT32 FirmwareRevision; /// A firmware vendor specific value that identifies the revision
                          /// of the system firmware for the platform.
 EFI_HANDLE ConsoleInHandle; /// The handle for the active console input device. This handle must support
                             /// EFI_SIMPLE_TEXT_INPUT_PROTOCOL and EFI_SIMPLE_TEXT_INPUT_EX_PROTOCOL.

 EFI_SIMPLE_TEXT_INPUT_PROTOCOL    *ConIn; /// A pointer to the EFI_SIMPLE_TEXT_INPUT_PROTOCOL interface t
                                           /// associated with ConsoleInHandle.
 EFI_HANDLE ConsoleOutHandle;              /// The handle for the active console output device.
 EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL   *ConOut;  /// A pointer to the EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL interfac
                                             /// that is associated with ConsoleOutHandle.
 EFI_HANDLE StandardErrorHandle;  /// The handle for the active standard error console device.
                                  /// This handle must support the EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL.
 EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL   *StdErr; /// A pointer to the EFI_SIMPLE_TEXT_OUTPUT_PROTOCOL interface
                                            /// that is associated with StandardErrorHandle.
 EFI_RUNTIME_SERVICES *RuntimeServices; /// A pointer to the EFI Runtime Services Table.
 EFI_BOOT_SERVICES *BootServices;       /// A pointer to the EFI Boot Services Table.
 UINTN NumberOfTableEntries; /// The number of system configuration tables in the buffer ConfigurationTabl
 EFI_CONFIGURATION_TABLE *ConfigurationTable;   /// A pointer to the system configuration tables.
                                                /// The number of entries in the table is NumberOfTableEnt
 EFI_SYSTEM_TABLE;
```

see many familiar protocols, such as ConIn, Conout, BootService, etc. It contains a series of pointers to the Console device, able, Boot Service Table, DXE Service Table and Configuration Table. RS and BS contain many basic functions, and Configura ntains ACPI, SMBIOS and other tables.

of Configuration Table is EFI_CONFIGURATION_TABLE, which is a set of GUID/Point pairs. The data structure is as follows

```
ypedef struct {
 ///
 /// The 128-bit GUID value that uniquely identifies the system configuration table.
 ///
 EFI_GUID                          VendorGuid;
 ///
 /// A pointer to the table associated with VendorGuid.
 ///
 VOID                              *VendorTable;
 EFI_CONFIGURATION_TABLE;
```

FI Spec, some UEFI configuration table GUIDs are given:

```
define EFI_ACPI_20_TABLE_GUID \
0x8868e871,0xe4f1,0x11d3,\
0xbc,0x22,0x00,0x80,0xc7,0x3c,0x88,0x81}}
define ACPI_TABLE_GUID \
0xeb9d2d30,0x2d88,0x11d3,\
0x9a,0x16,0x00,0x90,0x27,0x3f,0xc1,0x4d}}
define SAL_SYSTEM_TABLE_GUID \
0xeb9d2d32,0x2d88,0x11d3,\
0x9a,0x16,0x00,0x90,0x27,0x3f,0xc1,0x4d}}
define SMBIOS_TABLE_GUID \
0xeb9d2d31,0x2d88,0x11d3,\
0x9a,0x16,0x00,0x90,0x27,0x3f,0xc1,0x4d}}
define SMBIOS3_TABLE_GUID \
0xf2fd1544, 0x9794, 0x4a2c,\
0x99,0x2e,0xe5,0xbb,0xcf,0x20,0xe3,0x94})
define MPS_TABLE_GUID \
0xeb9d2d2f,0x2d88,0x11d3,\
0x9a,0x16,0x00,0x90,0x27,0x3f,0xc1,0x4d}}
/
/ ACPI 2.0 or newer tables should use EFI_ACPI_TABLE_GUID
/
define EFI_ACPI_TABLE_GUID \
0x8868e871,0xe4f1,0x11d3,\
0xbc,0x22,0x00,0x80,0xc7,0x3c,0x88,0x81}}

define EFI_ACPI_20_TABLE_GUID EFI_ACPI_TABLE_GUID?

define ACPI_TABLE_GUID \
0xeb9d2d30,0x2d88,0x11d3,\
0x9a,0x16,0x00,0x90,0x27,0x3f,0xc1,0x4d}}

define ACPI_10_TABLE_GUID ACPI_TABLE_GUID
```

the corresponding GUID, you can find the required ACPI Table pointer.

Implementation

le does not intend to explain the concepts of RSDP, FADT, etc. in ACPI, but prints out some information based on the data struc in EDK2. Through experiments, you can have a practical understanding of various ACPI tables, and the relationship between th uced in the next article.

nple code comes from the lab-z article introduced at the beginning of this article. I just slightly modified some statements. The fi enumerates all the items contained in the configuration table. The code is as follows:

```
OID ListConfigurationTable(VOID)

 UINTN i;
 EFI_CONFIGURATION_TABLE *configTab = NULL;

 Print(L"Number of Configuration Tables: %d\n",gST->NumberOfTableEntries);
 configTab = gST->ConfigurationTable;
 for(i=0; i<gST->NumberOfTableEntries;i++)
 {
   Print(L"No%d. %g\n",i+1, &configTab->VendorGuid); //%g - a pointer to a GUID structure.
   configTab++;
```

```
}
```

is to take out all the table items contained in the configuration table in SystemTable and print out their GUIDs.

function demonstrates how to find the DSDT table in ACPI through the configuration table. The code is as follows:

```
OID ListAcpiTable(VOID)

 UINTN     i,j,EntryCount;
 CHAR8 strBuff[20];
 UINT64    *EntryPtr;
 EFI_GUID  AcpiTableGuid  = ACPI_TABLE_GUID;
 EFI_GUID  Acpi2TableGuid = EFI_ACPI_TABLE_GUID;
 EFI_CONFIGURATION_TABLE   *configTab=NULL;
 EFI_ACPI_DESCRIPTION_HEADER           *XSDT,*Entry,*DSDT;
 EFI_ACPI_5_0_FIXED_ACPI_DESCRIPTION_TABLE   *FADT;
 EFI_ACPI_5_0_ROOT_SYSTEM_DESCRIPTION_POINTER  *Root;

 Print(L"List ACPI Table:\n");
 configTab=gST->ConfigurationTable;

 for (i=0;i<gST->NumberOfTableEntries;i++)
 {
   //Step1. Find the table for ACPI
   if ((CompareGuid(&configTab->VendorGuid,&AcpiTableGuid) == 0) ||
     (CompareGuid(&configTab->VendorGuid,&Acpi2TableGuid) == 0))
     {
       Print(L"Found table: %g\n",&configTab->VendorGuid);
       Print(L"Address: @[0x%p]\n",configTab);

       Root=configTab->VendorTable;
       Print(L"ROOT SYSTEM DESCRIPTION @[0x%p]\n",Root);
       ZeroMem(strBuff,sizeof(strBuff));
       CopyMem(strBuff,&(Root->Signature),sizeof(UINT64));
       Print(L"RSDP-Signature [%a] (",strBuff);
       for(j=0;j<8;j++)
         Print(L"0x%x ",strBuff[j]);
       Print(L")\n");
       Print(L"RSDP-Revision [%d]\n",Root->Revision);
       ZeroMem(strBuff,sizeof(strBuff));
       for (j=0;j<6;j++) { strBuff[j]= (Root->OemId[j] & 0xFF); }
       Print(L"RSDP-OEMID [%a]\n",strBuff);

       Print(L"RSDT address= [0x%p], Length=[0x%X]\n",Root->RsdtAddress,Root->Length);
       Print(L"XSDT address= [0x%LX]\n",Root->XsdtAddress);
       WaitKey();
       // Step2. Check the Revision, we olny accept Revision >= 2
       if (Root->Revision >= EFI_ACPI_5_0_ROOT_SYSTEM_DESCRIPTION_POINTER_REVISION)
       {
         // Step3. Get XSDT address
         XSDT=(EFI_ACPI_DESCRIPTION_HEADER *)(UINTN) Root->XsdtAddress;
         EntryCount = (XSDT->Length - sizeof(EFI_ACPI_DESCRIPTION_HEADER))
                     / sizeof(UINT64);
         ZeroMem(strBuff,sizeof(strBuff));
         CopyMem(strBuff,&(XSDT->Signature),sizeof(UINT32));
         Print(L"XSDT-Sign [%a]\n",strBuff);
         Print(L"XSDT-length [%d]\n",XSDT->Length);
```

```
        Print(L"XSDT-Counter [%d]\n",EntryCount);

        // Step4. Check the signature of every entry
        EntryPtr=(UINT64 *)(XSDT+1);
        for (j=0;j<EntryCount; j++,EntryPtr++)
        {

          Entry=(EFI_ACPI_DESCRIPTION_HEADER *)((UINTN)(*EntryPtr));

          // Step5. Find the FADT table
          if (Entry->Signature==0x50434146) { //'FACP'
            FADT = (EFI_ACPI_5_0_FIXED_ACPI_DESCRIPTION_TABLE *)(UINTN) Entry;
            Print(L"FADT->Dsdt = 0x%X\n",FADT->Dsdt);
            Print(L"FADT->xDsdt = 0x%LX\n",FADT->XDsdt);

            // Step6. Get DSDT address
            DSDT = (EFI_ACPI_DESCRIPTION_HEADER *) (FADT->Dsdt);
            Print(L"DSDT table @[%X]\n",DSDT);
            Print(L"DSDT-Length = 0x%x\n",DSDT->Length);
            Print(L"DSDT-Checksum = 0x%x\n",DSDT->Checksum);
          }
        }
      }
    }
  configTab++;
}
```

ionship between the ACPI entries will be described in detail in the next article.

there are many tables in ACPI, and the DSDT table is used to describe the fixed parts of the system, including power managem nanagement, and plug-and-play functions.

ram finds the DSDT table through the order of RSDP->XSDT->FADT->DSDT and prints out some information of interest.

n, it should be noted that EDK2 provides a large number of functions, data structures, GUIDs, etc. for ACPI table processing, w in the following header files:

```
include <Guid/Acpi.h>
include <IndustryStandard/Acpi10.h>
include <IndustryStandard/Acpi50.h>
```

y to compile it yourself and experiment on an actual machine.