Development Exploration 101 – PCD Exploration

ng4365 Posted on 2021-10-18 22:11:19 Read 5.5k Collection 43 Likes 14

gory Column: UEFI Development Article Tags: uefi UEFI Programming Practice bios Low-level application development

EFI Development This column includes this content 503 Subscribe 104 articles Su

o

This article introduces the concept, usage and types of UEFI Platform Configuration Database (PCD) in detail, and emphasizes the importan reuse and modularization. It shows how to declare and use PCD in DEC and INF files, and how to access and modify PCD variables in program o shares the problems and experiences encountered in practice, providing readers with a way to deeply understand PCD.

ary is generated by CSDN through intelligent technology

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

ration

ublication of "UEFI Programming Practice", a series of things have happened one after another, and finally today, it has come to an

period, I had many opportunities to discuss UEFI and BIOS with people in the industry. Reflecting on my understanding of this field was inadequate.

ho have read "UEFI Programming Practice" should understand that the book has relatively little explanation of the theoretical part o usually follow the framework of "raising questions-introducing UEFI related knowledge-providing examples" to conduct research aro c.

reasons is that I was originally aiming for practice, and gradually discussed the topics encountered in daily development. Another re ot go deep into the specific implementation of EDK2.

starting from the 101st chapter of UEFI development exploration, I want to gradually turn to the study of EDK2 code. Just like the pr CI Option ROM development, the goals set this time include:

the OvmfPkg source code to understand the firmware architecture, compilation process, and code implementation at each stage;

stand how Qemu uses firmware to boot and how to boot the operating system;

stand how Ovmf firmware provides various tables, runtime services, and even SMI Handlers required by the operating system. Und
UEFI operating system is integrated with BIOS.

d that this is a big goal and there will be many difficulties, but I am not particularly afraid of it because of my interest. My original ide
periments on the Raspberry Pi, but at the suggestion of lab-z (blog: https://www.lab-z.com/), I felt that OvmfPkg would be more c
ents.

pic is determined, I will fill in the gaps in the knowledge I am not familiar with later. Well, I will start with the core of EDK2 global cor

troduction

onfiguration Database (PCD) is the mechanism used by EDK2 for global configuration, and plays a huge role in code reuse and
tion.

ts the configurable options from the code, so when the platform needs to be modified, there is no need to modify the source code
n of its parameters can be done during the compilation process, at runtime, and even in the binary file.

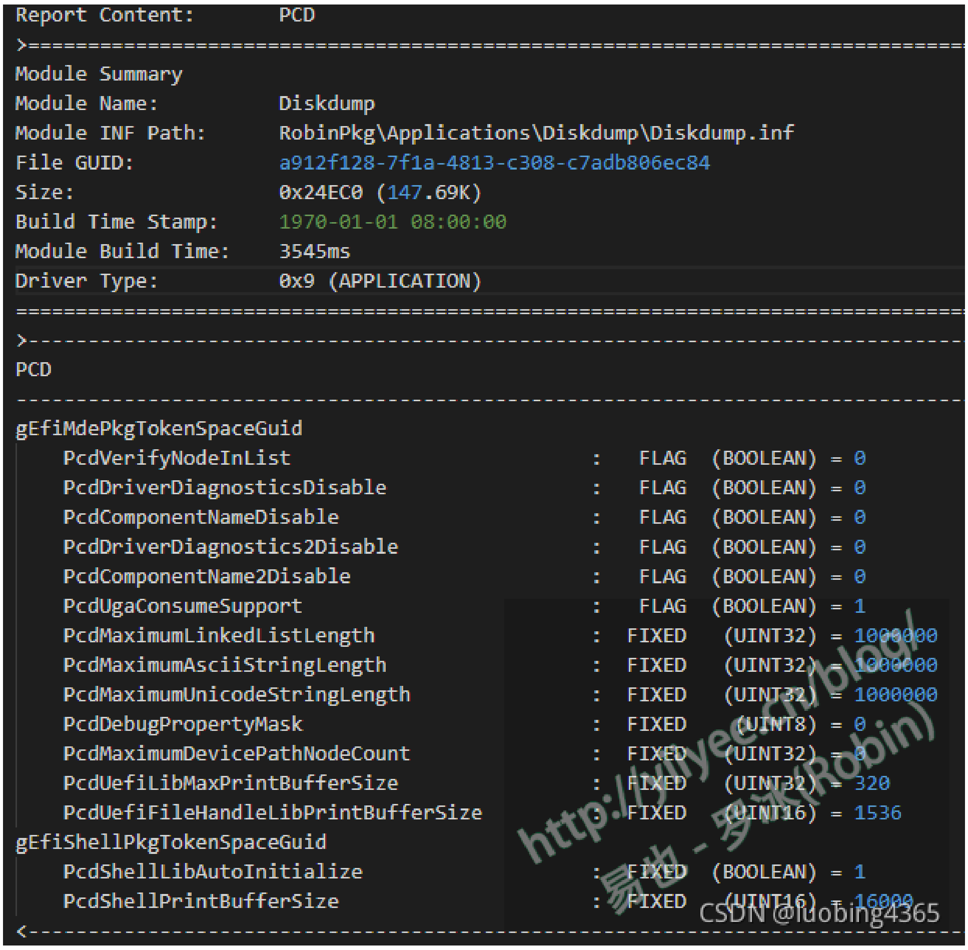
approach is quite fascinating, and it also makes customization easier and the code easier to maintain.

days, I always thought that PCD was like macros in C/C++, used to extract common code, which was wrong. The functions it provi
sive and more complex.

ake a look at the PCD used in ordinary programs. Take the Diskdump project in the previous articles as an example. Use the followi
o compile and extract the PCD information used:

```
vUDK2018\edk2>build -Y PCD -y pcd.log -p RobinPkg\RobinPkg.dsc -m RobinPkg\Applications\Diskdump\Diskdump.inf
```

information is stored in pcd.log. Check the log information:



```
Report Content:         PCD
>===========================================================================
Module Summary
Module Name:            Diskdump
Module INF Path:        RobinPkg\Applications\Diskdump\Diskdump.inf
File GUID:              a912f128-7f1a-4813-c308-c7adb806ec84
Size:                   0x24EC0 (147.69K)
Build Time Stamp:       1970-01-01 08:00:00
Module Build Time:      3545ms
Driver Type:            0x9 (APPLICATION)
===========================================================================
>---------------------------------------------------------------------------
PCD
---------------------------------------------------------------------------
gEfiMdePkgTokenSpaceGuid
    PcdVerifyNodeInList                         :   FLAG  (BOOLEAN) = 0
    PcdDriverDiagnosticsDisable                 :   FLAG  (BOOLEAN) = 0
    PcdComponentNameDisable                     :   FLAG  (BOOLEAN) = 0
    PcdDriverDiagnostics2Disable                :   FLAG  (BOOLEAN) = 0
    PcdComponentName2Disable                    :   FLAG  (BOOLEAN) = 0
    PcdUgaConsumeSupport                        :   FLAG  (BOOLEAN) = 1
    PcdMaximumLinkedListLength                  :  FIXED   (UINT32) = 1000000
    PcdMaximumAsciiStringLength                 :  FIXED   (UINT32) = 1000000
    PcdMaximumUnicodeStringLength               :  FIXED   (UINT32) = 1000000
    PcdDebugPropertyMask                        :  FIXED    (UINT8) = 0
    PcdMaximumDevicePathNodeCount               :  FIXED   (UINT32) = 0
    PcdUefiLibMaxPrintBufferSize                :  FIXED   (UINT32) = 320
    PcdUefiFileHandleLibPrintBufferSize         :  FIXED   (UINT16) = 1536
gEfiShellPkgTokenSpaceGuid
    PcdShellLibAutoInitialize                   :  FIXED  (BOOLEAN) = 1
    PcdShellPrintBufferSize                     :  FIXED   (UINT16) = 16000
<---------------------------------------------------------------------------
```

CD used in Diskdump

at it, you can see many PCD parameters that you didn't notice when programming. Diskdump uses the Protocol in MdePkg and She s related PCD.

ocuments, you can refer to:
e:
ePkg\Universal\PCD\Dxe\Pcd.inf
hub.com/tianocore/tianocore.github.io/wiki/EDK-II-Documents:
latform Configuration Database Infrastructure Description》
latform Description(DSC) File Specification》
ackage Declaration(DEC) File Format Specification》
uild Specification》
i.org/specifications:
Initialization(PI) Specification》

use PCD

e used for most of the time that UEFI exists, except in the SEC stage, the early PEI and DXE stages, and can basically be accesse need to understand the structure and type of PCD.

of PCD

of a PCD variable is a bit like a structure:

```
enSpaceGuidCName.PcdCName
```

m, TokenSpaceGuidCName is the GUID, and PcdCName is the variable name, and the combination of the two constitutes a PCD v

he following types of PCD.

ild Type

ined at compile time, is a static value, and cannot be changed at runtime or in binary form. It can be considered a macro.

g Type

the same type as FixedAtBuild and returns a Bool type (True or False), which can be used to judge the condition.

nModule Type

of this type of variable is determined at compile time, and it is modified using tools on the compiled binary file. Unlike FixedAtBuild, i module (scoped in one module).

ype, DynamicHii type, and DynamicVpd type

of the Dynamic type variable is the entire system. It is a dynamic PCD and can be modified during the UEFI operation.

icHii type and the Dynamic type are stored in different locations. The Dynamic type can be considered to exist in Memory, and the l be lost when loaded again; while the DynamicHii type exists in the Efi variable (NVRAM), and its modification is non-volatile.

d type variables are read-only and cannot be written, which is generally determined by the factory.

x Type

e Dynamic type, it is an enhanced version. The difference between it and the Dynamic type is whether to use the PCD in the binar you want to use the PCD variable in FSP, the PCD type in FSP must be set to ### DynamicEx type.

sing PCD variables

PCD variables, PEI provides PCD_PPI and EFI_PEI_PCD_PPI; DXE provides PCD_PROTOCOL and EFI_PCD_PROTOCOL.

r ease of use, EDK2 introduced the PCD Library to hide these access details. (MdePkg\Include\Library\PcdLib.h)

contains the following functions :

```
GetXX()
SetXX()
GetExXX()
SetExXX()
Token()
SetSku()
GetNextToken()
GetNextTokenSpace()
lBackOnSet()
celCallBack()
```

an be 8, 16, 32, Size, Ptr, or Boolean.

ration and Use of PCD

PCD can basically be carried out according to the following process.

information for declaring PCD variables in DEC files , such as:

```
dsFixedAtBuild, PcdsPatchableInModule, PcdsDynamic, PcdsDynamicEx]
  gEfiMdeModulePkgTokenSpaceGuid.PcdHelloWorldPrintTimes|1|UINT32|0x40000005
  gEfiMdeModulePkgTokenSpaceGuid.PcdHelloWorldPrintString|L"UEFI Hello World!\n"|VOID*|0x40000004
```

:

```
okenSpaceGuidCname.PcdCname|DefaultValue|DatumType|Token
```

ed above, PcdCname is the variable name, DefaultValue is its default value, DatumType is the data type of PCD, Token is a 32-bit i
CD in DEC has a unique Token.

can be of type BOOLEAN, UINT8, UINT16, UINT32, UINT64, or VOID *.

the value of PCD variable in DSC file
n set the value of the corresponding PCD variable in the DSC file, for example:

```
dsFixedAtBuild]
EfiMdePkgTokenSpaceGuid.PcdDebugPropertyMask|0x0f
```

process is not necessary. If not set, the default value in the DEC file is used.

ation in INF file
module's INF file, you need to declare the PCD variable before you can use it in the source code. For example:

```
d]
EfiMdeModulePkgTokenSpaceGuid.PcdHelloWorldPrintString
EfiMdeModulePkgTokenSpaceGuid.PcdHelloWorldPrintTimes
```

ed to list the PCD variable name, and no other information.

leting the above work, you can use PCD library functions to access PCD variables in the source code. The example is as follows: (E
odulePkg\Application\HelloWorld\HelloWorld.c)

```
 (FeaturePcdGet (PcdHelloWorldPrintEnable)) {
  for (Index = 0; Index < PcdGet32 (PcdHelloWorldPrintTimes); Index ++) {
    //
    // Use UefiLib Print API to print string to UEFI console
    //
      Print ((CHAR16*)PcdGetPtr (PcdHelloWorldPrintString));
  }
```

write an example

plication development or Option ROM development, PCD variables are basically not used. But it does not prevent them from being
cations. Let's try to use PCD variables in an Application of RobinPkg.

Diskdump project I developed before, renamed it to Pcdtouch, and tried to use PCD variables. Of course, you can choose any othe
ject happened to be in front of me, so I modified it.

cation steps are as follows:

RobinPkg.dec
e following statement:

```
ids]
RobinPkgPcdSampleGuid = { 0xe7e1efa6, 0x7607, 0x3a78, { 0xc7, 0xdd, 0x43, 0xe4, 0xbd, 0x72, 0xc1, 0x19 }}
 PcdsFixedAtBuild, PcdsPatchableInModule, PcdsDynamic, PcdsDynamicEx]
dsPatchableInModule, PcdsDynamic, PcdsDynamicEx]
RobinPkgPcdSampleGuid.PcdtouchValue|12345|UINT32|0x90000005
RobinPkgPcdSampleGuid.PcdtouchStr|L"Hello,UEFI World, this is robin!\n"|VOID*|0x90000004
```

modify the value of the PCD variable by modifying the Pcdtouch.inf
e do not need to modify it here, and we do not need to modify the DSC file.

dify the INF file and add the PCD variables that will be used in the source program:

```
aturePcd]
EfiMdeModulePkgTokenSpaceGuid.PcdHelloWorldPrintEnable   ## CONSUMES

d]
EfiMdeModulePkgTokenSpaceGuid.PcdHelloWorldPrintString   ## SOMETIMES_CONSUMES
EfiMdeModulePkgTokenSpaceGuid.PcdHelloWorldPrintTimes    ## SOMETIMES_CONSUMES
RobinPkgPcdSampleGuid.PcdtouchStr
RobinPkgPcdSampleGuid.PcdtouchValue
```

to the two PCD variables added in the DEC file, several PCD variables in MdeModulePkg are also declared, which will be used in tl

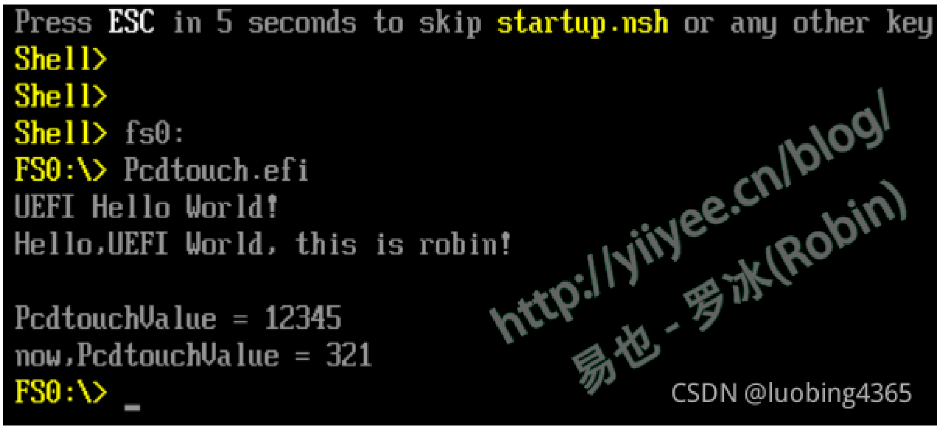
le to the source program Pcdtouch.c.
lify the main program:

```
n (
N int Argc,
N char **Argv
```

```
INT32 Index,myValue;

ndex = 0;
yValue = 0;
```

cation is now complete. After compiling, run it in the simulator, and the results are as follows:



```
Press ESC in 5 seconds to skip startup.nsh or any other key
Shell>
Shell>
Shell> fs0:
FS0:\> Pcdtouch.efi
UEFI Hello World!
Hello,UEFI World, this is robin!

PcdtouchValue = 12345
now,PcdtouchValue = 321
FS0:\> _
```

dtouch running results

riences gained during the writing process:

variable of type VOID * (string type) cannot be defined as only type PcdsDynamic, otherwise the compilation will fail. (Is it because dified? The specific reason is unclear);

CD variable using PcdSet32 cannot be defined as PcdsFixedAtBuild type, and the compilation will prompt that the PCD variable car (This is easy to understand, because the PcdsFixedAtBuild type is determined at compile time)

TokenSpaceGuidCname.PcdCname|DefaultValue|DatumType|Token
m, the PCD variable should be uniquely identified by TokenSpaceGuidCname and Token, and PcdCname is convenient for progran (to be confirmed);

CD types have a priority of FixedAtBuild>PatchableInModule>Dynamic>DynamicEx. They are identified in this order when there is r ation (to be confirmed).

SC file is not modified in the example. It should be noted that FixedAtBuild is defined under [PcdsFixedAtBuild] in both DEC and DS namic type is defined under [PcdsDynamic] in DEC and [PcdsDynamicDefault] (or PcdsDynamicHii, PcdsDynamicVpd) in DSC. Th ed, and you can check the DSC and DEC file specifications;

ue to deepen my understanding later.

ess: https://gitee.com/luobing4365/uefi-exolorer
e is located in: /FF RobinPkg/RobinPkg/Applications/Pcdtouch