






EFI Basic Tutorial (VI) - Simple Use of PROTOCOL

 xiaopangzi313

 Modified on 2024-10-30 22:53:51


 Read 4.8k


 Collection 14

 Likes 8

Copyright CC 4.0 BY-

Category columns: 15_Firmware Development Article Tags: UEFI FirmWare BIOS

 CSDN Learning Co... The article has been collected by the community

 15_Firmware Devel... This column includes this content

27 articles

Join the community

Subscribe to our column

摘要 This article describes how to use the Protocol mechanism to make UEFI applications easily access DXE driver resources. The specific steps include writing UEFI application and DXE driver code, compiling to generate EFI files, and running them in a virtual environment. With the help of gMyHelloWorldPEIGUID, the application can locate and call the driver service.

This summary is generated in [C Know](#), supported by DeepSeek-R1 full version, [go to experience](#)

Write source code

Write UEFI Application code C:\edkii\OvmfPkg\MyHelloWorldAppProtocol\MyHelloWorldAppProtocol.c,

C AI generated projects 登录复制

```
1  ...
2  static EFI_MYHELLOWORLD_PROTOCOL * gMyHelloWorldProtocol = NULL;
3
4  EFI_STATUS
5  EFIAPI
6  MyHelloWorldAppProtocolEntry(
7      IN EFI_HANDLE          ImageHandle,
8      IN EFI_SYSTEM_TABLE    *SystemTable
9  )
10 {
11     EFI_STATUS  Status = EFI_SUCCESS;
12     DEBUG ((EFI_D_ERROR, "[MyHelloWorldProtocol] MyHelloWorldAppProtocolEntry Start...\n"));
13     Print (L"[MyHelloWorldProtocol] MyHelloWorldAppProtocolEntry Start...\n");
14
15     Status = gBS->LocateProtocol(&gMyHelloWorldPEIGUID, NULL, (VOID **) &gMyHelloWorldProtocol);
16     if (EFI_ERROR(Status)){
17         Print(L"[MyHelloWorldProtocol] Locate Protocol gMyHelloWorldProtocol %r \n", Status);
18         return Status;
19     }
20     gMyHelloWorldProtocol->PrintMsg(gMyHelloWorldProtocol, L"Hello World App....\n");
21     DEBUG ((EFI_D_ERROR, "[MyHelloWorldProtocol] MyHelloWorldAppProtocolEntry End...\n"));
22     Print (L"[MyHelloWorldProtocol] MyHelloWorldAppProtocolEntry End ... \n");
23
24     return Status;
25 }
```

收起 ^

Write **UEFI DXE Driver** the code C:\edkii\OvmfPkg\MyHelloWorldDXEProtocol\MyHelloWorldDXEProtocol.c,

C AI generated projects 登录复制

```
1  ...
2  static EFI_MYHELLOWORLD_PROTOCOL  gMyHelloWorldProtocol ;
3
4  EFI_STATUS
5  EFIAPI
6  MyHelloWorldDXEProtocolEntry(
7      IN EFI_HANDLE          ImageHandle,
```

```

0      IN EFI_SYSTEM_TABLE  *SystemTable
9    )
10   {
11     EFI_STATUS              Status = EFI_SUCCESS;
12     DEBUG ((EFI_D_ERROR , "[MyHelloWorldProtocol]  MyHelloWorldDXEProtocolEntry Start..\n"));
13
14     gMyHelloWorldProtocol.PrintMsg = PrintHelloWorldMsg;
15     gMyHelloWorldProtocol.Revision = 1;
16
17     Status = gBS->InstallProtocolInterface(&ImageHandle,
18                                           &gMyHelloWorldPEIGUID,
19                                           EFI_NATIVE_INTERFACE,
20                                           &gMyHelloWorldProtocol
twen                                     );
twen
twen   if (!EFI_ERROR(Status)){
twen       DEBUG ((EFI_D_ERROR , "[MyHelloWorldProtocol]  MyHelloWorldDXEProtocolEntry Installed Protocol Successfully..\n"))
twen   }else{
25       DEBUG ((EFI_D_ERROR , "[MyHelloWorldProtocol]  MyHelloWorldDXEProtocolEntry Installed Protocol Faily..\n"));
26   }
27
28   DEBUG ((EFI_D_ERROR , "[MyHelloWorldProtocol]  MyHelloWorldDXEProtocolEntry End..\n"));
29   return Status;
30 }

```

收起 ^

Compile and generate EFI files

and **edksetup.bat** compile the entire OvmfPkg Package

Run DXE Driver **MyHelloWorldDXEProtocol** and UEFI APP **MyHelloWorldAppProtocol**

Copy **C:\edkii\Build\OvmfX64\DEBUG_VS2013x86\FV\OVMF.fd** to **C:\qemu** ; Copy

C:\edkii\Build\OvmfX64\DEBUG_VS2013x86\X64\OvmfPkg\MyHelloWorldAppProtocol\MyHelloWorldAppProtocol\OUTPUT\MyHelloWorldAppProtocol.efi to virtual disk **HDD_BOOT.img**

Execute , and then execute in , the result is as follows, **setup-qemu-x64.bat | findstr MyHelloWorldProtocol UEFI SHELL MyHelloWorldAppProtocol.efi**

```
QEMU - Press Ctrl+Alt+G to release grab
Machine View

FS0:\> ls
Directory of: FS0:\
04/27/2019  00:32                10,464  HelloWorld.efi
04/27/2019  20:45                11,136  MyHelloWorldAppProtocol.efi
                2 File(s)          21,600 bytes
                0 Dir(s)

FS0:\> MyHelloWorldAppProtocol.efi
[MyHelloWorldProtocol] MyHelloWorldAppProtocolEntry Start..
Hello World App.....
[MyHelloWorldProtocol] MyHelloWorldAppProtocolEntry End ...
FS0:\> _
```

<https://blog.csdn.net/xiaopangzi313>

```
C:\qemu>setup-qemu-x64.bat | findstr MyHelloWorldProtocol
WARNING: Image format was not specified for 'HDD_BOOT.img' and probing guessed raw.
Automatically detecting the format is dangerous for raw images, write operation
s on block 0 will be restricted.
Specify the 'raw' format explicitly to remove the restrictions.
[MyHelloWorldProtocol] MyHelloWorldDXEProtocolEntry Start..
[MyHelloWorldProtocol] MyHelloWorldDXEProtocolEntry Installed Protocol Successfully..
[MyHelloWorldProtocol] MyHelloWorldDXEProtocolEntry End..
[MyHelloWorldProtocol] MyHelloWorldAppProtocolEntry Start..
[MyHelloWorldProtocol] MyHelloWorldAppProtocolEntry Start..
[MyHelloWorldProtocol] PrintHelloWorldMsg Hello World App.....
[MyHelloWorldProtocol] MyHelloWorldAppProtocolEntry End..
[MyHelloWorldProtocol] MyHelloWorldAppProtocolEntry End .. https://blog.csdn.net/xiaopangzi313
```

Summary

With the help of Protocol mechanism, the application can easily access the resources of the driver (DXE Driver). In this article, the driver module `gHelloWorldDXEProtocol` registers the Protocol service at the program entry `gMyHelloWorldProtocol`, and the service is identified as `'HelloWorldPEIGUID'`; then the application can obtain the service registered in the driver `MyHelloWorldAppProtocol` in its entry function `GUID`, and then call the function in the service. Among them, there are multiple interfaces for creating Protocol, and you can create a single or multiple Protocols as needed. There are also multiple interfaces for obtaining Protocol, such as `LocateProtocol`, `HandleProtocol`, and `OpenProtocol`. The specific differences can be referred to in the spec.

In the Dxe/gRT/BDS stage, protocol plays a very important role, such as `PCIe`, `Opmem`, `SMM`, `MultiProcessor`, `Variable`, `driver model`, `Smbios`, `ACPI` and other subsystems or services. Specifically, if you want to operate a device `SATA/NVME/CXL/VGA/Serial port` in the `UEFI` environment, you don't need to directly care about the library implementation of these domains. You only need to get the corresponding Protocol instance through the `guid` given in the spec, and then call the corresponding function. On the other hand, the protocol mechanism protects the interests of manufacturers. For example, `Intel`, `IBV`, and `OEM` release some core drivers in the form of `xxx.efi` when releasing code. After users get them, they only need to locate the corresponding Protocol to complete the access to the device.

In short, the meaning of Protocol is as follows:

Achieve code isolation and remove dependencies on calling libraries

To protect the interests of manufacturers, for example, in the BIOS code released by Intel, ASPPED VGA driver, VMD driver, SATA controller driver, PFR driver, and network card driver are all distributed in the form of `Opmem`.

To realize the linkage of driver events,

1) For example, if you want driver B to execute after driver A, you can add a dependency to driver B as follows,

```
[depex]
```

```
gDriverAProtocol # The gDriverAProtocol is installed in driver A
```

2) For example, if you want a function in driver B to be triggered by a protocol in driver A
gBS->RegisterProtocolNotify (&gDriverAProtocol, ...callback) //Driver B
gBS->InstallProtocolInterface (&gDriverAProtocol, xxx) //Driver A, once the protocol is installed, the callback in B will be called
Protocol is widely used in edk2. If you want to quickly find out the usage of the protocol in the current code, you can use the following regex to sea
in VSCode,

Using gEfiSmmSwDispatch2ProtocolGuid regex search as an example, the following match can quickly search all related Protocol operations from t
k

AI generated projects 登录复

1	. *? \ ((? = (\ n [^ \]) * ?) g E f i S m m S w D i s p a t c h 2 P r o t o c o l G u i d)	// 查询所有的gEfiSmmSwDispatch2ProtocolGuid操作
2	I n s t a l l . * ? \ ((? = (\ n [^ \]) * ?) g E f i S m m S w D i s p a t c h 2 P r o t o c o l G u i d)	// 查询所有的gEfiSmmSwDispatch2ProtocolGuid Install操作
3	L o c a t e . * ? \ ((? = (\ n [^ \]) * ?) g E f i S m m S w D i s p a t c h 2 P r o t o c o l G u i d)	// 查询所有的gEfiSmmSwDispatch2ProtocolGuid Locate操作
4	R e g i s t e r . * ? \ ((? = (\ n [^ \]) * ?) g E f i S m m S w D i s p a t c h 2 P r o t o c o l G u i d)	// 查询所有的gEfiSmmSwDispatch2ProtocolGuid Register操作

install.*? \ ((? = (\ n | [^ \]) * ?) g E f i S m m S w D i s p a t c h 2 P r o t o c o l G u i d)

es to include

c

es to exclude

result in 1 file - Open in editor

C PchSmiDispatchSmm.c edk2\UefiPay... 1

Status = gSmst->SmmInstallProt...

edk2 > UefiPayloadPkg > PchSmiDispatchSmm > C PchSmiDispatchSmm.c > PchSmiDispatchSmm.c

397)

429

430 //

431 // Locate PI SMM CPU protocol

432 //

433 Status = gSmst->SmmLocateProtocol (&gEfiSmmCpuProtocolGuid, NULL);

434 ASSERT_EFI_ERROR (Status);

435

436 //

437 // Register a SMM handler to handle subsequent SW SMIs.

438 //

439 Status = gSmst->SmiHandlerRegister ((EFI_MM_HANDLER_ENTRY_POINT) NULL, NULL);

440 ASSERT_EFI_ERROR (Status);

441

442 //

443 // Publish PI SMM SwDispatch2 Protocol

444 //

445 ImageHandle = NULL;

446 Status = gSmst->SmmInstallProtocolInterface (&ImageHandle, &gEfiSmmSwDispatch2ProtocolGuid, EFI_NATIVE_INTERFACE, &gSmmSwDispatch2Protocol);

447

448

449

450

451

452 ASSERT_EFI_ERROR (Status);

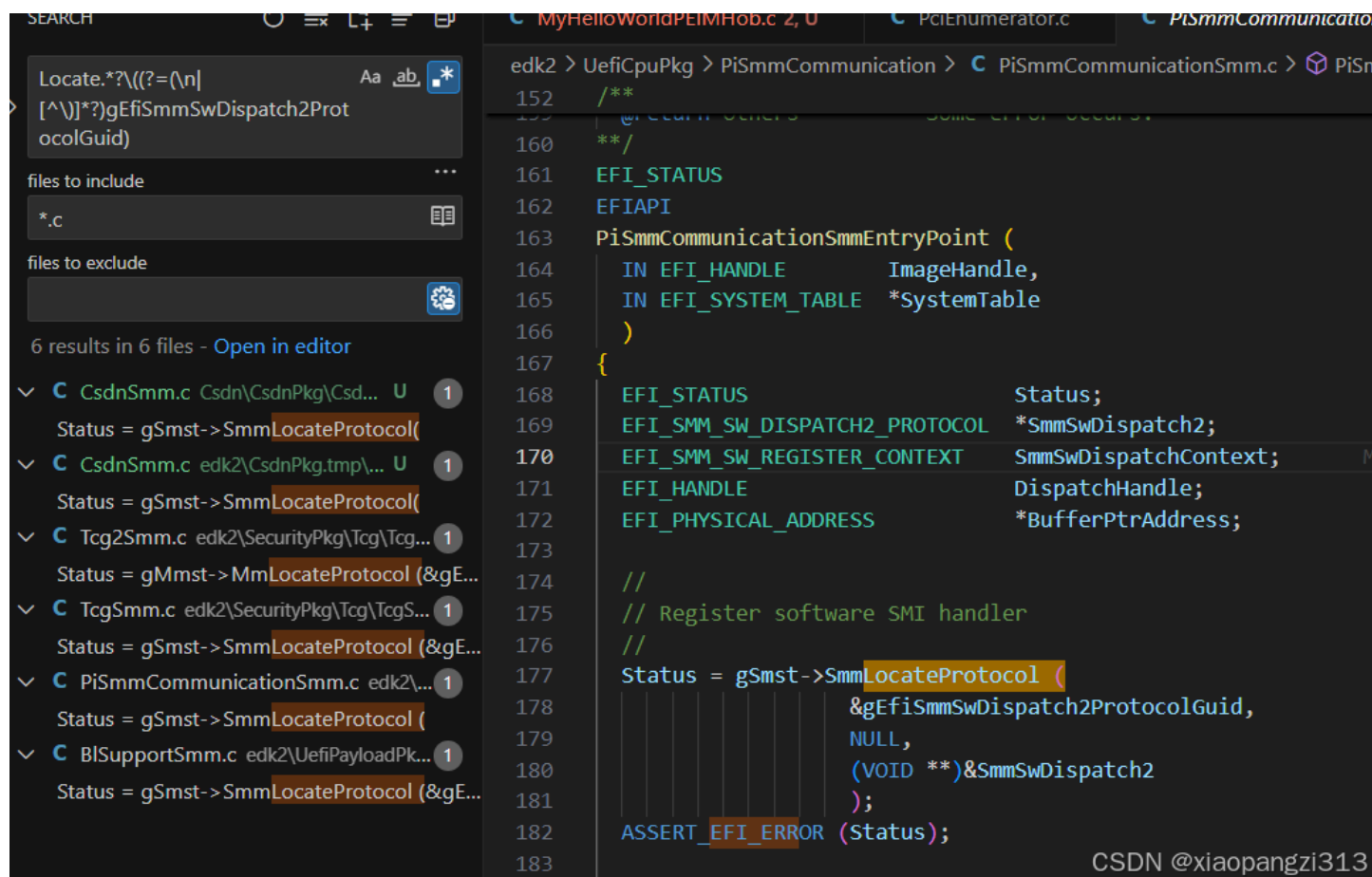
453

454 return Status;

455 }

456

CSDN @xiaopangzi313



3: The protocol is managed by gBS. So once the system boots to the [bootloader](#) (grub/lilo) when the ExitBootService Event is triggered, all gBS services will be discarded (DXE protocol will not be available). If you still want to use the service provided by BIOS, you can only use gRT or gSmm.

[\[EProtocol DEMO source code \]](#)

色无效搬运！Alex重构流量引擎

拆解：TikTok→独立站 双引擎→ 技术人优先预约通道

广告