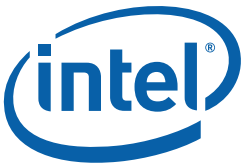


White Paper

Jenny Pelner
Firmware Architect
Intel Corporation

EDKII Platform Configuration Database Entries

An Introduction to PCD Entries

May 2011

325619-001

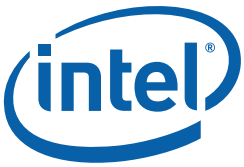
Executive Summary

The intent of this white paper is to simplify the concept of Platform Configuration Database (PCD) Entries as well as specify the definitions for the PCD types which were introduced for the EFI Development Kit II (EDKII).

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. http://www.intel.com/p/en_US/embedded.

§

Contents



PCD Introduction

Today, platforms often abstract configuration details associated with their platform and behavior. These configuration settings can be classified as two types:

1) Build-time generated platform settings
2) Run-time generation platform settings

A Platform Configuration Database (PCD) entry is a setting which is established during the time that the platform BIOS/Boot-loader is built. In the case of a UEFI compliant codebase, there are commonly defined interfaces for abstracting certain types of PCDs. The remainder of this whitepaper covers specifics associated with how one uses the PCDs and the roles associated with the differing types of PCDs.

A PCD Library Class provides a PCD usage macro interface for all PCD entry types. The PCD Library Class should be included in any module that uses PCD entries. For modules that use PCD entries, the following should be added to the module's DSC file:

```
[LibraryClasses]
        PcdLib|MdePkg/Library/DxePcdLib/DxePcdLib.inf
```

The file can optionally override any default values assigned to PCDs in PCD database holds all dynamic type PCD information.

A PCD entry is defined in a DEC file. The following section describes the types of PCD entries as well as the format for each. Each module must include any PCD entries used in the code in the INF file for that module or driver. The platform DSC either a module or driver's DEC or DSC file.

AutoGen.c and AutoGen.h

The PCD entries are translated to a variable or macro that is auto-generated by the build tool in each module's autogen.h and autogen.c files.

PCD Types

There are several types of PCD entries. Some are static in nature and are fixed in the final binary image at build time. Other PCD entries are dynamic and are used for configuration settings and are exposed in the Boot Setting File (BSF) file for use in the Intel® Boot Loader Development Kit (Intel® BLDK) Development Application.

Fixed PCDs

PcdsFeatureFlag

"Feature Flag" PCD entries are Boolean values that enable or disable a feature flag. These PCD entries are fixed at build time and are not included in the BSF file since they cannot be modified in the binary image.

The format of the PCD is defined in the DSC Spec as follows:
PcdTokenSpaceGuidCName.PcdCName|Value

Shown below is an example from the Crown Bay project. In this example, the CPU EIST feature flag is set to TRUE.
gEfiCpuTokenSpaceGuid.PcdCpuEistFlag|TRUE

The following is an example of the definition in the AutoGen.c file for CPU EIST feature flag PCD entry:

```
GLOBAL_REMOVE_IF_UNREFERENCED const BOOLEAN
_gPcd_FixedAtBuild_PcdCpuEistFlag = _PCD_VALUE_PcdCpuEistFlag;
```

The following is an example of the definition in the AutoGen.h file for CPU EIST feature flag PCD entry:

```
#define _PCD_TOKEN_PcdCpuEistFlag  489
#define _PCD_VALUE_PcdCpuEistFlag  ((BOOLEAN)1)
extern const  BOOLEAN  _gPcd_FixedAtBuild_PcdCpuEistFlag;
#define _PCD_GET_MODE_BOOL_PcdCpuEistFlag
_gPcd_FixedAtBuild_PcdCpuEistFlag
```

PcdsFixedAtBuild

"Fixed at Build" PCD entries cannot be changed in the IDE, since these PCD entries are fixed at build time. The PCD value is essentially a constant and a copy of the constant exists in each module that uses it.

The format of the PCD entry is defined in the DSC Spec as follows. If the DatumType is VOID *, then the MaximumDatumSize must be defined.

PcdCName|PcdTokenSpaceGuidCName|Value[|DatumType[|MaximumDatumSize or Token]

Shown below is an example from the Crown Bay project. In this example, the CPU AP Stack Size PCD entry value is set to 0x8000. The DatumType is defined to be UINT32 and the uniquely assigned token number is 0x30000003. The token is auto-generated so the token field is not necessary in this example.
gEfiCpuTokenSpaceGuid.PcdCpuApStackSize|0x8000|UINT32|0x30000003

The following is an example of the definition in the AutoGen.c file for the CPU AP Stack Size PCD entry:

```
GLOBAL_REMOVE_IF_UNREFERENCED const UINT32
_gPcd_FixedAtBuild_PcdCpuApStackSize =
_PCD_VALUE_PcdCpuApStackSize;
```

The following is an example of the definition in the AutoGen.h file for the CPU AP Stack Size PCD entry:

```
#define _PCD_TOKEN_PcdCpuApStackSize  485
#define _PCD_VALUE_PcdCpuApStackSize  0x8000
extern const  UINT32  _gPcd_FixedAtBuild_PcdCpuApStackSize;
#define _PCD_GET_MODE_32_PcdCpuApStackSize
_gPcd_FixedAtBuild_PcdCpuApStackSize
```

PcdsPatchableInModule

"Patchable in Module" PCD entries cannot be changed in the Intel® BLDK Development Application; however, there is a patch tool that can be used to modify the value of the PCD entries in the PE/COFF image. The PatchPcdValue tool is used to patch the PCD value. The GenPatchPcdTable tool is used to get the patchable PCD offset for the EFI image by parsing the map file. Both of these tools are available in the BaseTools Package available at http://sourceforge.net/apps/mediawiki/tianocore/index.php?title=BaseTools.

The "Patchable in Module" PCD entries can also be modified at runtime by modules. The format is the same as the FixedAtBuild PCDs. The PCD entry value is saved in the data segment of each module. The "Patchable in Module" PCD entries can have different values for the same PCD defined in the different modules.

If a module uses the "Patchable in Module" PCD type, then the module needs the library instance to produce the LibPatchPcdSetPtr() interface.

The format of the "Patchable in Module" PCD entry is the same as the "Fixed At Build" PCD entry as defined in the DSC specification. The following is an example from the Crown Bay project and illustrates the definition in the DEC file for the Debug Print Error Level PCD entry.

```
gEfiMdePkgTokenSpaceGuid.PcdDebugPrintErrorLevel|0x80000000|UINT3
2|0x00000006
```

The following is an example of the definition in the AutoGen.h file for the Debug Print Error Level PCD entry:

```
#define _PCD_TOKEN_PcdDebugPrintErrorLevel  420
extern UINT32 _gPcd_BinaryPatch_PcdDebugPrintErrorLevel;
#define _PCD_GET_MODE_32_PcdDebugPrintErrorLevel
_gPcd_BinaryPatch_PcdDebugPrintErrorLevel
#define _PCD_SET_MODE_32_PcdDebugPrintErrorLevel(Value)
(_gPcd_BinaryPatch_PcdDebugPrintErrorLevel = (Value))
```

Dynamic PCDs

PcdsDynamicDefault and PcdsDynamicExDefault

"Dynamic Default" and "Dynamic EX Default" PCD entries cannot be changed in the Intel® BLDK Development Application. The default value will be stored in flash and will be copied into memory; these PCD entries can only be changed at runtime and the changed values will be lost after the boot time memory is turned off.

The format of the PCD entry is defined in the DSC Spec as follows. If the DatumType is VOID *, then the MaximumDatumSize must be defined.
PcdCName|PcdTokenSpaceGuidCName|Value[|DatumType[|MaximumDatumSize or Token]

Shown below is an example from the Crown Bay project. In this example, the CPU Processor Feature Capability PCD entry value is set to 0. The DatumType is defined to be UINT32. The token number is defined as 0x40000002, although the token is auto-generated. A value is required for the Token field; if it is not defined, then the the build will fail.
[PcdsDynamic]
gEfiCpuTokenSpaceGuid.PcdCpuProcessorFeatureCapability|0|UINT32|0x40000002

The following is an example of the definition in the AutoGen.h file for a "Dynamic Default" or Dynamic PCD entry:

```
#define _PCD_TOKEN_PcdCpuProcessorFeatureCapability  14
#define _PCD_GET_MODE_32_PcdCpuProcessorFeatureCapability
LibPcdGet32(_PCD_TOKEN_PcdCpuProcessorFeatureCapability)
#define _PCD_SET_MODE_32_PcdCpuProcessorFeatureCapability(Value)
LibPcdSet32(_PCD_TOKEN_PcdCpuProcessorFeatureCapability, (Value))
```

PcdsDynamicHii and PcdsDynamicExHii

"Dynamic HII" PCD entries cannot be changed in the Intel® BLDK Development Application. These PCD entries use the HII database and are stored in the flash. If the EFI variable is not already defined, then a new non-volatile variable will be created and the default value specified in the DSC file for this PCD entry will be used.

The format of the PCD is defined in the DSC Spec as follows.
PcdTokenSpaceGuidCName.PcdTokenName|VariableName|VariableGuid \ | VariableOffset | HiiDefaultValue [|MaximumDatumSize]

Shown below is an example from the Crown Bay project. In this example, the Platform Boot Time Out PCD entry has the HII Variable Name set to L"Timeout", the Variable Guid is set to gEfi GlobalVariableGuid, the HII Variable Offset is 0, and the HII Default Value is 5.

```
[PcdsDynamicHii.common.DEFAULT]
gEfiIntelFrameworkModulePkgTokenSpaceGuid.PcdPlatformBootTimeOut|
L"Timeout"|gEfiGlobalVariableGuid|0x0|5
```

The following is an example of the definition in the AutoGen.h file for a "Dynamic HII" PCD entry:

```
#define _PCD_TOKEN_PcdPlatformBootTimeOut  7
#define _PCD_GET_MODE_16_PcdPlatformBootTimeOut
LibPcdGet16(_PCD_TOKEN_PcdPlatformBootTimeOut)
#define _PCD_SET_MODE_16_PcdPlatformBootTimeOut(Value)
LibPcdSet16(_PCD_TOKEN_PcdPlatformBootTimeOut, (Value))
```

PcdsDynamicVpd and PcdsDynamicExVpd

"Dynamic VPD" PCD entries are the only PCD entries that can be changed in the Intel® BLDK Development Application and are stored in the VPD database (which is stored in flash). These PCD entries are read-only and can't be modified at runtime. Accessing these PCD entries will be redirected to the flash VPD region.

The format of the PCD is defined in the DSC Spec as follows.

```
PcdTokenSpaceGuidCName.PcdTokenName|VpdOffset[|<MaxDatumSize> |
<Token>]
```

Shown below is an example from the Crown Bay project. In the example below, PcdIgdPreAllocSize is at VpdOffset 4 and has an initial value of 2.

```
[PcdsDynamicVpd.common.DEFAULT]
gEfiE6xxTokenSpaceGuid.PcdIgdPreAllocSize | 4 | 0x02
```

In the below example, PcdTCPcieRootPortConfiguration is at Vpd offset 0x1bd, has a maximum size of 16 and the initial Vpd value is in the between the brackets.

```
gEfiE6xxTokenSpaceGuid.PcdTCPcieRootPortConfiguration | 0x1bd |
16 | {0x00, 0x00, 0x01, 0x00, 0x00, 0x01, 0x02, 0x00, 0x00, 0x01,
0x03, 0x00, 0x00, 0x01, 0x04, 0x00}
```

The following is an example of the definition in the AutoGen.h file for a "Dynamic VPD" PCD entry:

```
#define _PCD_TOKEN_PcdTCPcieRootPortConfiguration  5
#define _PCD_GET_MODE_PTR_PcdTCPcieRootPortConfiguration
LibPcdGetPtr(_PCD_TOKEN_PcdTCPcieRootPortConfiguration)
#define
_PCD_SET_MODE_PTR_PcdTCPcieRootPortConfiguration(SizeOfBuffer,
Buffer)  LibPcdSetPtr(_PCD_TOKEN_PcdTCPcieRootPortConfiguration,
(SizeOfBuffer), (Buffer))
```

PcdsDynamic vs PcdsDynamicEx

If a module is released in source code and it will be built with the platform DSC file, then the dynamic PCD entries used by this module can be accessed

as PcdGetxx(PcdSampleDynamicPcd). When building the platform, the build tools will translate PcdSampleDynamicPcd to the Token Space GUID and the Token Number for this PCD entry.

If a module is released as a binary image and will not be part of the platform build, then the dynamic PCD entry used by the binary module must be accessed as PcdGetxxEx(gEfiMyTokenspaceGuid, PcdSampleDynamicPcd). In this case, the developer needs to explicitly give the Token Space GUID and the Token Number as a parameter when the source code is written.

The token field is only used for "Dynamic Ex" PCD entries since the Token Number must be specified as a parameter when accessing the PCD entry.

The format of the PCD entry is defined the same as "Dynamic Default" and "Dynamic EX Default" PCD entries.

PCD Usage

Multiple PCD Type Definitions

Some DEC files define multiple PCD types, but only one type can be used and will be chosen by the platform DSC file.

The MdePkg.dec file has the following definition:
[PcdsFixedAtBuild,PcdsPatchableInModule,PcdsDynamic]

gEfiMdePkgTokenSpaceGuid.PcdPciExpressBaseAddress|0xE0000000|UINT64|0x0000000a

The CrownBayPlatform.dsc file has the following definition:

[PcdsFixedAtBuild]
gEfiMdePkgTokenSpaceGuid.PcdPciExpressBaseAddress|0xE0000000

The definition in the IntelFrameworkPkg.dsc follows:

```
IntelFrameworkPkg.dsc
[PcdsPatchableInModule]
  gEfiMdePkgTokenSpaceGuid.PcdDebugPrintErrorLevel|0x80000000
```

This example illustrates that the type and definition of the same PCD is different in different modules and is selectable by the DSC file.

References

- EDK II DSC File Specification v1.22 http://sourceforge.net/projects/edk2/files/
- EDK II DEC File Specification v1.22 http://sourceforge.net/projects/edk2/files/
- UDK2010.UP3, MdeModulePkg/Universal/PCD/Dxe/Pcd.inf http://sourceforge.net/apps/mediawiki/tianocore/index.php?title=UDK2010

Conclusion

This white paper was intended to clarify the types and definitions for the Platform Configuration Database entries.

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. http://www.intel.com/p/en_US/embedded.

Author

Jenny M Pelner is a Firmware Architect with the Embedded and Communications Group at Intel Corporation.

Acronyms

Intel® BLDK	Intel® Boot Loader Development Kit
BSF	Boot Setting File
DEC	EDKII Declaration File
DSC	EDKII Platform Description File
EDC	Embedded Design Center
EFI	Extensible Firmware Interface
EIST	Enhanced Intel SpeedStep® Technology
EDKII	EFI Development Kit II
GUID	Globally Unique Identifier
HII	Human Interface Infrastructure
PCD	Platform Configuration Data
VPD	Vital Product Data

§