tianocore

UEFI & EDK II Training

EDK II Build Process and Environment

tianocore.org

tianocore

LESSON OBJECTIVE

- Define EDK II
- Describe EDK II's elements including file extensions, rectories, modules, packages, and libraries
- Explain the EDK II build process
- Explain the Build tools

EDK II OVERVIEW

The Edk II Infrastructure

PHILOSOPHY OF EDK II

Support UEFI & PI needs

PHILOSOPHY OF EDK II

Support UEFI & PI needs

Separate tool & source code

PHILOSOPHY OF EDK II

Support UEFI & PI needs

Separate tool & source code

Package Definition File: DEC

PHILOSOPHY OF EDK II

Support UEFI & PI needs

Separate tool & source code

Package Definition File: DEC

FLASH Mapping Tool

PHILOSOPHY OF EDK II

Support UEFI & PI needs

Separate tool & source code

Package Definition File: DEC

FLASH Mapping Tool

Move as much code to C

PHILOSOPHY OF EDK II

Support UEFI & PI needs

Separate tool & source code

Package Definition File: DEC

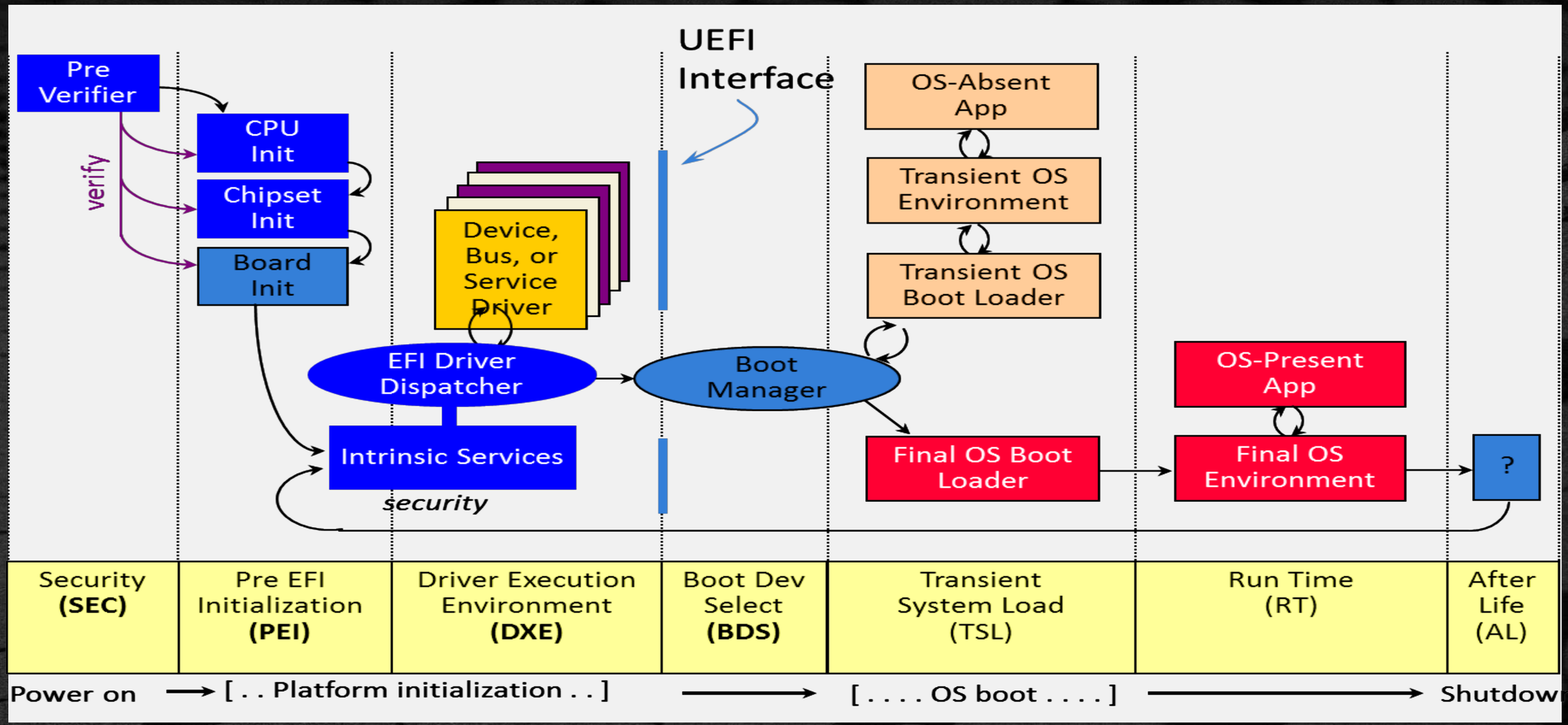
FLASH Mapping Tool

Move as much code to C

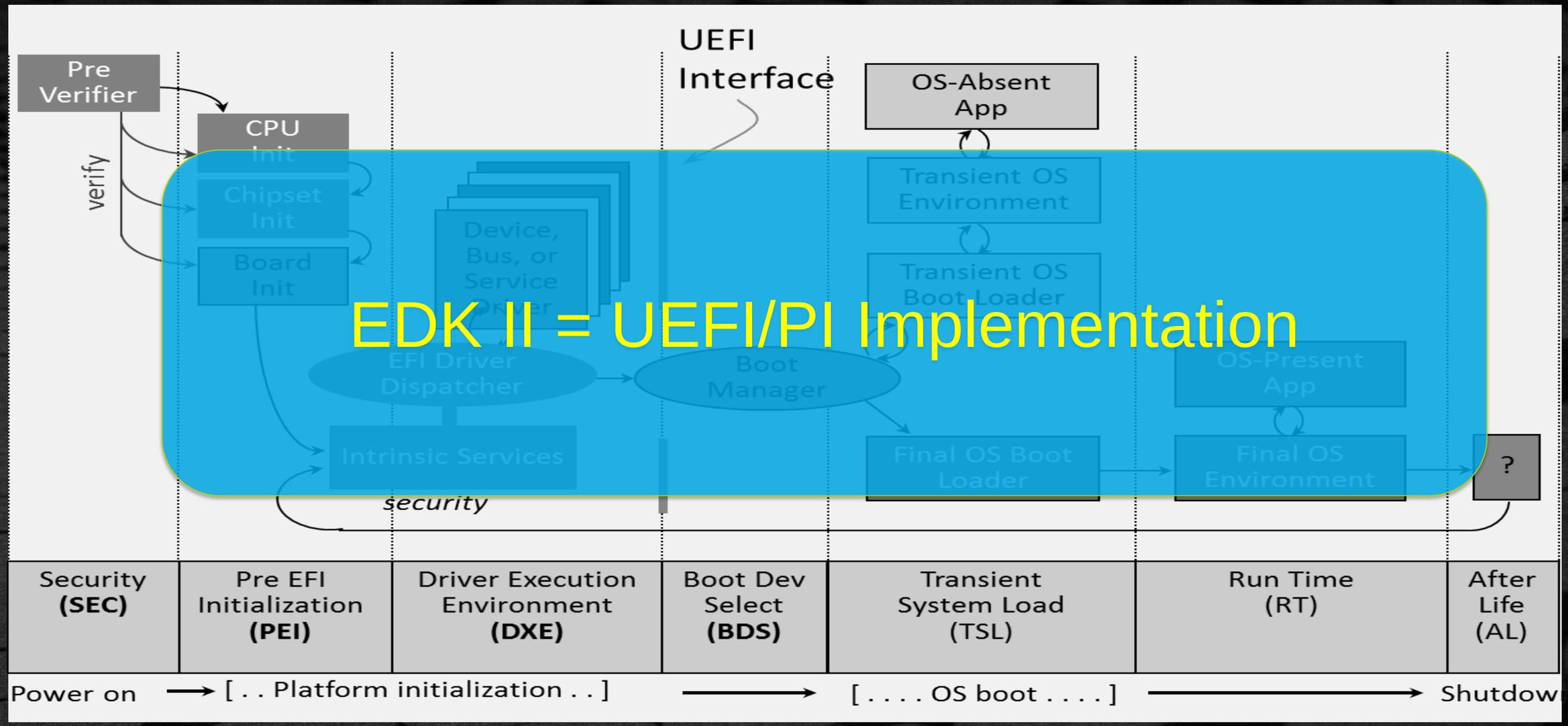
Open source EDK II on

tianocore.org

IMPLEMENTATION OF EDK II



IMPLEMENTATION OF EDK II



FILE EXTENSIONS

- Located on tianocore.org project edk2

.DSC file - Platform Description

.DEC file - Package Declaration

.INF file - Module Definition (define a component)

.FDF file - Flash Description File

FILE EXTENSIONS

- Located on tianocore.org project edk2

.DSC file - Platform Description

.DEC file - Package Declaration

.INF file - Module Definition (define a component)

.FDF file - Flash Description File

.VFR file - Visual Forms Representation for User interface

.UNI file - String text file for ease of localization

.c & . h files - Source code files

.DXS file - Dependency expression file – now [DEPEX]

.FV file - Firmware Volume image file

FILE EXTENSIONS

- Located on tianocore.org project edk2

.DSC file - Platform Description

.DEC file - Package Declaration

.INF file - Module Definition (define a component)

.FDF file - Flash Description File

EDK II Spec

.VFR file - Visual Forms Representation for User interface

.UNI file - String text file for ease of localization

.c & . h files - Source code files

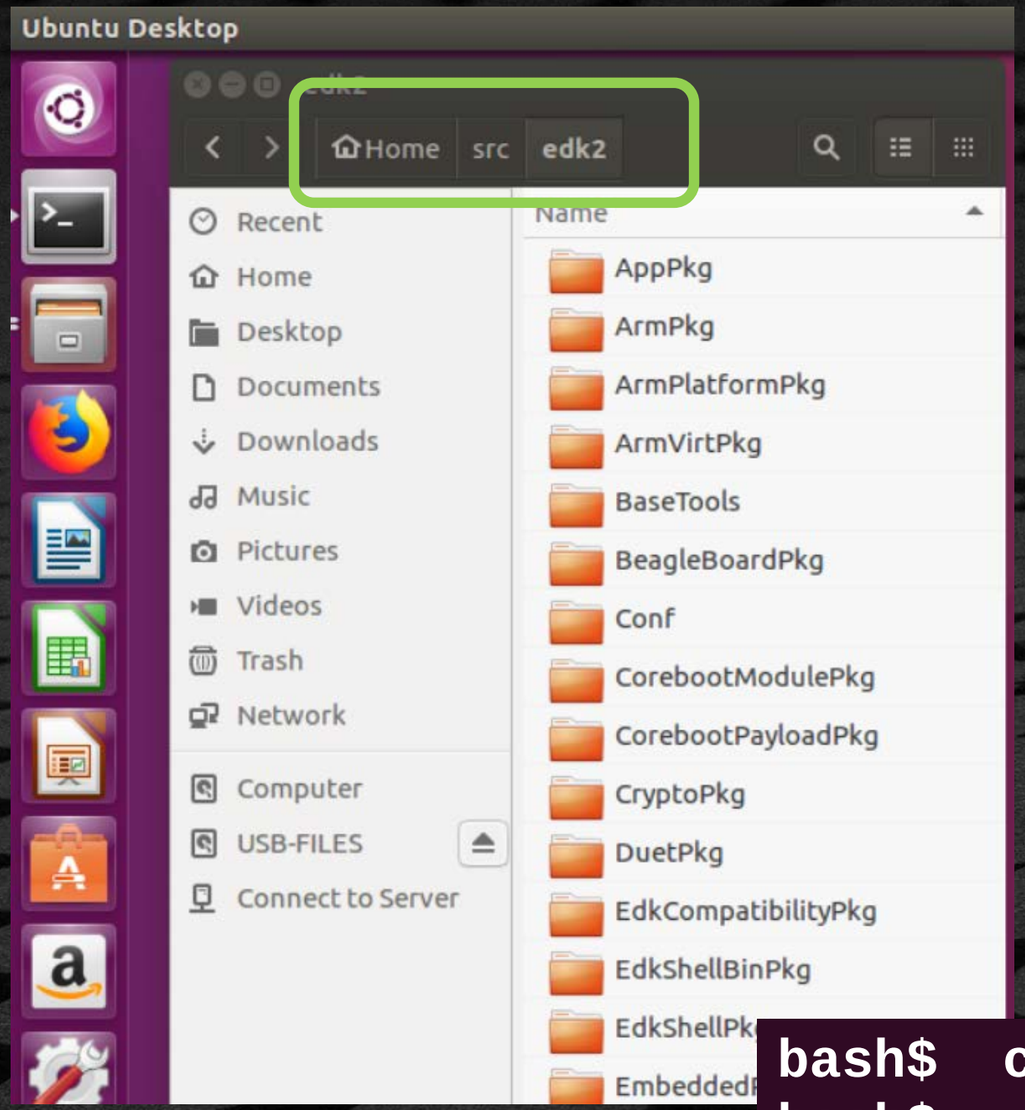
Source

.DXS file - Dependency expression file – now [DEPEX]

.FV file - Firmware Volume image file

Output

EDK II DIRECTORY STRUCTURE



- Package concept for each EDK II sub-directory
- Platforms are contained in an EDK II package
- EDK II build process reflects the package
- Concept of “Work Space”

$HOME/src/edk2

```
bash$   cd $HOME/src/edk2
bash$   . edksetup.sh
bash$   build
```

DIRECTORY STRUCTURE- REAL PLATFORM

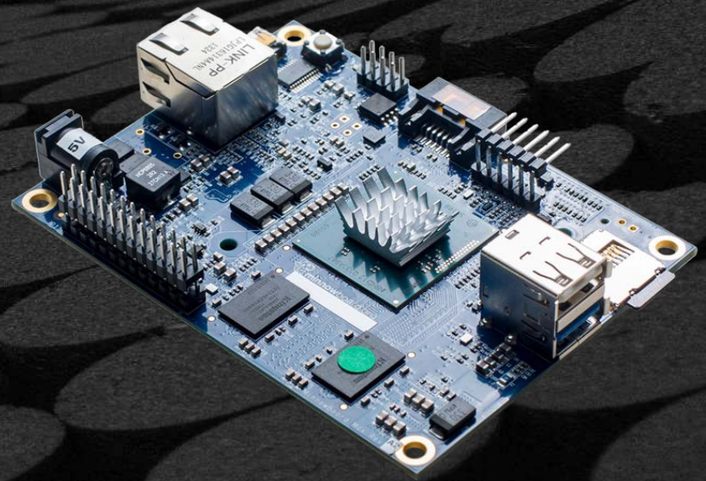
Open Source Directories

```
MyWorkSpace/
    BaseTools/
    Conf/
    CryptoPkg/
    EdkCompatibilityPkg/
    FatBinPkg/
    IntelFrameworkModulePkg/
    IntelFrameworkPkg/
    IntelFsp2Pkg
    MdeModulePkg/
    MdePkg/
    NetworkPkg/
    Nt32Pkg
    OptionRomPkg/
    OvmPkg
    PcAtChipsestPkg/
    ShellBinPkg/
    Etc.
```

DIRECTORY STRUCTURE- REAL PLATFORM

Open Source Directories

```
MyWorkSpace/
    BaseTools/
    Conf/
    CryptoPkg/
    EdkCompatibilityPkg/
    FatBinPkg/
    IntelFrameworkModulePkg/
    IntelFrameworkPkg/
    IntelFsp2Pkg
    MdeModulePkg/
    MdePkg/
    NetworkPkg/
    Nt32Pkg
    OptionRomPkg/
    OvmPkg
    PcAtChipsestPkg/
    ShellBinPkg/
    Etc.
```



minnowboard.org

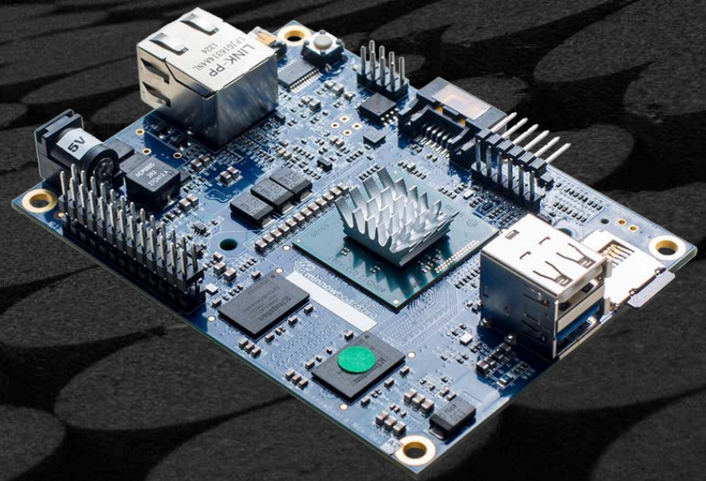
DIRECTORY STRUCTURE- REAL PLATFORM

Open Source Directories

```
MyWorkSpace/
    BaseTools/
    Conf/
    CryptoPkg/
    EdkCompatibilityPkg/
    FatBinPkg/
    IntelFrameworkModulePkg/
    IntelFrameworkPkg/
    IntelFsp2Pkg
    MdeModulePkg/
    MdePkg/
    NetworkPkg/
    Nt32Pkg
    OptionRomPkg/
    OvmPkg
    PcAtChipsestPkg/
    ShellBinPkg/
    Etc.
```

Platform and Silicon Directories

```
IA32FamilyCpuPkg/
Vlv2DeviceRefCodePkg /
Vlv2TbltMiscPkg /
Vlv2TbltDevicePkg /
```



minnowboard.org

DIRECTORY STRUCTURE- REAL PLATFORM

Open Source Directories

```
MyWorkSpace/
     Build/
     edk2/
       - "edk2 Open Source"
       - . . .
     edk2-platforms/
       Vlv2DeviceRefCodePkg/
          AcpiTablesPCAT/
          Include/
          ValleyView2Soc/
               CPU/
               NorthCluster/
               SouthCluster
       Vlv2TbltDevicePkg/
     Silicon/
       IA32FamilyCpuPkg/
       Vlv2BinaryPkg
       Vlv2MiscBinariesPkg/
```

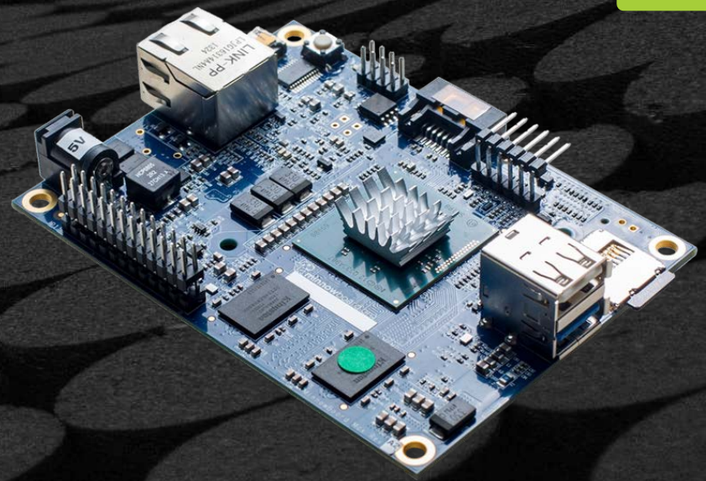
Platform and Silicon Directories

Key

Open Source

Silicon/Chipset

Platform



minnowboard.org

MODULES

Modules: Building blocks of EDK II

MODULES

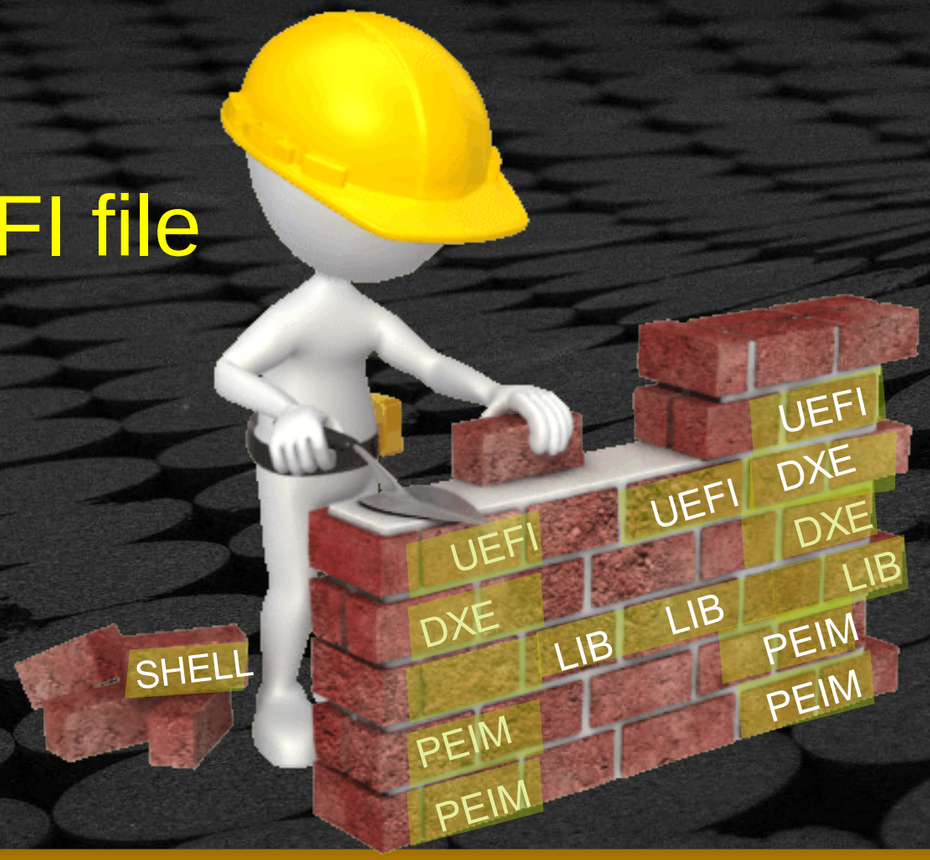
Smallest separate object compiled in EDK II

Modules: Building blocks of EDK II

MODULES

Smallest separate object compiled in EDK II

Compiles to .EFI file



Modules: Building blocks of EDK II

MODULES

Smallest separate object compiled in EDK II

Compiles to .EFI file

UEFI/DXE Driver

PEIM

UEFI App.

Library

Modules: Building blocks of EDK II

PACKAGES

- EDK II projects are made up of packages
- Make your own packages
- Package contains only the necessities
- Remove packages from projects when not required

EDK II PACKAGE EXAMPLES: SPECS

MdePkg

Include files and libraries for Industry Standard Specifications

EDK II PACKAGE EXAMPLES: SPECS

MdePkg

Include files and libraries for Industry Standard Specifications

MdeModulePkg

Modules only definitions from the Industry Standard Specification are defined in the MdePkg

EDK II PACKAGE EXAMPLES: SPECS

MdePkg

Include files and libraries for Industry Standard Specifications

MdeModulePkg

Modules only definitions from the Industry Standard Specification are defined in the MdePkg

IntelFrameworkPkg

Include files and libraries for those parts of the Intel Platform Innovation Framework for EFI specifications not adopted “as is” by the UEFI or PI specifications

EDK II PACKAGE EXAMPLES: SPECS

MdePkg

Include files and libraries for Industry Standard Specifications

MdeModulePkg

Modules only definitions from the Industry Standard Specification are defined in the MdePkg

IntelFrameworkPkg

Include files and libraries for those parts of the Intel Platform Innovation Framework for EFI specifications not adopted “as is” by the UEFI or PI specifications

IntelFrameworkModulePkg

Contains modules that make reference to one or more definitions in the IntelFrameworkPkg

EDK II ADDITIONAL PACKAGES

- Platforms — `Nt32Pkg & OvmfPkg`

EDK II ADDITIONAL PACKAGES

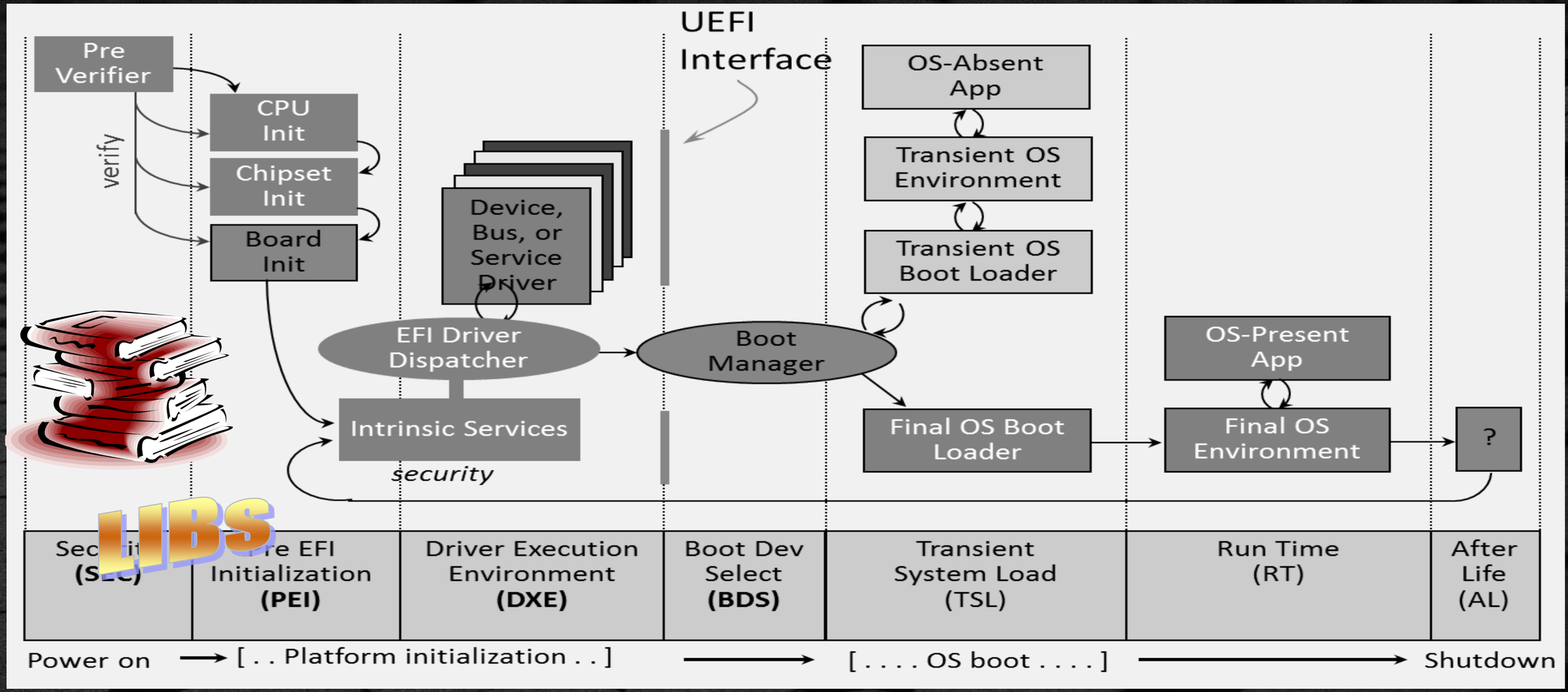
- Platforms — **Nt32Pkg & OvmfPkg**
- Chipset/Processor — **IA32FamilyCpuPkg & BroxtonSiPkg**

EDK II ADDITIONAL PACKAGES

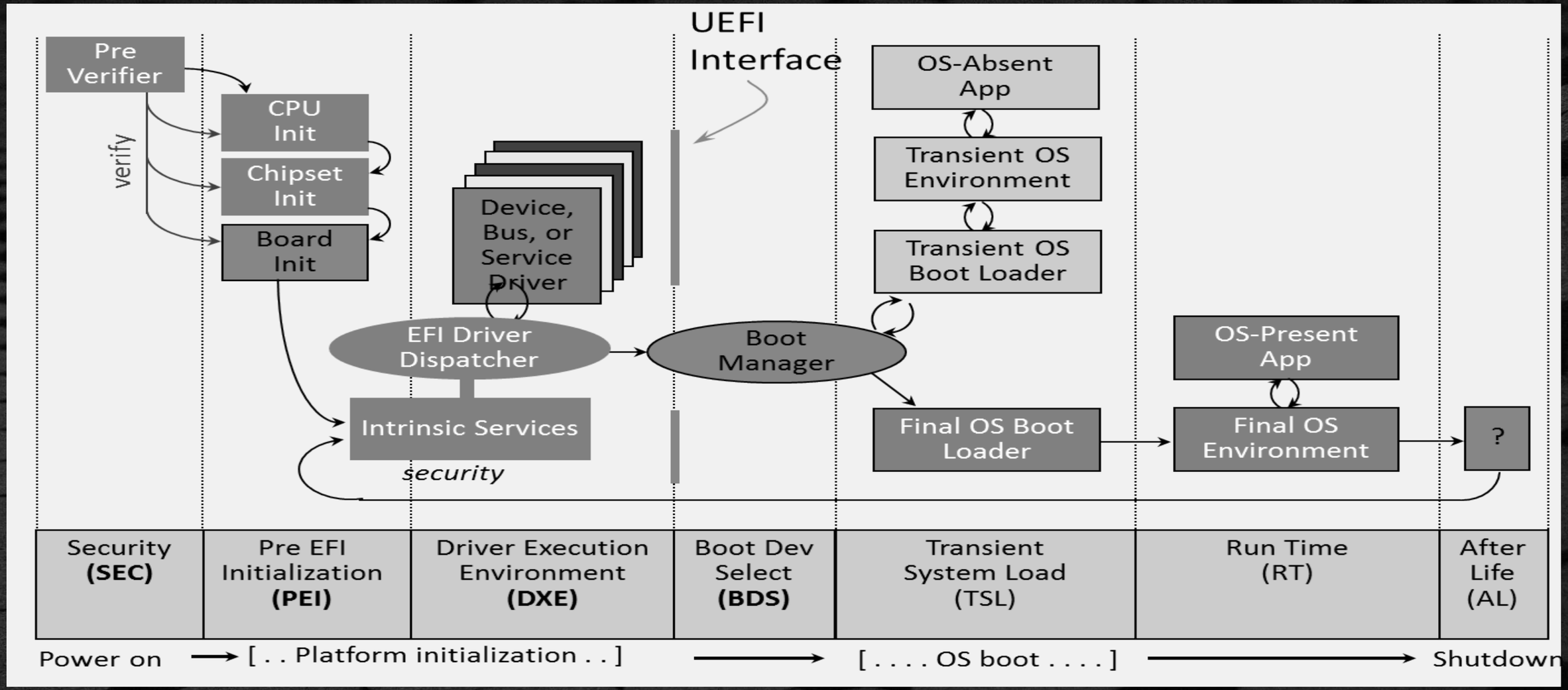
- Platforms — **Nt32Pkg & OvmfPkg**
- Chipset/Processor — **IA32FamilyCpuPkg & BroxtonSiPkg**
- Functionality — **ShellPkg & NetworkPkg**

tianocore

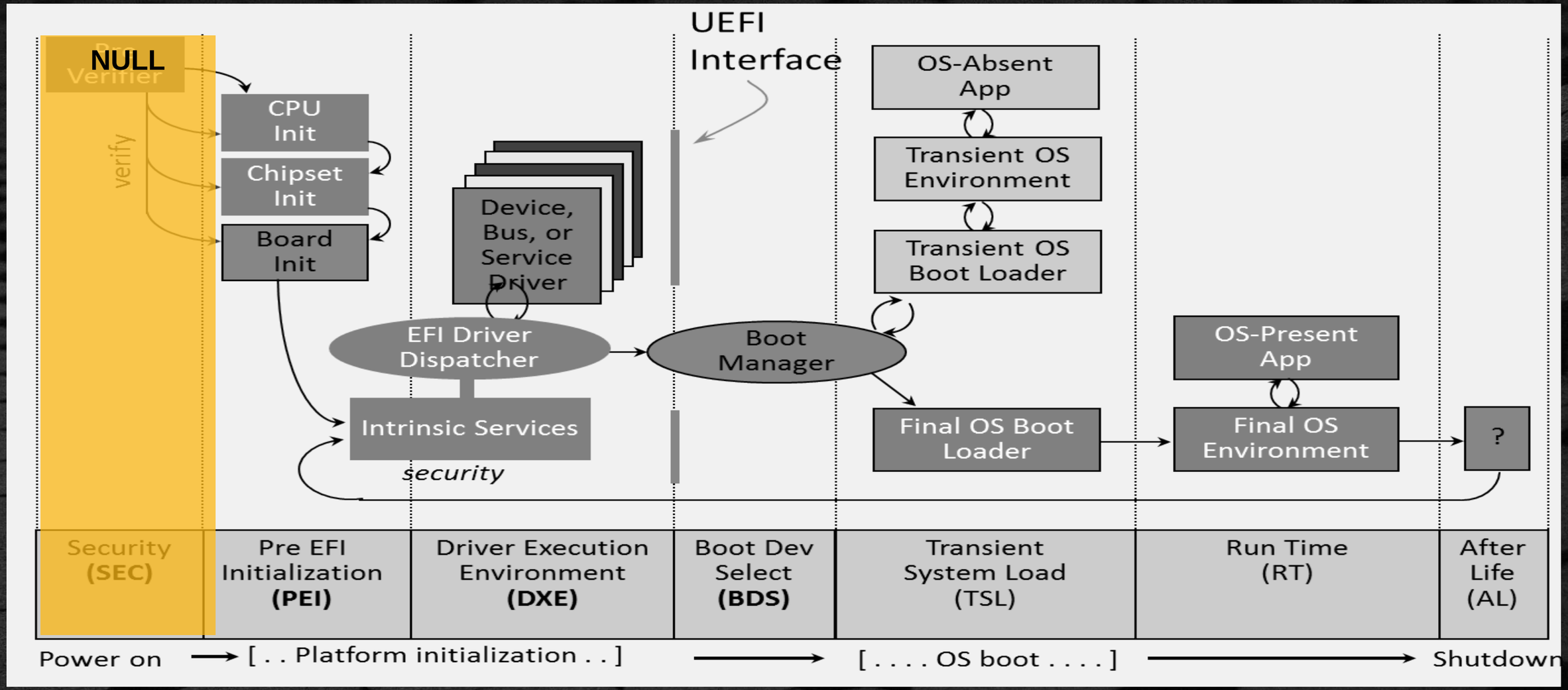
LIBRARIES



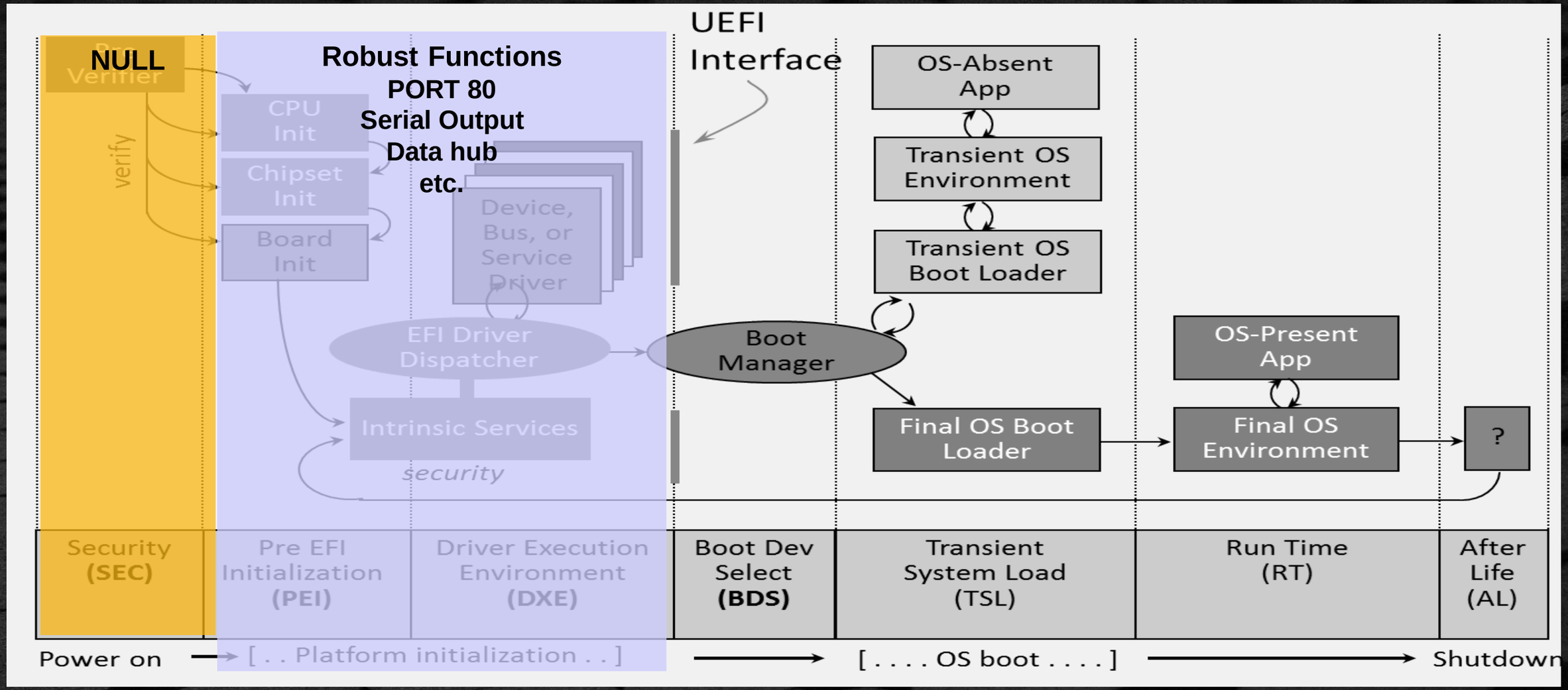
EXAMPLE – LIBRARY "DEBUGLIB"



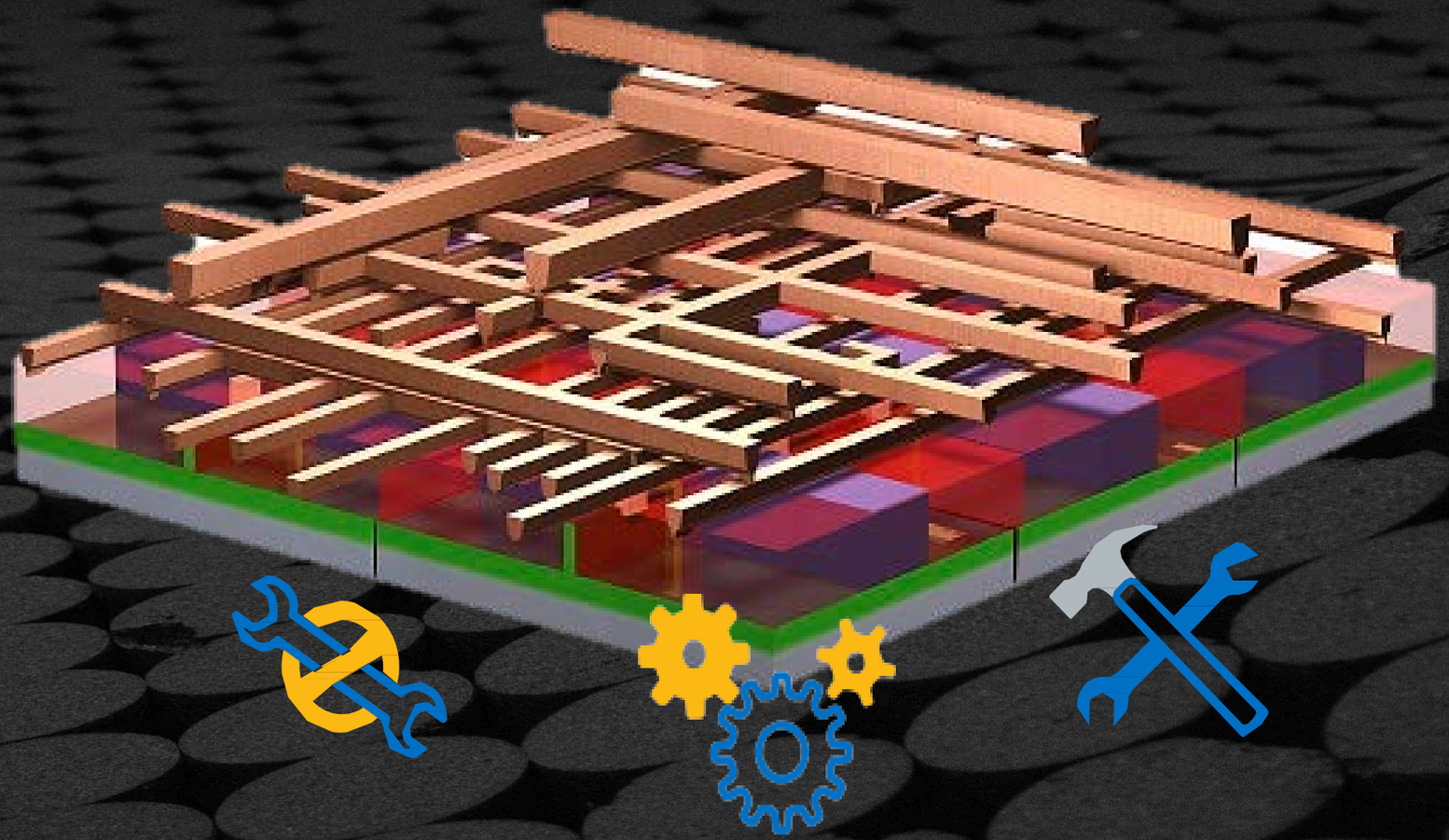
EXAMPLE – LIBRARY "DEBUGLIB"



EXAMPLE – LIBRARY "DEBUGLIB"



PLATFORM CONFIGURATION DATABASE (PCD)



PLATFORM CONFIGURATION DATABASE (PCD)

Goals

Define module parameters

- Store module / platform configurations

PLATFORM CONFIGURATION DATABASE (PCD)

Goals

Define module parameters	• Store module / platform configurations
Reduce edits of source code	• Maximize module reuse across platforms

PLATFORM CONFIGURATION DATABASE (PCD)

Goals

Define module parameters	• Store module / platform configurations
Reduce edits of source code	• Maximize module reuse across platforms
Remove #define	• No searching for “magic” **`#define`** statements

PLATFORM CONFIGURATION DATABASE (PCD)

Goals

Define module parameters	• Store module / platform configurations
Reduce edits of source code	• Maximize module reuse across platforms
Remove #define	• No searching for “magic” **`#define`** statements
API functions	• Get and Set functions for access to PCD variable DB

PLATFORM CONFIGURATION DATABASE (PCD)

Advantages

Binary Modularity

- Offers a way to configure firmware settings by patching binaries without building

PLATFORM CONFIGURATION DATABASE (PCD)

Advantages

Binary Modularity	• Offers a way to configure firmware settings by patching binaries without building
Configure	• Provide for options to configure features

PLATFORM CONFIGURATION DATABASE (PCD)

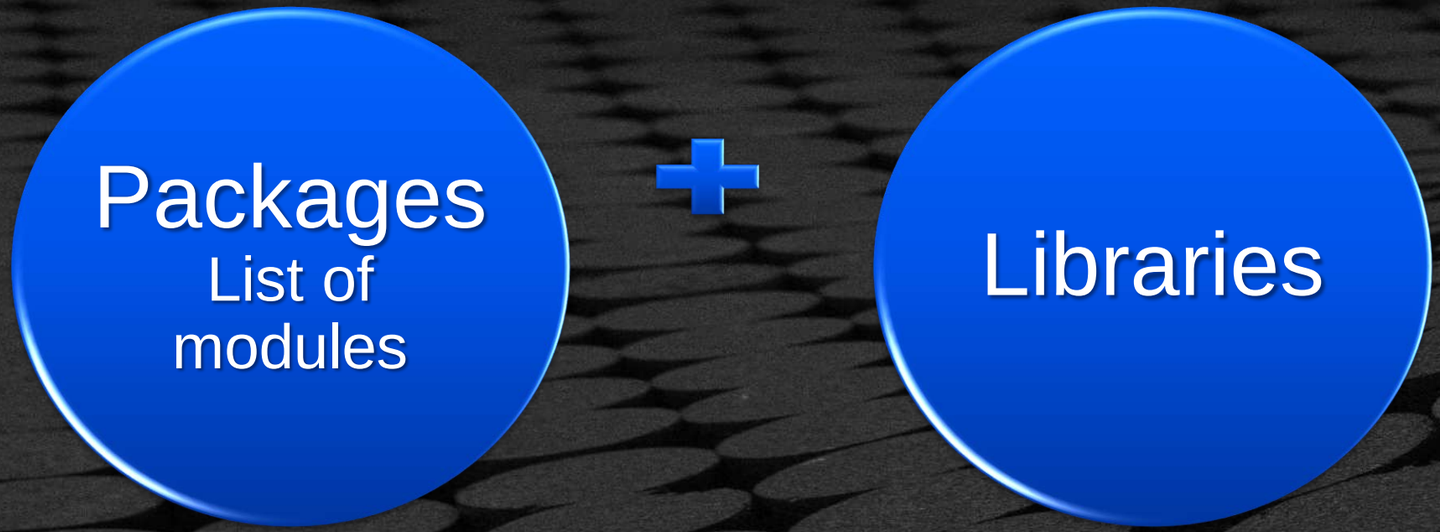
Advantages

Binary Modularity	• Offers a way to configure firmware settings by patching binaries without building
Configure	• Provide for options to configure features
Patching	• Simplify the patching process

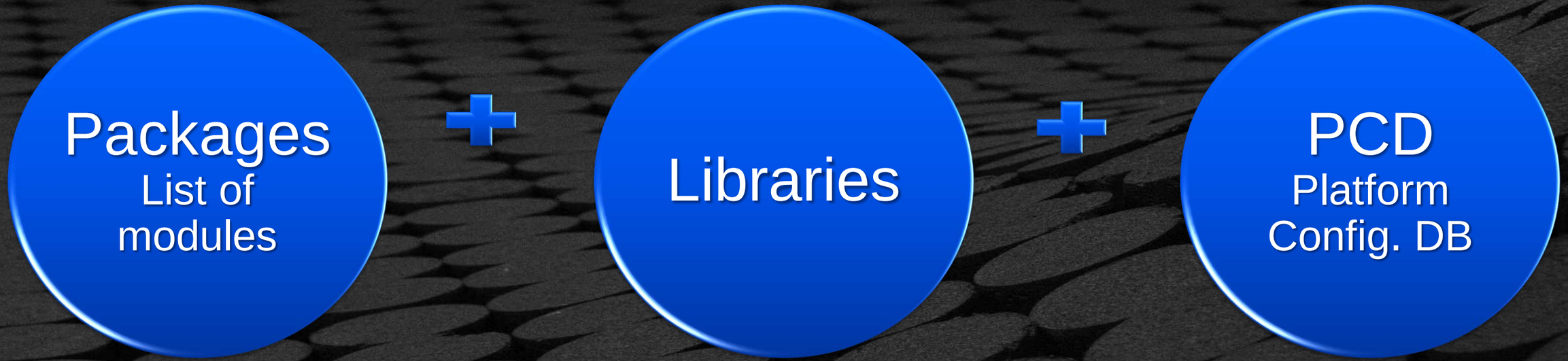
EDK II INFRASTRUCTURE SUMMARY

Packages
List of modules

EDK II INFRASTRUCTURE SUMMARY



EDK II INFRASTRUCTURE SUMMARY



EDK II VS. UDK (2010| 2017 .. 2018)

UEFI Developer's Kit 2018 (UDK2018)

Stable build of the EDK II project

Neither contain Intel silicon or platform code

wiki on tianocore.org Differences between UDK - EDK II

BUILD TOOLS

EDK Build Tools and Configuration Files

DEVELOPMENT ENVIRONMENT

Compiler Tool Chains

- Microsoft Visual Studio (VS2015, VS2013, VS2012, VS2010, etc.)
- Microsoft WDK
- Intel C/C++ compiler
- Intel C EFI Byte Code (EBC) compiler
- GCC V5.x or later

Operating Systems

- Microsoft Windows XP/7/8/10
- Apple Mac OS X
- RedHat Enterprise Linux
- Novell SuSE Linux
- Ubuntu

ENVIRONMENT VARIABLES

Set by **`edksetup`**

Windows = .bat

Linux = .sh

1. EDK_TOOLS_PATH
2. PATH
3. WORKSPACE
4. PACKAGES_PATH *(optional)*
5. EFI_SOURCE/EDK_SOURCE

CONFIGURATION FILES - SCRIPTS

1. **edksetup.bat** or **edksetup.sh**

```
bash@usid:~/src/edk2
bash@usid:~/src/edk2$ . edksetup.sh
```

CONFIGURATION FILES - SCRIPTS

1. **edksetup.bat** or **edksetup.sh**

```
bash@usid:~/src/edk2
bash@usid:~/src/edk2$ . edksetup.sh
```

2. First time use will set up configuration files:
   - Conf/build_rule.txt
   - Conf/target.txt
   - Conf/tools_def.txt

CONFIGURATION FILES - SCRIPTS

1. **edksetup.bat** or **edksetup.sh**

```
bash@usid:~/src/edk2
bash@usid:~/src/edk2$ . edksetup.sh
```

2. First time use will set up configuration files:
   - Conf/build_rule.txt
   - Conf/target.txt
   - Conf/tools_def.txt

3. Setup & verify a developer's workspace

CONFIGURATION FILES - SCRIPTS

1. **edksetup.bat** or **edksetup.sh**

```
bash@usid:~/src/edk2
bash@usid:~/src/edk2$ . edksetup.sh
```

2. First time use will set up configuration files:
   - Conf/build_rule.txt
   - Conf/target.txt
   - Conf/tools_def.txt
3. Setup & verify a developer's workspace
4. Ensure environment variables set correctly
   - WORKSPACE, EDK_TOOLS_PATH, …

Multiple Workspace Environment Variable

PACKAGES_PATH

WORKSPACE

PACKAGES_PATH – *Optional*

Multiple paths that will be searched when attempting to resolve the location of packages.

Multiple Workspace Environment Variable

PACKAGES_PATH

WORKSPACE

PACKAGES_PATH – *Optional*

Multiple paths that will be searched when attempting to resolve the location of packages.

- Highest search Priority / Build Directory
- Additional Paths in priority order. Must be set before **edksetup** and ***not*** set by **edksetup**

Multiple Workspace Environment Variable

PACKAGES_PATH

WORKSPACE

PACKAGES_PATH – *Optional*

- Highest search Priority / Build Directory
- Additional Paths in priority order. Must be set before **edksetup** and ***not*** set by **edksetup**

Multiple paths that will be searched when attempting to resolve the location of packages.

Example:

```
set WORKSPACE=%CWD%/MyWorkspace
set \
PACKAGES_PATH=%WORKSPACE%/edk2;%WORKSPACE%/Platform;\
%WORKSPACE%/Silicon
```

Multiple Workspace Environment Variable

PACKAGES_PATH

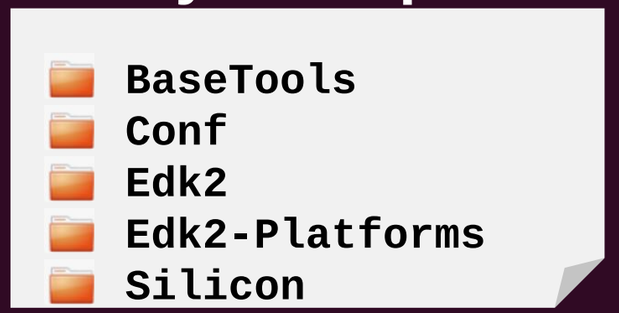
WORKSPACE

PACKAGES_PATH – *Optional*

- Highest search Priority / Build Directory
- Additional Paths in priority order. Must be set before **edksetup** and ***not*** set by **edksetup**

Multiple paths that will be searched when attempting to resolve the location of packages.

$HOME/MyWorkspace



Example:

```
set WORKSPACE=%CWD%/MyWorkspace
set \
PACKAGES_PATH=%WORKSPACE%/edk2;%WORKSPACE%/Platform;\
%WORKSPACE%/Silicon
```

Multiple Workspace Environment Variable

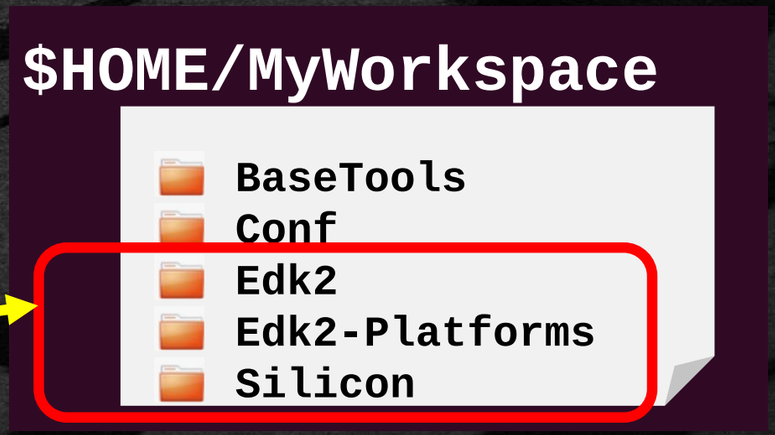
PACKAGES_PATH

WORKSPACE

PACKAGES_PATH – *Optional*

- Highest search Priority / Build Directory
- Additional Paths in priority order. Must be set before **edksetup** and ***not*** set by **edksetup**

Multiple paths that will be searched when attempting to resolve the location of packages.



Example:

```
set WORKSPACE=%CWD%/MyWorkspace
set \
PACKAGES_PATH=%WORKSPACE%/edk2;%WORKSPACE%/Platform;\
%WORKSPACE%/Silicon
```

tianocore

USING TARGET.TXT

Tag	Description
ACTIVE_PLATFORM	Pointer to DSC file being built
TARGET	Build mode: DEBUG or RELEASE
TARGET_ARCH	Build architecture (IA32, IPF, X64, EBC, ARM)
TOOL_CHAIN_CONF	Path to tools_def.txt
TOOL_CHAIN_TAG	Compiler/tool set to use, based on definitions in **tools_def.txt**
MAX_CONCURRENT_THREAD_NUMBER	Number of threads available to the build process (multi-threaded build)

USING TARGET.TXT

Tag	Description
ACTIVE_PLATFORM	Pointer to DSC file being built
TARGET	Build mode: DEBUG or RELEASE
TARGET_ARCH	Build architecture (IA32, IPF, X64, EBC, ARM)
TOOL_CHAIN_CONF	Path to tools_def.txt
TOOL_CHAIN_TAG	Compiler/tool set to use, based on definitions in **tools_def.txt**
MAX_CONCURRENT_THREAD_NUMBER	Number of threads available to the build process (multi-threaded build)

USING TARGET.TXT

Tag	Description
ACTIVE_PLATFORM	Pointer to DSC file being built
TARGET	Build mode: DEBUG or RELEASE
TARGET_ARCH	Build architecture (IA32, IPF, X64, EBC, ARM)
TOOL_CHAIN_CONF	Path to tools_def.txt
TOOL_CHAIN_TAG	Compiler/tool set to use, based on definitions in **tools_def.txt**
MAX_CONCURRENT_THREAD_NUMBER	Number of threads available to the build process (multi-threaded build)

USING TARGET.TXT

Tag	Description
ACTIVE_PLATFORM	Pointer to DSC file being built
TARGET	Build mode: DEBUG or RELEASE
TARGET_ARCH 	Build architecture (IA32, IPF, X64, EBC, ARM)
TOOL_CHAIN_CONF	Path to tools_def.txt
TOOL_CHAIN_TAG	Compiler/tool set to use, based on definitions in **tools_def.txt**
MAX_CONCURRENT_THREAD_NUMBER	Number of threads available to the build process (multi-threaded build)

USING TARGET.TXT

Tag	Description
ACTIVE_PLATFORM	Pointer to DSC file being built
TARGET	Build mode: DEBUG or RELEASE
TARGET_ARCH	Build architecture (IA32, IPF, X64, EBC, ARM)
TOOL_CHAIN_CONF	Path to tools_def.txt
TOOL_CHAIN_TAG	Compiler/tool set to use, based on definitions in `tools_def.txt`
MAX_CONCURRENT_THREAD_NUMBER	Number of threads available to the build process (multi-threaded build)

USING TARGET.TXT

Tag	Description
ACTIVE_PLATFORM	Pointer to DSC file being built
TARGET	Build mode: DEBUG or RELEASE
TARGET_ARCH	Build architecture (IA32, IPF, X64, EBC, ARM)
TOOL_CHAIN_CONF	Path to tools_def.txt
TOOL_CHAIN_TAG	Compiler/tool set to use, based on definitions in **tools_def.txt**
MAX_CONCURRENT_THREAD_NUMBER	Number of threads available to the build process (multi-threaded build)

USING TARGET.TXT

Tag	Description
ACTIVE_PLATFORM	Pointer to DSC file being built
TARGET	Build mode: DEBUG or RELEASE
TARGET_ARCH	Build architecture (IA32, IPF, X64, EBC, ARM)
TOOL_CHAIN_CONF	Path to tools_def.txt
TOOL_CHAIN_TAG	Compiler/tool set to use, based on definitions in **tools_def.txt**
MAX_CONCURRENT_THREAD_NUMBER	Number of threads available to the build process (multi-threaded build)

USING TOOLS_DEF.TXT

Paths for compilers, assemblers, and linkers

- Comes with definitions for all compilers

USING TOOLS_DEF.TXT

Paths for compilers, assemblers, and linkers

- Comes with definitions for all compilers

Only modify this file when …

- Tools are installed in a non-default location
- Different compilers/tools need to be added

USING TOOLS_DEF.TXT

Paths for compilers, assemblers, and linkers

- Comes with definitions for all compilers

Only modify this file when …

- Tools are installed in a non-default location
- Different compilers/tools need to be added

Default values are set by **`edksetup`** script

- Default values will cover most compiler needs
- If there are problems with the file after editing, just delete and re-run **`edksetup`** (restores default)

BUILD PROCESS

EDK Build Files

BUILD PROCESS OVERVIEW

tools_def.txt
target.txt

Source Files

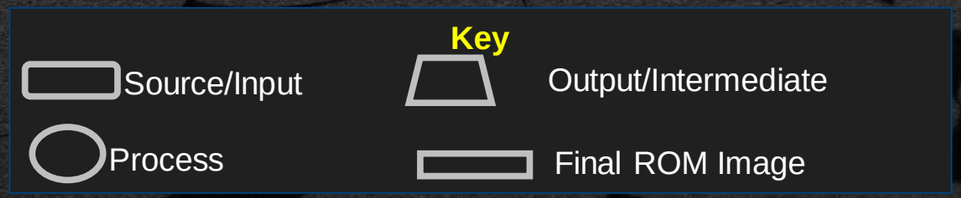
INF Files

DEC Files

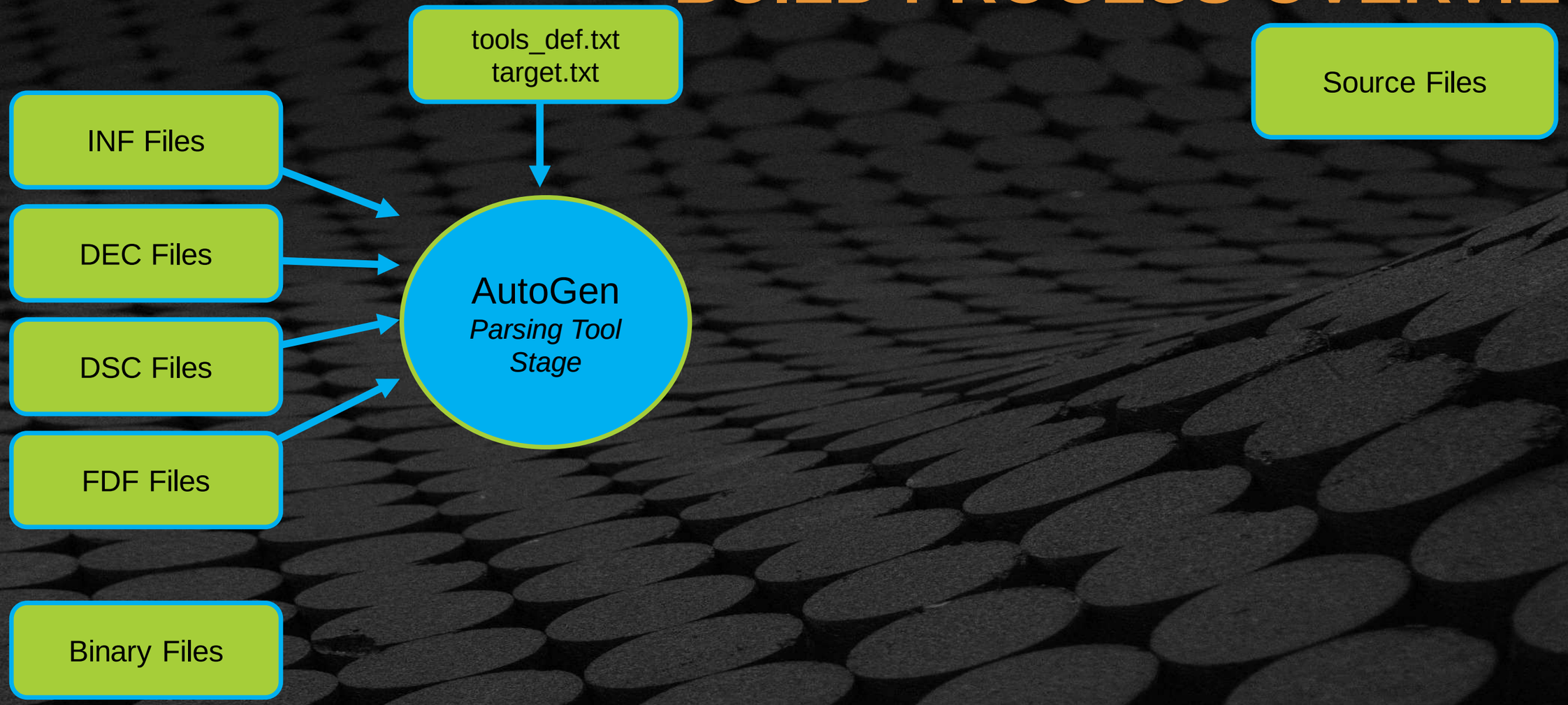
DSC Files

FDF Files

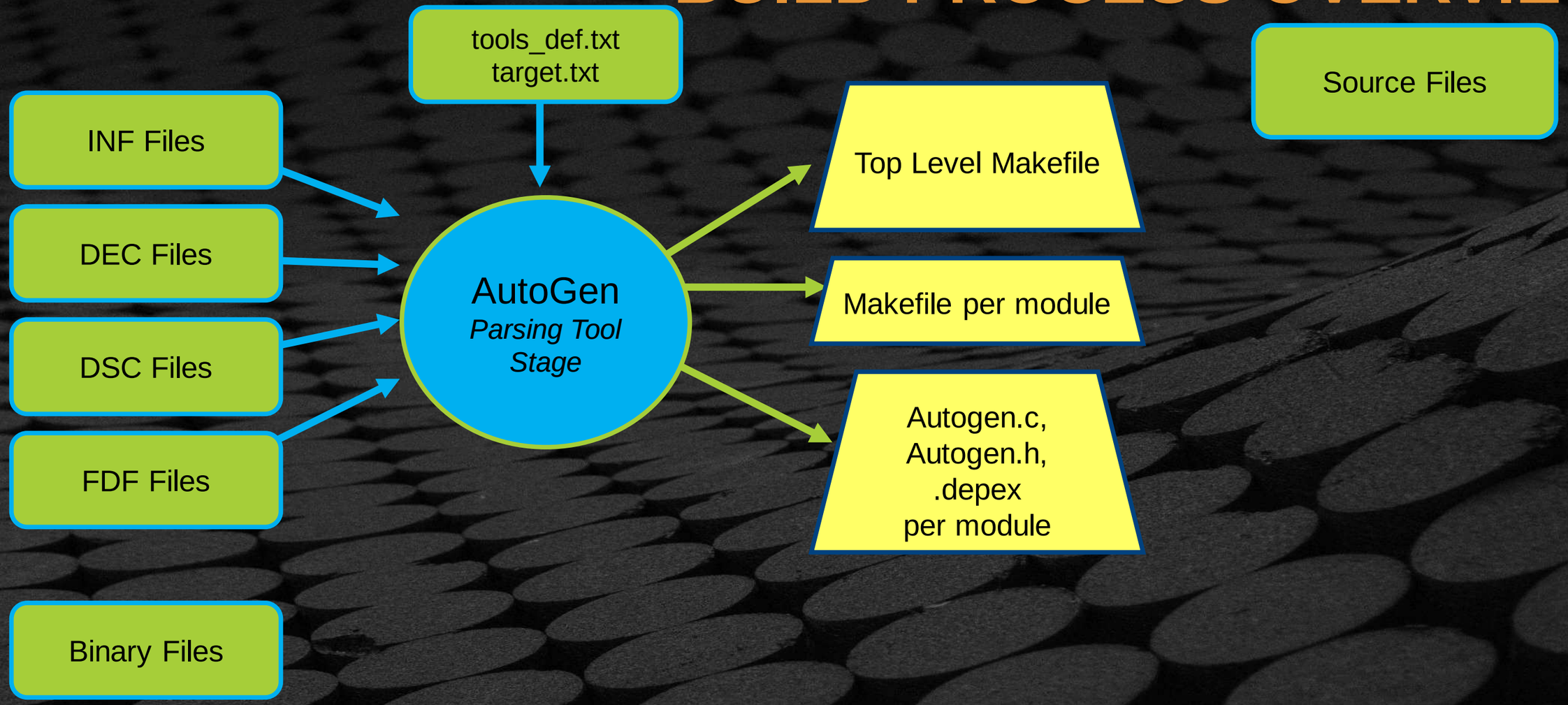
Binary Files



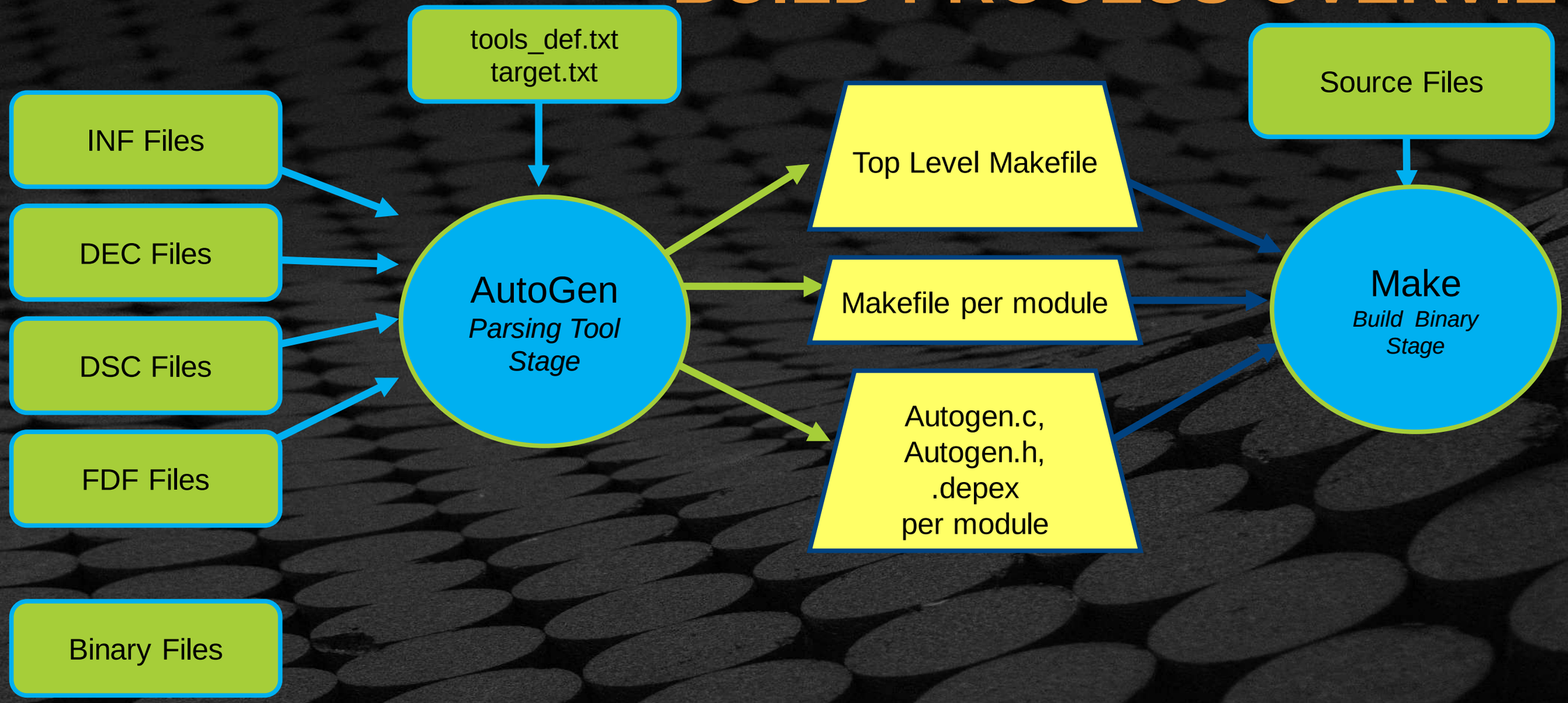
BUILD PROCESS OVERVIEW



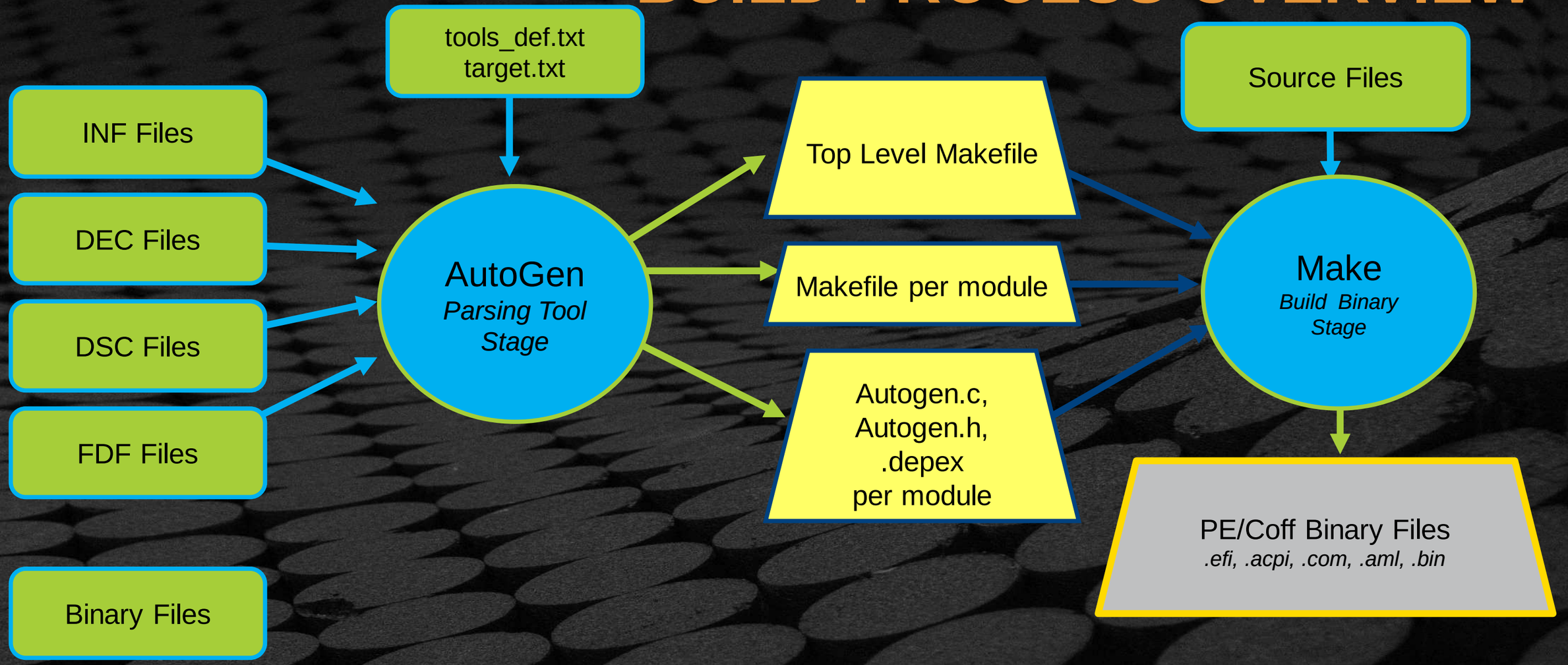
BUILD PROCESS OVERVIEW



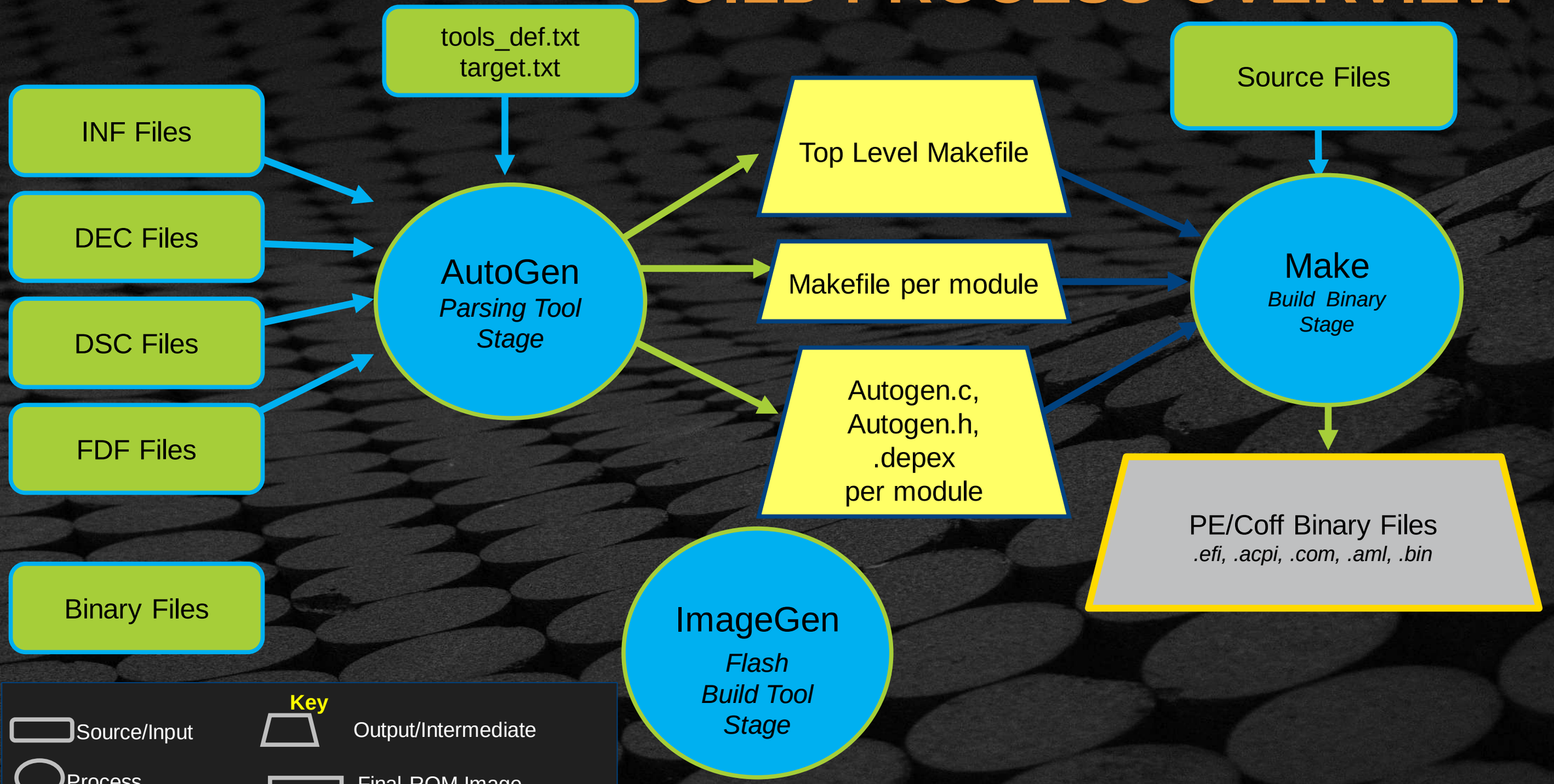
BUILD PROCESS OVERVIEW



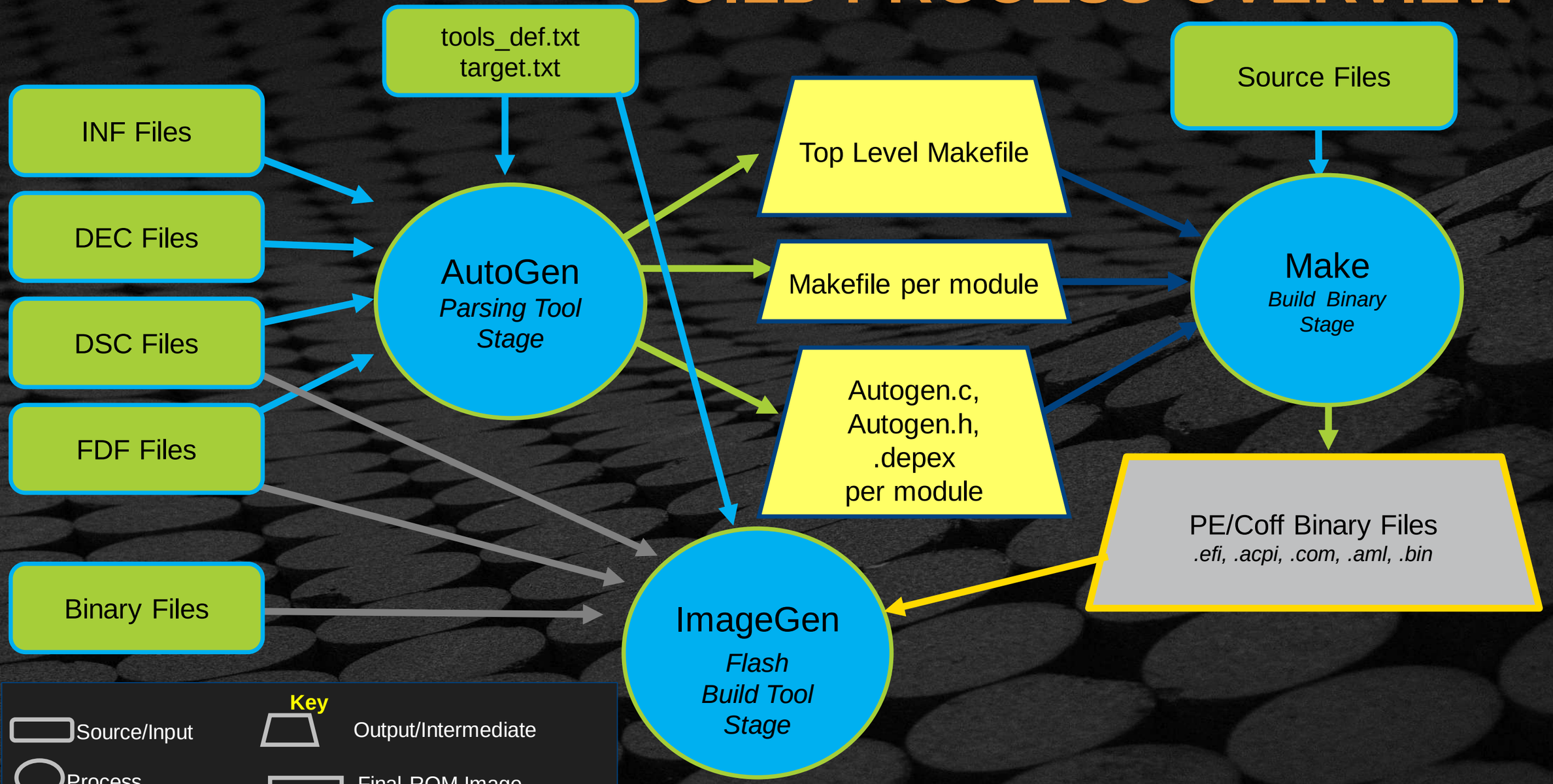
BUILD PROCESS OVERVIEW



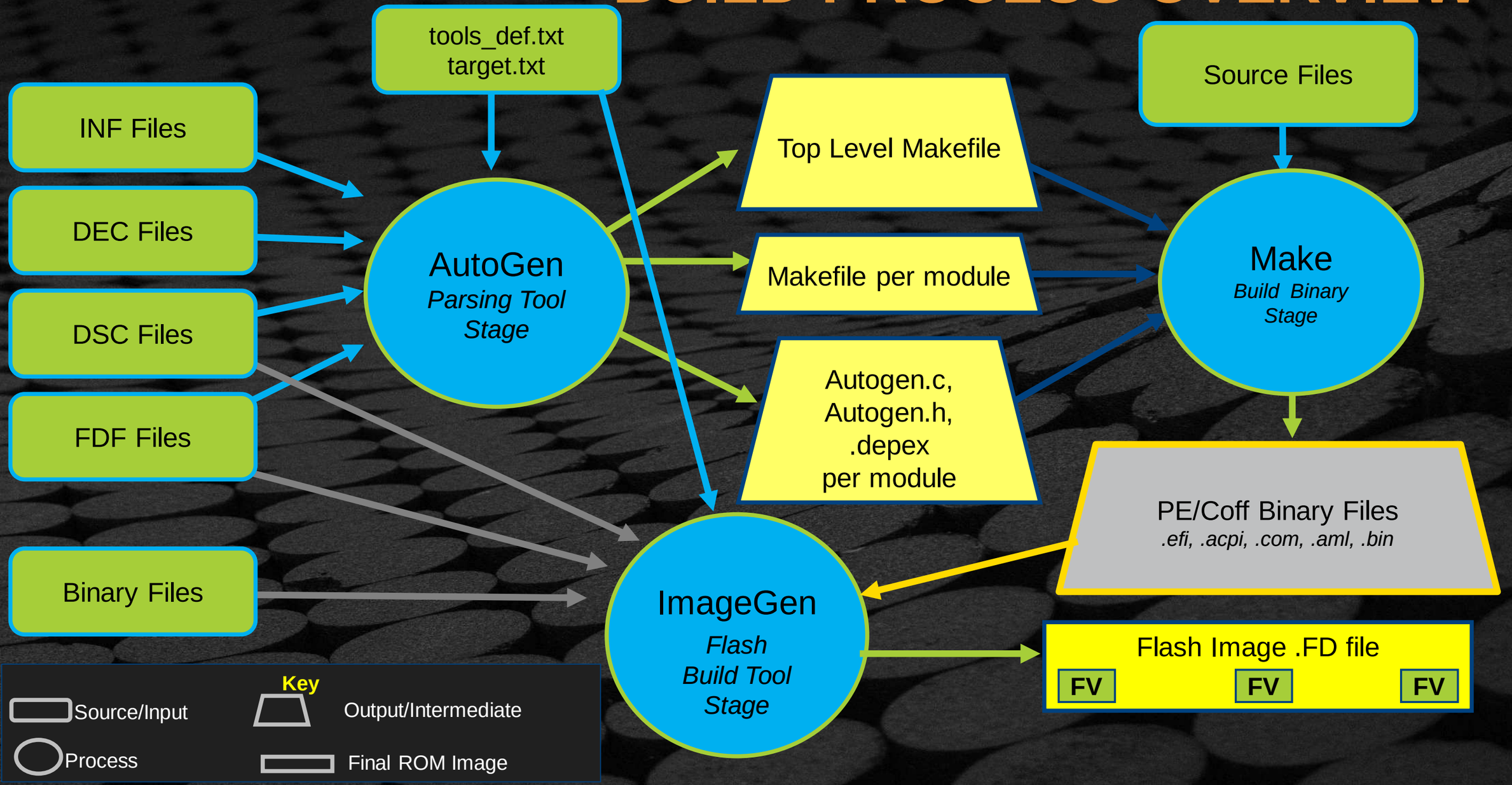
BUILD PROCESS OVERVIEW



BUILD PROCESS OVERVIEW



BUILD PROCESS OVERVIEW



BASIC BUILD STEPS

Platform

1. Open Terminal prompt
2. Navigate to root of EDK II workspace
3. Run **`edksetup`**
4. Edit **`Conf/target.txt`**
5. Run **`build`**
6. Output: firmware image (FD) file under **`Build`** directory

Module

1. Open Terminal prompt
2. Navigate to root of EDK II workspace
3. Run **`edksetup`**
4. Change to a directory with the proper INF
5. Run **`build`**
6. Output: .EFI files under **`Build`** directory

Note: Module .inf in .dsc components

BUILD OUTPUT LOCATION

Build
Build
Build

Path Element	Description	Notes
Build	Build directory	This is default.
Ovmfpkg	platform being used	
DEBUG_MYTOOLS	build mode and tool chain	From target.txt
FV	contains final image	Both FV and FD images
IA32	processor architecture	Contains platform makefile
Pkg/ModuleName	path to INF file	One for each INF
Foo	name of INF file (Module)	Contains module makefile
OUTPUT	.EFI file location	
DEBUG	Autogen files	

BUILD OUTPUT LOCATION

```
Build /OvmfX64
Build /Ovmf[1]
Build  /Ovmf[1]
```

Path Element	Description	Notes
Build	Build directory	This is default.
Ovmfpkg	platform being used	
DEBUG_MYTOOLS	build mode and tool chain	From target.txt
FV	contains final image	Both FV and FD images
IA32	processor architecture	Contains platform makefile
Pkg/ModuleName	path to INF file	One for each INF
Foo	name of INF file (Module)	Contains module makefile
OUTPUT	.EFI file location	
DEBUG	Autogen files	

[1] IA32 or X64

BUILD OUTPUT LOCATION

```
Build /OvmfX64 /DEBUG_MYTOOLS
Build /Ovmf[1]/DEBUG_MYTOOLS
Build  /Ovmf[1] /DEBUG_MYTOOLS
```

Path Element	Description	Notes
Build	Build directory	This is default.
Ovmfpkg	platform being used	
DEBUG_MYTOOLS	build mode and tool chain	From target.txt
FV	contains final image	Both FV and FD images
IA32	processor architecture	Contains platform makefile
Pkg/ModuleName	path to INF file	One for each INF
Foo	name of INF file (Module)	Contains module makefile
OUTPUT	.EFI file location	
DEBUG	Autogen files	

[1] IA32 or X64

BUILD OUTPUT LOCATION

```
Build /OvmfX64 /DEBUG_MYTOOLS /FV
Build /Ovmf[1]/DEBUG_MYTOOLS
Build  /Ovmf[1] /DEBUG_MYTOOLS
```

Path Element	Description	Notes
Build	Build directory	This is default.
Ovmfpkg	platform being used	
DEBUG_MYTOOLS	build mode and tool chain	From target.txt
FV	contains final image	Both FV and FD images
IA32	processor architecture	Contains platform makefile
Pkg/ModuleName	path to INF file	One for each INF
Foo	name of INF file (Module)	Contains module makefile
OUTPUT	.EFI file location	
DEBUG	Autogen files	

[1] IA32 or X64

BUILD OUTPUT LOCATION

```
Build /OvmfX64 /DEBUG_MYTOOLS /FV
Build /Ovmf[1]/DEBUG_MYTOOLS /IA32
Build /Ovmf[1] /DEBUG_MYTOOLS /IA32
```

Path Element	Description	Notes
Build	Build directory	This is default.
Ovmfpkg	platform being used	
DEBUG_MYTOOLS	build mode and tool chain	From target.txt
FV	contains final image	Both FV and FD images
IA32	processor architecture	Contains platform makefile
Pkg/ModuleName	path to INF file	One for each INF
Foo	name of INF file (Module)	Contains module makefile
OUTPUT	.EFI file location	
DEBUG	Autogen files	

[1] IA32 or X64

BUILD OUTPUT LOCATION

```
Build /OvmfX64 /DEBUG_MYTOOLS /FV
Build /Ovmf[1]/DEBUG_MYTOOLS /IA32 /Pkg /ModuleName
```


```
Build /Ovmf[1] /DEBUG_MYTOOLS /IA32 /Pkg /ModuleName
```


Path Element	Description	Notes
Build	Build directory	This is default.
Ovmfpkg	platform being used	
DEBUG_MYTOOLS	build mode and tool chain	From target.txt
FV	contains final image	Both FV and FD images
IA32	processor architecture	Contains platform makefile
Pkg/ModuleName	path to INF file	One for each INF
Foo	name of INF file (Module)	Contains module makefile
OUTPUT	.EFI file location	
DEBUG	Autogen files	

[1] IA32 or X64

BUILD OUTPUT LOCATION

```
Build /OvmfX64 /DEBUG_MYTOOLS /FV
Build /Ovmf[1]/DEBUG_MYTOOLS /IA32 /Pkg /ModuleName/Foo
Build /Ovmf[1] /DEBUG_MYTOOLS /IA32 /Pkg /ModuleName /Foo
```

Path Element	Description	Notes
Build	Build directory	This is default.
Ovmfpkg	platform being used	
DEBUG_MYTOOLS	build mode and tool chain	From target.txt
FV	contains final image	Both FV and FD images
IA32	processor architecture	Contains platform makefile
Pkg/ModuleName	path to INF file	One for each INF
Foo	name of INF file (Module)	Contains module makefile
OUTPUT	.EFI file location	
DEBUG	Autogen files	

[1] IA32 or X64

BUILD OUTPUT LOCATION

```
Build /OvmfX64 /DEBUG_MYTOOLS /FV
Build /Ovmf[1]/DEBUG_MYTOOLS /IA32 /Pkg /ModuleName/Foo
Build /Ovmf[1] /DEBUG_MYTOOLS /IA32 /Pkg /ModuleName /Foo /OUTPUT
```

Path Element	Description	Notes
Build	Build directory	This is default.
Ovmfpkg	platform being used	
DEBUG_MYTOOLS	build mode and tool chain	From target.txt
FV	contains final image	Both FV and FD images
IA32	processor architecture	Contains platform makefile
Pkg/ModuleName	path to INF file	One for each INF
Foo	name of INF file (Module)	Contains module makefile
OUTPUT	.EFI file location	
DEBUG	Autogen files	

[1] IA32 or X64

BUILD OUTPUT LOCATION

```
Build /OvmfX64 /DEBUG_MYTOOLS /FV
Build /Ovmf[1]/DEBUG_MYTOOLS /IA32 /Pkg /ModuleName/Foo /DEBUG
```

```
Build  /Ovmf[1]  /DEBUG_MYTOOLS    /IA32  /Pkg  /ModuleName   /Foo   /OUTPUT
                                                                    /DEBUG
```

Path Element	Description	Notes
Build	Build directory	This is default.
Ovmfpkg	platform being used	
DEBUG_MYTOOLS	build mode and tool chain	From target.txt
FV	contains final image	Both FV and FD images
IA32	processor architecture	Contains platform makefile
Pkg/ModuleName	path to INF file	One for each INF
Foo	name of INF file (Module)	Contains module makefile
OUTPUT	.EFI file location	
DEBUG	Autogen files	

[1] IA32 or X64

EDK II BUILD PROCESS STAGES

AutoGen
Parsing Tool Stage

Parse meta-data files to generate some C source code files and the make files

EDK II BUILD PROCESS STAGES

AutoGen
Parsing Tool Stage

Parse meta-data files to generate some C source code files and the make files

Make
Build Binary Stage

Process source code files to create PE32/PE32+/COFF images processed to UEFI format using $(MAKE) tool

EDK II BUILD PROCESS STAGES

AutoGen
Parsing Tool Stage

Parse meta-data files to generate some C source code files and the make files

Make
Build Binary Stage

Process source code files to create PE32/PE32+/COFF images processed to UEFI format using $(MAKE) tool

ImageGen
Flash Build Tool Stage

Takes the UEFI format files, creates UEFI “FLASH” images, UEFI apps, or UEFI PCI option ROMs

EDK II BUILD: AUTOGEN STAGE

EDK II Open Source

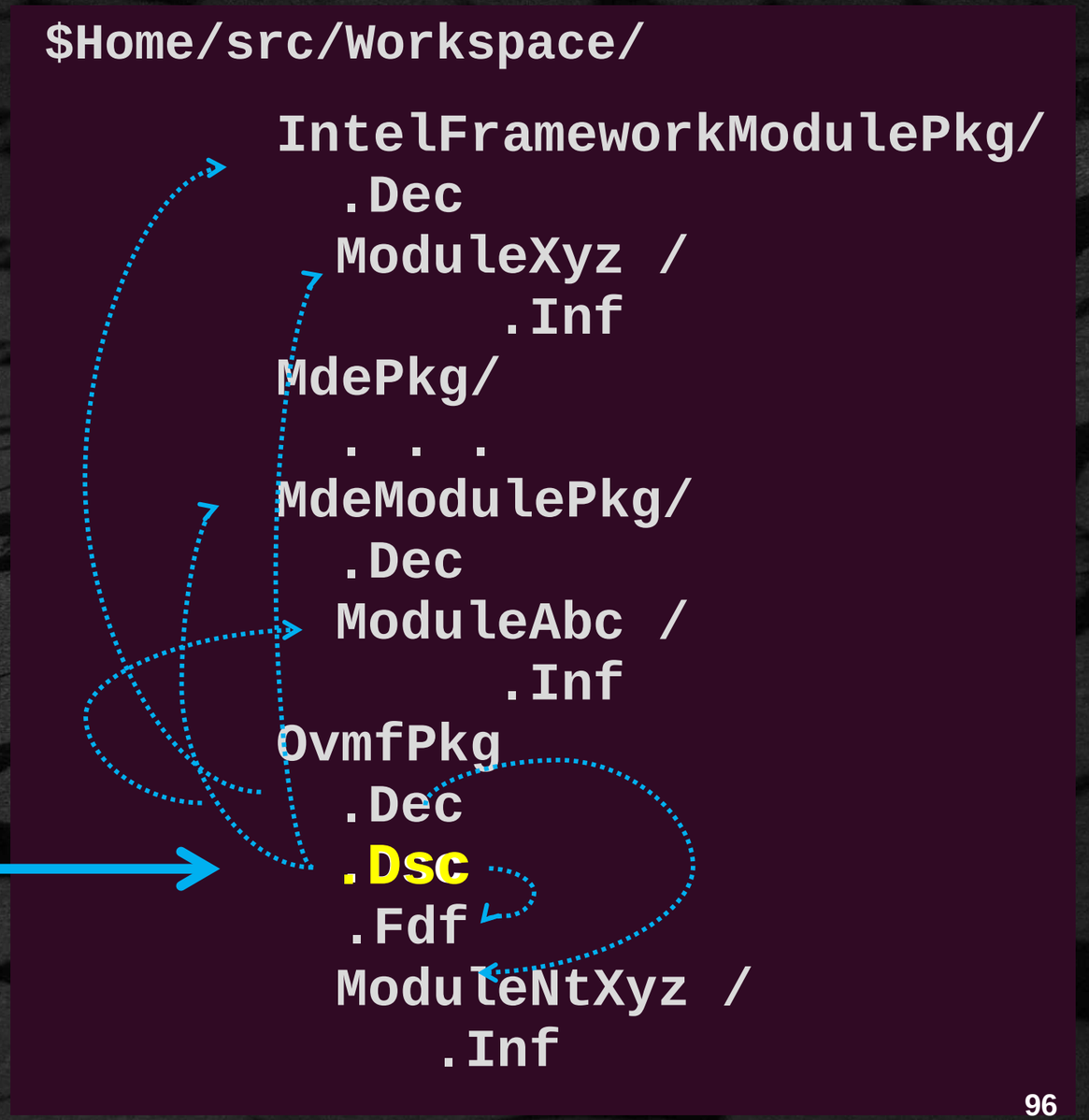
```
Build –p OvmfPkg/OvmfX64Pkg.dsc
```

```
$Home/src/Workspace/
        IntelFrameworkModulePkg/
          .Dec
          ModuleXyz /
                .Inf
        MdePkg/
          . . .
        MdeModulePkg/
          .Dec
          ModuleAbc /
                .Inf
        OvmfPkg
          .Dec
          .Dsc
          .Fdf
          ModuleNtXyz /
             .Inf
```

EDK II BUILD: AUTOGEN STAGE

EDK II Open Source

```
Build –p OvmfPkg/OvmfX64Pkg.dsc
```



EDK II BUILD: MAKE STAGE

Uses assemblers/compilers/linkers to generate PE32/PE32+ COFF image file

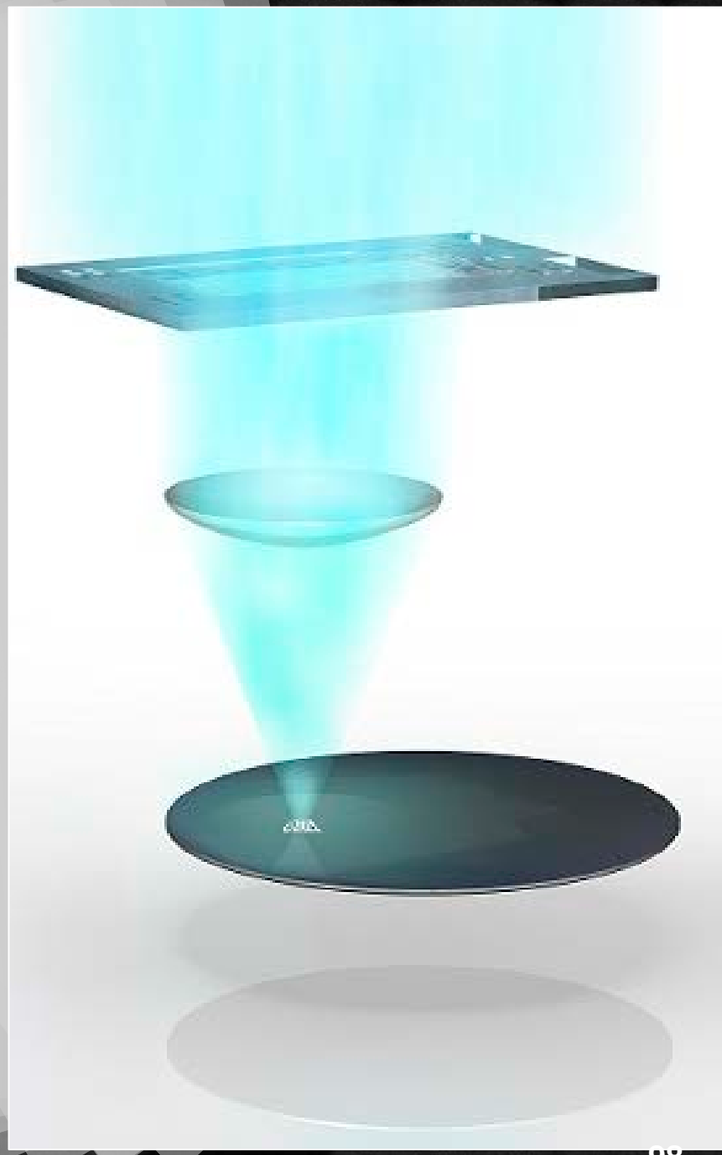
Uses ImageGen tools to modify PE32/PE32+/COFF image file; creates UEFI file (EFI_IMAGE_SECTION_HEADER structure)

GenFW

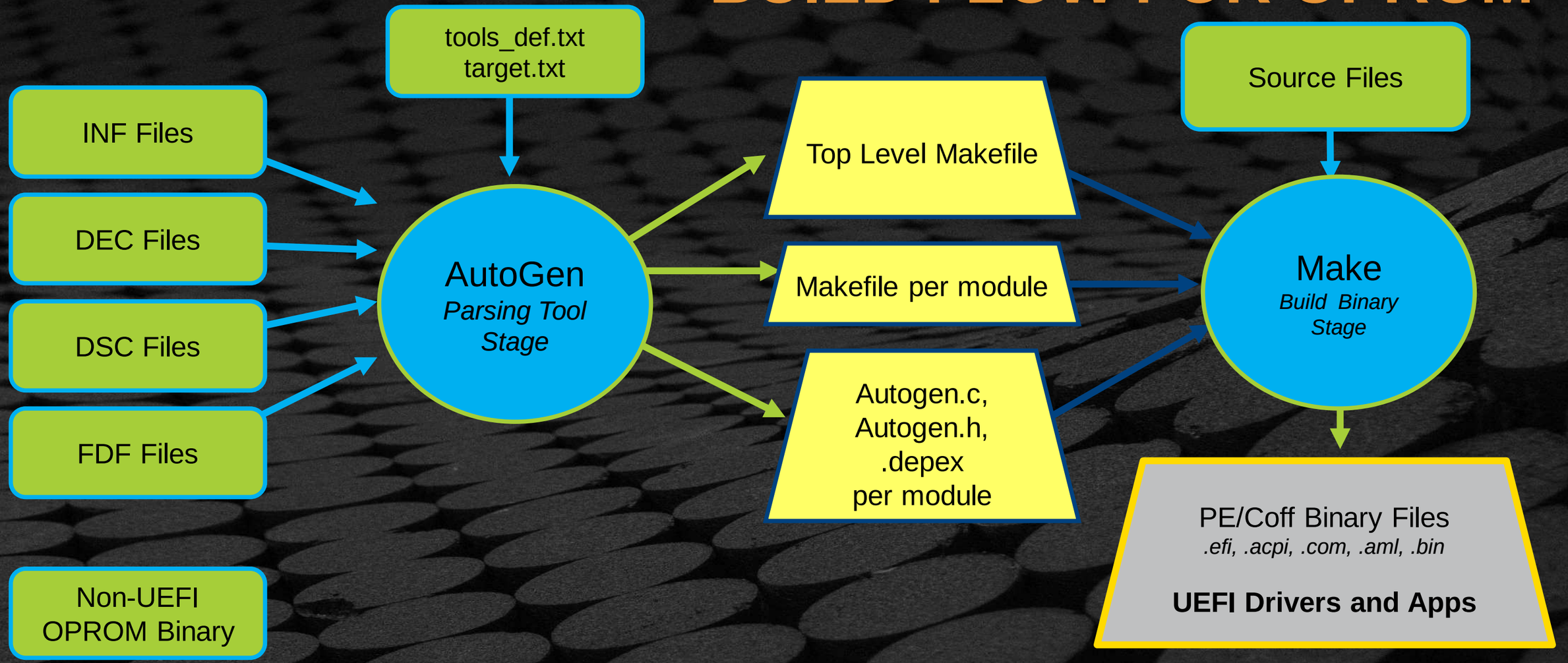
GenFds

EDK II BUILD: IMAGEGEN STAGE

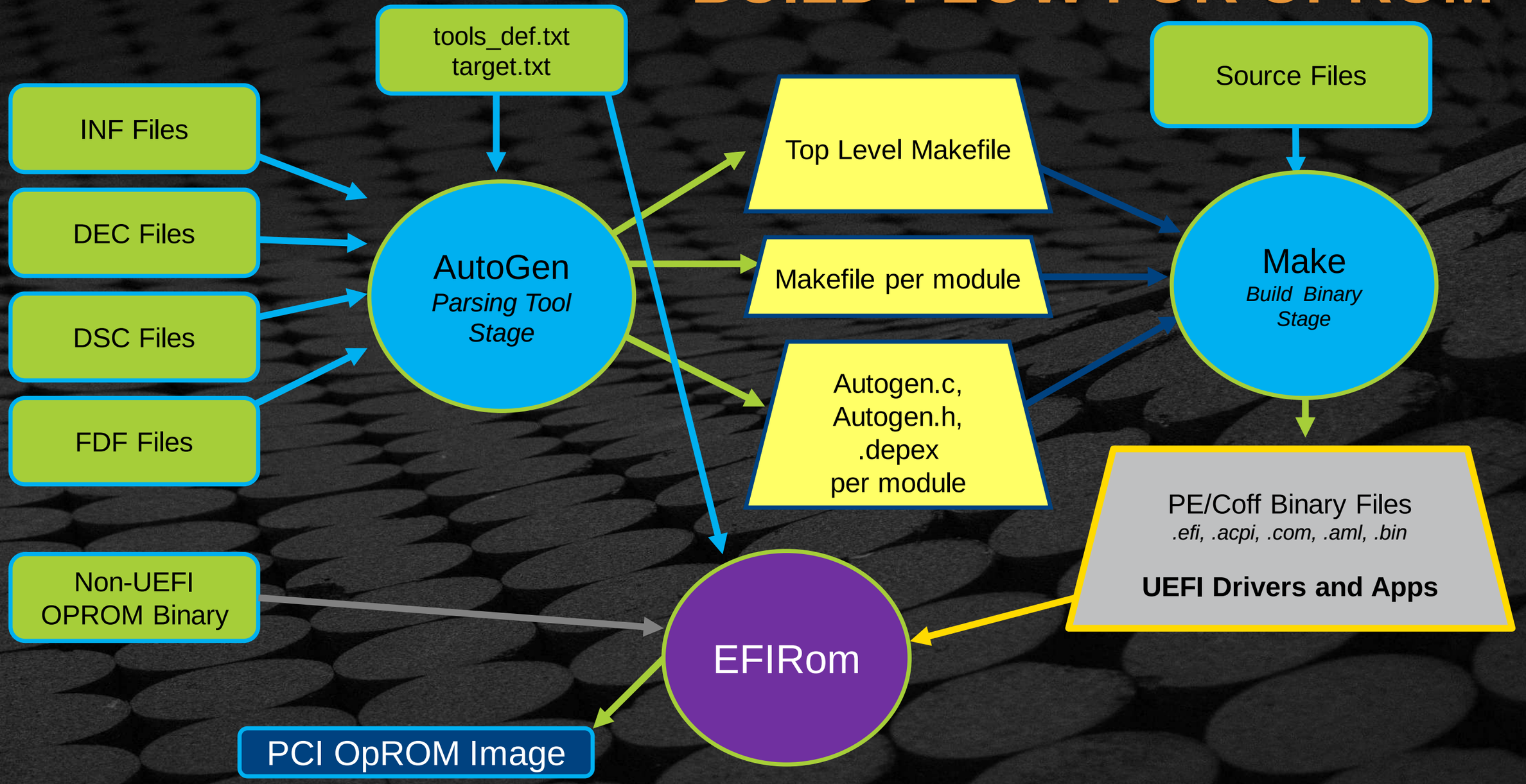
- Builds one image for each specified firmware volume (FV)
- The FDF file supports all syntax available in the PI Specification Vol. 3



BUILD FLOW FOR OPROM



BUILD FLOW FOR OPROM



tianocore

THE BUILD COMMAND

- Accepts command line arguments to support scripted builds
- Overrides most settings found in **`target.txt`**
- Overrides DSC with a minimal INF build
- Overrides some settings in DSC file (.FDF)
- Choose settings from the FDF file **`(ROMIMAGE, FVIMAGE)`**
- Choose **`$(make)`** options (silent, verbose, quiet)

USING EDK II BUILD COMMAND

```
Usage: build.exe [options] [all|fds|genc|genmake|clean|cleanall|cleanlib|modules|libraries|run]

Copyright (c) 2007 - 2017, Intel Corporation  All rights reserved.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -a TARGETARCH, --arch=TARGETARCH
                        ARCHS is one of list: IA32, X64, IPF, ARM or EBC,
                        which overrides target.txt's TARGET_ARCH definition
                        To specify more archs, please repeat this option.
  -p PLATFORMFILE, --platform=PLATFORMFILE
                        Build the platform specified by the DSC file name
                        argument, overriding target.txt's ACTIVE_PLATFORM
                        definition.
  -m MODULEFILE, --module=MODULEFILE
                        Build the module specified by the INF file name
                        argument.
```

bash$ Build -h

USING EDK II BUILD COMMAND

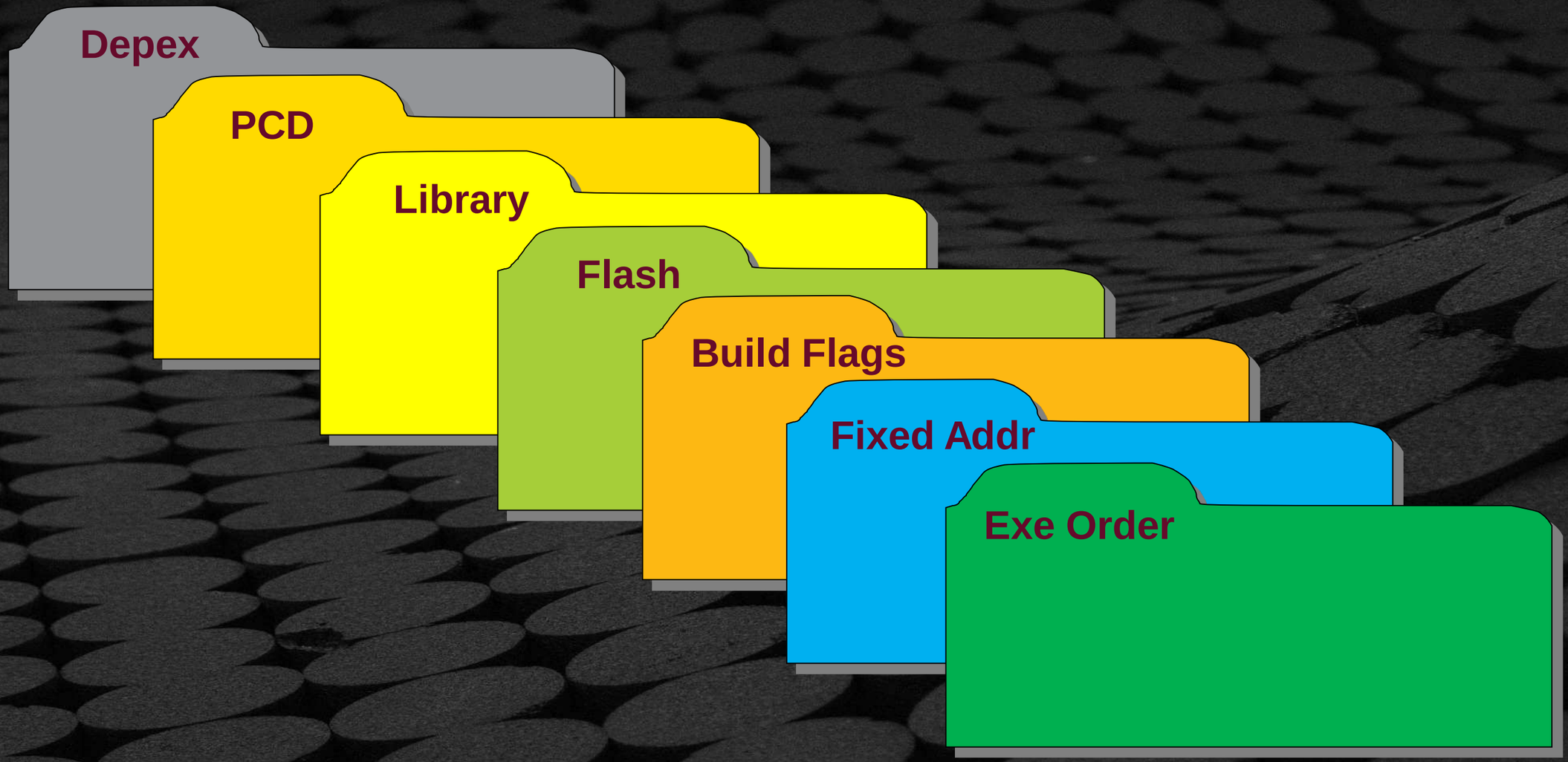
```
Usage: build.exe [options] [all|fds|genc|genmake|clean|cleanall|cleanlib|modules|libraries|run]

Copyright (c) 2007 - 2017, Intel Corporation  All rights reserved.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -a TARGETARCH, --arch=TARGETARCH
                        ARCHS is one of list: IA32, X64, IPF, ARM or EBC,
                        which overrides target.txt's TARGET_ARCH definition
                        To specify more archs, please repeat this option.
  -p PLATFORMFILE, --platform=PLATFORMFILE
                        Build the platform specified by the DSC file name
                        argument, overriding target.txt's ACTIVE_PLATFORM
                        definition.
  -m MODULEFILE, --module=MODULEFILE
                        Build the module specified by the INF file name
                        argument.
```

bash$ Build -h

build -h

https://gitpitch.com/Laurie0131/EDK_II_Build_Process_Pres_2/master#/36/2

USING BUILD -Y COMMAND




```
build  -Y  PCD -y pcd.log
```

USING BUILD -Y FOR REPORTS

- Scroll through examples of reports from the Build -Y commands
- Link to on line presentation

build -Y DEPEX

https://gitpitch.com/Laurie0131/EDK__II_Build_Process_Pres_2/master#/37/2

build -Y PCD

https://gitpitch.com/Laurie0131/EDK__II_Build_Process_Pres_2/master#/37/3

build -Y LIBRARY

https://gitpitch.com/Laurie0131/EDK__II_Build_Process_Pres_2/master#/37/4

build -Y FLASH

https://gitpitch.com/Laurie0131/EDK__II_Build_Process_Pres_2/master#/37/5

build -Y BUILD_FLAGS

https://gitpitch.com/Laurie0131/EDK__II_Build_Process_Pres_2/master#/37/6

build -Y FIXED_ADDRESS

https://gitpitch.com/Laurie0131/EDK__II_Build_Process_Pres_2/master#/37/7

build -Y EXECUTION_ORDER

https://gitpitch.com/Laurie0131/EDK__II_Build_Process_Pres_2/master#/37/8

Local Report.html is generated on the host build machine - pop up this in the Browser window.

Link: Link to Report.html on local machine

BUILD TOOL BINARIES

Utility	Description
Build.exe	Tool is written in Python and calls AutoGen.exe, then it calls $(MAKE) –f Makefile.out, and finally, it calls GenFds.exe
EfiRom.exe	used to build an option ROM image
PatchModule.exe	used to patch a binary module that has a PCD of type PATCHABLE_IN_MODULE
PatchPlatform.exe	used to modify either the PCD Database or the VPD settings in a flash device image

tianocore

SUMMARY

- Define EDK II
- Describe EDK II’s elements including file extensions, rectories, modules, packages, and libraries
- Explain the EDK II build process
- Explain the Build tools

tianocore

Questions?


tianocore