

UEFI & EDK II Training

EDK II Build Process and Environment

tianocore.org

LESSON OBJECTIVE

- ★ Define EDK II
- ★ Describe EDK II's elements including file extensions, directories, 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**

**Build tools &
source code
– added CI¹**

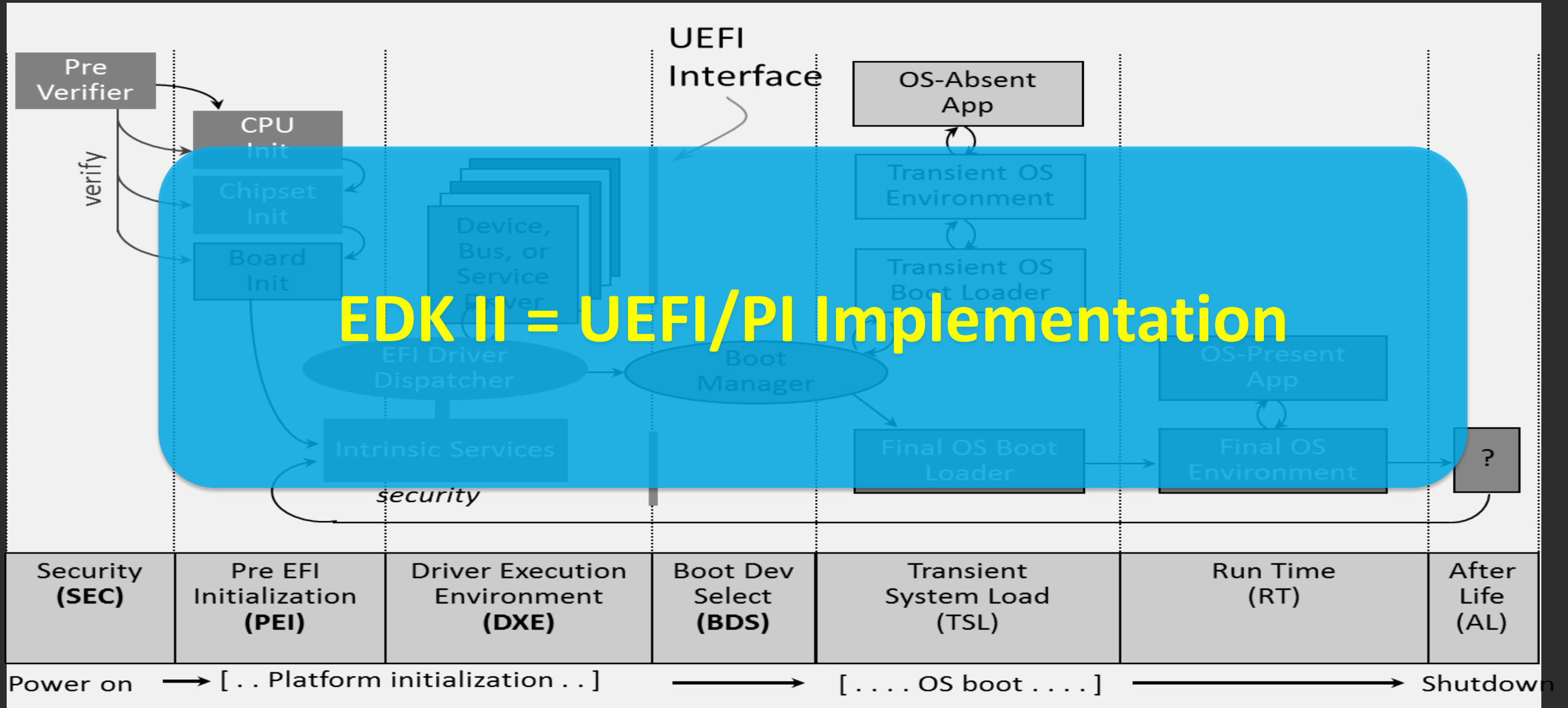
**Package
Definition file:
DEC**

**Flash Mapping
Tool**

**Move as much
Code to C**

**Open source
EDK II on
tianocore.org**

IMPLEMENTATION OF EDK II



EDK II File Extensions

- Located on tianocore.org project edk2

.DSC	- Platform Description
.DEC	- Package Declaration
.INF	- Module Definition <i>define a component</i>
.FDF	- Flash Description

EDK II File Extensions

- Located on tianocore.org project edk2

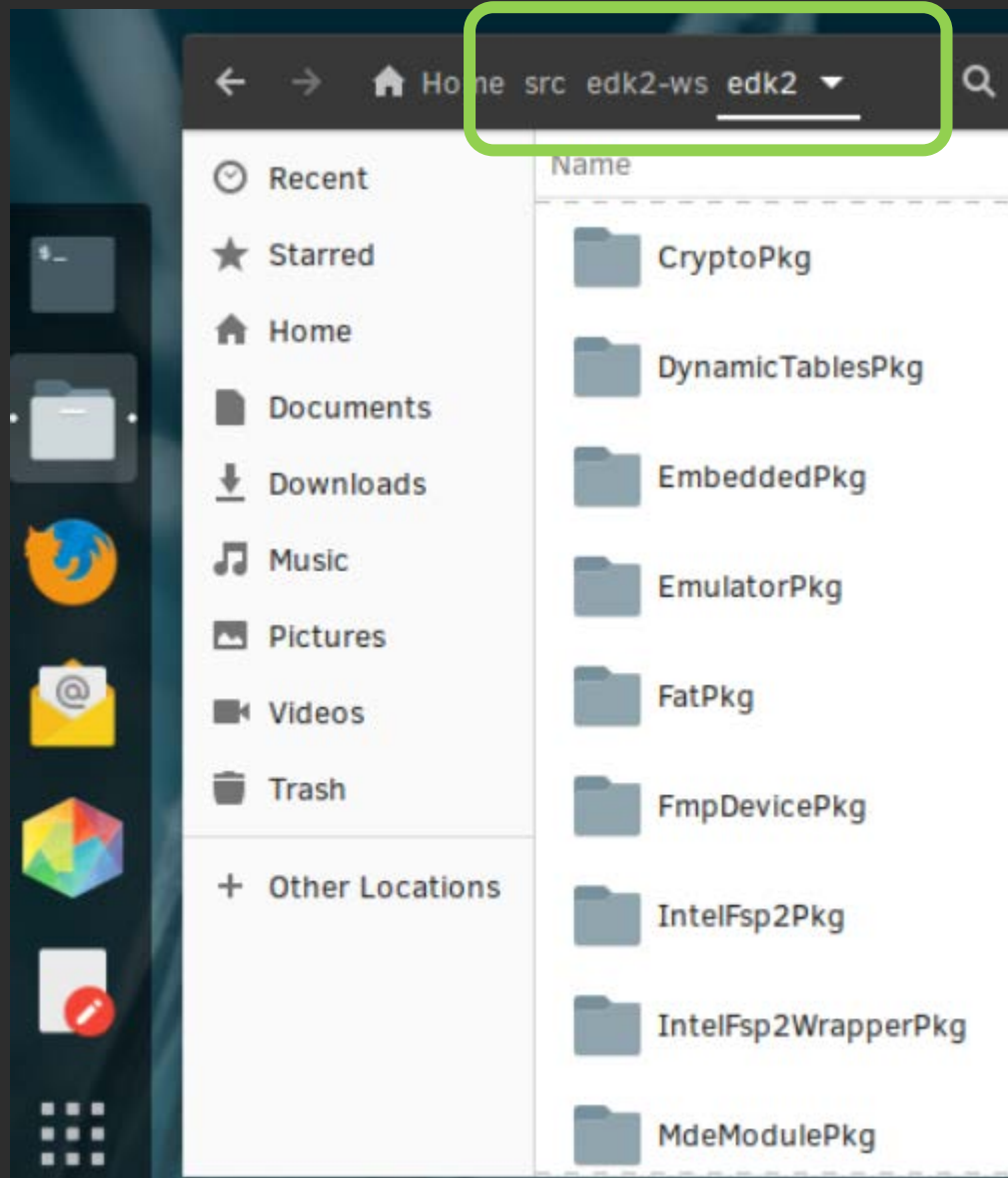
.DSC .DEC .INF .FDF	<ul style="list-style-type: none"> - Platform Description - Package Declaration - Module Definition <i>define a component</i> - Flash Description
.VFR .UNI .c & .h	<ul style="list-style-type: none"> - Visual Forms Representation for User interface - Unicode String text files w/ ease of localization - Source code files
.FD .FV	<ul style="list-style-type: none"> - Final Flash Device Image - Firmware Volume File

EDK II
Spec

Source

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-ws`

```
bash$ cd $HOME/src/edk2-ws/edk2
bash$ . edksetup.sh
bash$ make -C BaseTools/
bash$ build
```


Organization Directory Structure

Common

- No direct HW requirements, Features, Interface defs

Platform

- Enable a specific platform's capabilities.

Board

- Board specific code

Silicon

- Hardware specific code

Features

- Advanced features of platform functionality that is non-essential for "basic OS boot"

EDK II Open Board Directory Structure

edk2/ <https://github.com/tianocore/edk2> ← **Common** - KabyLake w/ Intel® FSP

...
edk2-platforms/ <https://github.com/tianocore/edk2-platforms>

Platform/

Intel/

BoardModulePkg

KabylakeOpenBoardPkg

KabylakeRvp3

MinPlatformPkg

UserInterfaceFeaturePkg

← **Common (sharable)**

← **Platform (family)**

← **Board (instance)**

← **Platform (common)**

← **Advanced Feature**

Silicon/

Intel/

KabylakeSiliconPkg

← **Silicon**

edk2-non-os/ <https://github.com/tianocore/edk2-non-os>

Silicon/

Intel/

KabylakeSiliconBinPkg

PurleySiliconBinPkg

← **Silicon**

FSP/ <https://github.com/IntelFsp/FSP>

KabylakeFspBinPkg

← **Silicon**

MinPlatformPkg Example

Key

Silicon/Chipset

Platform

Repository

Smallest separate object compiled in EDK II

Compiles to
.EFI file



UEFI/DXE Driver
PEIM
UEFI App. or
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
- Contain Multiple Modules



EDK II PACKAGE EXAMPLES: SPECS

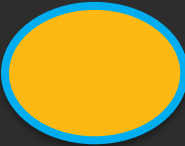

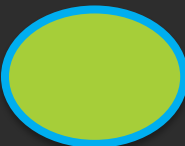
MdePkg

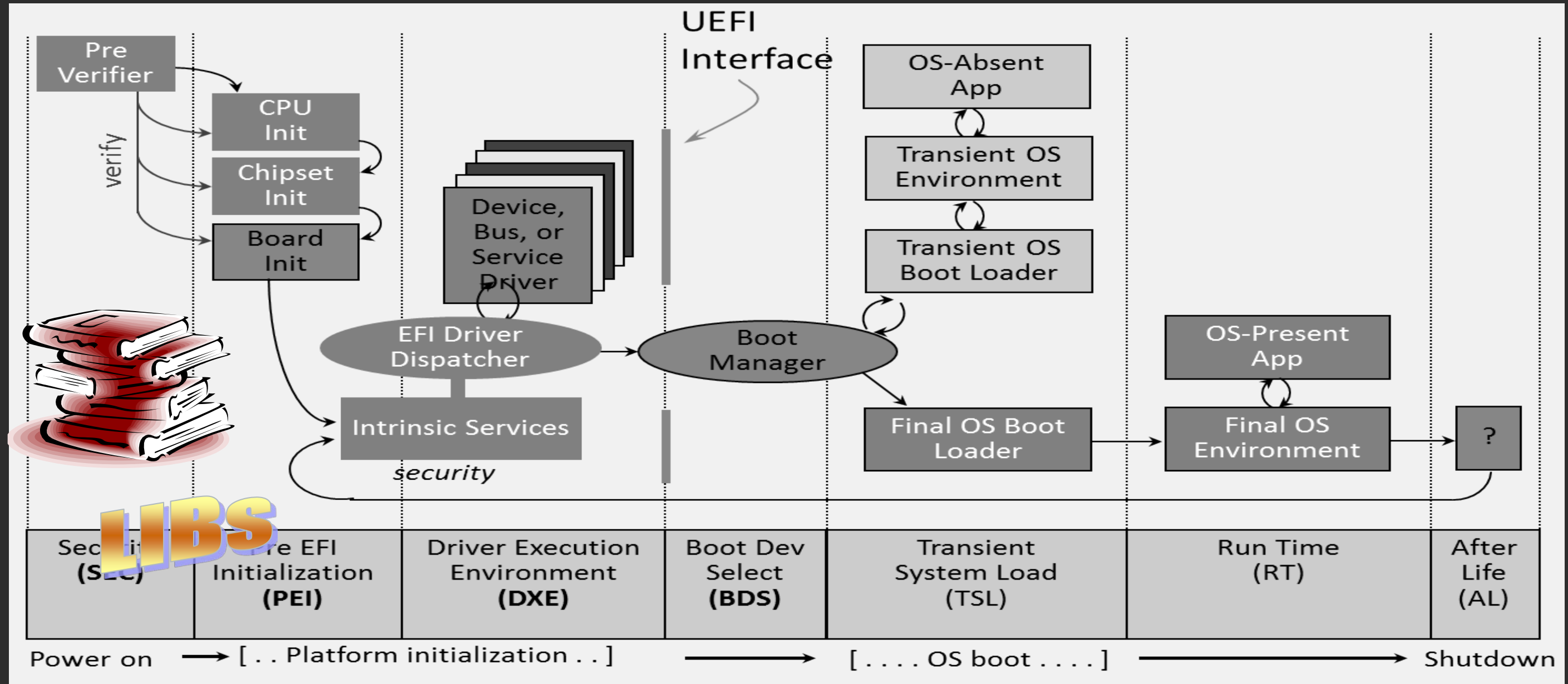
Include files and
libraries for Industry
Standard Specifications

MdeModulePkg

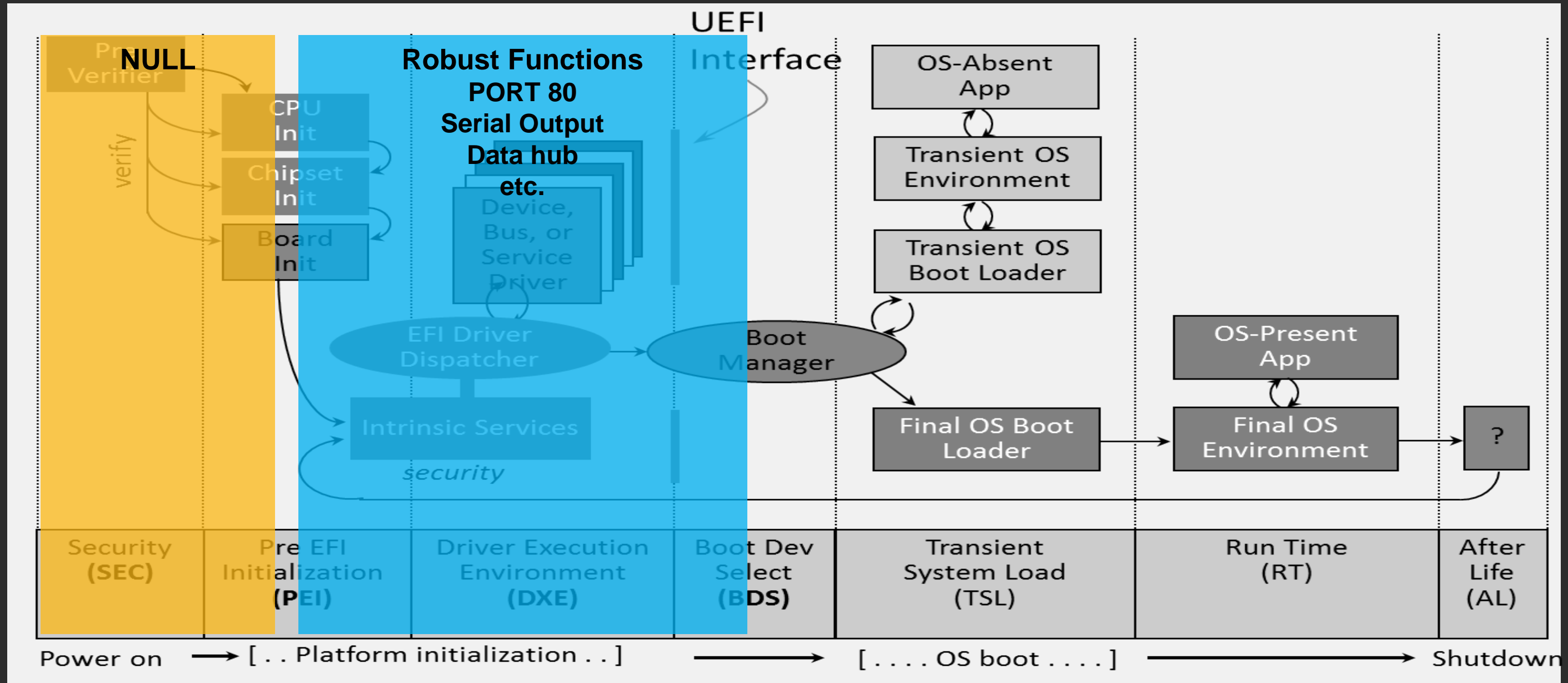
Modules only definitions
from the Industry
Standard Specification
are defined in the
MdePkg

ADDITIONAL EDK II PACKAGE EXAMPLES:

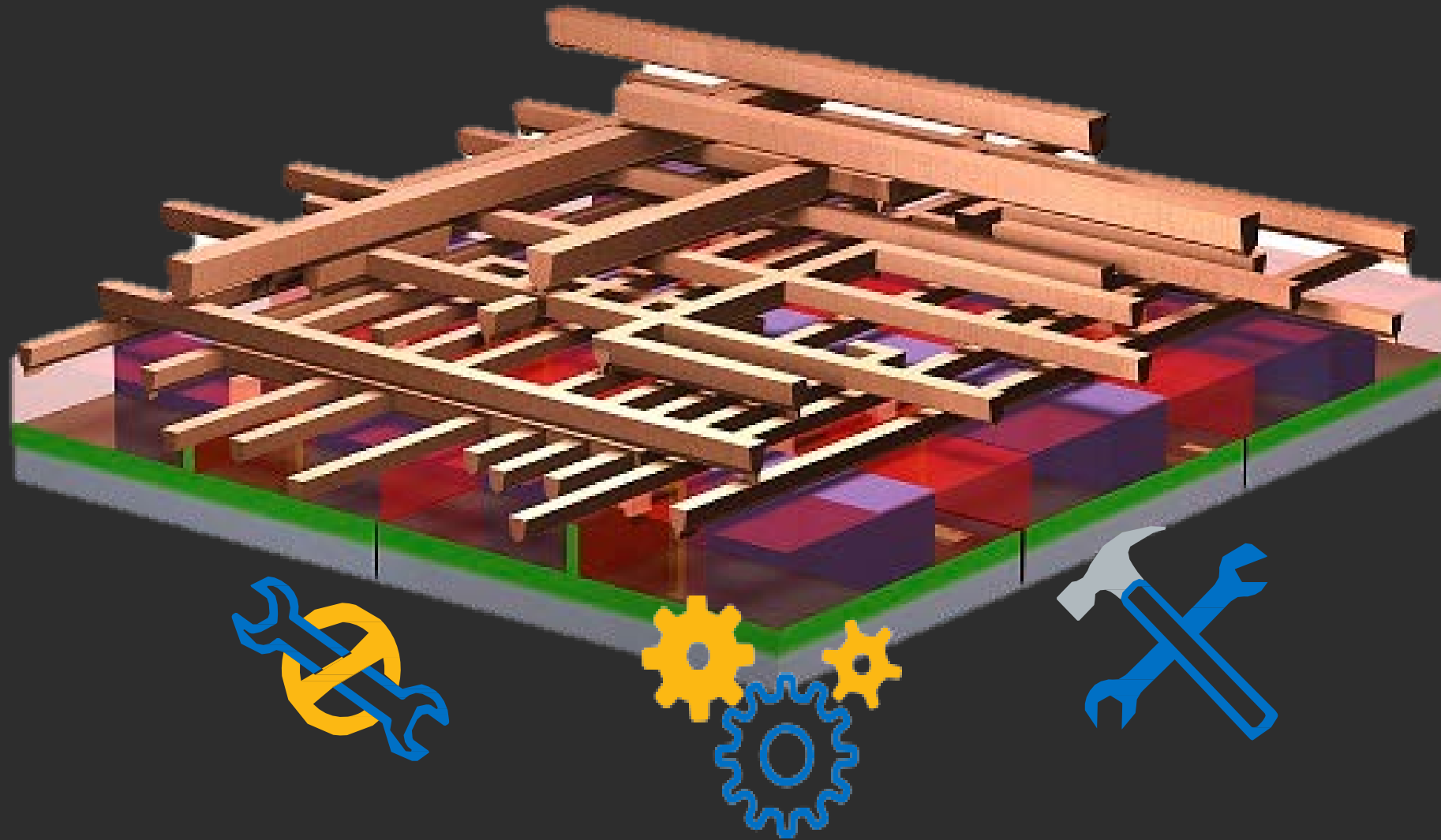
-  Platforms EmulatorPkg & OvmfPkg
-  Chipset/Processor IntelSiliconPkg
KabylakeSiliconPkg
KabylakeFspBinPkg
-  Functionality ShellPkg & NetworkPkg



EXAMPLE – LIBRARY “DEBUGLIB”



PLATFORM CONFIGURATION DATABASE (PCD)



Goals

Define module parameters

Store module / platform configurations

Reduce source edits

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

Advantages

Binary Modularity

Configure firmware settings in binaries without building

Configure

Provide for options to configure firmware features

Patching

Simplify the binary patching process

EDK II INFRASTRUCTURE SUMMARY

+

+

Packages

List of related
modules

Libraries

Same name &
interface

PCDs

Platform
Config. DB

BUILD TOOLS

EDK II Build Tools and Configuration Files

EDK II With Continuous Integration (CI) Tools

- Python tools (pytool) and extensions for building and maintaining an EDK II based UEFI firmware code
- Designed to easily and consistently support running locally and in a cloud CI environment
- Uses a dynamic Python module to customize a global configuration file
- Documentation: [pytool CI Tools](#)



Stuart

Stuart CI Development Environment

- Windows 10:
 - Visual Studio VS2017 or VS2019
 - Windows SDK (for rc)
- Ubuntu 18.04 or Fedora
- GCC5 or greater
- Python 3.7.x or greater on Path
- Git on Path

Typical Stuart CI Commands

```
$ pip install pip-requirements
$ stuart_setup
$ stuart_update
$ python BaseTools\Edk2ToolsBuild.py
$ stuart_ci_build
$ stuart_build
```

To Pass macros to build use:

```
BLD_*_[Macro-to-pass]=[Value]
```

Example Output From Stuart CI Build

```
INFO - Cmd to run is: build -p EmulatorPkg/EmulatorPkg.dsc -b DEBUG -t VS2019 -a X64  
-D WIN_HOST_BUILD=TRUE -D BUILD_X64=TRUE
```

```
INFO -  
INFO - -----Cmd Output Starting-----  
INFO - -----
```

```
INFO - Build environment: Windows-10-10.0.18362-SP0
```

```
INFO - Build start time: 10:30:55, Aug.27 2020
```

• • •

```
PROGRESS - Running Post Build
```

```
DEBUG - Plugin Success: Windows RC Path Support
```

```
DEBUG - Plugin Success: Windows Visual Studio Tool Chain Support
```

```
INFO - Writing BuildToolsReports to
```

```
D:\FW\edk2-ws\edk2\Build\EmulatorX64\DEBUG_VS2019\BUILD_TOOLS_REPORT
```

```
DEBUG - Plugin Success: Build Tools Report Generator
```

```
PROGRESS - End time: 2020-08-27 10:17:41.147836 Total time Elapsed: 0:01:42
```

```
SECTION - Log file is located at: D:\FW\edk2-ws\edk2\Build\BUILDLOG_EmulatorPkg.txt
```

```
SECTION - Summary
```

```
PROGRESS - Success
```

ERROR – Red

WARNING - Yellow

Non-Stuart CI Development Environment

Compiler Tool Chains

- Microsoft Visual Studio (VS2019, VS2017, VS2015, VS2013, VS2012, etc.)
- Microsoft WDK
- Intel C/C++ compiler
- Intel C EFI Byte Code (EBC) compiler
- GCC V5.x or later

Python 3.7.*n* & Nasm & IASL

Operating Systems

- Microsoft Windows XP/7/8/10
- Apple Mac OS X
- RedHat Enterprise Linux
- Novell SuSE Linux
- Ubuntu 18.04
- Fedora
- Clear Linux* Project

ENVIRONMENT VARIABLES

Set by
edksetup
Windows = .bat
Linux = .sh

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

CONFIGURATION FILES - SCRIPTS

● edksetup.bat or edksetup.sh

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

● First time use will set up configuration files:

Conf/**build_rule**.txt

Conf/**target**.txt

Conf/**tools_def**.txt

● Setup & verify a developer's workspace

Multiple Workspace Environment Variable

PACKAGES_PATH

WORKSPACE

PACKAGES_PATH – *Optional*

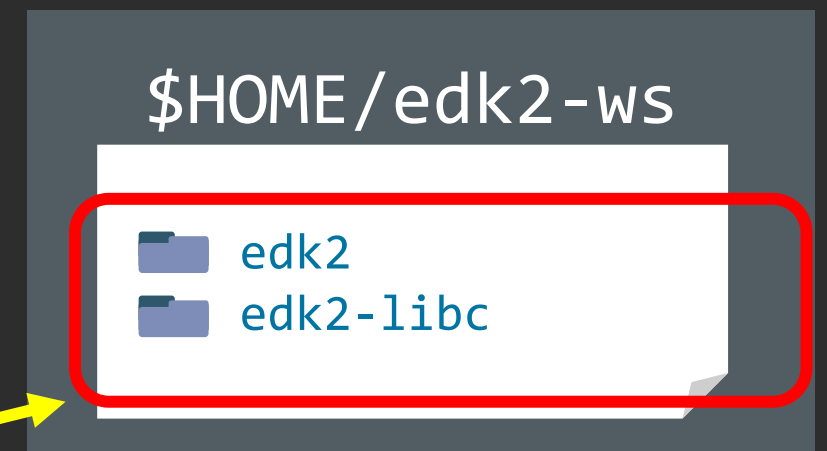
Multiple paths that will be searched when attempting to resolve the location of packages.

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

Example:

```
$> set WORKSPACE=%CWD%
```

```
$> set PACKAGES_PATH=%WORKSPACE%/edk2;%WORKSPACE%/edk2-libc
```



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



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)

First Make BaseTools

BaseTools

The first step is to make / “nmake” the “BaseTools” with the host OS & compiler environment.

For  Linux GCC5 the command is:

```
bash$ make -C BaseTools
```

For  Windows Visual Studio w/ Python 3.7 the command is:

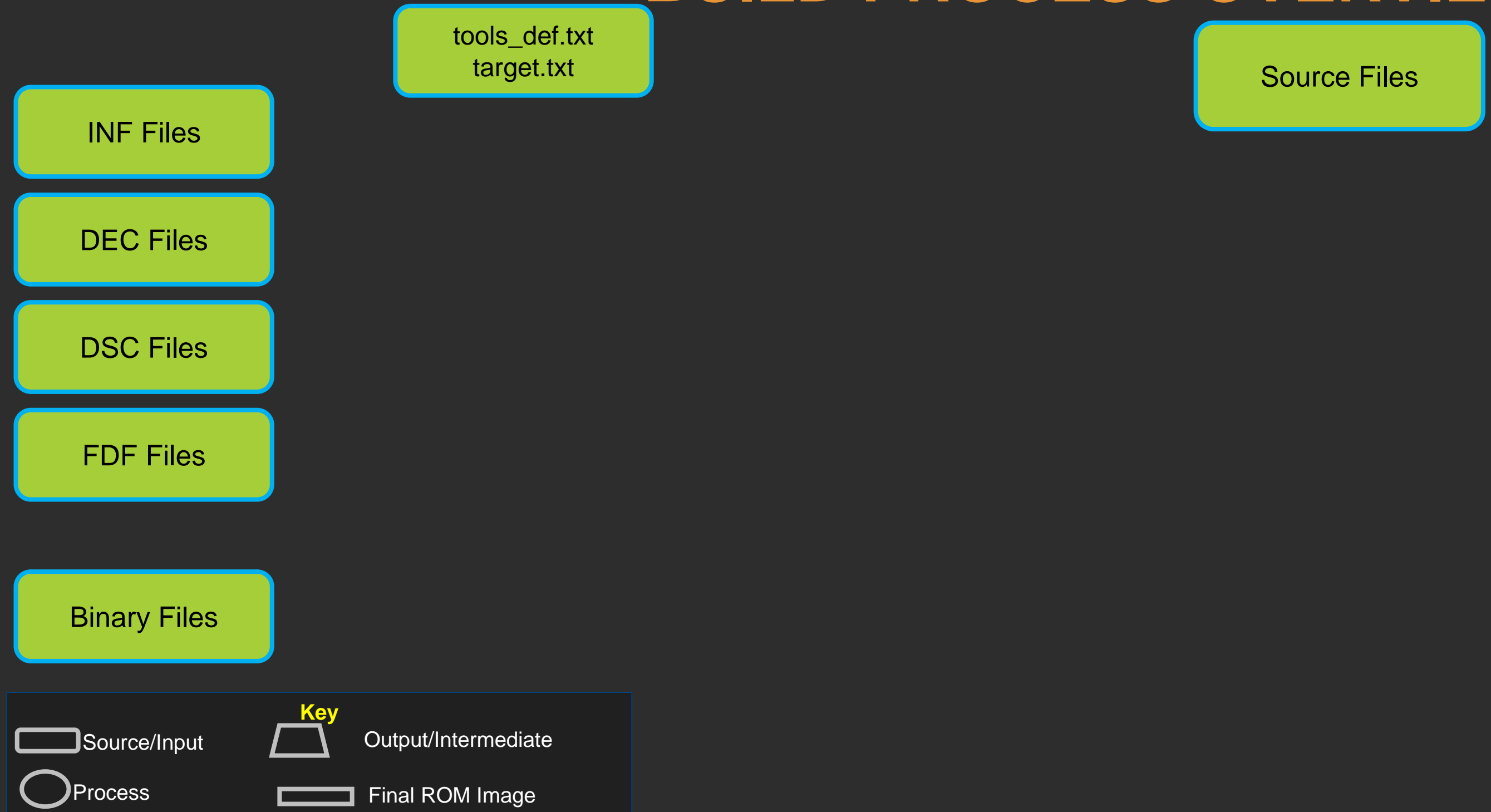
```
> edksetup.bat Rebuild
```

Building BaseTools only needs to be done once

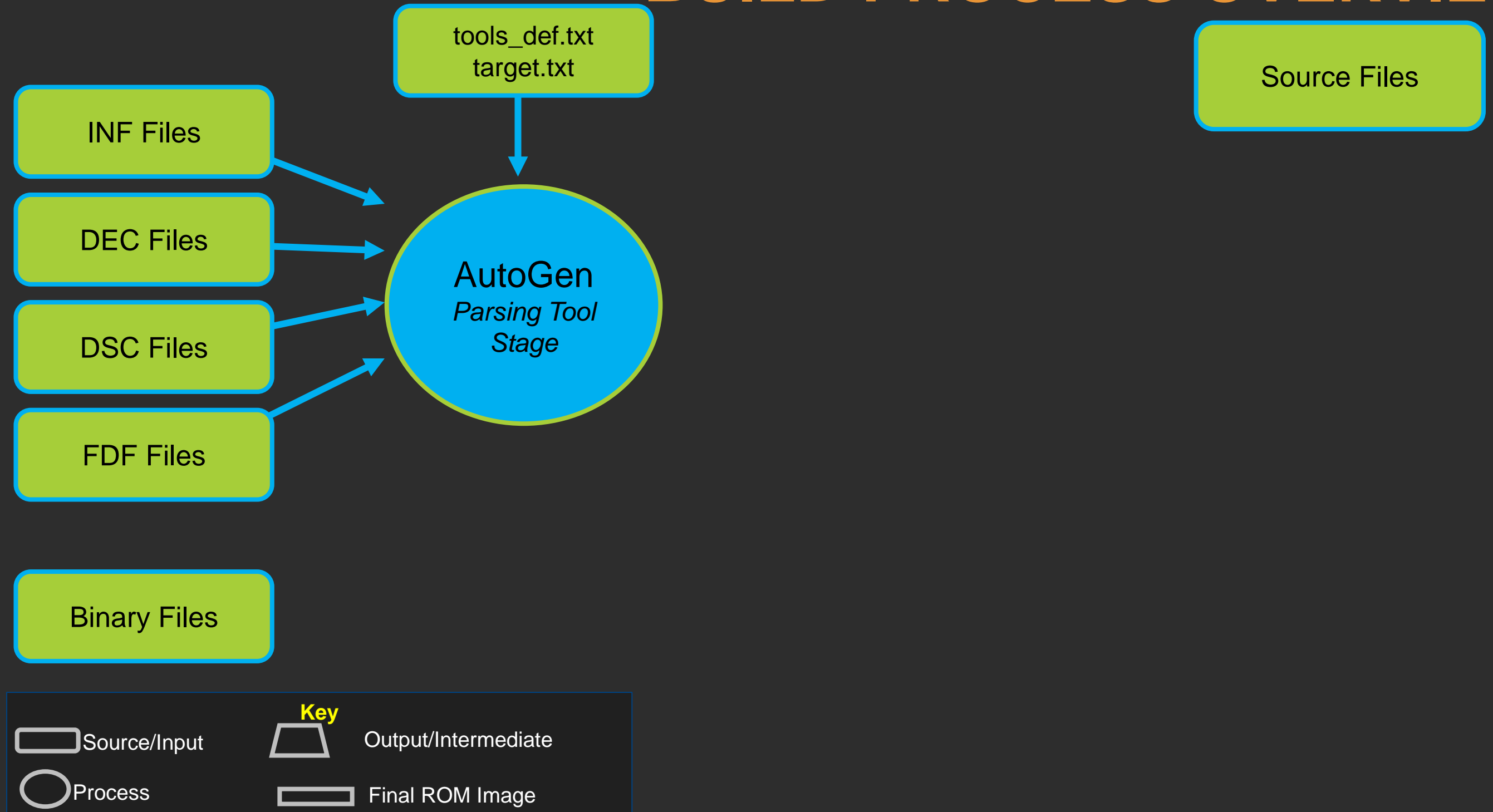
BUILD PROCESS

EDK II Process and Build Text Files

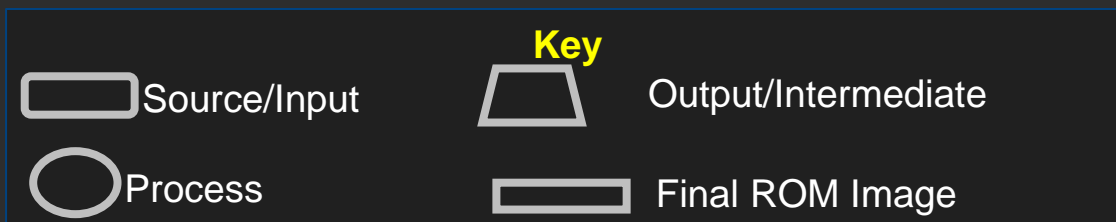
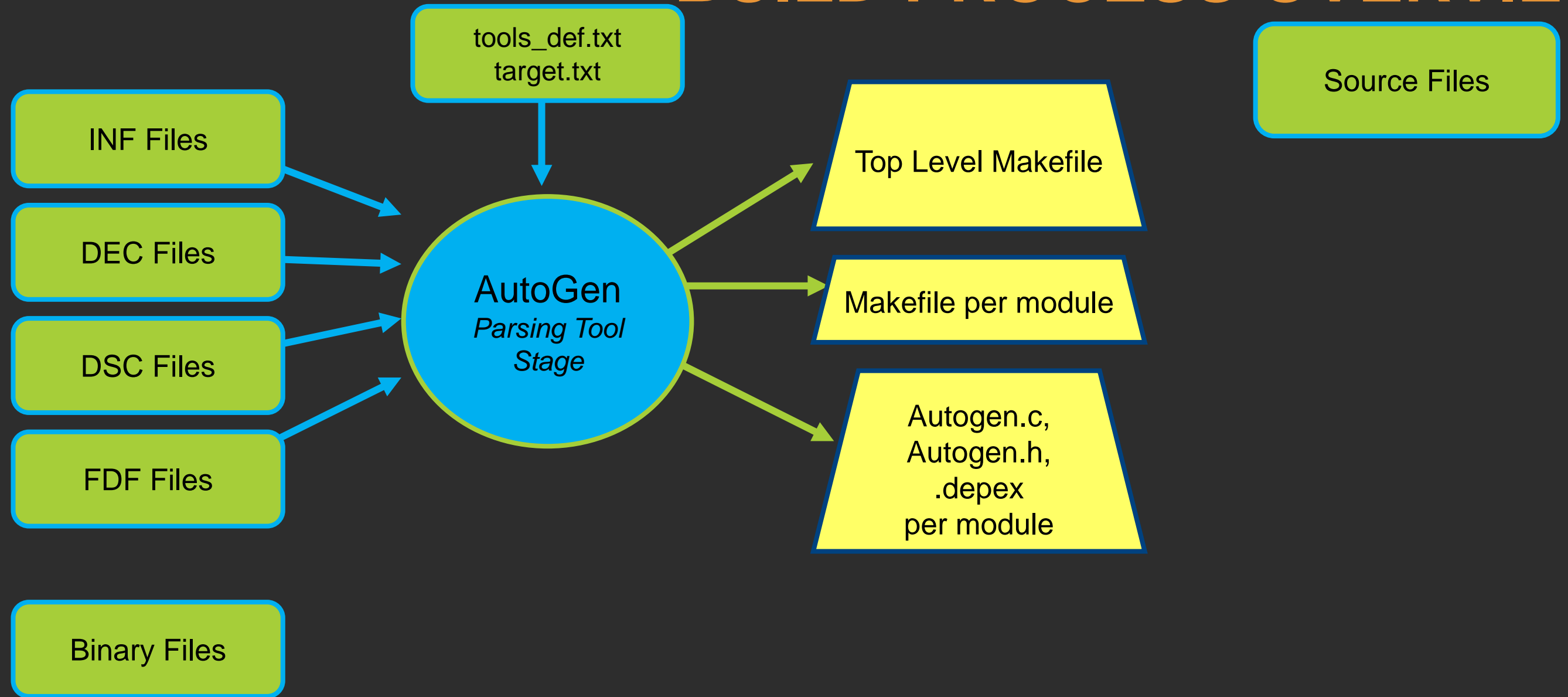
BUILD PROCESS OVERVIEW



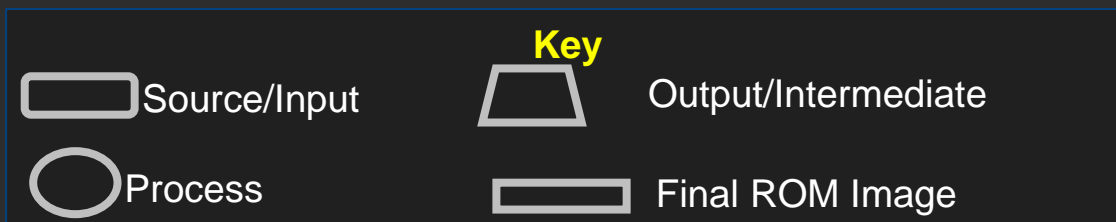
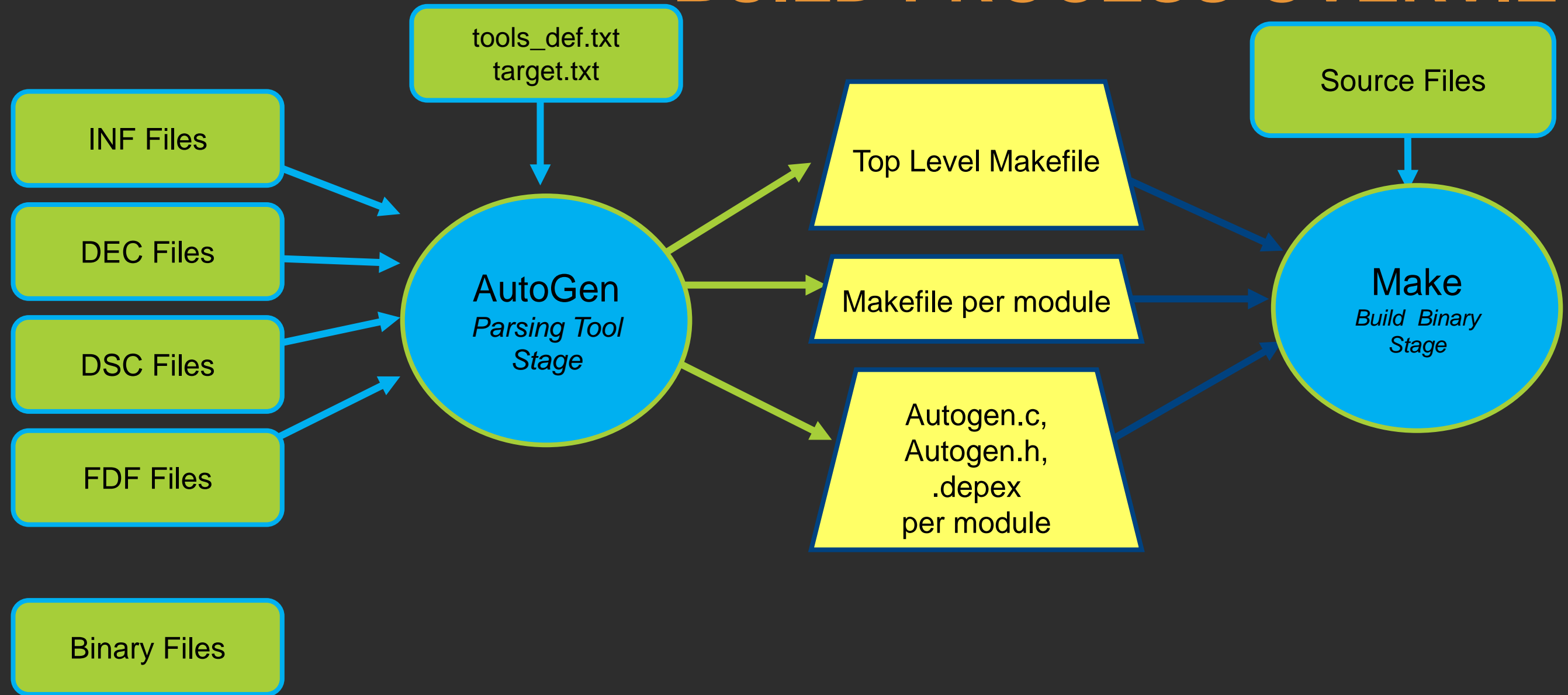
BUILD PROCESS OVERVIEW



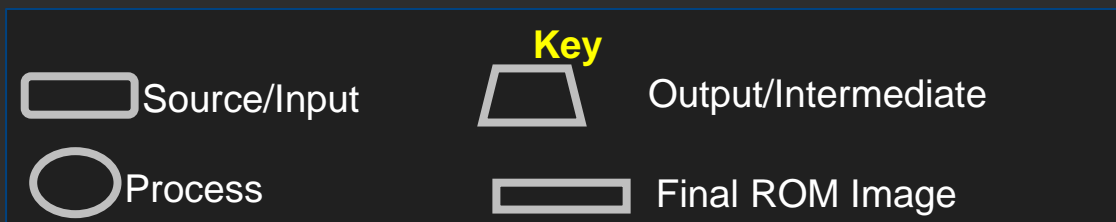
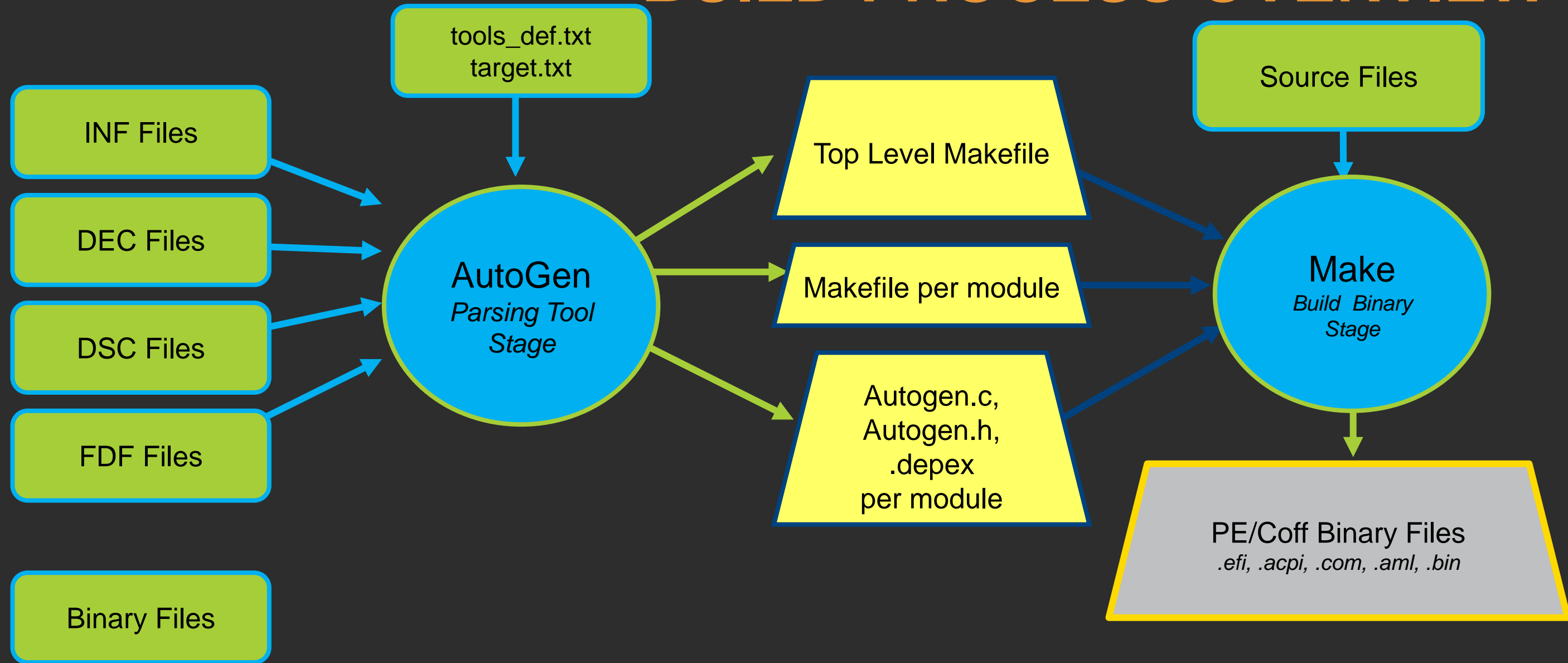
BUILD PROCESS OVERVIEW



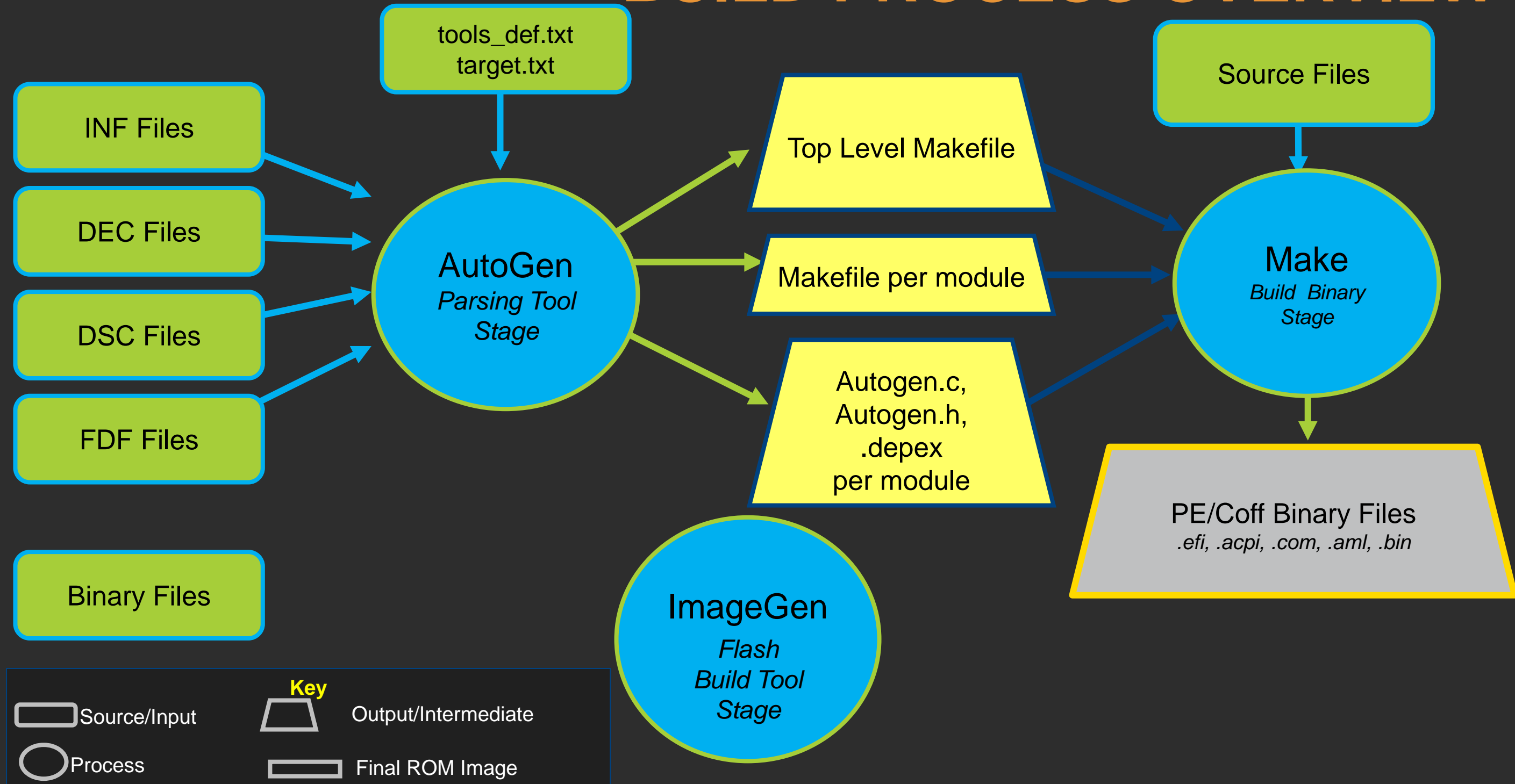
BUILD PROCESS OVERVIEW



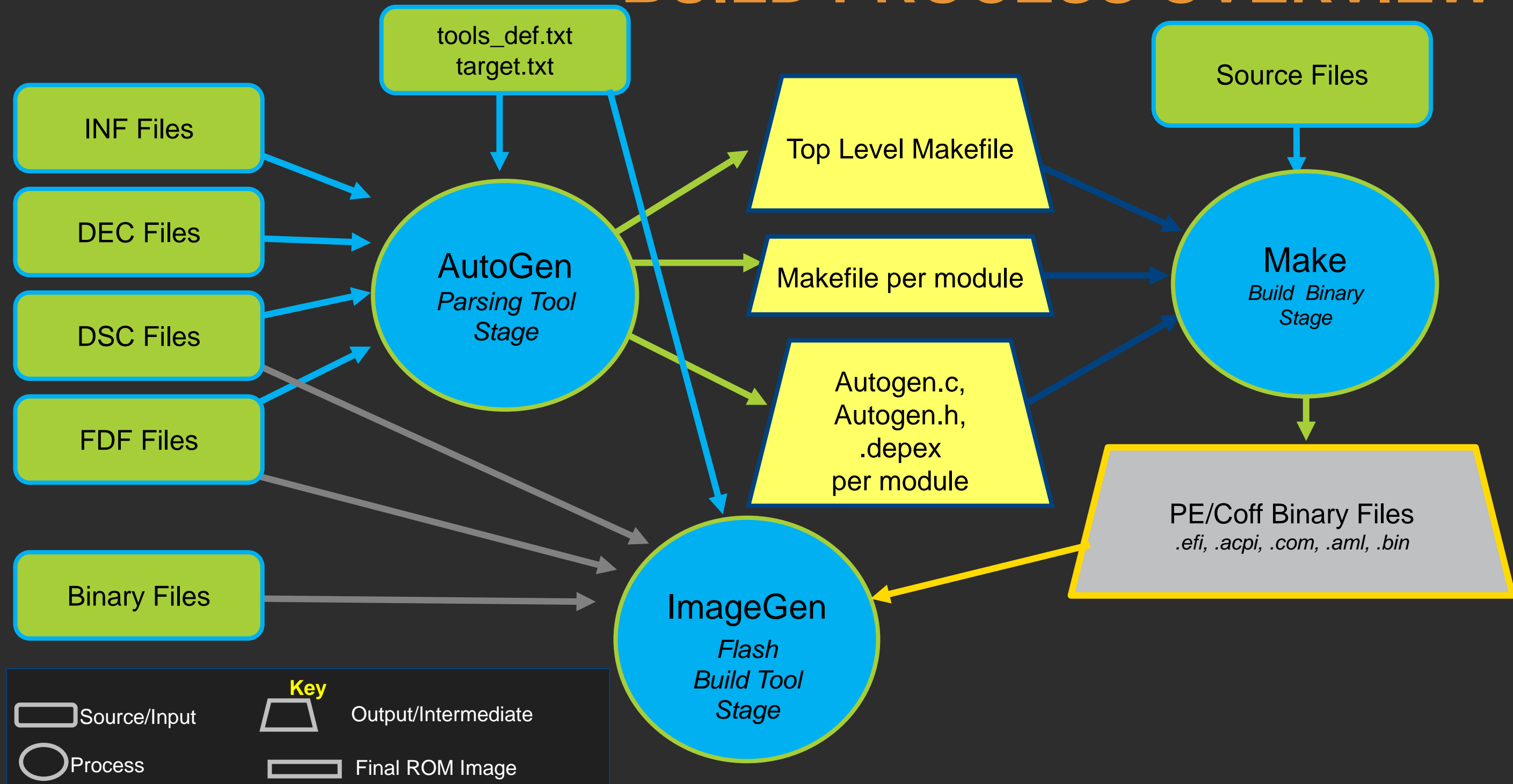
BUILD PROCESS OVERVIEW



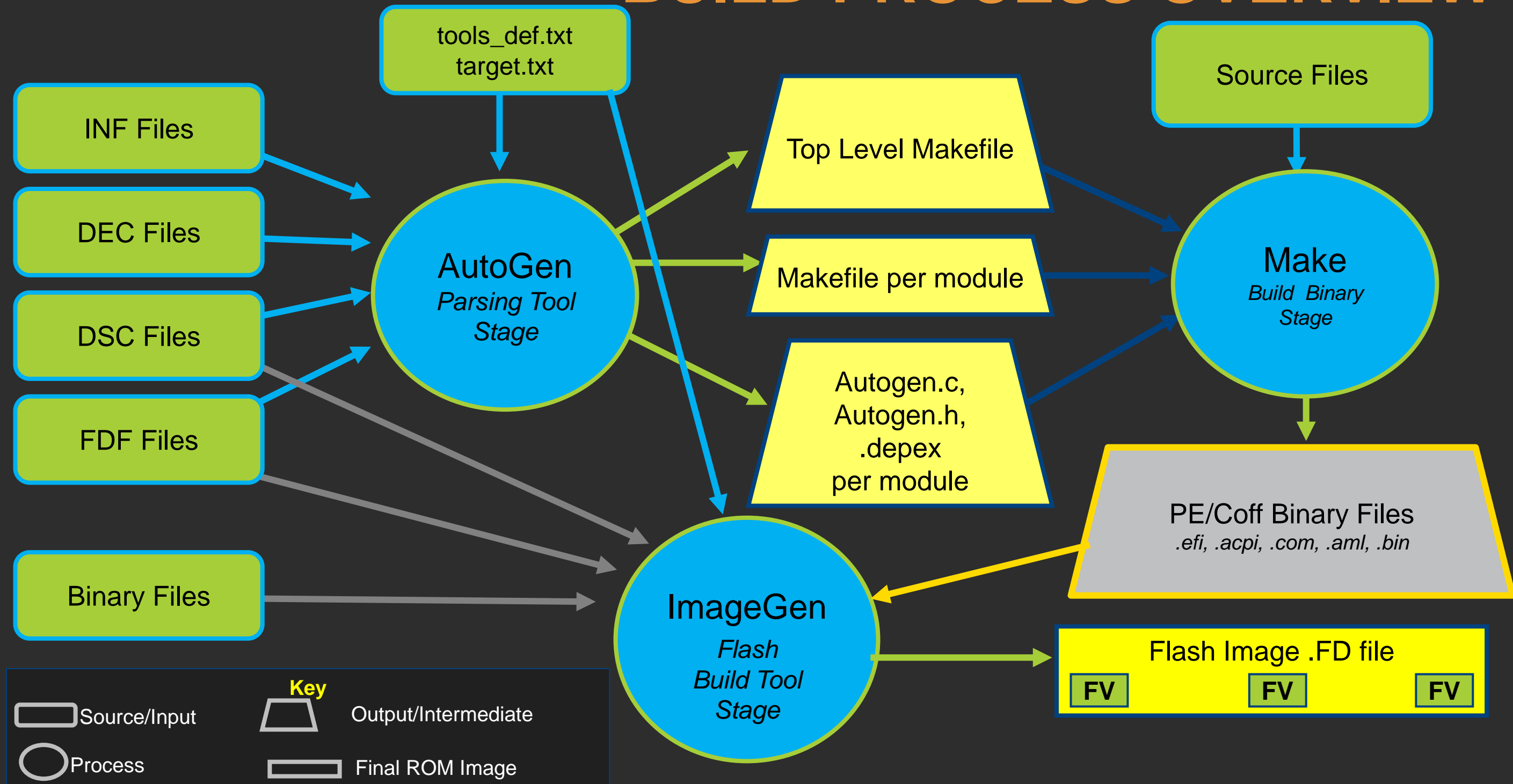
BUILD PROCESS OVERVIEW



BUILD PROCESS OVERVIEW



BUILD PROCESS OVERVIEW



BASIC BUILD STEPS

Platform

1. Navigate to root of EDK II workspace
2. Make the BaseTools
3. Run **edksetup**
4. Run **build**
5. **Output:** firmware image (FD) file under **Build** directory

Module

1. Navigate to root of EDK II workspace
2. Make the BaseTools
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 must be in .dsc components

BUILD OUTPUT LOCATION

```
Build /OvmfX64 /DEBUG_MYTOOLS /FV
```

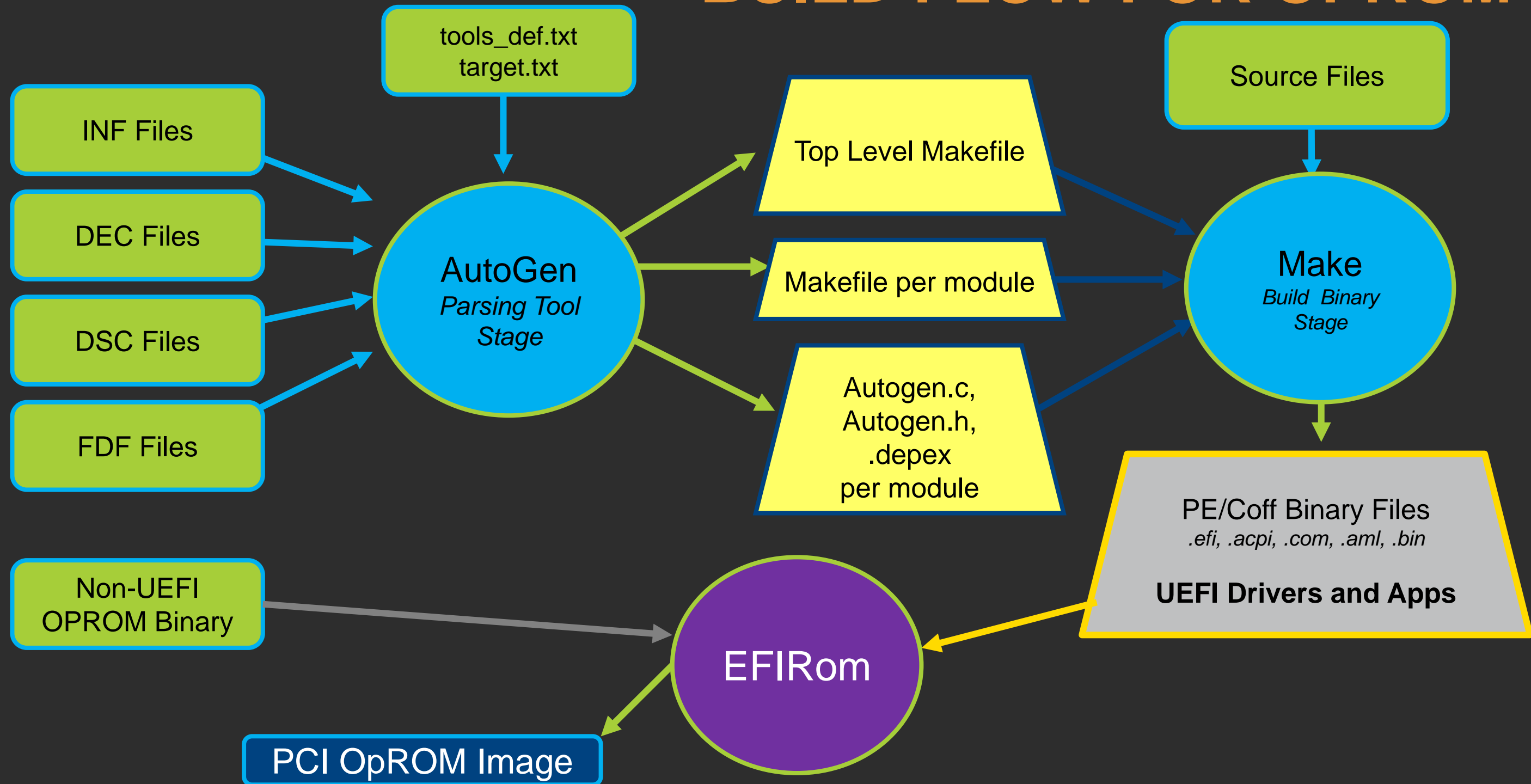
```
Build /Ovmf1 /DEBUG_MYTOOLS /IA321/Pkg /ModuleName /Foo /DEBUG
```

```
Build /Ovmf1 /DEBUG_MYTOOLS /IA321 /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 X64	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 FLOW FOR OPROM



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
```

#build--h-command

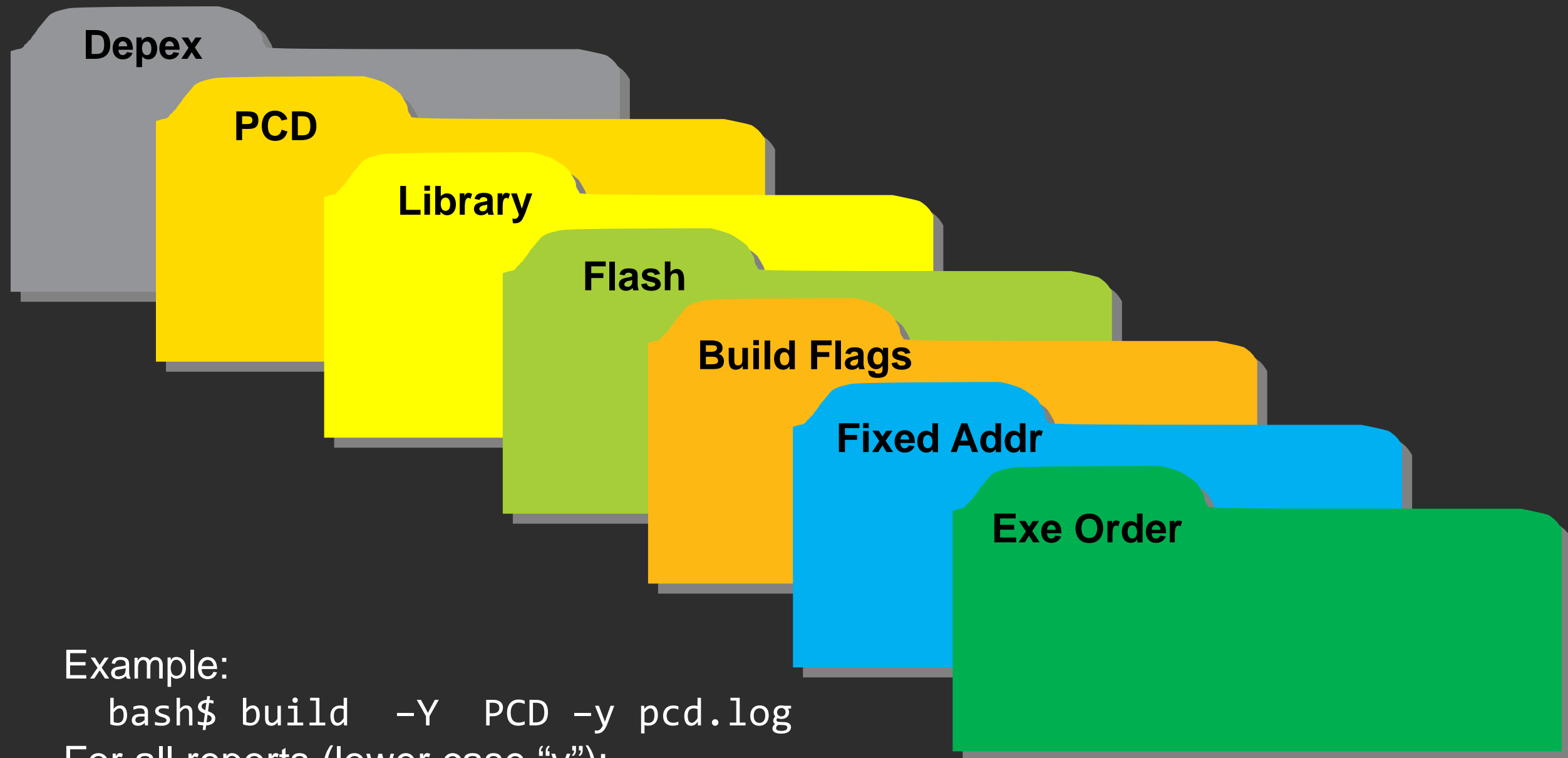
Usage: build.exe [options] [all|fds|genc|genmake|clean|cleanall|cleanlib|modules|libraries|run]

Copyright (c) 2007 - 2018, 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, ARM, AARCH64, RISCV64
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.

USING BUILD -Y COMMAND



Example:

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

For all reports (lower case “y”):

```
bash$ build -y MyReport.log
```

USING BUILD -Y FOR REPORTS

- Scroll through examples of reports from the Build -Y commands

B_01_depex.log.md#build--y-depex

Platform Summary

Platform Name:	EmulatorPkg
Platform DSC Path:	c:\fw\edk2-ws\edk2\EmulatorPkg\EmulatorPkg.dsc
Architectures:	X64
Tool Chain:	VS2015x86
Target:	DEBUG
SKUID:	DEFAULT
DefaultStore:	STANDARD
Output Path:	c:\fw\edk2-ws\Build\EmulatorX64
Build Environment:	Windows-10-10.0.18362-SP0
Build Duration:	00:00:42
AutoGen Duration:	00:00:14
Make Duration:	00:00:27
GenFds Duration:	00:00:02
Report Content:	DEPEX

B_01_Pcd.log.md#build--y-pcd

Platform Summary

Platform Name:	EmulatorPkg
Platform DSC Path:	c:\fw\edk2-ws\edk2\EmulatorPkg\EmulatorPkg.dsc
Architectures:	X64
Tool Chain:	VS2015x86
Target:	DEBUG
SKUID:	DEFAULT
DefaultStore:	STANDARD
Output Path:	c:\fw\edk2-ws\Build\EmulatorX64
Build Environment:	Windows-10-10.0.18362-SP0
Build Duration:	00:00:41
AutoGen Duration:	00:00:11
Make Duration:	00:00:27
GenFds Duration:	00:00:02
Report Content:	PCD

B_01_Library.log.md#build--y-library

Platform Summary

Platform Name:	EmulatorPkg
Platform DSC Path:	c:\fw\edk2-ws\edk2\EmulatorPkg\EmulatorPkg.dsc
Architectures:	X64
Tool Chain:	VS2015x86
Target:	DEBUG
SKUID:	DEFAULT
DefaultStore:	STANDARD
Output Path:	c:\fw\edk2-ws\Build\EmulatorX64
Build Environment:	Windows-10-10.0.18362-SP0
Build Duration:	00:00:36
AutoGen Duration:	00:00:11
Make Duration:	00:00:22
GenFds Duration:	00:00:02
Report Content:	LIBRARY

B_01_flash.log.md#build--y-flash

Platform Summary

Platform Name:	EmulatorPkg
Platform DSC Path:	c:\fw\edk2-ws\edk2\EmulatorPkg\EmulatorPkg.dsc
Architectures:	X64
Tool Chain:	VS2015x86
Target:	DEBUG
SKUID:	DEFAULT
DefaultStore:	STANDARD
Output Path:	c:\fw\edk2-ws\Build\EmulatorX64
Build Environment:	Windows-10-10.0.18362-SP0
Build Duration:	00:00:36
AutoGen Duration:	00:00:11
Make Duration:	00:00:23
GenFds Duration:	00:00:02
Report Content:	FLASH

B_01_Build_Flags.log.md#build--y-build_flags

Platform Summary

Platform Name:	EmulatorPkg
Platform DSC Path:	c:\fw\edk2-ws\edk2\EmulatorPkg\EmulatorPkg.dsc
Architectures:	X64
Tool Chain:	VS2015x86
Target:	DEBUG
SKUID:	DEFAULT
DefaultStore:	STANDARD
Output Path:	c:\fw\edk2-ws\Build\EmulatorX64
Build Environment:	Windows-10-10.0.18362-SP0
Build Duration:	00:00:37
AutoGen Duration:	00:00:11
Make Duration:	00:00:23
GenFds Duration:	00:00:02
Report Content:	BUILD_FLAGS

build -Y FIXED_ADDRESS

B_01_Fixed_Adress.log.md#build--y-fixed_address

Platform Summary

Platform Name:	EmulatorPkg
Platform DSC Path:	c:\fw\edk2-ws\edk2\EmulatorPkg\EmulatorPkg.dsc
Architectures:	X64
Tool Chain:	VS2015x86
Target:	DEBUG
SKUID:	DEFAULT
DefaultStore:	STANDARD
Output Path:	c:\fw\edk2-ws\Build\EmulatorX64
Build Environment:	Windows-10-10.0.18362-SP0
Build Duration:	00:00:34
AutoGen Duration:	00:00:11
Make Duration:	00:00:21
GenFds Duration:	00:00:02
Report Content:	FIXED_ADDRESS

build -Y EXECUTION_ORDER

[B_01_Execution_Order.log.md#build--y-execution_order](#)

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](#)

Platform Summary

Platform Name:	EmulatorPkg
Platform DSC Path:	c:\fw\edk2-ws\edk2\EmulatorPkg\EmulatorPkg.dsc
Architectures:	X64
Tool Chain:	VS2015x86
Target:	DEBUG
SKUID:	DEFAULT
DefaultStore:	STANDARD
Output Path:	c:\fw\edk2-ws\Build\EmulatorX64
Build Environment:	Windows-10-10.0.18362-SP0
Build Duration:	00:00:34
AutoGen Duration:	00:00:12
Make Duration:	00:00:21
GenFds Duration:	00:00:02
Report Content:	EXECUTION_ORDER

build -y MyReport.log

B_01_MyReport.log.md#build--y-myreportlog

Platform Summary

Platform Name:	EmulatorPkg
Platform DSC Path:	c:\fw\edk2-ws\edk2\EmulatorPkg\EmulatorPkg.dsc
Architectures:	X64
Tool Chain:	VS2015x86
Target:	DEBUG
SKUID:	DEFAULT
DefaultStore:	STANDARD
Output Path:	c:\fw\edk2-ws\Build\EmulatorX64
Build Environment:	Windows-10-10.0.18362-SP0
Build Duration:	00:00:35
AutoGen Duration:	00:00:12
Make Duration:	00:00:21
GenFds Duration:	00:00:02
Report Content:	PCD, LIBRARY, BUILD_FLAGS, DEPEX, HASH, FLASH, FIXED_ADDRESS

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
GenPatchPcdTable	Tool works together with PatchPcdValue tool to set the specific value of a patchable PCD into the binary EFI image
PatchPcdValue	used to Patch the specific value into the binary

SUMMARY

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

Questions?



Return to Main Training Page



Return to Training Table of contents for next presentation

[Link](#)



ACKNOWLEDGEMENTS

Redistribution and use in source (original document form) and 'compiled' forms (converted to PDF, epub, HTML and other formats) with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code (original document form) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.

Redistributions in compiled form (transformed to other DTDs, converted to PDF, epub, HTML and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS DOCUMENTATION IS PROVIDED BY TIANOCORE PROJECT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL TIANOCORE PROJECT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 2021, Intel Corporation. All rights reserved.

BACKUP

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](#)

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/edk2-ws/edk2/`

`MdePkg/`

`. . .`

`MdeModulePkg/`

`.Dec`

`ModuleAbc /`

`.Inf`

`OvmfPkg /`

`.Dec`

`.Dsc`

`.Fdf`

`ModuleNtXyz /`

`.Inf`

`ModuleAbc /`

`.Inf`

EDK II BUILD: MAKE STAGE

Uses assemblers/compiler/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

