

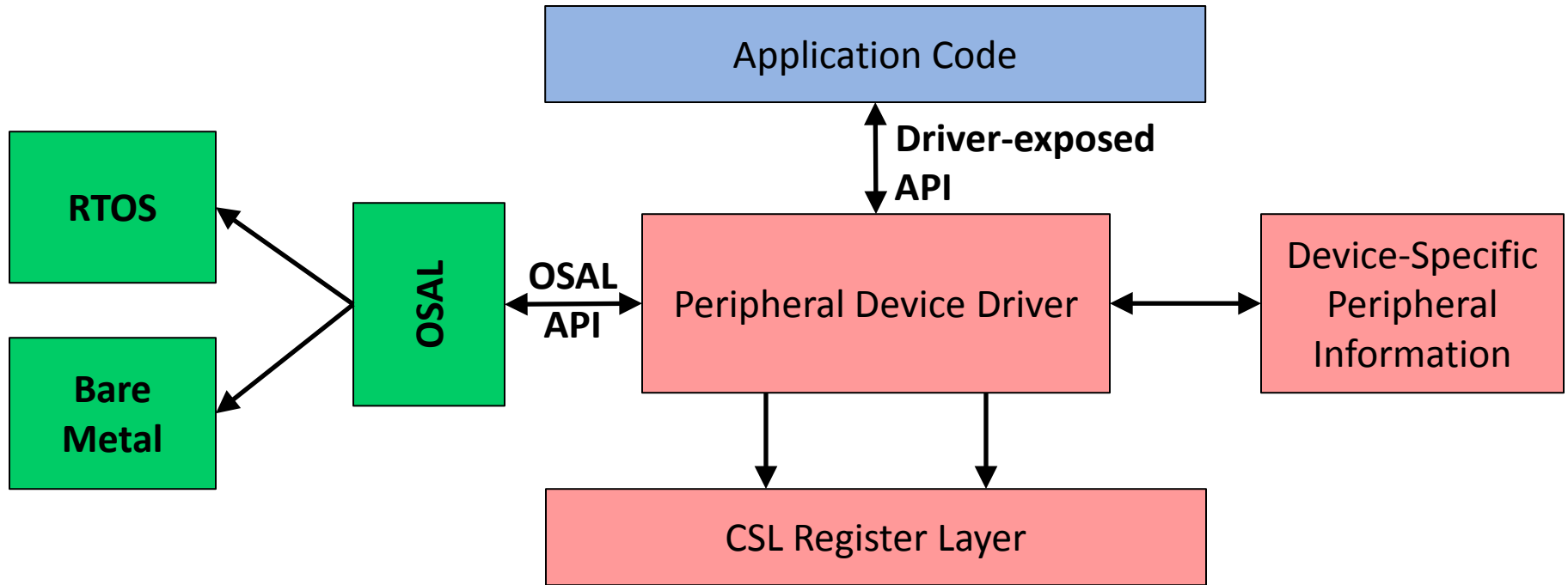
# Introduction to Processor SDK RTOS Part 2

SoC (System on Chip) Drivers

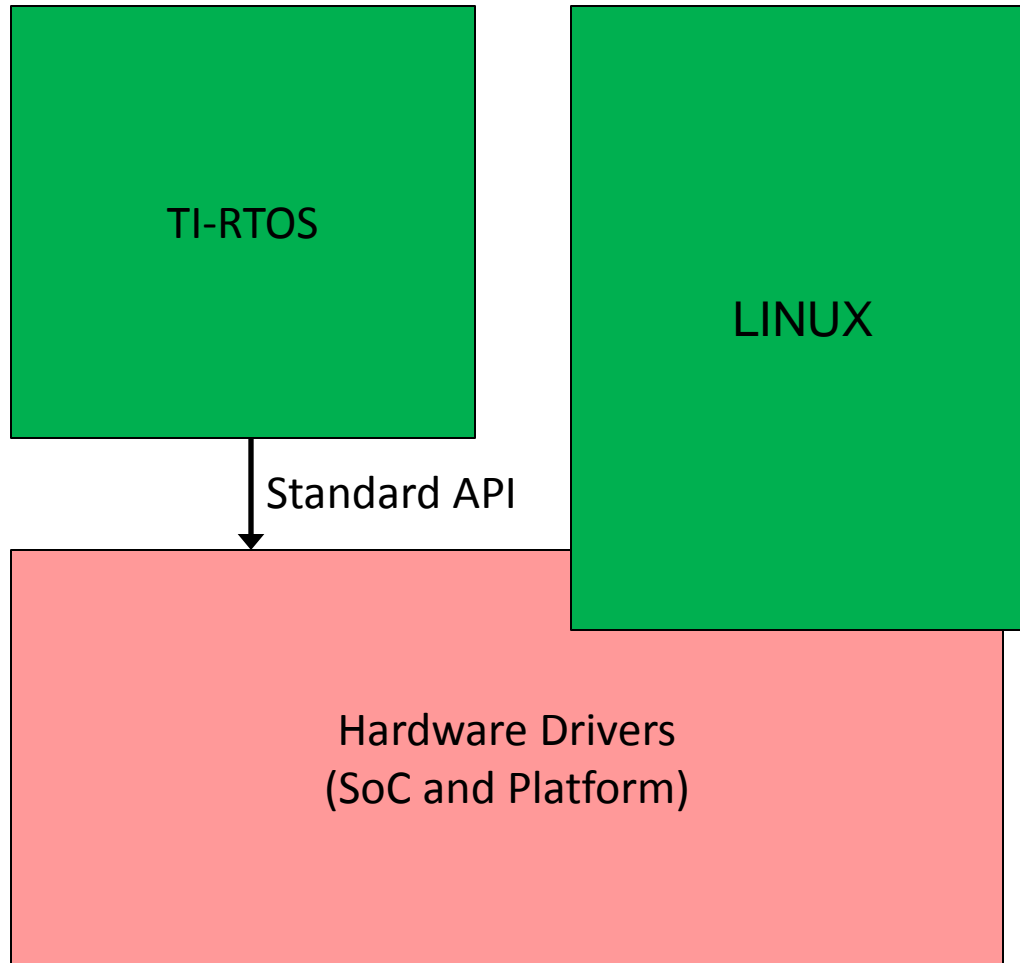
# Agenda

- Drivers Overview
- Chip Support Library (CSL) Layer
- Low Level Driver (LLD) Layer
- OS Abstraction Layer (OSAL)
- Board Library
- Secondary Boot Loader (SBL)
- Board Diagnostics

# SoC Driver Interfaces



# SoC Drivers & Operating System

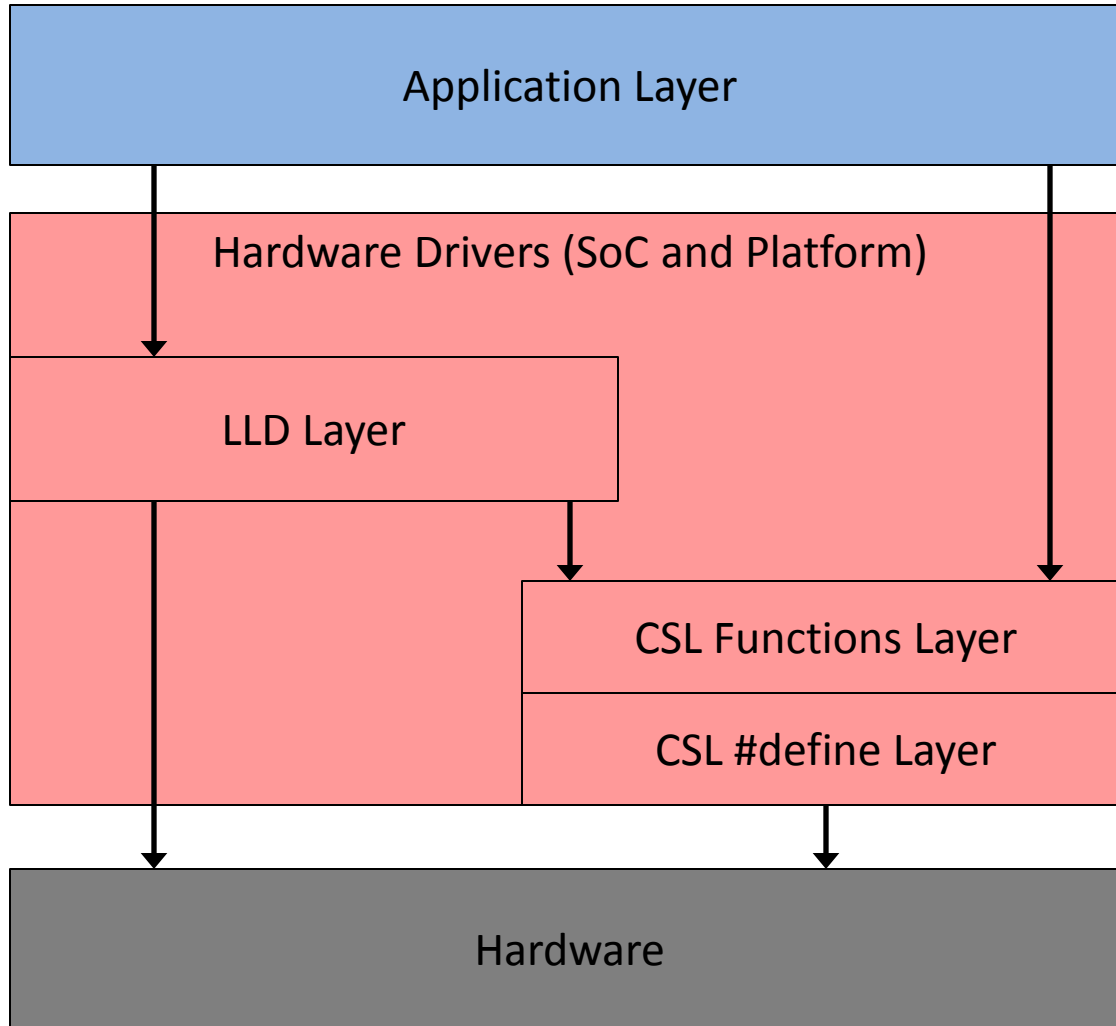


- TI-RTOS exposes the physical drivers to the applications.
- Linux does not.

# CSL Layer

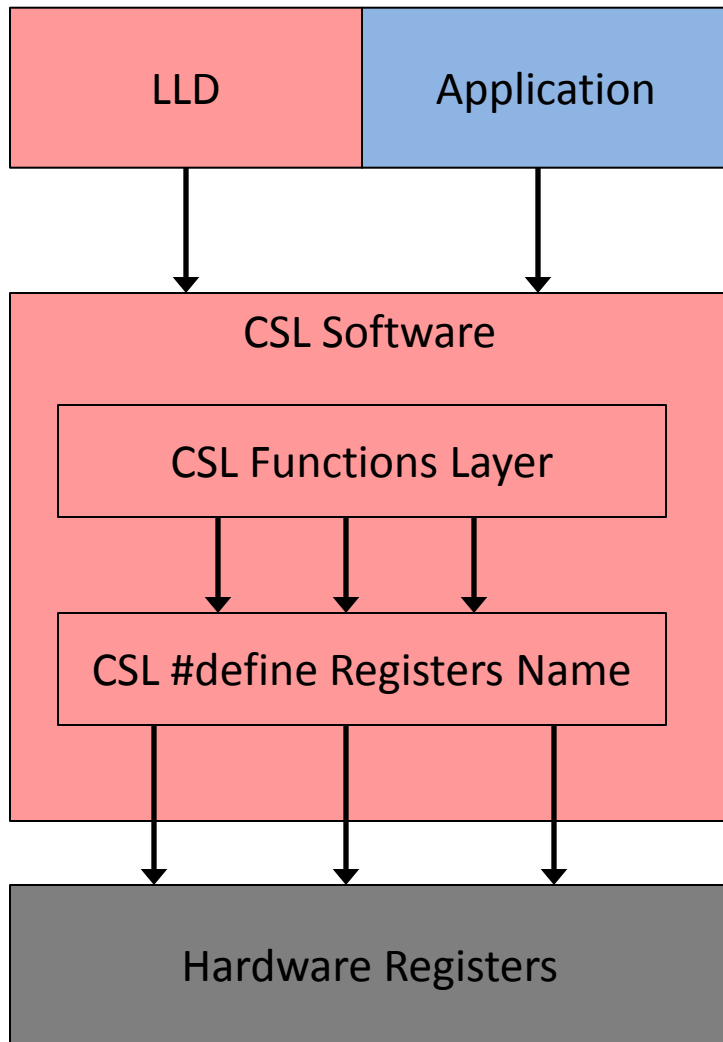
Introduction to Processor SDK RTOS Part 2

# LLD Structure & Chip Support Library (CSL)



- Low Level Drivers (LLD) hide the details of CSL from the application.
- Some IP and peripherals do not have LLD. The application uses CSL directly.
- Some LLD can access the hardware directly (and not via CSL).

# Chip Support Library (CSL) Overview



- (Almost) All peripherals are controlled by Memory Mapped Registers (MMR).
- The MMR may have different addresses for different (future) SOCs.
- The CSL has two layers:
  - The first layer assigns a standard name to MMR.
  - The second layer is a set of functions to manipulate these registers.
- The application or LLD needs only to know the API of the CSL functions.

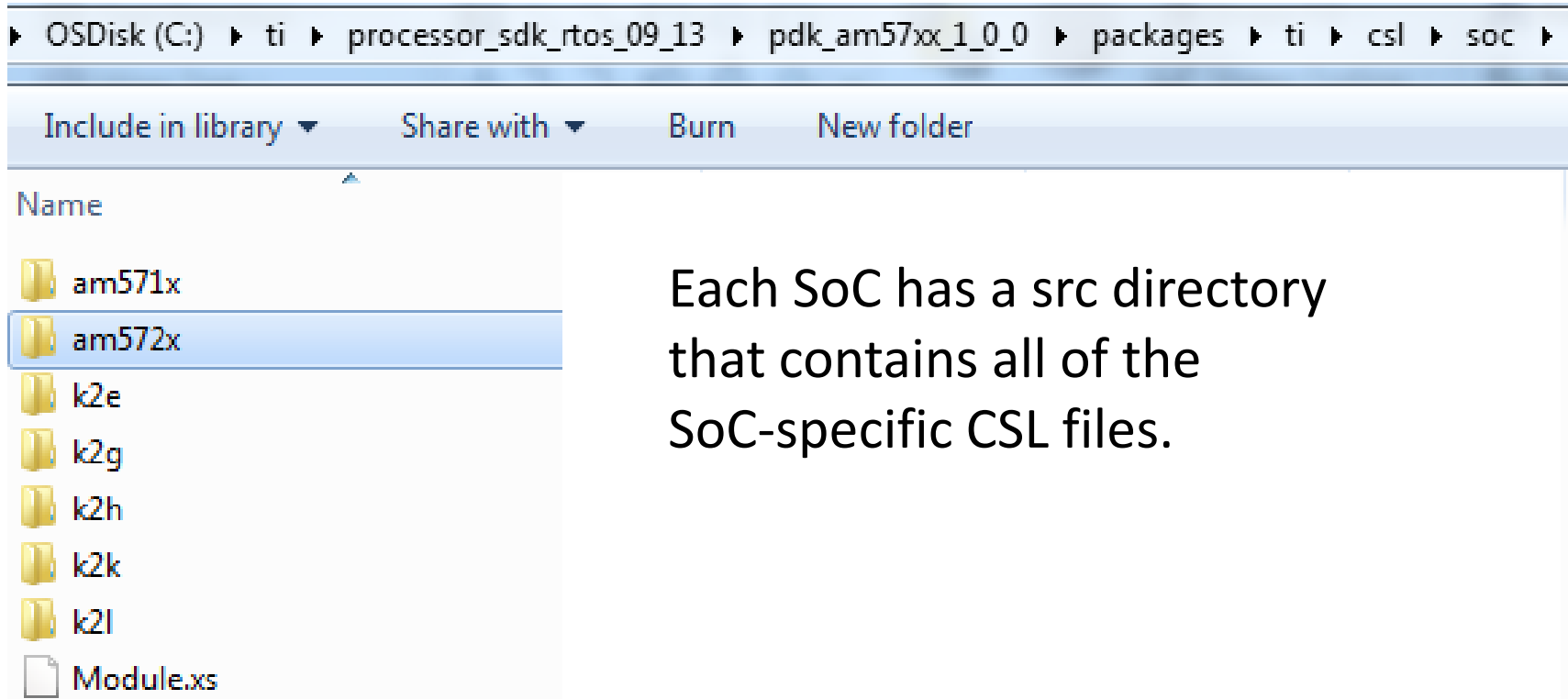
# CSL Registers #define

- The MMR address depends on the SoC family.
- The include file and SoC-specific CSL support files are located in a directory like the one shown:  

```
processor_sdk_rtos_09_13\pdk_am57xx_1_0_0\packages\ti\cs1
```
- SoC-specific CSL support files are located in one of the src directories (see next slide).



# CSL Registers #define SoC-dependent



Each SoC has a src directory that contains all of the SoC-specific CSL files.

# Example cs1r\_soc\_ipu\_baseaddress.h

C:\ti\processor\_sdk\_rtos\_09\_13\pdk\_am57xx\_1\_0\_0\packages\ti\cs1\soc\am572x\src

```
#define CSL_IPU_IPU1_C1_DWT_REGS (0xe0001000U)
#define CSL_IPU_IPU1_C1_DWT_SIZE (0x1000U)
#define CSL_IPU_IPU1_C0_DWT_REGS (0xe0001000U)
#define CSL_IPU_IPU1_C0_DWT_SIZE (0x1000U)
#define CSL_IPU_IPU1_C1_FPB_REGS (0xe0002000U)
#define CSL_IPU_IPU1_C1_FPB_SIZE (0x1000U)
#define CSL_IPU_IPU1_C0_FPB_REGS (0xe0002000U)
#define CSL_IPU_IPU1_C0_FPB_SIZE (0x1000U)
#define CSL_IPU_IPU1_C0_INTC_REGS (0xe000e000U)
#define CSL_IPU_IPU1_C0_INTC_SIZE (0x1000U)
#define CSL_IPU_IPU1_C1_INTC_REGS (0xe000e004U)
#define CSL_IPU_IPU1_C1_INTC_SIZE (0xffcU)
#define CSL_IPU_IPU1_C0_ICECRUSHER_REGS (0xe0042000U)
```

Gives the location of the IPU interrupt registers' block.



# Interrupt Interface Structures & Constant:

## CSL\_cpInt.h

```
#ifndef _CSL_CPINTC_H_
#define _CSL_CPINTC_H_

#ifdef __cplusplus
extern "C" {
#endif

#include <ti/csl/soc.h>
#include <ti/csl/csl.h>
#include <ti/csl/cslr_cpintc.h>

/**
 @defgroup CSL_CPINTC_SYMBOL CPINTC Symbols Defined
 @ingroup CSL_CPINTC_API
 */
/**
 @defgroup CSL_CPINTC_DATASTRUCT CPINTC Data Structures
 @ingroup CSL_CPINTC_API
 */
/**
 @defgroup CSL_CPINTC_FUNCTION CPINTC Functions
 @ingroup CSL_CPINTC_API
 */

/** @addtogroup CSL_CPINTC_DATASTRUCT
 @{ */

/** @brief Register Overlay Memory map for the CPINTC0 Registers. */
typedef volatile CSL_CPINTCRegs* CSL_CPINTC_RegsOvly;
```

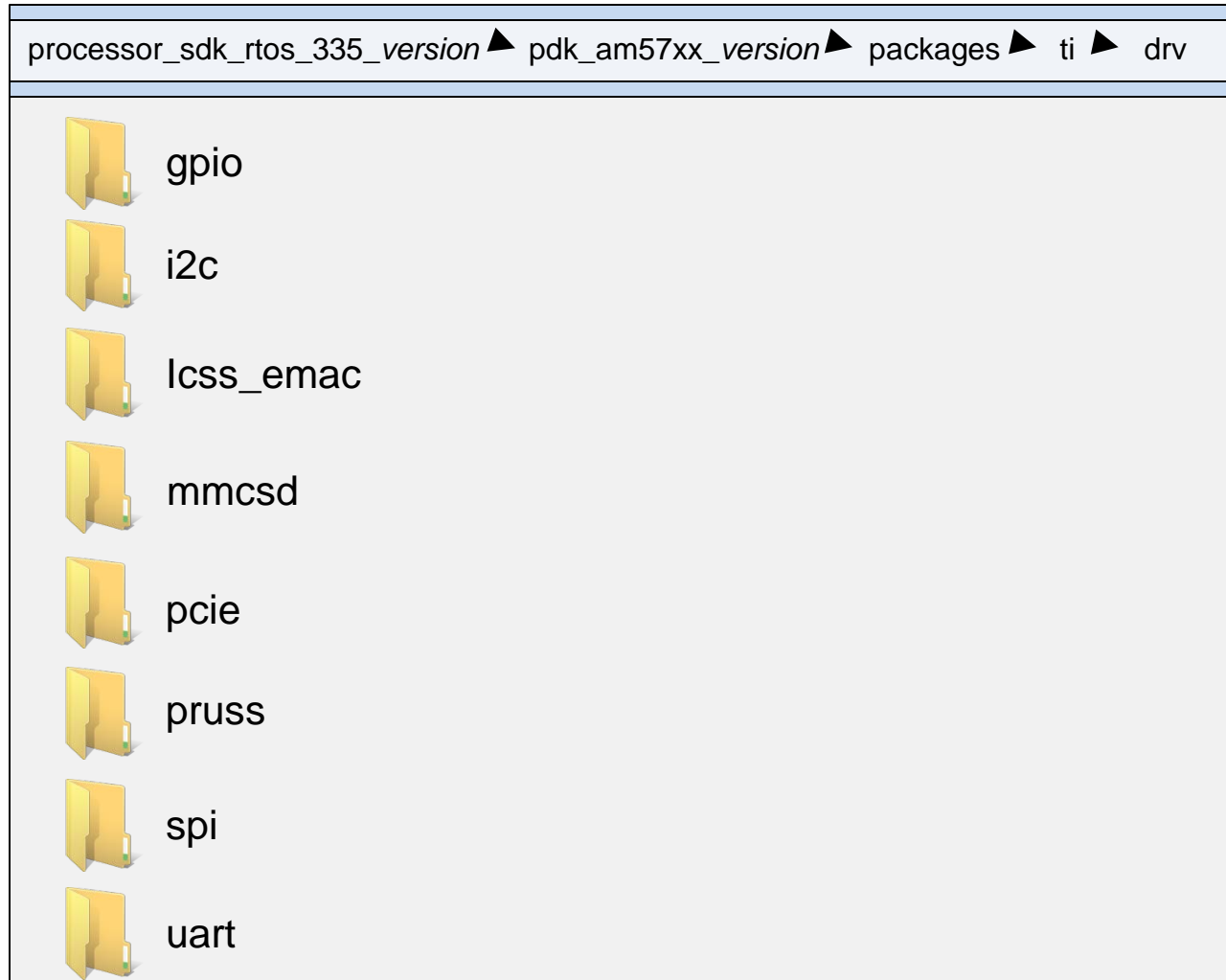
# LLD Layer

Introduction to Processor SDK RTOS Part 2

# Understanding the LLD

- LLDs are hardware drivers that talk directly or via CSL to the hardware registers.
- LLDs are used to abstract hardware implementation (or CSL) details from the user.
- The Real-Time Software Component (RTSC) system enforces a fixed structure. All LLDs are RTSC-compatible, but not dependent on RTSC.
- Most (but not all) LLD code and pre-build libraries are located in the pdk (platform development kit) directory.

# Core-dependent LLD in pdk Directory (SoC-dependent)



# Example Directory Structure: GPIO (1)

The top level directory includes:

- Sub-directories
- Files that are visible to the application
- XDC files that help with XDC building (auto-building) projects using configuration file

Name
build
docs
example
lib
package
soc
src
test
config.bld
GPIO.h
GPIO_osal.h
GPIOver.h
GPIOver.h.xdt
makefile
makefile_armv7
package.bld
package.xdc
package.xs
Settings.h
Settings.xdc
Settings.xdc.xdt



# Example Directory Structure: GPIO (2)

- **build** contains make files to build the generic libraries.
- **docs** contains all user documentation:
  - Software manifests (licensing, export control, etc.)
  - Release notes
  - The Doxygen subdirectory contains a collection of linked HTML documentation files that are generated from the code.
  - Module-specific documents
- **example** contains code that support the example projects.

Name

- build
- docs
- example
- lib
- package
- soc
- src
- test
- config.bld
- GPIO.h
- GPIO\_osal.h
- GPIOver.h
- GPIOver.h.xdt
- makefile
- makefile\_armv7
- package.bld
- package.xdc
- package.xs
- Settings.h
- Settings.xdc
- Settings.xdc.xdt

# Example Directory Structure: GPIO (3)

- **lib** contains libraries for different cores.
- **package** contains files that are used during the XDC building of the module
- **src** contains the SOC-independent source and include files.
- **soc** contains the SOC-dependent source and include files.
- **test** contains files that are part of the example test.

Name

- build
- docs
- example
- lib
- package
- soc
- src
- test
- config.bld
- GPIO.h
- GPIO\_osal.h
- GPIOOver.h
- GPIOOver.h.xdt
- makefile
- makefile\_armv7
- package.bld
- package.xdc
- package.xs
- Settings.h
- Settings.xdc
- Settings.xdc.xdt

# Example GPIO Include File: gpio.h

The include file gpio.h has all the external information needed by the application:

- #define of all values that may be needed by application in order to use GPIO
- Defines all structures that may be used by the application
- Defines the APIs for all the functions

# gpio.h Functions API

```
extern void GPIO_clearInt(unsigned int index);  
extern void GPIO_disableInt(unsigned int index);  
extern void GPIO_enableInt(unsigned int index);  
extern void GPIO_init();  
extern unsigned int GPIO_read(unsigned int index);  
extern void GPIO_setCallback(unsigned int index, GPIO_CallbackFxn callback);  
extern void GPIO_setConfig(unsigned int index, GPIO_PinConfig pinConfig);  
extern void GPIO_toggle(unsigned int index);  
extern void GPIO_write(unsigned int index, unsigned int value);
```

# Developing Application Using LLD Code

- Get a resource (open, create).
- Configure the resource.
  - Understand the structure of the parameters of the configuration function (example to follow)
- If there are dependencies, configure dependencies.
- Use in run time.
  - Refer to the Processor SDK examples to understand what needs to be done.

# API Flow for Generic LLD: UART (1/2)

Configure the board specific interface parameters from the SOC directory.

For UART, the UART\_soc.c file sets the RX and TX pin for communication.

Use init function to create an handle for the instance of the LLD.

```
UART_init()  
From directory:  
pdk_am57xx_1_0_0\packages\ti\drv\uart\src
```

Configure the parameters of the LLD

```
UART_Params_init()  
From directory:  
pdk_am57xx_1_0_0\packages\ti\drv\uart\src
```

# API Flow for Generic LLD: UART (2/2)

Open an instance of the LLD and make it ready to start working.

```
UART_open(instance,param)  
Open the handle for the UART  
instance
```

Run-time functions:  
Read, Write

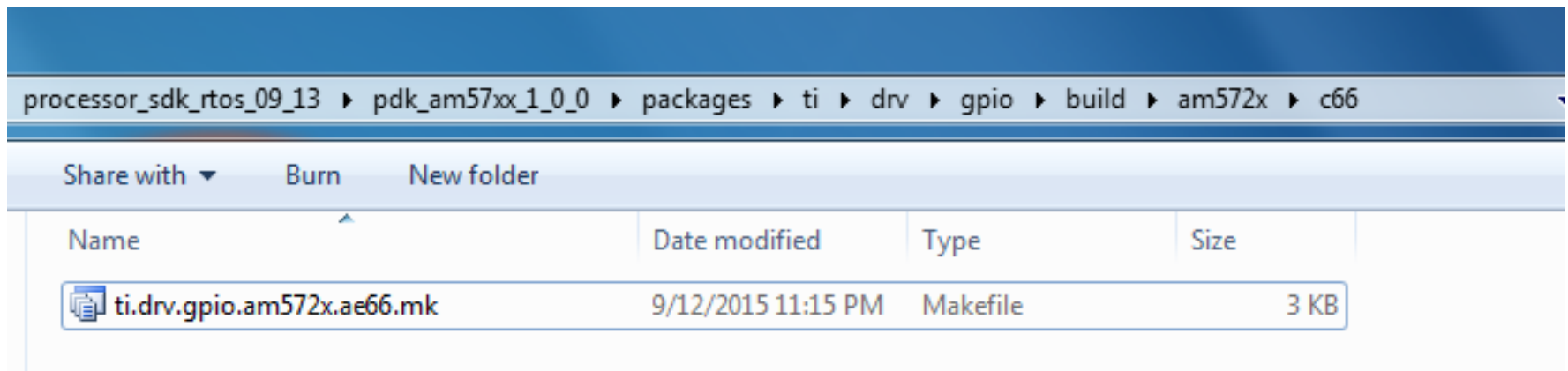
```
UART_read(UART_handle handle,  
void *buffer, size_t size)  
UART_write(UART_handle handle,  
void *buffer, size_t size)
```

Close the LLD instance.

```
UART_close(UART_handle handle)
```

# Building the LLD Library

Each LLD for each SOC and each core has a makefile that builds the LLD library.



In the makefile, use `-g` and no optimization to debug driver code and use optimization for production build.

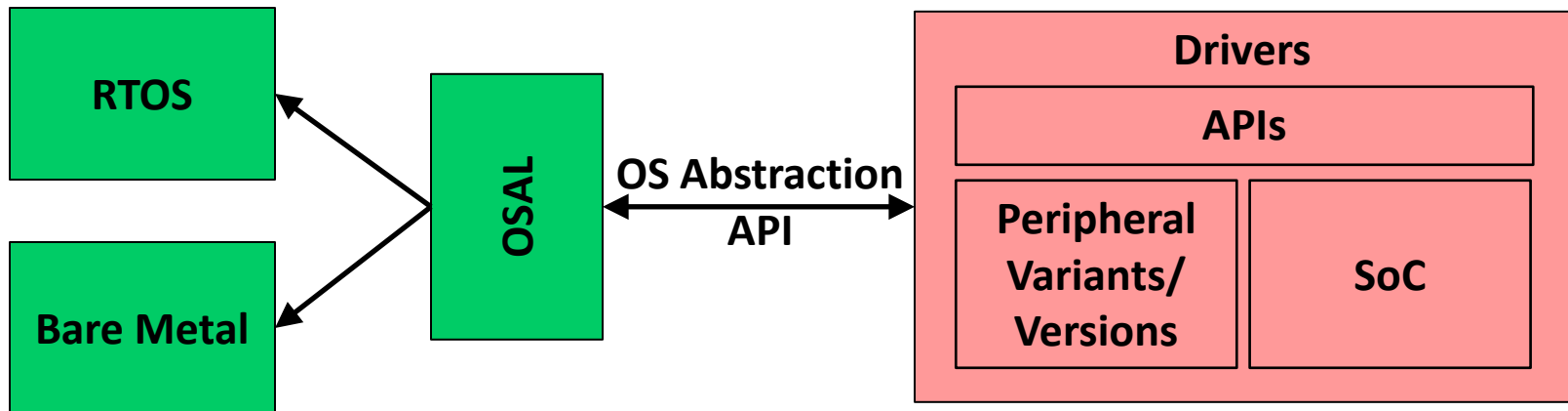


# Operating System Abstraction Layer (OSAL)

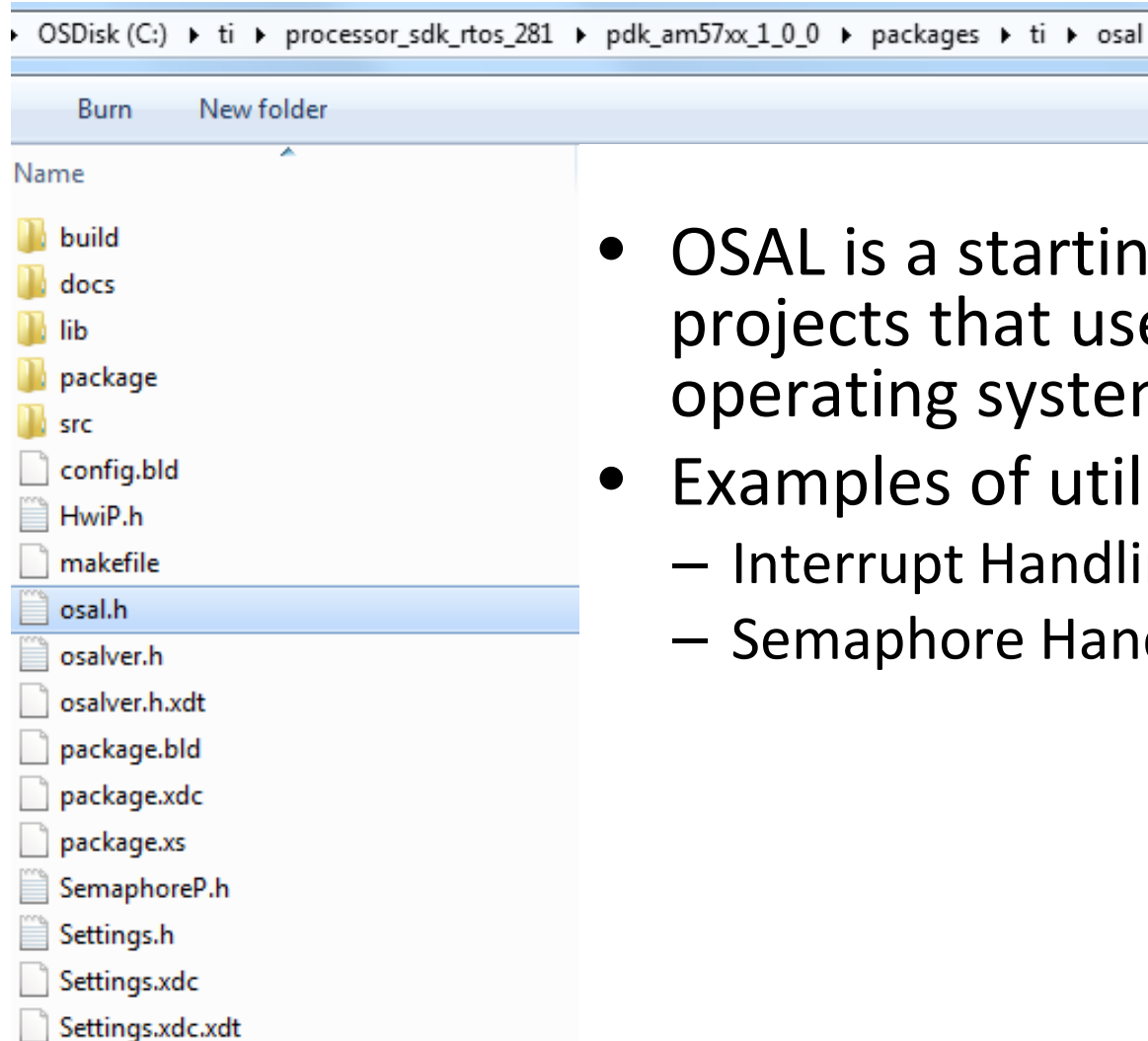
Introduction to Processor SDK RTOS Part 2

# OSAL: Makes Driver OS-independent

- When a driver requires an operating system utility, OSAL provides a standard interface to any OS.
- Operating system can be TI-RTOS or any generic OS ... or even bare metal.



# OSAL: Makes Driver OS-independent (2/2)

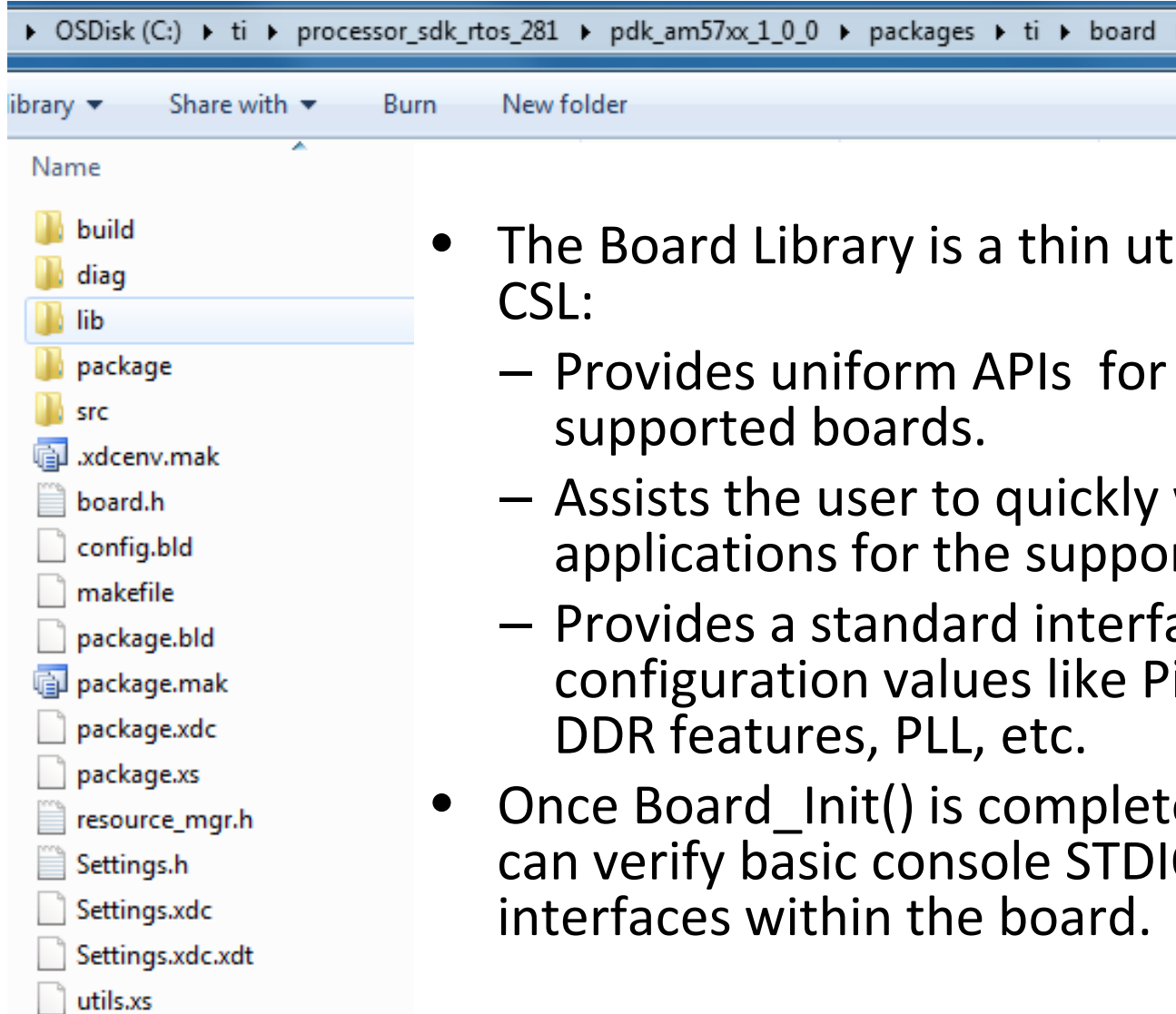


- OSAL is a starting point for projects that use a different operating system.
- Examples of utilities:
  - Interrupt Handling
  - Semaphore Handling

# Board Library

Introduction to Processor SDK RTOS Part 2

# Board Library

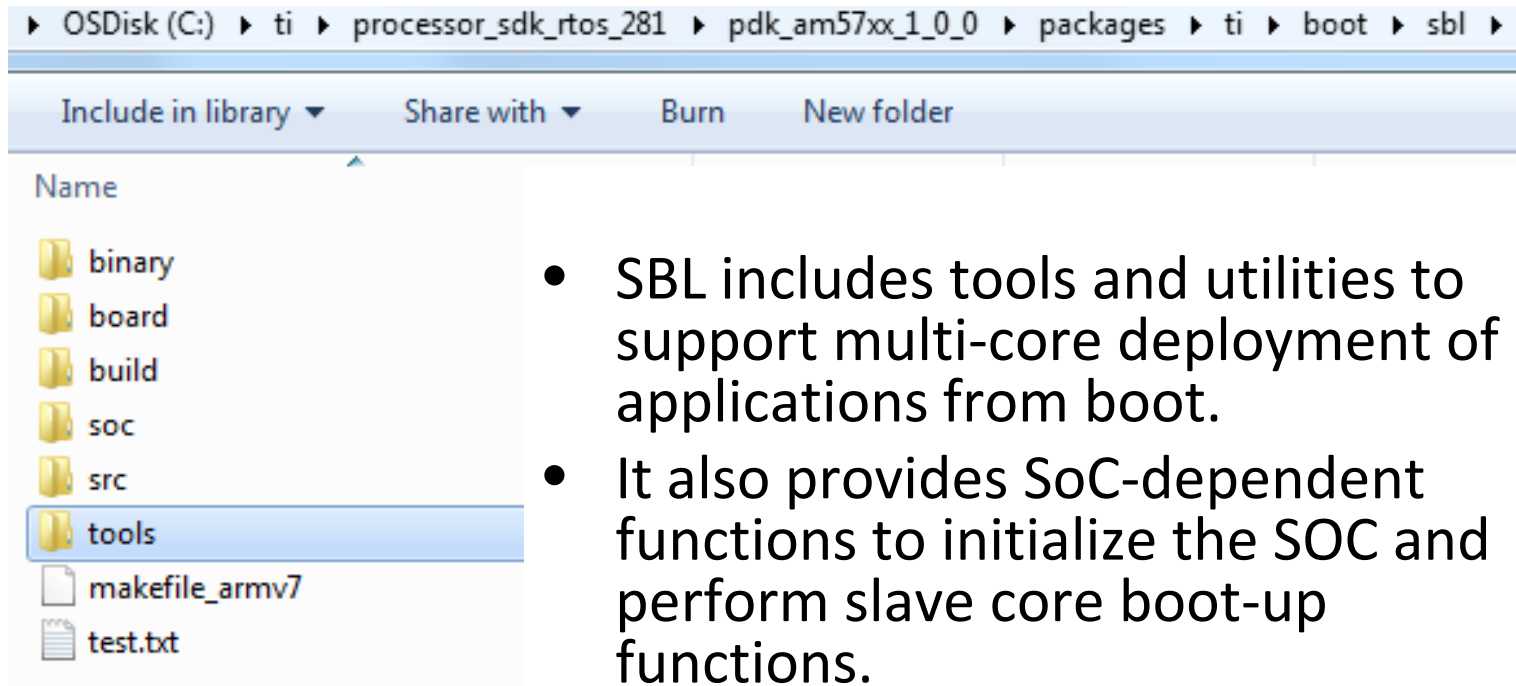


- The Board Library is a thin utility layer on top of CSL:
  - Provides uniform APIs for configuration of all supported boards.
  - Assists the user to quickly write portable applications for the supported boards.
  - Provides a standard interface to basic board configuration values like Pinmux, clocking, DDR features, PLL, etc.
- Once Board\_Init() is complete, the application can verify basic console STDIO and I2C interfaces within the board.

# Boot Support: Secondary Boot Loader (SBL)

Introduction to Processor SDK RTOS Part 2

# Secondary Boot Loader (SBL)



OSDisk (C:) > ti > processor\_sdk\_rtos\_281 > pdk\_am57xx\_1\_0\_0 > packages > ti > boot > sbl

Include in library ▾ Share with ▾ Burn New folder

Name

- binary
- board
- build
- soc
- src
- tools**
- makefile\_armv7
- test.txt

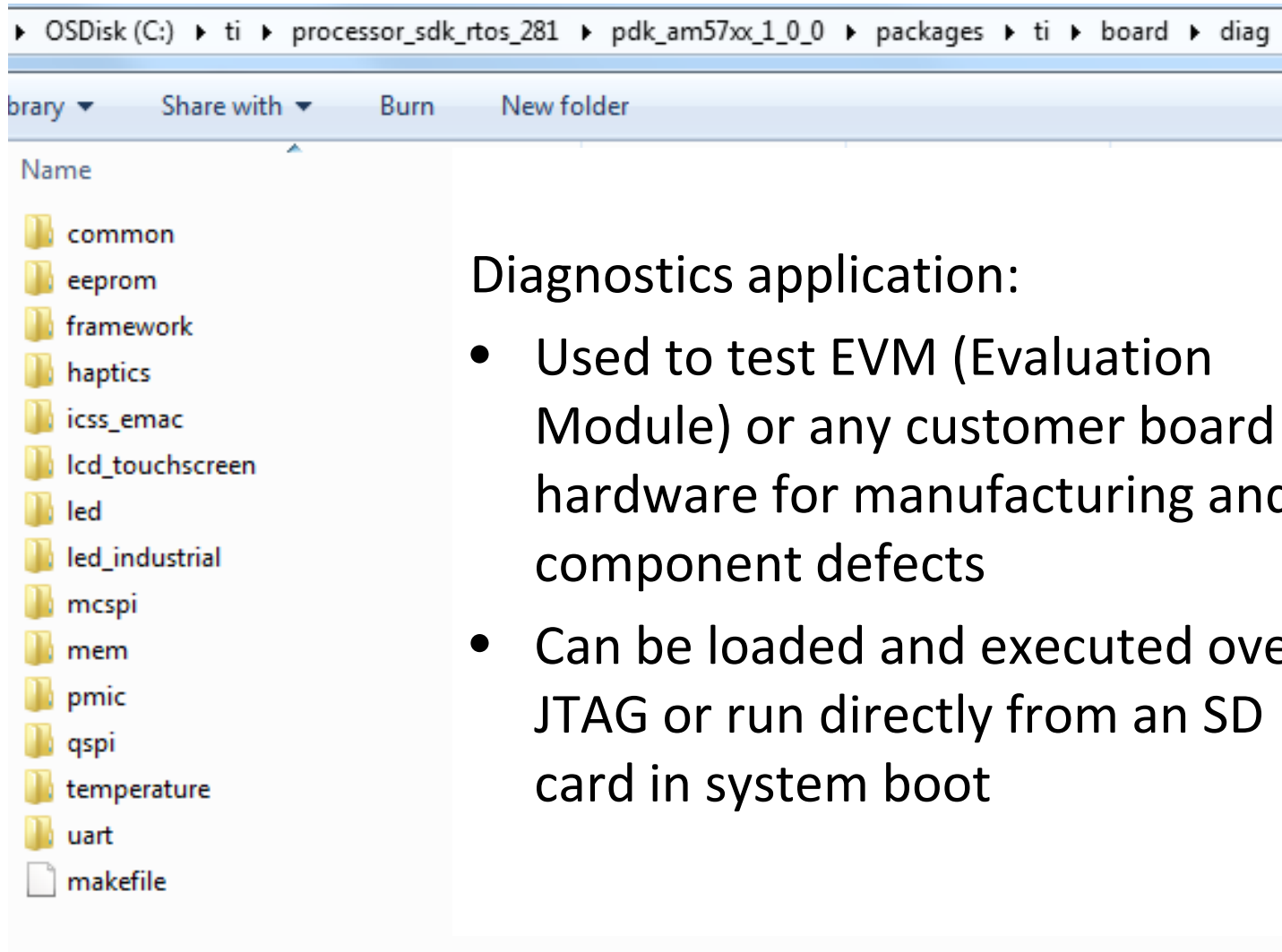
- SBL includes tools and utilities to support multi-core deployment of applications from boot.
- It also provides SoC-dependent functions to initialize the SOC and perform slave core boot-up functions.

# Diagnostics

Introduction to Processor SDK RTOS Part 2



# Diagnostics Software



## Diagnostics application:

- Used to test EVM (Evaluation Module) or any customer board hardware for manufacturing and component defects
- Can be loaded and executed over JTAG or run directly from an SD card in system boot

# Diagnostics Software: More Info

Diagnostics application:

- Instructions on how to build the diagnostics code are in a SoC wiki: [http://ap-fpdsp-swapps.dal.design.ti.com/index.php/Processor\\_SDK\\_RTOS\\_Software\\_Developer\\_Guide#Board\\_Library\\_and\\_Diagnostic\\_Examples](http://ap-fpdsp-swapps.dal.design.ti.com/index.php/Processor_SDK_RTOS_Software_Developer_Guide#Board_Library_and_Diagnostic_Examples)
- Instructions on how to connect a board to CCS and run diagnostics are in a SoC wiki like this one (for AM572X): [http://ap-fpdsp-swapps.dal.design.ti.com/index.php/GSG:AM572x\\_General\\_Purpose\\_EVM\\_Hardware\\_Setup](http://ap-fpdsp-swapps.dal.design.ti.com/index.php/GSG:AM572x_General_Purpose_EVM_Hardware_Setup)

```
*****
*                               AM57xx HW Diagnostics                               *
*****

Diagnostics                               Pass       Fail
-----
0 - Auto run tests 1-17
1 - ID Memory Programming                 1         0
2 - DDR Test                             1         0
3 - Ethernet Test                        1         0
3 - QSPI Test                            1         0
4 - PMIC Test                            1         0
5 - PRU ICSS IEP Test                    1         0
6 - eMMC Test                            1         0
7 - LED Test                             1         0
8 - LED driver Test                      1         0
9 - Haptics Test                         1         0
10 - Serializer Test                     1         0
11 - Misc Header Test                    1         0
12 - Touch Screen Test                   1         0
13 - PCIe EP Test                        1         0
14 - PCIe RC App                         1         0

Enter desired option: 12

*****
*                               Touchscreen Test                               *
*****

Input 9 touches to exit test

Touch  t1      t2      t3      t4      t5      t6      t7      t8      t9
1      538, 754  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095
1      538, 754  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095
1      1433, 438  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095
3      704, 788   975, 571  1286, 389  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095
2      619, 755  1126, 599  1266, 380  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095
3      1271, 641  643, 362   381, 701  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095
2      571, 392  1574, 607  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095
3      538, 370  1614, 667  1391, 449  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095  4095,4095
```

User Interface

# For More Information

- [Processor SDK RTOS Getting Started Guide](#)
- [Processor SDK Training Series](#)
- Additional training:
  - [TI-RTOS Kernel Workshop](#)
  - [Processor SDK RTOS Overview P1](#)
- For questions regarding topics covered in this training, visit the [Sitara Processor](#) support forum at the TI E2E Community website.