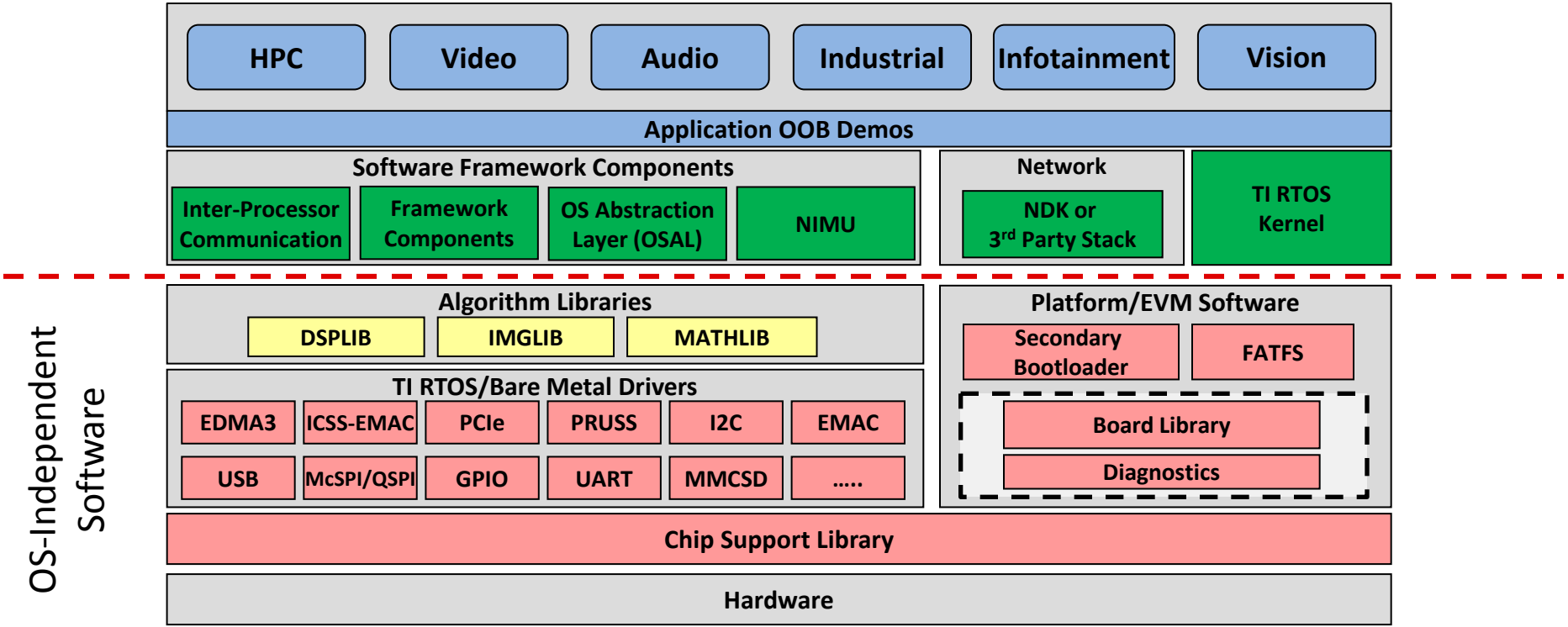
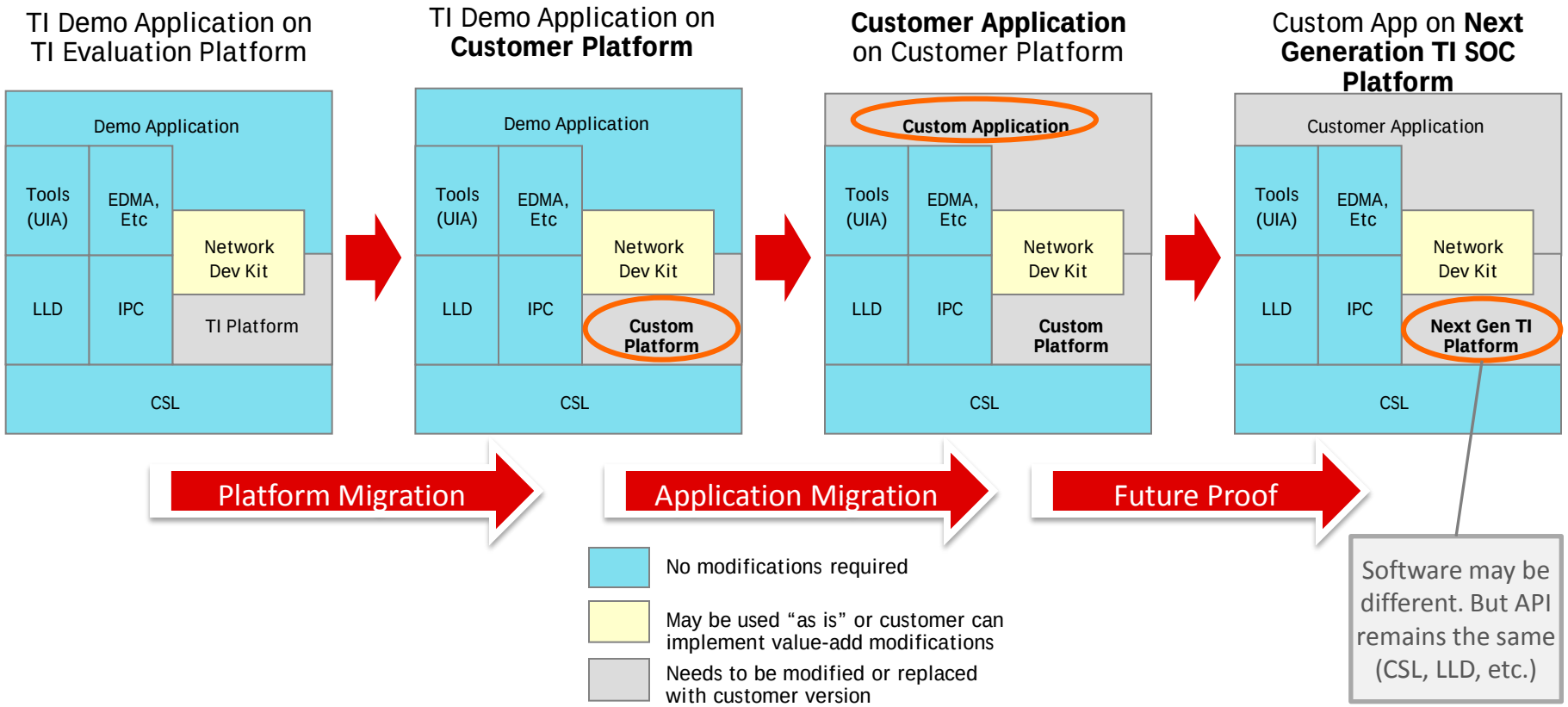
Application Development Using Processor SDK RTOS

Processor SDK RTOS: Software Stack



Processor SDK RTOS: Maximize Software Reuse



Processor SDK RTOS: Typical Development Flow

Customize	Customize application software
Port	Custom hardware bring up
Develop	Develop application
Run	Run SDK demo applications
Start	Boot RTOS O/S, Start UART, Network, USB
Setup	Setup EVM Connect UART, Network Cable, USB, Power

EVM Kit, Processor SDK RTOS Package

Processor SDK RTOS: Setup

Application Development Using
Processor SDK RTOS

Customize

Port

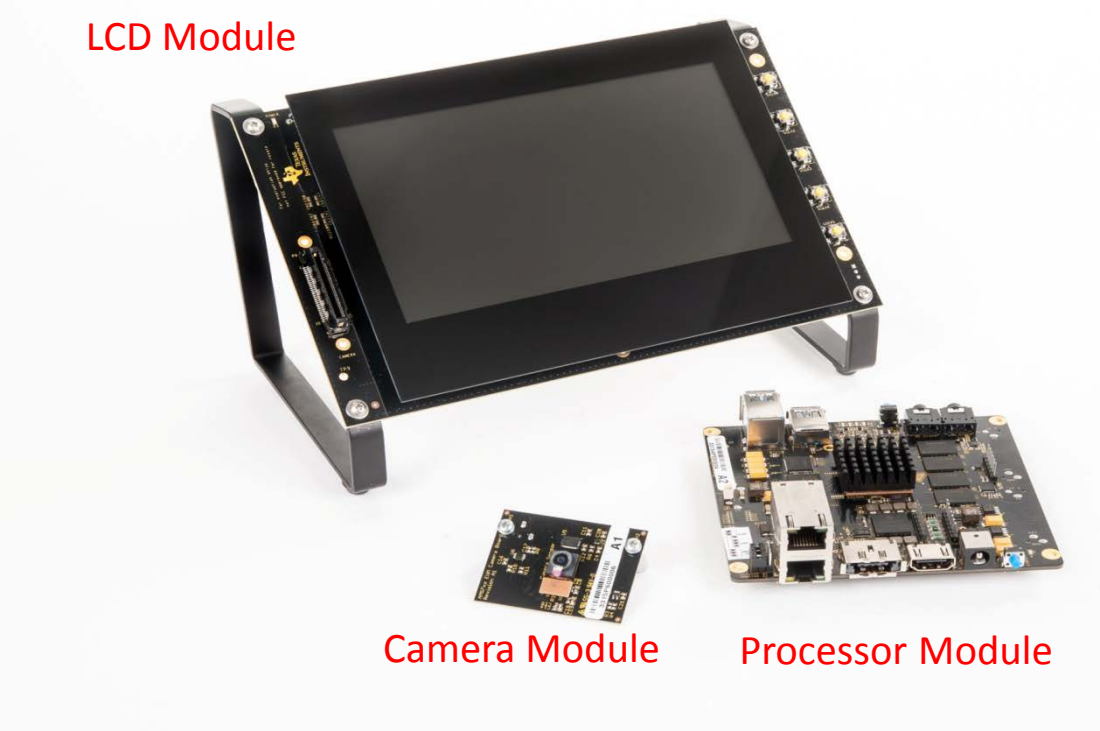
Develop

Run

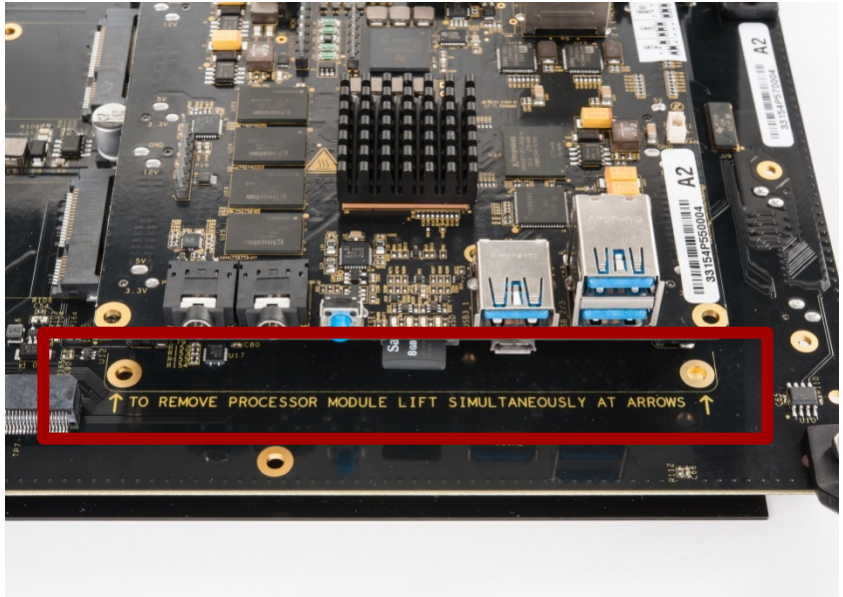
Start

Setup

Processor SDK RTOS: AM572x GP EVM



Removing the Processor Module from the LCD



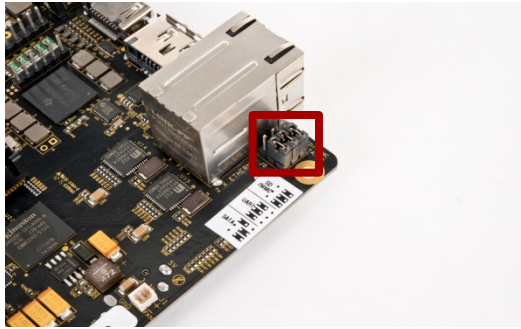
NOTE: This is mandatory to connect an external emulator to the AM572x GP EVM.

Processor SDK RTOS: AM572x GP EVM Setup

Emulation and boot settings

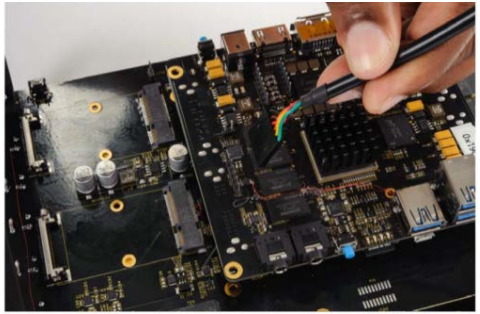
Connect emulator (only for debugging)

Configure boot jumpers



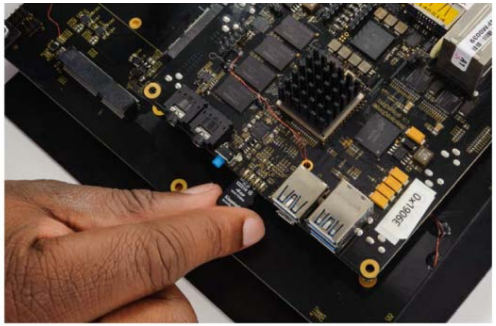
Optional peripheral connections

Plug in FTDI cable for UART console out



Connect Ethernet cable

Insert SD card (only for SD boot & mass storage)



For EVM-specific instructions, select **Setup EVM Hardware** in the *Processor SDK RTOS Getting Started Guide*:
http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_Getting_Started_Guide

CAUTION: EVM Power Up/Down Sequence (AM572x EVM Only)

Safe power up/ power down sequence:

Refer to wiki article for safe power up/down sequence:
[AM572x_General_Purpose_EVM_HW_User_Guide]

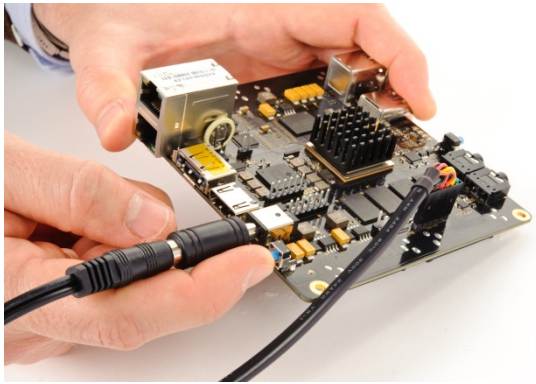
PMIC shutdown in 7 seconds:

- PMIC on the TMDXEVM5728 turns off the board in 7 seconds due to a hardware errata.
- Software needs to write to PMIC register to keep it on.
- GEL files and board library provide board configuration.

Errata: http://www.ti.com/product/AM5728

Connect power jack

Push power button



Processor SDK RTOS: Software Setup

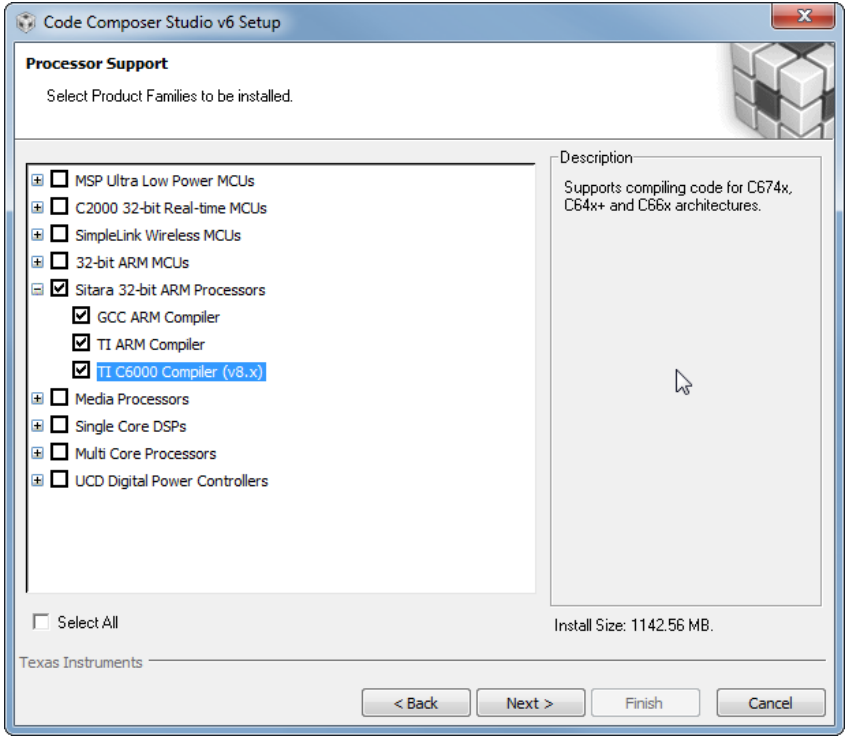
Recommended host setup supported:

- Windows: Windows 7 on 64-bit machine
- Linux: Ubuntu 14.04 or 12.04 on 64-bit machine

Setting up the development environment:

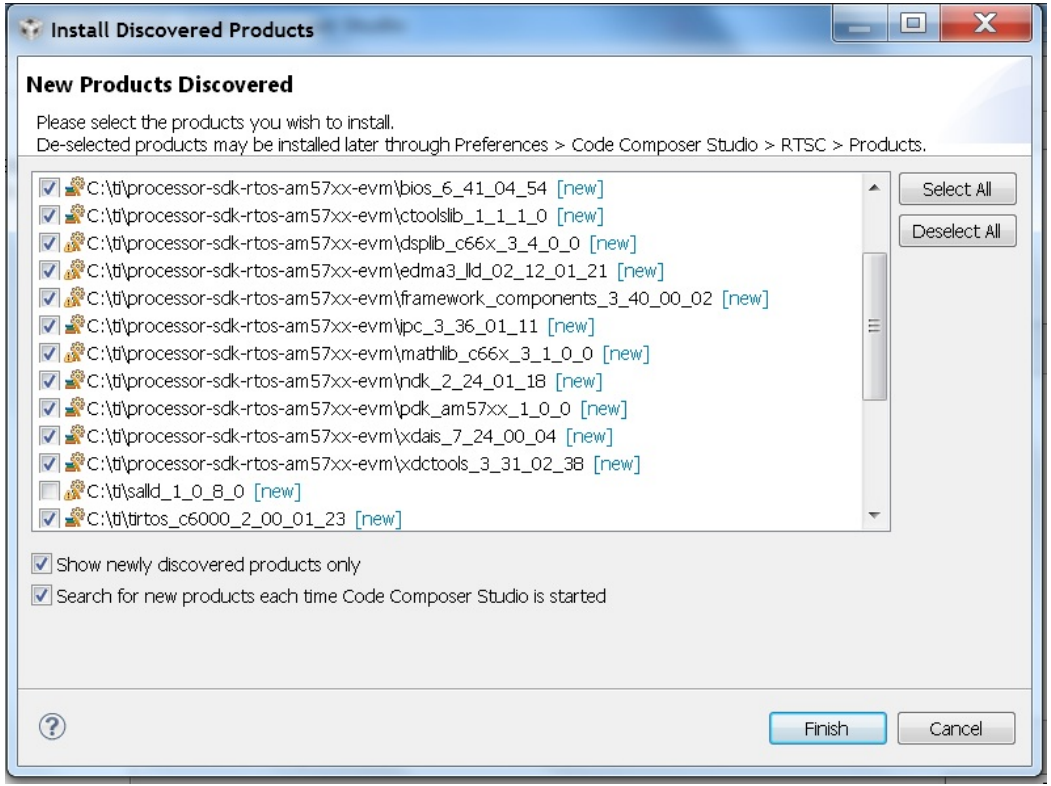
- Processor-SDK RTOS installer
- Code Composer Studio v6.1.1 or later

CPU	Tool	TI- Software Package
A15	Linaro GCC Toolchain	CCS
A8	Linaro GCC Toolchain	CCS
A9	Linaro GCC Toolchain	CCS
M4	TI ARM Toolchain	CCS
C66x	TI CGT Toolchain	CCS



For software instructions, select **Setup Software** in the *Processor SDK RTOS Getting Started Guide*:
http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_Getting_Started_Guide

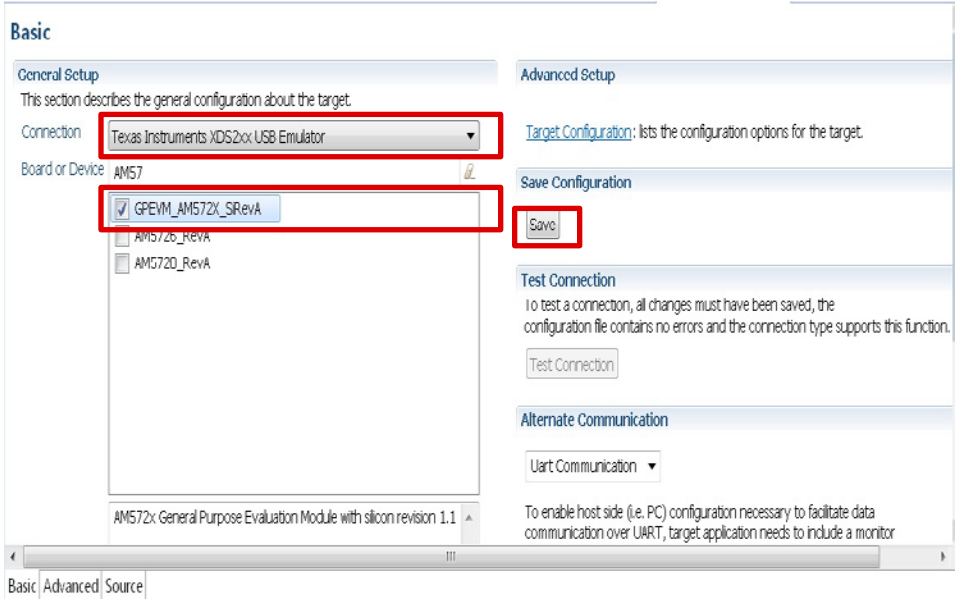
Processor SDK RTOS: CCSv6 Product Discovery



NOTE: Mandatory CCS restart is required for product discovery to take effect.

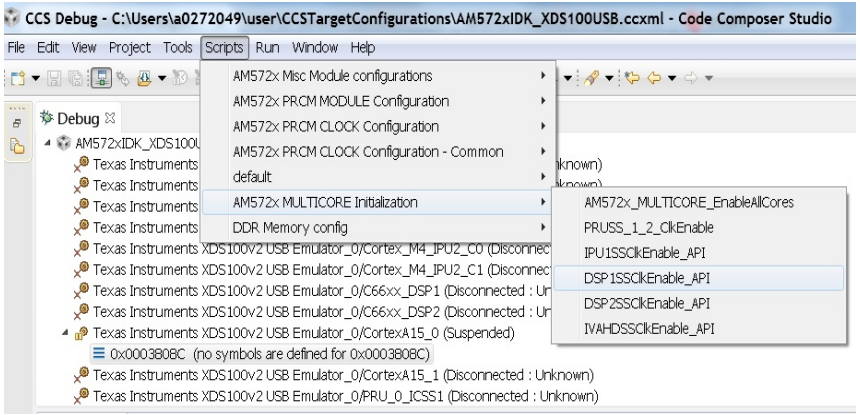
Processor SDK RTOS: CCSv6 Target Configuration

CCS Edit View: File ->New->Target Configuration



- Gel files for A15, C66x, M4 are auto-populated.
- Test connection option available.
- Advanced options allows customization.

CCS Debug View: Launch target configuration



- Connect to CortexA15_0
- GEL initializes SoC clocks, DDR, PMIC
- All slave cores are in reset and need wake up

For EVM-specific instructions, select **Setup EVM Hardware** in the *Processor SDK RTOS Getting Started Guide*: http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_Getting_Started_Guide

Processor SDK RTOS: Start

Application Development Using
Processor SDK RTOS

- Customize
- Port
- Develop
- Run
- **Start**
- Setup

Processor SDK RTOS: Start

Init O/S, Interrupts, Timers

Start UART

Start Ethernet Driver

Start USB

Start RTOS Tasks

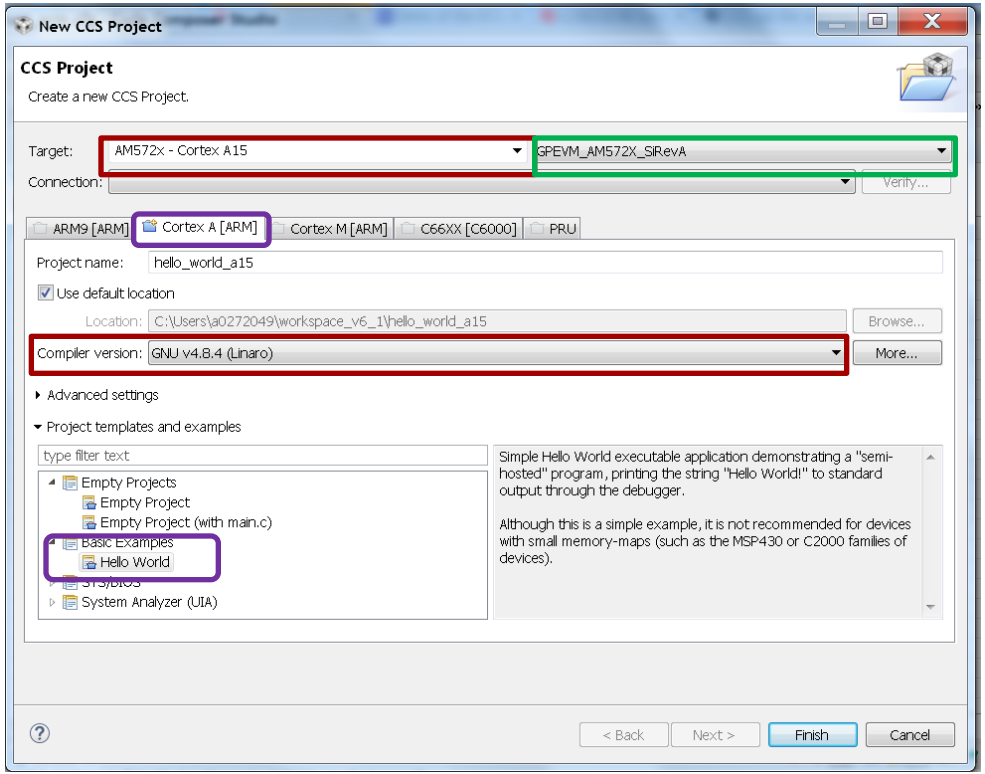
Boot RTOS O/S, Start UART, Network, USB

Processor SDK RTOS: Bare Metal Hello World Example

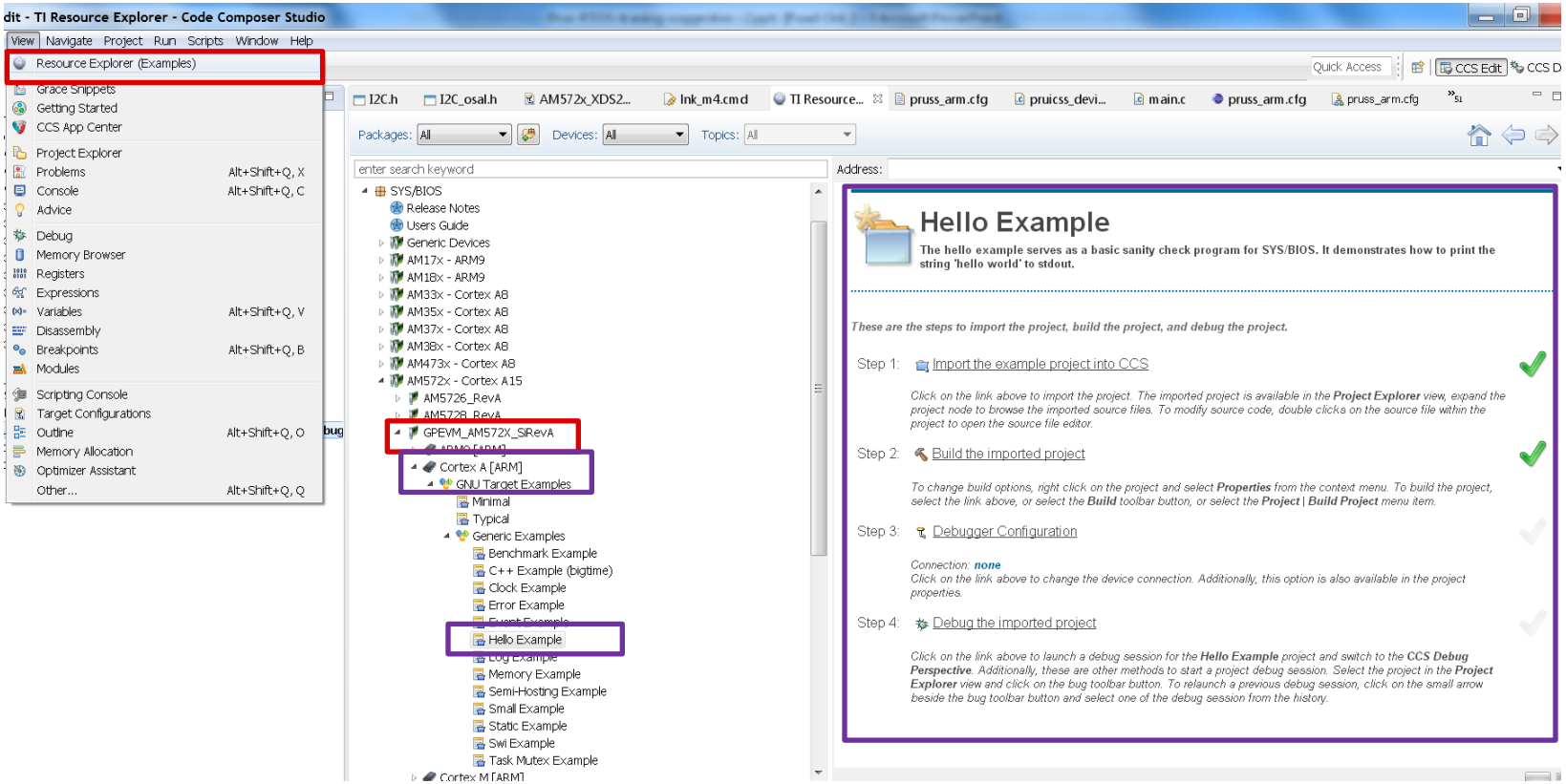
- CCS “Hello World” template available.
- Template provided for all cores on SoCs.

For more details:

[Processor SDK Bare Metal Examples]



Processor SDK RTOS: SYSBIOS Hello World Example



Wiki Link: http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_Examples

Set Up Build Environment to Build PDK Components

Build instructions:

- Navigate to processor_sdk_rtos_<soc>_2_xx_xx_xx>
- Set environment variables:
  - **SDK_INSTALL_PATH** is SDK and CCS installation path.
  - Default sets it to “C:\TI” (Windows) & “/home/[user]/ti” (Linux).
- Run the script setup.bat (Windows) and source setupenv.sh (Linux)

Build all components:
make clean
make all

For other build target options:
http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_Building_The_SDK
Custom installation options:
http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_Install_In_Custom_Path

Script to Create Unit Tests for Device Drivers

pdkProjectCreate.bat [soc] [board] [endian] [module] [processor] [pdkDir] **(Windows)**

pdkProjectCreate.sh [soc] [board] [endian] [module] [processor] [pdkDir] **(Linux)**

File location: {PDK_INSTALL_DIR}\packages

Description:
soc – eg. am335x
board – refer ${PDK_INSTALL_DIR}\package\ti\board\lib
endian - little
module - all – eg uart
processor – eg arm, dsp
pdkDir - THIS FILE LOCATION

Example:
pdkProjectCreate.bat am572x evmAM572x little uart arm

Refer to **PDK Example and Test Project Creation** in the *RTOS Software Developer Guide*: http://processors.wiki.ti.com/index.php/Rebuilding_The_PDK

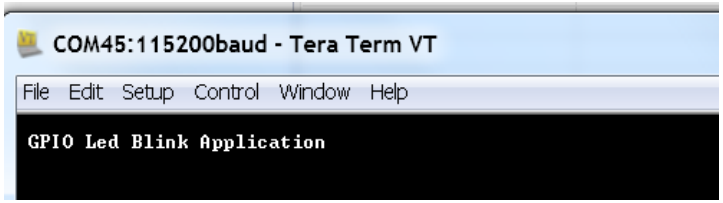
Processor SDK RTOS: Set Up GPIO LED Example

GPIO example location:
pdk_1_x_x/packages/exampleProjects/GPIO_LedBlink_<soc>_evm_armExampleProject

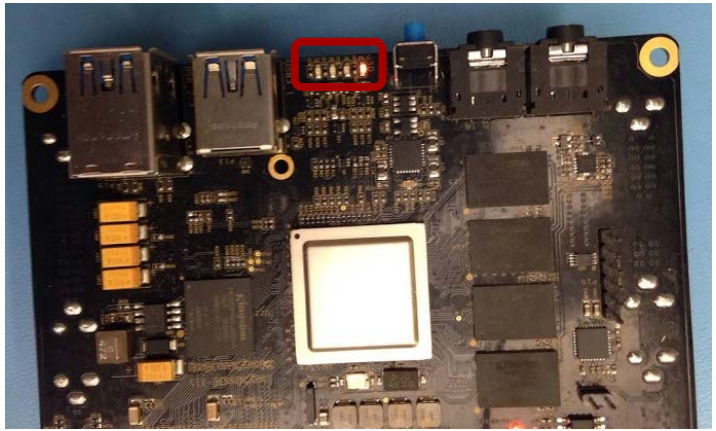
- Import the project in CCSv6 and build the project.
- Connect the serial cable on host to view console.
- Host setup for serial console software:

```
*Baud rate: 115,200
*Data bits: 8
*Parity: None
*Stop bits: 1
*Flow control: None
```

UART console output



User LED blink output



GPIO LLD and example documentation:
http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_GPIO

Processor SDK RTOS: Set Up UART

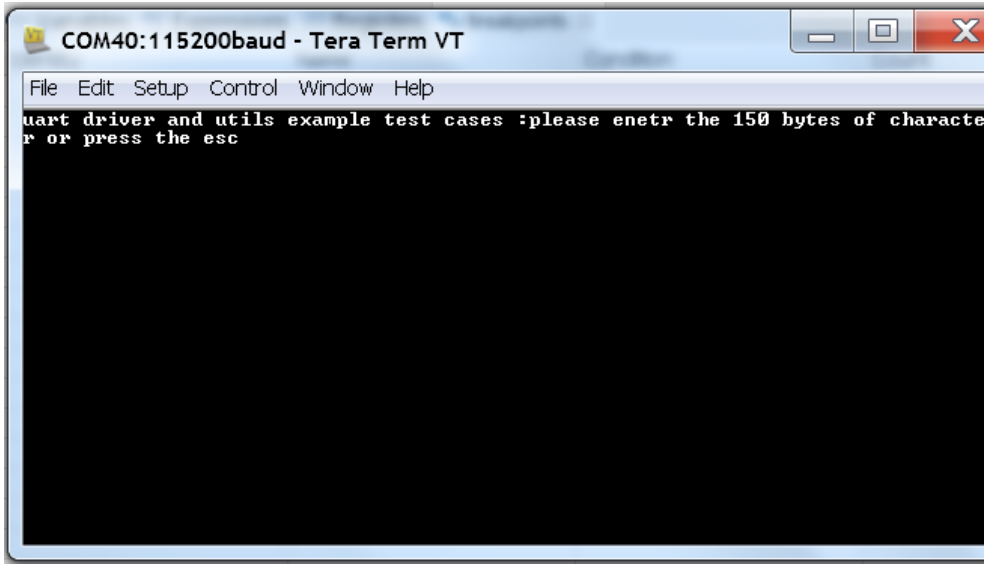
Locate UART example:

pdk_1_0_0/packages/exampleProjects/*UART_BasicExample_<SOC>_armTestproject*

- Import the project in CCSv6 and build the project.
- Connect UART using FTDI or microUSB cable.
- Configure the serial terminal on host to view console.
- Host setup for Teraterm:

```
*Baud rate: 115,200
*Data bits: 8
*Parity: None
*Stop bits: 1
*Flow control: None
```

Example output



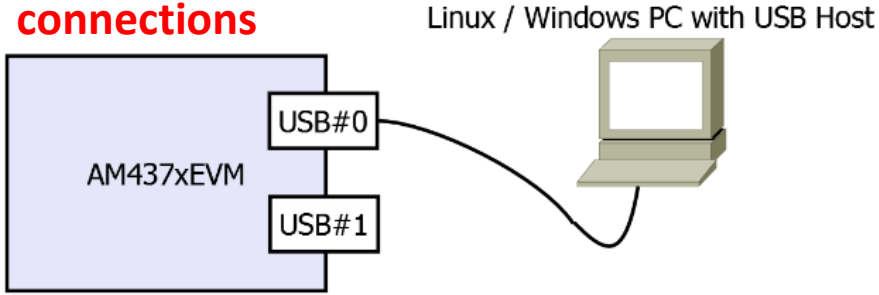
UART LLD and example documentation:

http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_UART

Processor SDK RTOS: Set Up USB Device

- USB device instance will behave like a USB thumb drive.
- EVM DDR memory acts as storage to external host.
- Compile and run project under pdk/packages/exampleProjects usb_dev_msc_<BoardName>_arm_project
- Connect USB cable to USB device port on EVM and to USB port on the PC.
- Hook up UART cable to PC to view console logs.
- PC detects the EVM hardware as USB mass storage and prompts user to format disk before using the device.

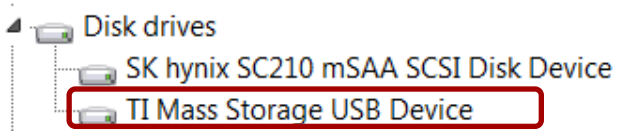
USB device mode connections



UART console

```
0:>
RTOS USB Dev MSC example!!
Done configuring USB and its interrupt
```

Host view of the AMXX hardware

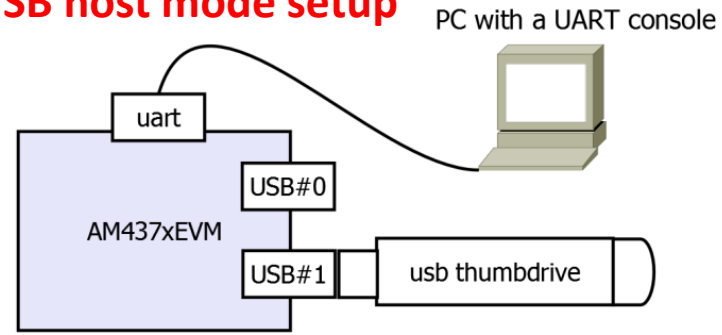


Wiki Link:
http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_USB

Processor SDK RTOS: Set Up USB HOST (MSC)

- USB instance acts as USB host communicating with a USB mass storage class device.
- Compile and run the following project under pdk/packages/exampleProjects usb_host_msc_<BoardName>_arm_project
- Plug in USB flash driver (FAT formatted) in the USB host port (USB0/1 on AM437x EVM).
- Connect UART cable to view example console prompt. Screenshot of example console is shown.
- Example demonstrates mass storage class functionality of the USB driver.

USB host mode setup

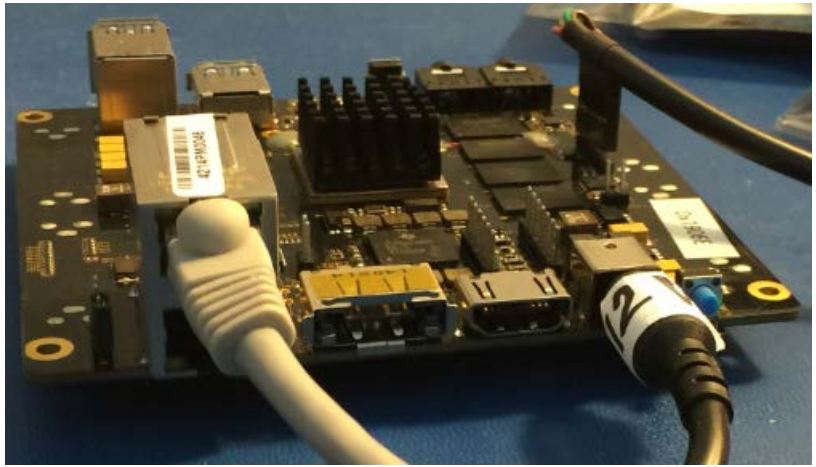


UART console

```
0:>help
Available commands
------------------
help : Display list of commands
ls   : Display list of files
cd: Change directory
mkdir: Create directory
rm   : Delete a file or an empty directory
pwd  : Show current working directory
cat  : Show contents of a text file : cat <FILENAME>
       Write to a file : cat <INPUTFILE> > <OUTPUTFILE>
       Read from UART : cat dev.UART
       Write from UART : cat dev.UART > <OUTPUTFILE>
0:>lsD---- 2013/03/18 14:06          0  DRIVER~1
----A 2013/11/22 10:14     233984  SOCKET~1.DOC
----A 2013/11/22 10:16      75520  UDP~1.PCA
----A 2013/11/20 17:12      50456  CAP~1.PCA
----A 2013/11/20 17:12       1100  mylog.txt
----A 2013/10/15 13:45       1734  README.TXT
```

Wiki Link:
http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_USB

Processor SDK RTOS: Set Up Networking



Example Application:

NIMU_BasicExample_<SOC>_Evm_armExampleproject

- Import project into CCSv6 and build unit test.
- Load unit test via CCS using emulator.
- Example configures IP address 192.168.1.2 on the target.
- Before running:
  - Create interface on PC with static address 192.168.1.x
  - Hook up Ethernet cable from PC to Ethernet port on EVM.
    e.g., ETH0 interface. (top Ethernet port) on AM572x GP EVM
- To verify, ping 192.168.1.2 IP address (EVM board) from your host.

Wiki Link: http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_NDK

CSL Examples

- Chip Support Library (CSL):
  - Provides a set of well-defined APIs
  - Abstracts low-level interface details of underlying SoC
  - Allow users to configure, control (start/stop, etc.) and read/write from peripherals
- User can use the CSL layer to create examples and custom drivers.
- **Example location:** (TI_PDK_INSTALL_DIR)\packages\ti\csl\test

Example Name	Description
WDT (Watchdog timer)	The application resets the A15 CPU0 core.
RTC (Real Time Clock)	The application prints date and time on UART console.
GMAC(External PHY)	The application prints on console the configuration of PHY.

Wiki Link:
http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_CSL

pdk_am57xx_1_0_0 ▸ packages ▸ ti ▸ csl ▸ soc ▸ am572x ▸ src

Name	Date modified
csl_device_pllc.c	9/11/2015 1:57 PM
csl_device_pllc.h	9/11/2015 1:57 PM
csl_device_xbar.c	9/11/2015 1:57 PM
csl_device_xbar.h	9/11/2015 1:57 PM
csl_version.c	9/11/2015 1:57 PM
cslr_cam_cm_core.h	9/11/2015 1:57 PM
cslr_cam_prm.h	9/11/2015 1:57 PM
cslr_cgem_system.h	9/11/2015 1:57 PM
cslr_ckgen_cm_core.h	9/11/2015 1:57 PM
cslr_ckgen_cm_core_aon.h	9/11/2015 1:57 PM
cslr_ckgen_prm.h	9/11/2015 1:57 PM
cslr_cmi.h	9/11/2015 1:57 PM
cslr_control_core.h	9/11/2015 1:57 PM
cslr_control_core_pad.h	9/11/2015 1:57 PM
cslr_control_core_pad_io.h	9/11/2015 1:57 PM
cslr_control_core_sec.h	9/11/2015 1:57 PM
cslr_control_core_wkup.h	9/11/2015 1:57 PM
cslr_control_core_wkup_pad.h	9/11/2015 1:57 PM
cslr_control_intr_dma.h	9/11/2015 1:57 PM
cslr_core_cm_core.h	9/11/2015 1:57 PM
cslr_core_prm.h	9/11/2015 1:57 PM
cslr_coreaon_cm_core.h	9/11/2015 1:57 PM
cslr_coreaon_prm.h	9/11/2015 1:57 PM
cslr_custefuse_cm_core.h	9/11/2015 1:57 PM

Processor SDK RTOS: Run

Application Development Using
Processor SDK RTOS

- Customize
- Port
- Develop
- **Run**
- Start
- Setup

Creating SD Card to Boot SDK Demos

Script location in Processor SDK:

<SDK INSTALL DIR>/bin/create-sdcard.sh (Linux host only)

Notes:

- Linux script formats, partitions and loads the boot images to the SD card.
- Windows requires formatting, partitioning and copying of boot image using Win32 Disk Imager.

Location of prebuilt binaries for OOB demo images and sd-card image:

<SDK INSTALL DIR>\demos\oob\<SOC_EVM>\sd_card_img

Reference: Processor_SDK_RTOS_Creating_a_SD_Card_with_Windows

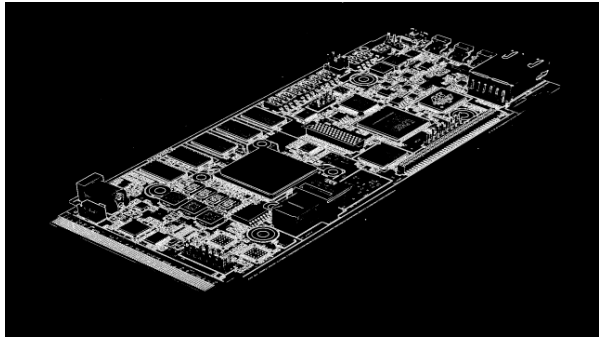
Processor_SDK_RTOS_create_SD_card_script for Linux

Processor SDK Demonstration: Image Processing Demo

- TI RTOS kernel based OOB demo demonstrates :
  - Booting from SD card using SBL,
  - UART, SD/MMC drivers
  - IPC messaging between ARM and DSP
  - IMGLIB functionality
- Application flow:
  - ARM reads the input image from SD card.
  - ARM partitions image across DSP cores.
  - ARM sends messages to DSP cores via IPC MessageQ.
  - DSP cores process partitioned images concurrently using IMGLIB edge detection functions.
  - DSP stores resulting image in DDR and notifies ARM cores.
  - ARM writes the resulting image into the SD card.
- Demo supports UART console logs and user input.

Input image

Output image



Processor SDK RTOS: Develop

Application Development Using
Processor SDK RTOS

- Customize
- Port
- **Develop**
- Run
- Start
- Setup

Processor SDK RTOS: Develop (Source Reference)

Link to UART LLD source to enable console output

Link to USB LLD location in package

Link to EMAC LLD, NIMU and NDK source location in package

Adding filesystem support to the application

Booting an application

IPC code to enable slave cores.

Boot RTOS O/S, Start UART, Network, USB

Processor SDK RTOS: Enabling UART

API Header Files:

ti/drv/uart/UART_stdio.h
board.h
board_cfg.h

Sample Source Code:

```
main(){
Board_initCfg boardCfg;
 boardCfg = BOARD_INIT_UART_STDIO;

Board_init(boardCfg);
UART_printf(" Text to output ");
}
```

Wiki Link:

http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_UART

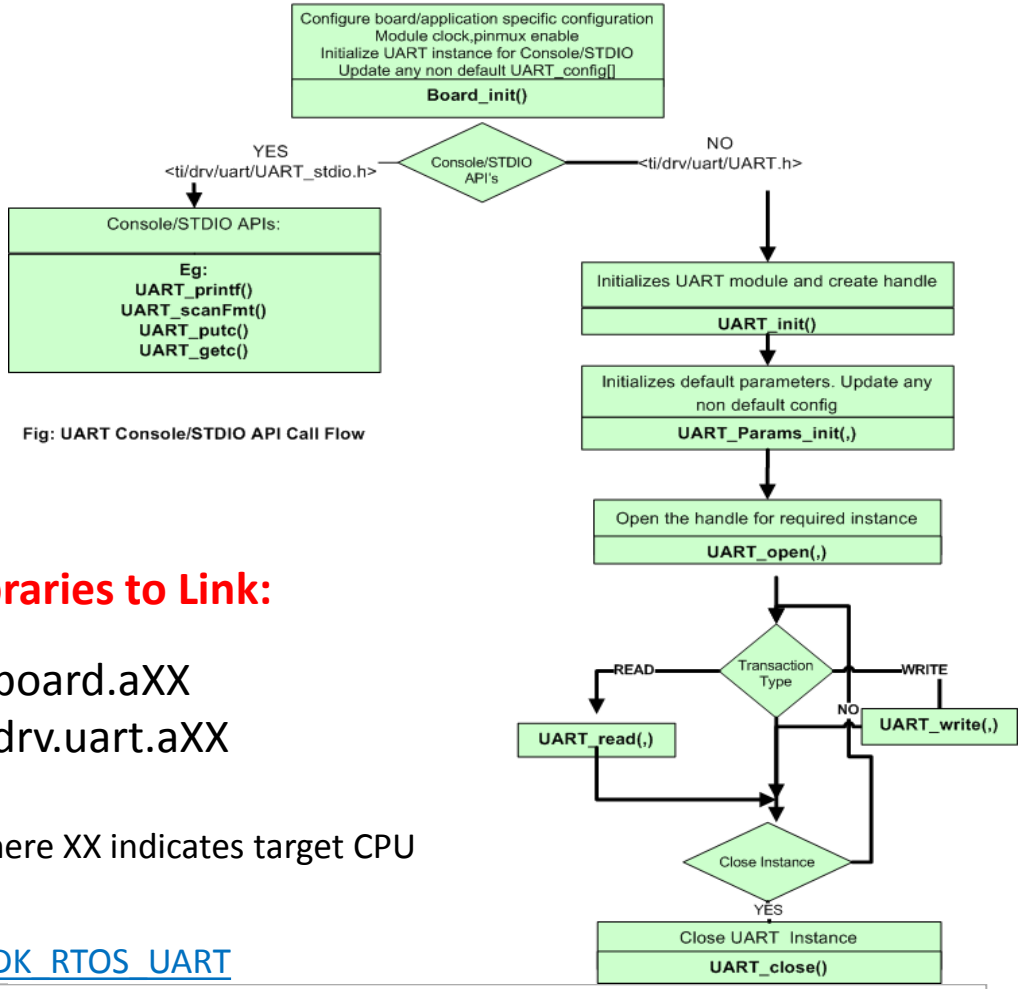


Fig: UART Console/STDIO API Call Flow

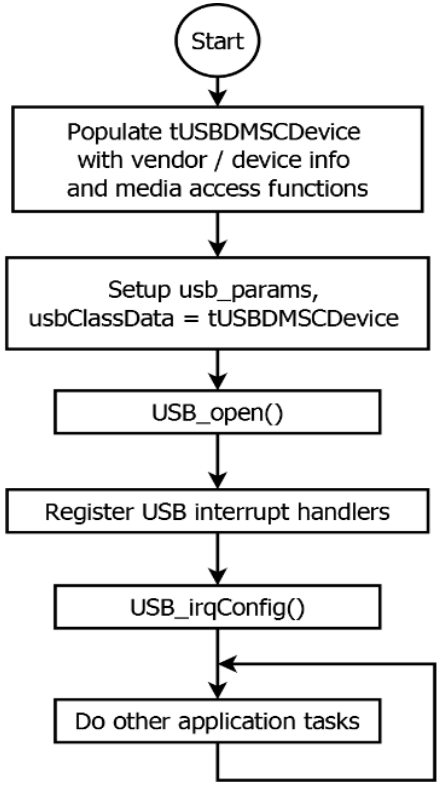
Libraries to Link:

ti.board.aXX
ti.drv.uart.aXX

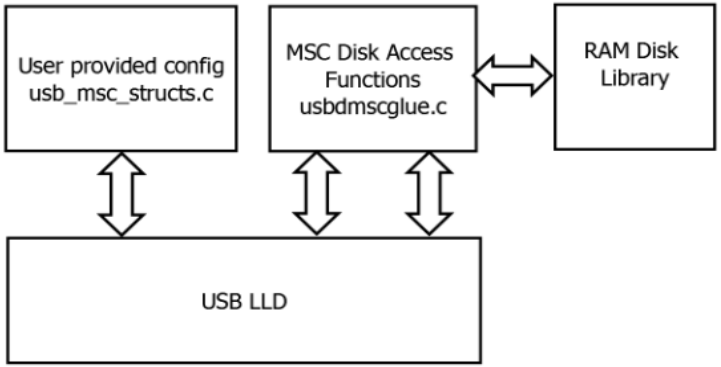
Where XX indicates target CPU

Processor SDK RTOS: Enabling USB Device

Sequence of APIs used to enable USB device



USB device implementation in PDK



API header file:

usb_drv.h

usbdmsc.h

Libraries to link:

ti.board.aXX

ti.drv.usb.aXX

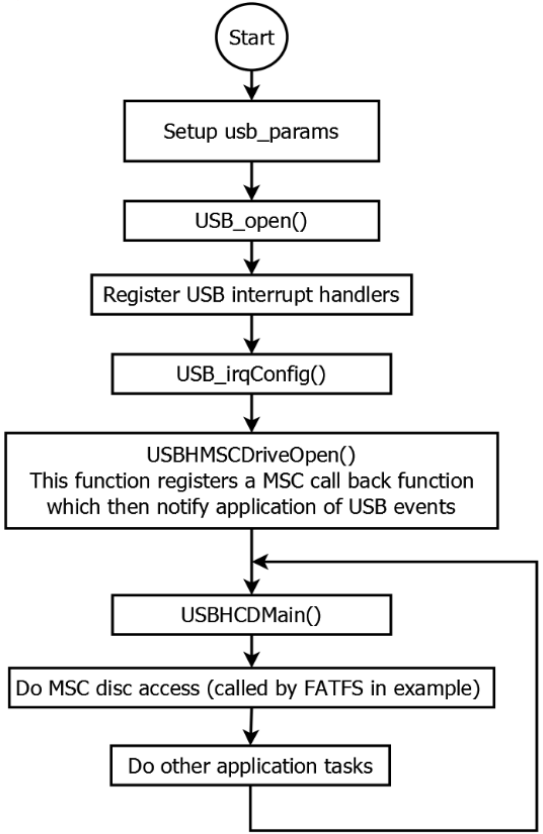
Where XX indicates target CPU

Wiki Link:

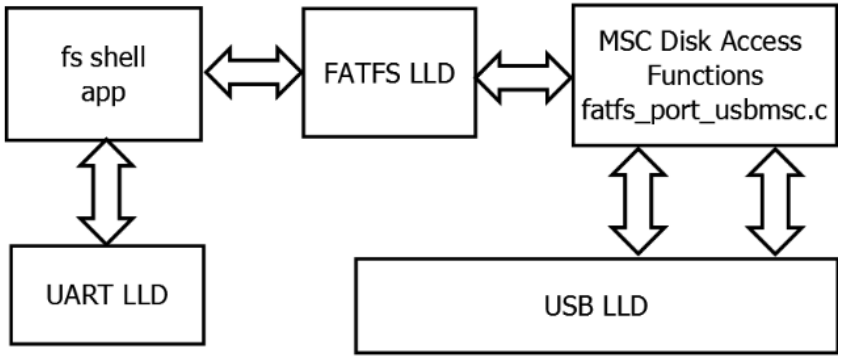
http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_USB

Processor SDK RTOS: Enabling USB Host

Sequence of APIs used to enable USB host



USB Host Mode example implementation in PDK



API header file:

usb_drv.h

usbhmsc.h

Libraries to link:

ti.board.aXX

ti.drv.usb.aXX

Where XX indicates target CPU

Wiki Link:

http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_USB

Processor SDK RTOS: Enabling Networking

NIMU/EMAC header files:

ti/transport/ndk/nimu/nimu_eth.h

NDK header files:

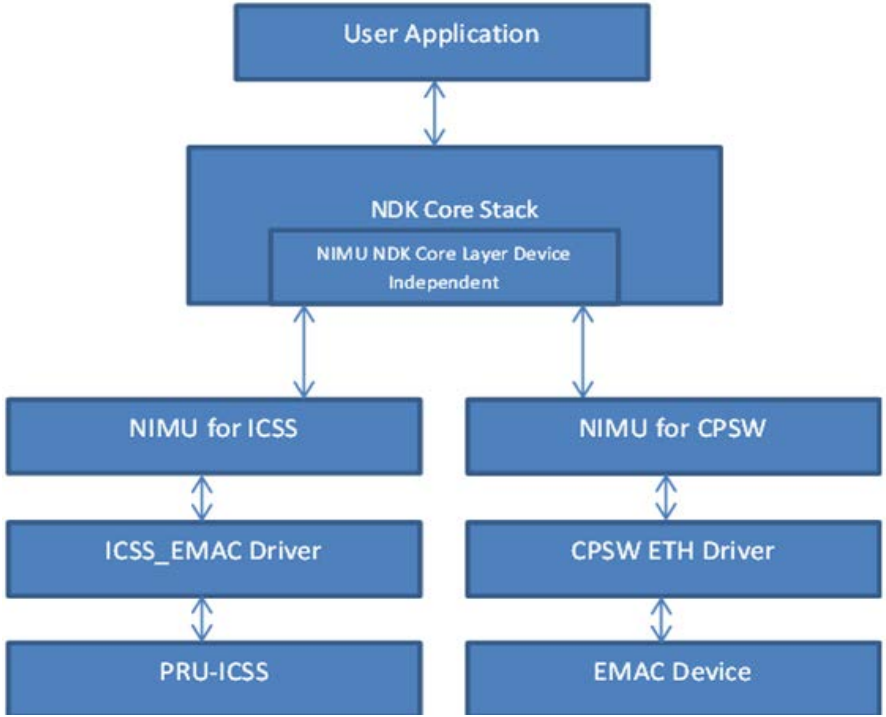
ti/ndk/inc/netmain.h
ti/ndk/inc/stkmain.h

Libraries to link:

ti.transport.ndk.nimu.aXX
ti.ndk.config.<NDKModule>

Where XX indicates target CPU

NDK Software Architecture

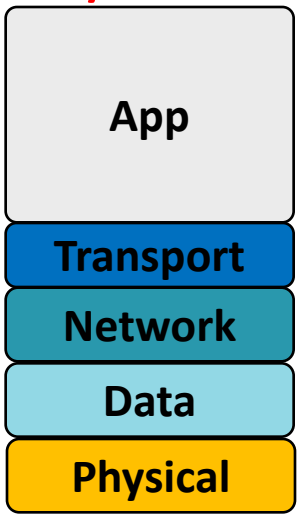


Wiki Link: http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_NDK

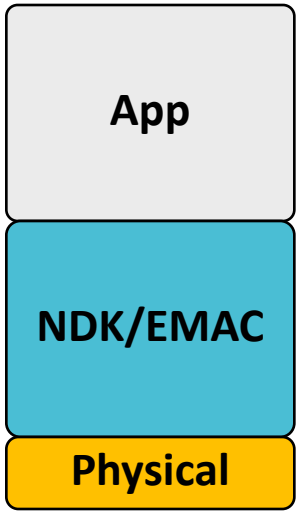
Network Development Kit (NDK)

- NDK is a *set of libraries + example code* that initialize/configure/operate the hardware (EMAC) & perform all of the TCP/IP functionality through a set of "socket" programming APIs(e.g. socket, bind, send, recv, etc.)
- Provides a seamless interface to the physical layer (EMAC/PHY)

TCP/IP Model



NDK Model



- HTTP
- TFTP
- Telnet
- DHCP
- PPP
- DNS
- PPPoE
- many others

- Sockets Programming Services
- Internal stack functions
- Configures stack/services and configures the EMAC

What does the user touch?

- Configuration

Do you know all of the details of what is going on underneath?

- No

Would you like an example to play with?

- NIMU example in PDK…

Network Stack (NDK) System Overview

BIOS configuration file for NDK example:

Global Initializations

var Global = xdc.useModule('ti.ndk.config.Global');

Network layer modules:

var Ip = xdc.useModule('ti.ndk.config.Ip');

Transport layer modules:

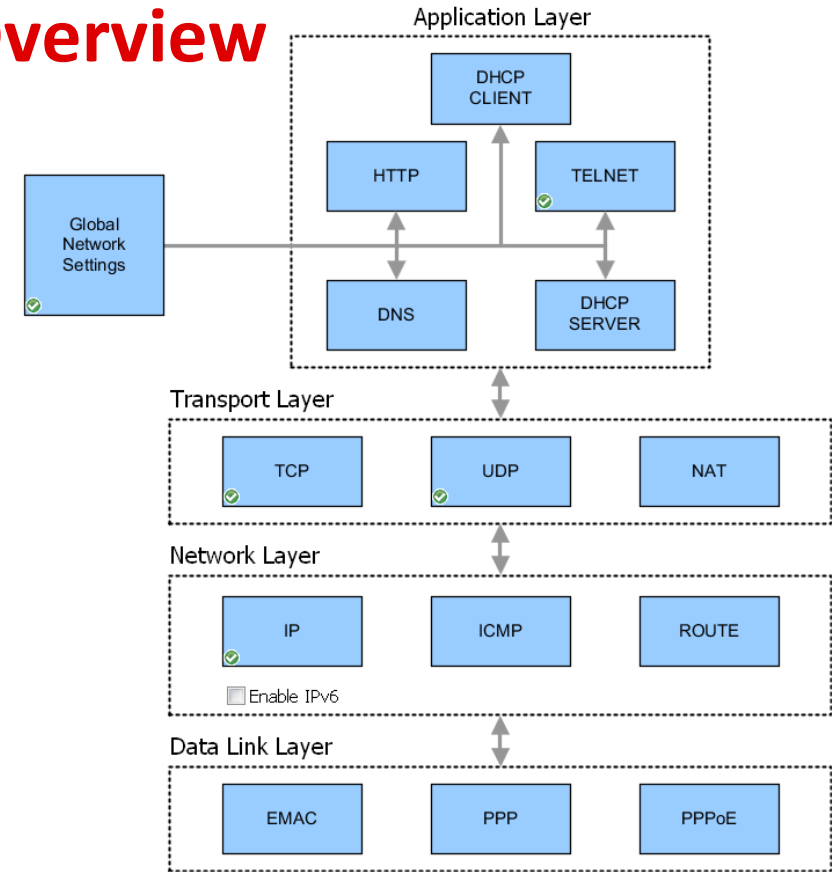
var Tcp = xdc.useModule('ti.ndk.config.Tcp');
var Udp = xdc.useModule('ti.ndk.config.Udp');

Application layer modules:

var Telnet = xdc.useModule('ti.ndk.config.Telnet');

NDK Transport device driver(specific to device)

var Nimu = xdc.loadPackage('ti.transport.ndk.nimu');



Wiki Link: http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_NDK

System Overview of NDK Example

Processor SDK RTOS: FATFS Filesystem Support

FATFS module driver enables device interface with FAT file system compatible device via the MMCSD, USB, etc

Header files:

ti/drv/FATFS/FATFS.h
ti/drv/FATFS/ff.h

Libraries to link:

ti.fs.fatfs.aXX

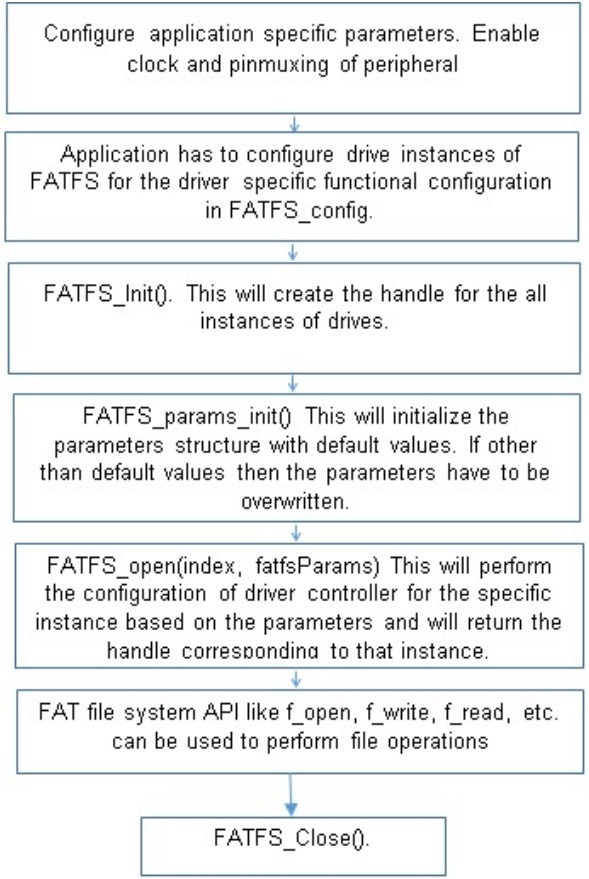
XX indicates the target CPU

Wiki Link:

http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_FATFS

Examples:

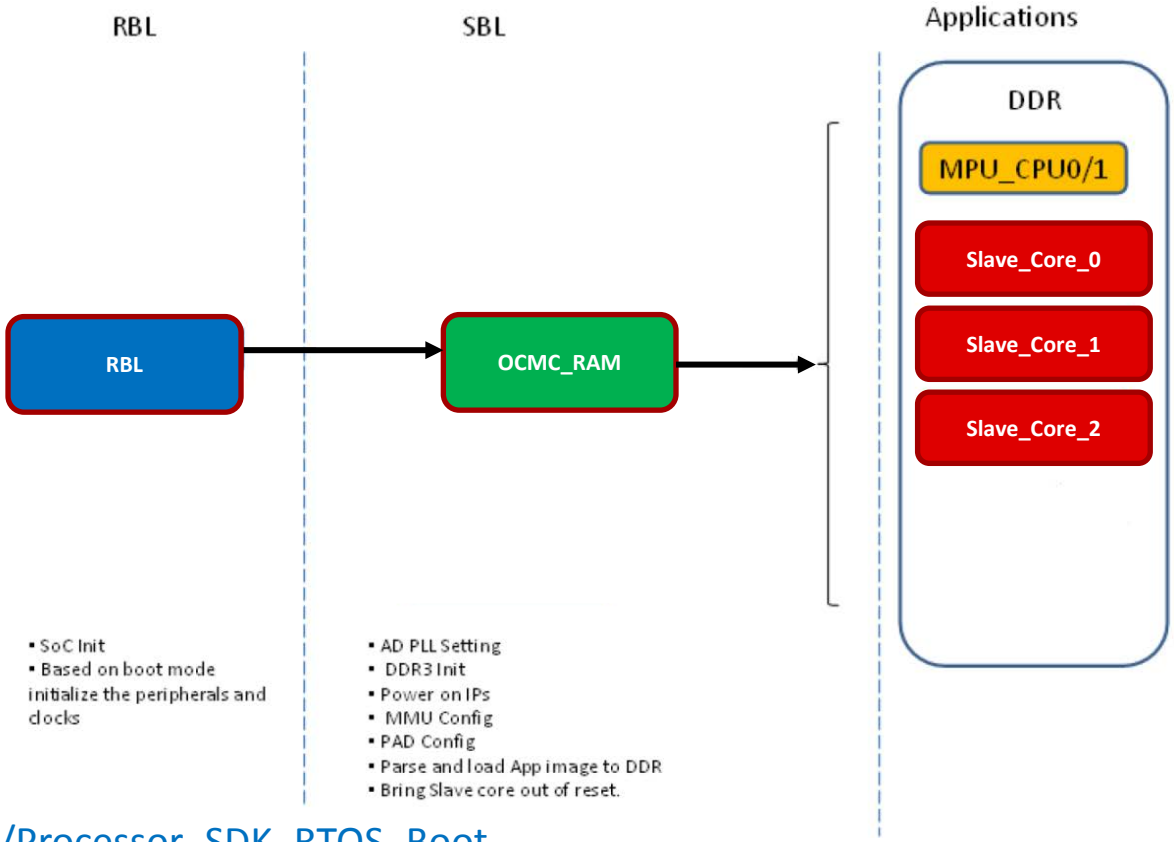
$(PDK_INSTALL_PATH)/packages/exampleProjects/FATFS_Console_<SOC>_Evm_armExampleProject



Processor SDK RTOS: Bootloader

SBL functions:

- Sets up the PLL clock, pinmux
- Powers on the I/O Peripherals, initializes the DDR
- Loads the application image from memory device into DDR
- Brings the slave cores out of reset



Wiki Link:

http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_Boot

Bootloader: Multicore Application Image Creation

AM57xImageGen script for creation of bootable multi-core:

Location: $(PDK_INSTALL_DIR)/packages/ti/boot/sbl/tools/scripts

Step 1: Set BIN_PATH variable in environment for output.

Step 2: Set path to ARM , DSP and M4 binaries:

- **App_MPU_CPU0:** Path to location of A15 MPU application .out
- **App_IPU1_CPU0:** Path to location of M4 core 1 application .out
- **App_DSP1:** Path to location of DSP core 1 application .out

Step 3: Run the script to create app.out

Tools used for image generation:

- Convert ELF Images of application binary to rprc format.

  out2rprc.exe <App_In_name(elf or coff)> <App_out_name>

- Multi-core image generator:

  MulticoreImageGen.exe <ENDIAN> <Dev Id> <App out file> <Core Id 1> <RPRC in file for Core Id 1> [<Core Id n> <RPRC in file for Core Id n> ...]

Bootloader: Boot Media-Specific Details

SD/MMC boot:

1. Create a primary FAT partition on MMC/SD card (FAT32 format with sector size 512).
2. Rename the SBL image as MLO (RBL requirement)and copy to the SD card.
3. Rename the Application multicore image file as “app” and copy to the SD card.
4. Copy the MLO and application to the bootable SD card.

NOTE: SD card formatting tool is not included in SDK.

For other boot media-specific details:

http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_Boot

Processor SDK RTOS: IPC Examples

SOC IPC examples path:

IPC_DIR\examples\<SOC>_bios_elf

List of Examples:

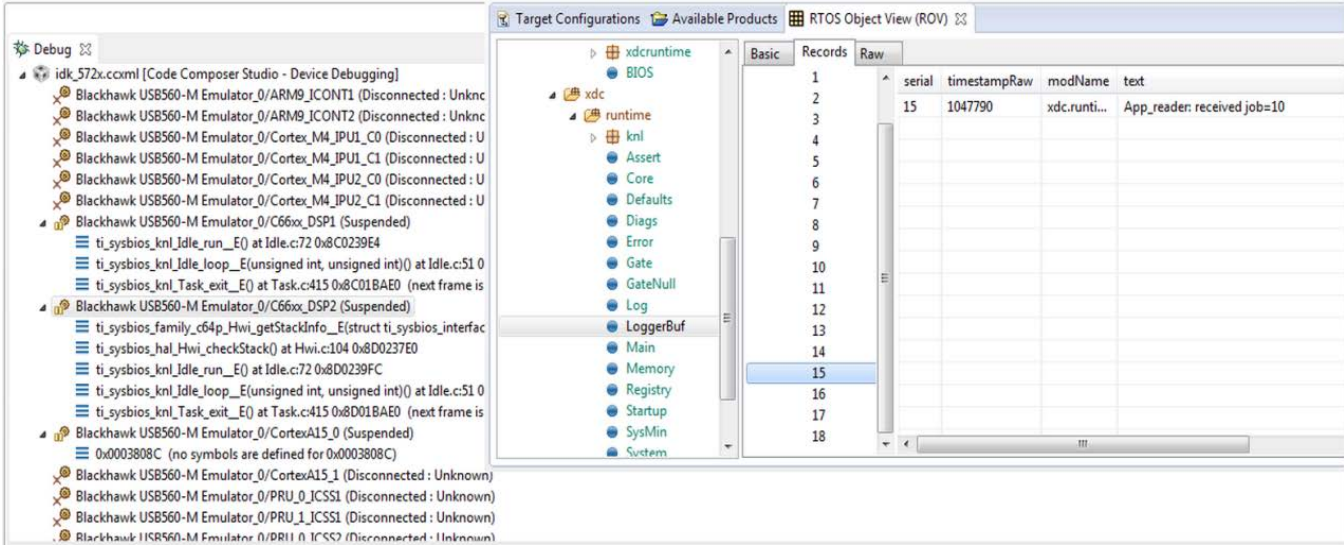
MessageQ: Send round-trip message from client to server and back

Ping: Send a message between all cores in the system

NotifyPeer: Use notify to communicate to a peer processor

Hello Example: Send one-way messages from writer to reader

CCS RTOS ROV Viewer for IPC Hello Example



Wiki: http://processors.wiki.ti.com/index.php/Running_IPC_Examples_on_DRA7xx/AM572x

IPC User Guide: http://processors.wiki.ti.com/index.php/IPC_Users_Guide

Processor SDK RTOS: Port

Application Development Using
Processor SDK RTOS

Customize

Port

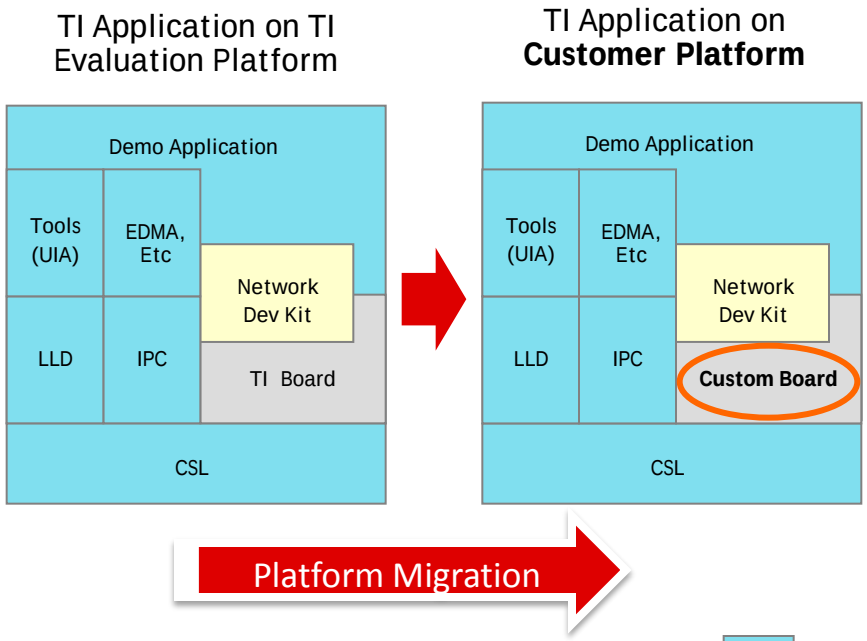
Develop

Run

Start

Setup

Processor SDK RTOS: Port



No modifications required

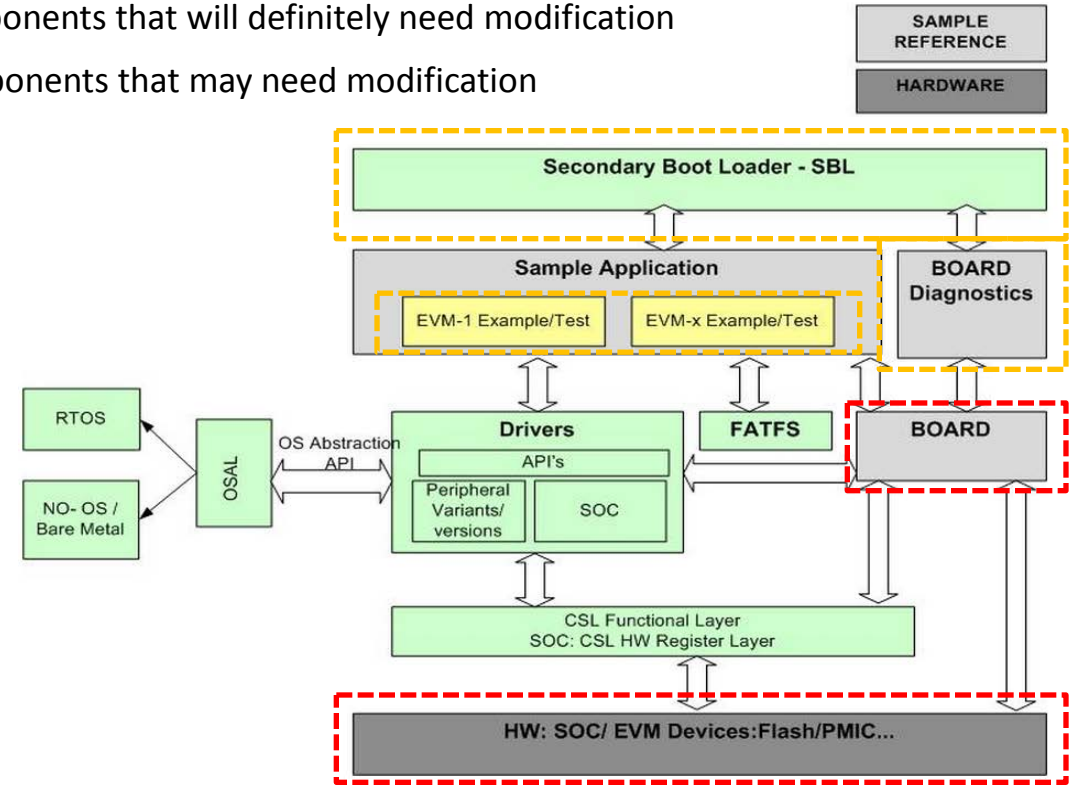
May be used "as is" or customer can implement value-add modifications

Needs to be modified or replaced with customer version

Processor SDK RTOS: Functional View

Components that will definitely need modification

Components that may need modification



Board Library: Configuration Options

Example code:

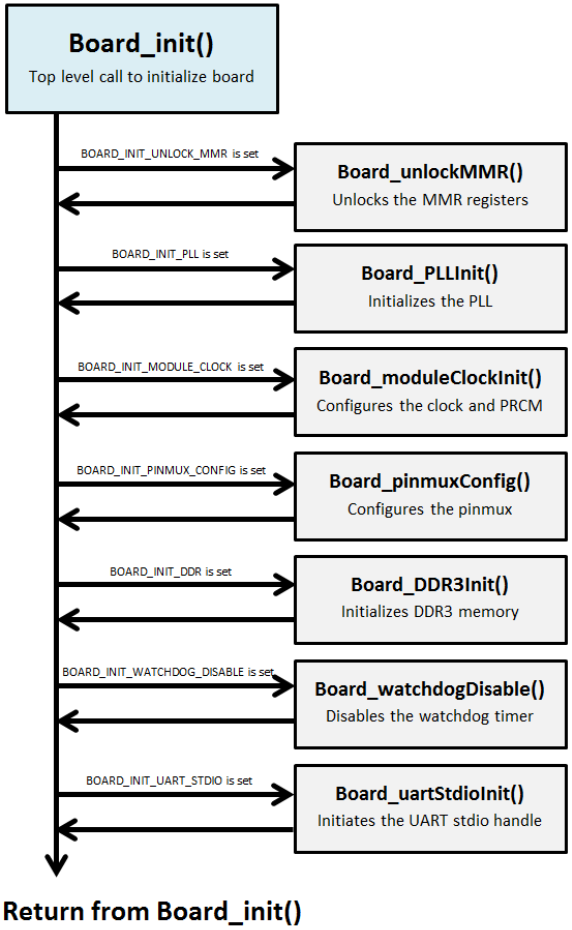
```
//Setting up for pinmux and uart
Board_STATUS ret;
Board_initCfg boardCfg;
boardCfg = BOARD_INIT_MODULE_CLOCK |
           BOARD_INIT_PINMUX_CONFIG |
           BOARD_INIT_UART_STDIO;
ret = Board_init(boardCfg);
```

API Header file:

"ti/board/board.h"

Library to link:

ti.board.aXX



See **Application Integration for AM5x** in the *RTOS Software Developer Guide*:
http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_Board_Support

See **Custom Board Addition** in the *RTOS Software Developer Guide*:
http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_Board_Support

Board Library: Modifying Source for Custom Platform

- PinMux
- Clocking
- DDR configuration
- IO configuration
- External components
- Board initialization

board ▸ src ▸ idkAM572x ▸ device

External components

enet_phy.c	
enet_phy.h	
qspi_flash.c	
qspi_flash.h	

pdk_am57xx_1_0_0 ▸ packages ▸ ti ▸ board ▸ src ▸ evmAM572x ▸

Name	Date modified	Type
device	9/15/2015 4:22 PM	File folder
include	9/15/2015 4:22 PM	File folder
boardPadDelay.h	9/11/2015 2:07 PM	C/C++ Header
boardPadDelayDevice.c	9/11/2015 2:07 PM	C Source
boardPadDelayInit.c	9/11/2015 2:07 PM	C Source
boardPadDelayTune.h	9/11/2015 2:07 PM	C/C++ Header
evmAM572x.c		
evmAM572x_clock.c		
evmAM572x_ddr.c		
evmAM572x_info.c	9/11/2015 2:07 PM	C Source
evmAM572x_lld_init.c	9/11/2015 2:07 PM	C Source
evmam572x_pinmux.c	9/11/2015 2:07 PM	C Source
evmAM572x_pll.c	9/11/2015 2:07 PM	C Source
iodelay_config.c	9/11/2015 2:07 PM	C Source
iodelay_config.h	9/11/2015 2:07 PM	C/C++ Header

PinMux

DDR configuration

Board Initialization

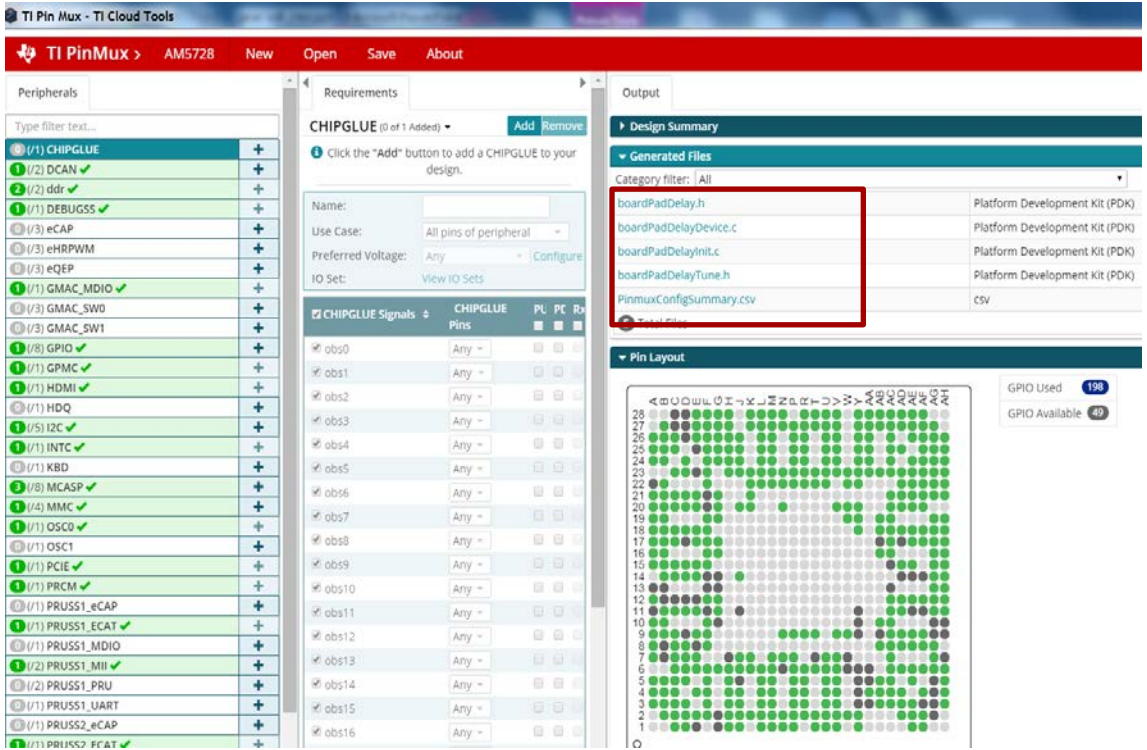
Clocking

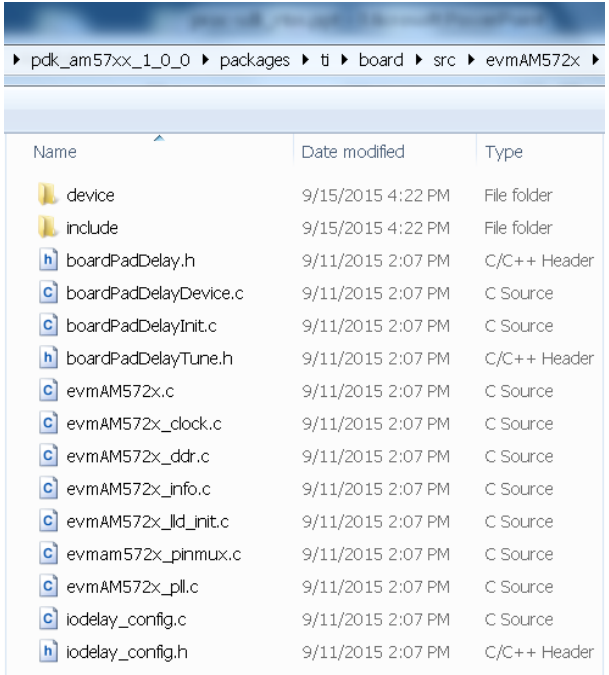
pdk_am57xx_1_0_0 ▸ packages ▸ ti ▸ board ▸ src ▸ evmAM572x ▸ include

IO configuration

Name	Date modified	Type
board_cfg.h	9/11/2015 2:07 PM	C/C++ Header
board_internal.h	9/11/2015 2:07 PM	C/C++ Header
evmam572x_pinmux.h	9/11/2015 2:07 PM	C/C++ Header

Processor SDK RTOS: Modifying Board PinMux Settings



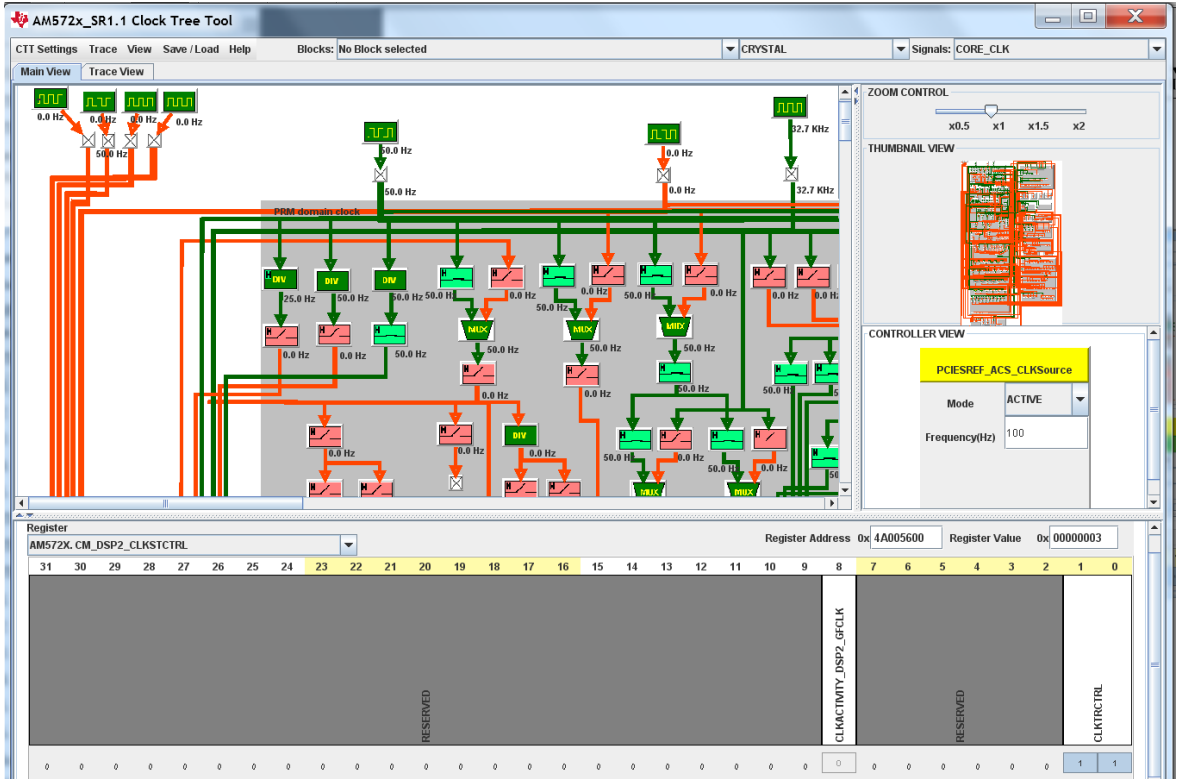


Pinmux Utility Download: http://www.ti.com/tool/PINMUXTOOL

AM57xx Sitara IO Configuration Requirements : http://www.ti.com/lit/an/sprac44/sprac44.pdf

Board Library: Clock Tree Tool to Simulate SoC Clocks

- Interactive Clock Tree Tool (CTT) for configuration:
  - Helps with visualization of the device clock tree
  - Allows users to customize clock tree as per specific use-case
- The CTT GUI is composed of 5 sub-views:
  - Main View
  - Thumbnail View
  - Controller View
  - Register View
  - Trace View
- Allows users to save register settings that can then be used to configure the software.



Clock Tree Tool Download: http://www.ti.com/tool/CLOCKTREETOOL

DDR Configuration Tools

Platform Parameters

CS1 populated	Yes	
Chip-select interleaving	Yes	If both chip-selects are populated, enabling the chip-select interleaving will modify the address decoding so that incrementing addresses will first hit pages in CS0, then CS1, back to CS0, and so on.
Turnaround time between chip-selects, in SDRAM clock cycles	3	Applies to read-to-read and write-to-write transitions from one chip-select to the other
SDRAM Data Bus Width	32bits	
SDRAM low-power mode	None	Automatic low-power mode settings.
Clock Stop Timer	Immediate	In number of clock cycles
Power-down Timer	Immediate	In number of clock cycles
Self-refresh Timer	Immediate	In number of clock cycles
Deep power-down enable	No	Enabling this bit will put the SDRAM in deep power-down. All memory contents will be lost. This mode is not available in DDR3.
Max Number of LL Transactions in the Command FIFO	0	Must be a value between 0 and 10
Max Number of MPU Transactions in the Command FIFO	10	Must be a value between 0 and 10
Max Number of SYS Transactions in the Command FIFO	10	Must be a value between 0 and 10
SDRAM Clock Frequency	1333MHz	
Operating Performance Point	OPP_NOM	Ratio programmed in the PRCM - DDR3 may only use OPP_NOM
Time between 2 short ZQ calibration commands	50ms	
Perform a long ZQ calibration when exiting self-refresh	Yes	
Perform a long ZQ calibration when exiting power down	No	
Perform ZQ calibration on both channels simultaneously	No	Enabling this feature requires separate calibration resistors for each chip select
Enable ZQ calibration for CS0	No	Only disables calibration runs after initialization. ZQINIT is still performed at boot time.
Enable ZQ calibration for CS1	No	Only disables calibration runs after initialization. ZQINIT is still performed at boot time.
Maximum number of SDRAM clock cycles to unlock the DDR PHY DLL	128	

Cover page | Platform Parameters | Performance Settings | DDR3 Leveling Configuration | Delay Lines Configuration | SDRAM Par

Timing Parameters

SDRAM Type	DDR3	
SDRAM Geometry	Using 2x 4Gb pieces (256M x16) per rank per channel	
Data Bus Width	32bits	
Row Addresses	R0-R14	
Column Addresses	C0-C9	
Bank Addresses	BA0-BA2	
Bank Interleaving	Full	
SDRAM Refresh Rate	Normal	
SDRAM Timings	Timings for an Elpida DDR3-1333 (EDJ4216EBBG-AE-F)	
CL	9	
CWL	7	
tREFI	7.8us	
tWTR	7.5ns	
tRTW	6.0ns	
tFAW	40.0ns	Changed this to 40ns per Micron Data sheet
tRRD	7.5ns	Changed this to 7.5ns per Micron Data sheet
tRC	48.8ns	Changed this per Micron Data sheet
tRAS	35.0ns	
tRASmax	70.2us	
tWR	15.0ns	
tRCD	13.8ns	
tRP	13.8ns	
tCKE	5.6ns	
tRTP	7.5ns	
tXSRD	512 tCK	
tXSNR	270.0ns	
tXP	6.0ns	

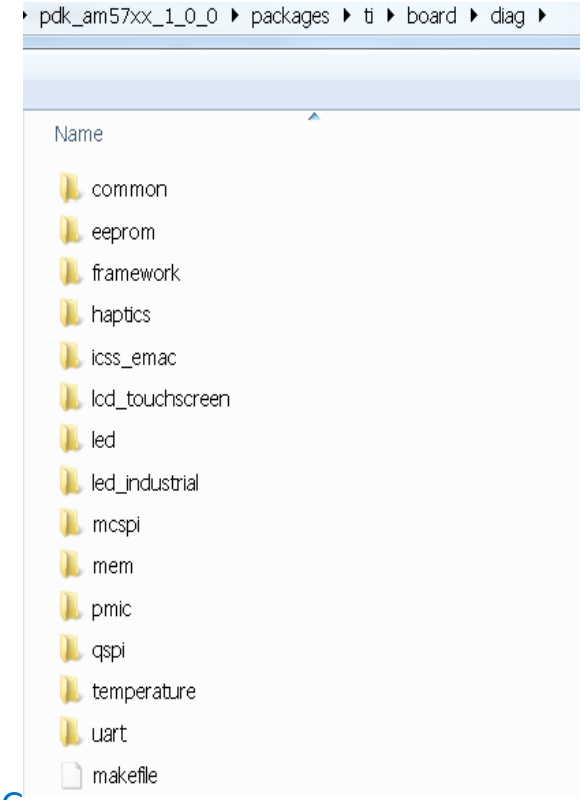
Hardware Leveling

Enable write leveling	Yes	
Enable read gate training	Yes	
Enable read data eye training	Yes	
Incremental write leveling interval	0us	Only applicable to DDR3, set to 0 to disable this incremental training.
Incremental read DQS training interval	0us	Only applicable to DDR3, set to 0 to disable this incremental training.
Incremental read data eye training interval	0us	Only applicable to DDR3, set to 0 to disable this incremental training.
DQS gate training number of samples	8	Number of samples used during each training iteration.
Write leveling number of samples	7	Number of samples used during each leveling iteration.

Refer to the **AM57x EMIF Tools** application note: http://www.ti.com/lit/an/sprac36/sprac36.pdf

Diagnostics: Tests to Bring up Custom Hardware

- Software to verify the functionality of on-board peripherals and external interfaces of each board.
- Constitute of ARM based bare metal (non-OS) code designed to validate TI EVM hardware
- Tests can be adapted to test new boards and/or peripherals.
- Validation suite utilizes:
  - board library for hardware configuration
  - UART drivers for standard output
  - relevant peripheral drivers for which the test are designed.
- Tests can be manually executed over an emulator or can be run off a SD card.



Wiki Link: http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_DIAG

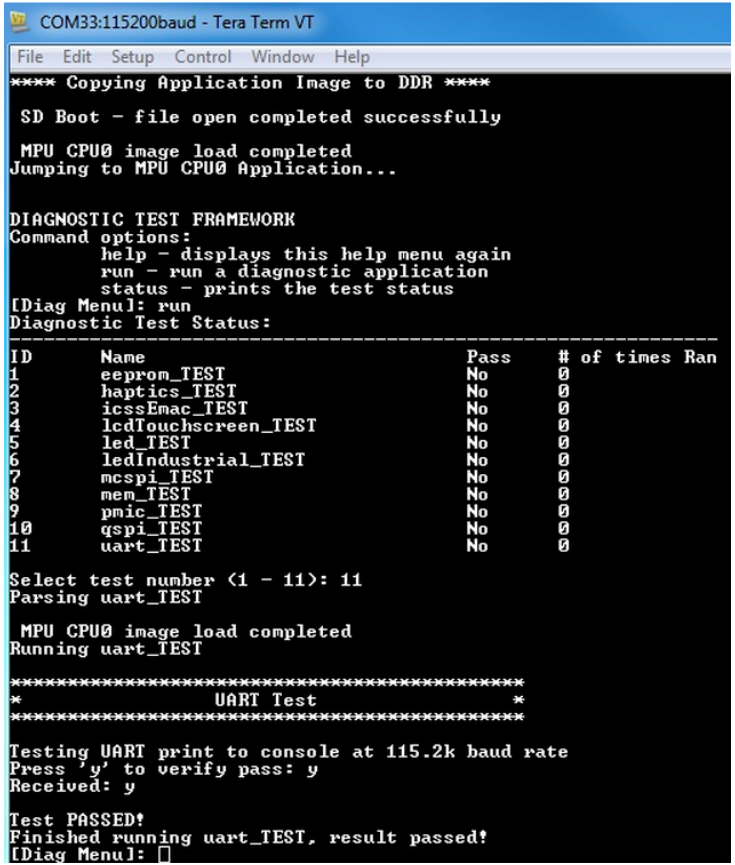
Diagnostics: Tests in the Board Package

Common tests:

- **UART**: Testing UART standard IO by sending/receiving characters at 115.2k baud
- **GPIO LEDs:** Flash the LEDs connected to GPIO on board
- **I2C LEDs:** Flash the LEDs connected to I2C on board
- **EEPROM**: Read/write to eeprom connected to I2C
- **DDR read/write:** Writes and reads back bits in the DDR memory
- **MCSPI**: Similar to QSPI, multichannel SPI also reads/writes to connected memory

For complete list of diagnostics for your SoC, refer to:

http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_DIAG



Processor SDK RTOS: Customize

Application Development Using
Processor SDK RTOS

- **Customize**
- Port
- Develop
- Run
- Start
- Setup

Processor SDK RTOS: Application Customization

Start with the example template of Image Processing demo

Add ARM or DSP algorithms, processing, tasks code

Customize and Run

Develop and run custom application

Example Application Template: Image Processing Demo

- Typical RTOS Application development starts from an existing template.
- CCS provide SYS BIOS application template with typical or minimal configurations.

Example application template for training:
processor_sdk_rtos_am57xx_2_xx_xx_xx\demos\image_processing

Steps for building a custom application:

- Include header files for all drivers and OS dependencies
- Configure the BIOS configuration file to link to required driver libraries.
- Creation of task for adding application functionality.
- Porting and optimizing IPC configuration for communication with slave cores.
- Add algorithm for processing.

Application Development: Includes and Initialization

Include required header files:

```
/* TI CSL Header files */
#include <ti/csl/cslr_device.h>

/* SD/MMC and FAT FS Header files */
#include "MMCSD_log.h"
#include <ti/fs/fatfs/diskio.h>
#include <ti/fs/fatfs/FATFS.h>
#include <ti/drv/mmcsd/MMCSD.h>

/* UART COnsole IO header files */
#include <ti/drv/uart/UART.h>
#include <ti/drv/uart/UART_osal.h>
#include <ti/drv/uart/UART_stdio.h>

#include <ti/board/board.h>
```

Add headers for other drivers here.

Board initialization:

```
Board_initCfg boardCfg;

boardCfg = BOARD_INIT_PINMUX_CONFIG |
           BOARD_INIT_MODULE_CLOCK | BOARD_INIT_UART_STDIO;

Board_init(boardCfg);
```

Create application tasks and custom algorithms here.

```
/* Start BIOS */
BIOS_start();
return (0);
```

NOTE: Slide does not include SYSBIOS and XDC-related includes.

Application Development: Create Tasks to Add Features

SYS/BIOS ▸ Scheduling ▸ Task - Instance Settings

Module Instance Advanced

Tasks

echo

Add ... Remove

Required Settings

Handle	echo
Function	gpio_test
Priority	1

Use the vital flag to prevent system exit until this thread exits

☑ Task is vital

Stack Control

Stack size	4096
Stack memory section	.bss
Stack pointer	null
Stack heap	null

Thread Context

Argument 0	0
Argument 1	0
Environment pointer	null

type filter text

- BIOS
- Cache (ti.sysbios.hal)
- Cache (ti.sysbios.family.arm.a15)
- Clock
- Defaults
- Diags
- Error
- Hwi (ti.sysbios.family.arm.gic)
- Hwi (ti.sysbios.hal)
- IntXbar
- Log
- LoggerBuf
  - logger0
- Main
- Memory
- Mmu
- Program
- Semaphore
- SemiHostSupport
- Swi
- SysMin
- System
- Task
  - New Task...
  - Stop Using Task
  - Help
  - Build Configurations
- Te

Add function gpio_test to the application source.

Application Development: Modifying Configuration Script

IPC libraries:

```
xdc.useModule('ti.sdo.ipc.Ipc');
xdc.useModule('ti.sdo.ipc.MessageQ');
xdc.useModule('ti.sdo.ipc.SharedRegion');
xdc.useModule('ti.sdo.utils.MultiProc');
var HeapBufMP     = xdc.useModule('ti.sdo.ipc.heaps.HeapBufMP');
```

Add other IPC modules here.

OSAL libraries for TI RTOS:

```
/* Load the OSAL package */
var osType = "tirtos"
var Osal = xdc.useModule('ti.osal.Settings');
Osal.osType = osType;
```

Change default SYSBIOS settings here.

SoC platform and board libraries to link:

```
/* Load the Board package and set the board name */
var Board = xdc.loadPackage('ti.board');
/* Board.Settings.boardName = "idkAM572x"; */
Board.Settings.boardName = "evmAM572x";
```

Driver libraries to link:

```
/* Load the MMCSD package */
var Mmcsd = xdc.loadPackage('ti.drv.mmcsd');
var Fatfs = xdc.loadPackage('ti.fs.fatfs');
var UART = xdc.loadPackage('ti.drv.uart');
```

Add other drivers to link here.

Wiki Link: http://processors.wiki.ti.com/index.php/IPC_Users_Guide/Porting_IPC

Application Development: Customize And Run

- Driver instance and interrupt configuration
- Memory configuration
- Debugging

Application Development: Customize Driver Instance

<Module>_soc.c binds driver with Default Driver Attributes on the board.

Hardware attributes includes base address, interrupt number, etc.
Module behavior can be configured statically … or dynamically during runtime.

For Static configuration:

```
/* Number of GPIO ports */
#define CSL_GPIO_PER_CNT    8U

/* GPIO Driver hardware attributes */
GPIO_v1_HwAttrs GPIO_v1_hwAttrs[CSL_GPIO_PER_CNT] = {
    {
        CSL_MPU_GPIO1_REGS,
#ifdef _TMS320C6X
        15,
#else
        61,
#endif
        0,
        55,
        0
    },
```

Dynamic Runtime Configuration

```
GPIO_v1_HwAttrs *hwAttrs = NULL;
uint32_t portNum = 1;
hwAttrs   = (GPIO_v1_HwAttrs *)&GPIO_v1_hwAttrs[(portNum - 1U)];
hwAttrs->line1IntNum = 62;
```

NOTE: The example shown refers to an ARM application.

Define Application Memory Map

SoC memory requires partitioning to allow all cores to have their own memory space and also to set up shared memory regions for cores.

Example: Application Memory Map

Memory Segment	Start Address	Length	Comments
OCMC_SBL	0x40300000	112KB	SBL reserved L3
OCMC_0	0x4031C000	400KB	Shared L3 section 1
OCMC_1	0x40400000	1MB	Shared L3 section 2
OCMC_2	0x40500000	1MB	Shared L3 section 3
DDR3_Shared1	0x80000000	50MB	Shared DDR region
DDR3_MPU	0x83200000	50MB	ARM code/data
DDR3_DSP	0x86400000	50MB	DSP code/data
DDR3_M4	0x89600000	50MB	M4 code/data

Creating Custom RTSC Platform For BIOS Applications

Platform Definition in BIOS: $BIOS_INSTALL_DIR\packages\ti\platforms\<PlatformName>

New Platform

Page 2 of 2 - Device Page

Enter Details for device

Device Details

Device Name	DRA7XX
Device Family	cortexa15
Clock Speed (MHz)	1500.0

Import...

Device Memory

Name	Base	Length	Space	Access
OCMC_RAM2	0x40400000	0x00100000	code/data	RWX
OCMC_RAM1	0x40300000	0x00080000	code/data	RWX
OCMC_RAM3	0x40500000	0x00100000	code/data	RWX

Customize Memory

External Memory

Name	Base	Length	Space	Access
DDR_Shared	0x80000000	0x03200000	code/data	RWX
DDR_ARM	0x83200000	0x03200000	code/data	RWX

Memory Sections

Code Memory: DDR_ARM Data Memory: DDR_ARM Stack Memory: DDR_ARM

< Back Next > Finish Cancel

New Platform

Page 2 of 2 - Device Page

Enter Details for device

Device Details

Device Name	DRA7XX
Device Family	c6000
Clock Speed (MHz)	600

Import...

Device Memory

Name	Base	Length	Space	Access
OCMC_RAM2	0x40400000	0x00100000	code/data	RWX
OCMC_RAM1	0x4031c000	0x00064000	code/data	RWX
OCMC_RAM3	0x40500000	0x00100000	code/data	RWX

L2 Cache: 128k L1D Cache: 32k L1P Cache: 32k

Customize Memory

External Memory

Name	Base	Length	Space	Access
DDR_Shared	0x80000000	0x03200000	code/data	RWX
DDR_DSP	0x86400000	0x32000000	code/data	RWX

Memory Sections

Code Memory: DDR_DSP Data Memory: DDR_DSP Stack Memory: DDR_DSP

< Back Next > Finish Cancel

New Platform

Page 2 of 2 - Device Page

Enter Details for device

Device Details

Device Name	DRA7XX
Device Family	cortexm4
Clock Speed (MHz)	212.8

Import...

Device Memory

Name	Base	Length	Space	Access
OCMC_RAM2	0x40400000	0x00100000	code/data	RWX
OCMC_RAM1	0x4031c000	0x00064000	code/data	RWX
L2_ROM	0x00000000	0x00004000	code/data	RWX

Customize Memory

External Memory

Name	Base	Length	Space	Access
DDR_Shared	0x80000000	0x03200000	code/data	RWX
DDR_M4	0x89600000	0x03200000	code/data	RWX

Memory Sections

Code Memory: DDR_M4 Data Memory: DDR_M4 Stack Memory: DDR_M4

< Back Next > Finish Cancel

Debugging SYSBIOS Applications

- SYSBIOS and IPC generate a highly optimized, minimally debug-able custom SYS/BIOS library that will link to your application.
- Building Debug-able SYSBIOS library in configuration file for your application:

```
var BIOS = xdc.useModule('ti.sysbios.BIOS');
BIOS.libType = BIOS.LibType_Debug; // build custom BIOS library.
BIOS.customCCOpts = BIOS.customCCOpts.replace("-o3", "-o0"); //change optimization level
BIOS.customCCOpts = BIOS.customCCOpts.replace("--opt_for_speed=2", ""); // For ARM only
```

- All PDK prebuilt libraries are built to support single-stepping into drivers and board libraries.
- In addition to single-stepping, ROV tools, RTOS analyzer and System Analyzer tools in CCS can be used to view logs, task execution logs, and benchmark applications.

For More Information

Processor SDK Downloads:

AM335x AM437x AM572x

C667x C665x 66AK2Ex 66AK2Gx 66AK2Hx 66AK2Lx

Software Documentation:

Processor_SDK_RTOS_Software_Developer_Guide

Hardware Wikis:

AM335x EVM AM437x EVM AM572x EVM

C6678 EVM C6657 EVM 66AK2Ex EVM 66AK2Gx EVM 66AK2Hx EVM 66AK2Lx EVM

Tools and Utilities:

PINMUX Utility Clocking Tree Utility DDR Timing & Hardware Leveling PRU_ICSS

TI RTOS Trainings:

TI RTOS Workshop Processor_SDK_RTOS Overview