Integrating several Processor SDK RTOS examples into one application

The Processor SDK RTOS provides fundamental platform software and tools for development, deployment and execution of RTOS-based applications. Driver examples are available in the package as a reference to users to understand driver API callflow and functionality testing. Those driver examples are standalone SYSBIOS projects. It is natural the users need to deal with how to integrate multiple drivers into their application after they exercised individual driver examples. However, such information is not given in the processor RTOS package.

This note illustrates how to integrate several SYSBIOS driver example projects, namely, EMAC, USB and PCIE into one application, and test on A15 core of AM572x IDK EVM, as an example. The same method also applies to other projects, cores or platforms.

1. Create CCS project

The Processor SDK RTOS uses a batch file called pdkProjectCreate to detect all CCS projects specified in text file format then creates the corresponding CCS projects, which in turn can be imported into CCS to build, load and run.

To show how to integrate these projects together, one may create a folder called “merge” (e.g. under packages\ti\drv) and a text file (e.g. Merge_BasicExample_idkAM572x_armBiosExampleProject.txt) in it. To avoid creating all the examples, the batch file “set module_list” can be changed to add this “merge” module and use “pdkProjectCreatet [soc] [board] [endian] merge [processor] [pdkDir]” to create the CCS project:

```
set module_list=all aif2 bcp cppi csl dfe emac fatfs fm fftc gpio gpmc hyplnk i2c icss_emac iqn2 mcasp mcbsp mmap
mmcsd nimu nimu_icss nwal pa pcie pktlib pruss qm rm sa serdes_diag spi srio tcp3d tfw transportqmss transportsrio
tsip uart usb wdtimer vps merge
```

The text file includes several parts:

- Source files
- configuration file
- Complier, linker options and linked libraries

Those are combined from individual SYSBIOS project text files. In this example, EMAC_BasicExample_idkAM572x_armBiosExampleProject, usb_host_msc_idkAM572x_arm_project and PCIE_idkAM572x_wSoCFile_armExampleProject are merged, from PROCESSOR-SDK-RTOS-AM57X 03_01_00_06 release. The same method is expected to work for future releases. The next sub-sections discuss how to merge the different parts of the text file.

1.1 Select source code files

Each individual project text file indicates the source code files used in the project. They can be copied into the “merge” folder and modified there, avoiding touch the originals. A main.c can also be created to replace the individual main files. Those files can be grouped under different subfolders for cleanness.

1.2 Merge SYSBIOS configuration file

The configuration file is also a merger of individual ones from all the projects. The typical configuration file has the follows entries:

- General configuration: SYSBIOS modules loaded
- BIOS configuration
- Task configuration
- Driver configuration: PDK driver modules loaded, this example includes:
    - EMAC, USB, PCIE
    - EDMA
    - CSL, OSAL, UART, I2C, BOARD, FATFS, GPIO
- Cache and MMU configuration
- Memory sections configuration: place pinmux in OCMC_RAM

The PDK driver loading order matters: the I2C driver has to be loaded before board library, as the latter calls into some I2C functions:

```
/* Load the I2C package - board package needs this to read EVM ID */
var I2c = xdc.loadPackage('ti.drv.i2c');

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

Otherwise, you may get errors when linking:

```
C:\ti\PDK_AM~4\packages\ti\board/src/idkAM572x/idkAM572x_lld_init.c:59: undefined reference to `I2C_init'
C:\ti\PDK_AM~4\packages\ti\board/src/idkAM572x/idkAM572x_lld_init.c:61: undefined reference to
`I2C_Params_init'
C:\ti\PDK_AM~4\packages\ti\board/src/idkAM572x/idkAM572x_lld_init.c:63: undefined reference to `I2C_open'
C:\ti\PDK_AM~4\packages\ti\board/src/idkAM572x/idkAM572x_lld_init.c:77: undefined reference to `I2C_transfer'
….
```

Cache configuration controls enabling cache or not. MMU configuration sets up the physical address to virtual address translation. It also sets up memory properties: peripheral configuration space is typically strongly ordered and non-cacheable, normal memory space are R/W cacheable and R/W allocatable.

1.3 Merge compiler/linker options and linked libraries

The compiler options are typical common across different projects, so it is easy to keep. The same is true when merging linker options from multiple projects.

The linked libraries from different projects need to be added in the merge text file.

1.4 Create multiple SYSBIOS task in main function

Each individual project has a main function already, which typically has following:

```
{
        Board initialization
        Other initialization routines (e.g. interrupt crossbar)
        Task creation
        BIOS starts
}
```

In the merged one, the main routine has the similar functions to replace all the individual main functions:

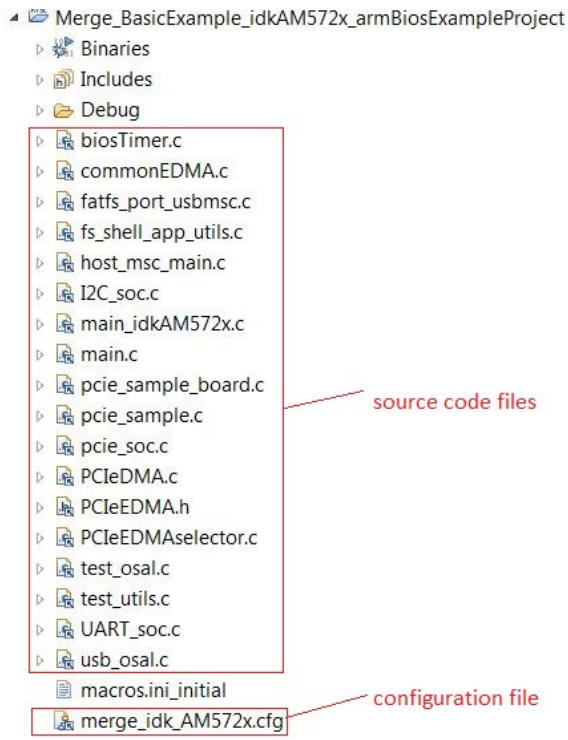
```
{
        Board initialization
        Initialization routines for project 1
        Initialization routines for project 2
        …
        Task creation for project 1
        Task creation for project 2
        …
        BIOS starts
}
```

The board initialization has to be called once only, also there has to be one BIOS_start() only. BIOS_start() starts up task threads which then takes over the application's flow of execution. All the application code after the BIOS_start() can't be executed.

2. Build and test

2.1 Import and build the CCS project

After finishing the file creations, the CCS project can be created and imported into CCS. The project has the follow files:



Compiler options:

Summary of flags set:

```
-mcpu=cortex-a15 -mtune=cortex-a15 -marm -mfloat-abi=hard -Dam5728 -Dcore0 -DSOC_AM572x -
DIDK_AM572X -Dgcc -I"C:/ti/pdk_am57xx_1_0_4/packages/ti/drv/usb" -I"C:/ti/pdk_am57xx_1_0_4/packages" -
I"C:/ti/pdk_am57xx_1_0_4/packages/ti/drv/usb/example/usb_host/msc" -
I"C:/ti/pdk_am57xx_1_0_4/packages/ti/drv/usb/src/usb_func/include" -
I"C:/ti/pdk_am57xx_1_0_4/packages/ti/drv/usb/src/include" -
I"C:/ti/pdk_am57xx_1_0_4/packages/ti/drv/usb/example/shell" -
I"C:/ti/pdk_am57xx_1_0_4/packages/ti/drv/pcie/example/sample/src" -
I"C:/ti/pdk_am57xx_1_0_4/packages/ti/drv/pcie/example/sample/am57x/src" -
I"C:/ti_6_1_3/ccsv6/tools/compiler/gcc-arm-none-eabi-4_9-2015q3/arm-none-eabi/include" -g -gdwarf-3 -
gstrict-dwarf -Wall -c -finstrument-functions -MMD -MP -mno-unaligned-access
```

Linker options:

Summary of flags set:

```
-mfloat-abi=hard -Dam5728 -Dcore0 -DSOC_AM572x -DIDK_AM572X -Dgcc -g -gdwarf-3 -gstrict-dwarf -
Wall -Wl,-Map,"Merge_BasicExample_idkAM572x_armBiosExampleProject.map" -nostartfiles -static -Wl,--gc-
sections -L"C:/ti/bios_6_45_01_29/packages/gnu/targets/arm/libs/install-native/arm-none-eabi/lib/fpu" -
L"$(BIOS_INSTALL_PATH)/packages/gnu/targets/arm/libs/install-native/arm-none-eabi/lib/fpu" -Wl,--
defsym,CORE0=1 -Wl,--defsym,STACKSIZE=0x10000 -Wl,--defsym,HEAPSIZE=0x400
```

An .out file called Merge_BasicExample_idkAM572x_armBiosExampleProject should be built.

2.2 Hardware setup

The testing of the integrated application is no different from the testing of individual applications, with the same hardware setup:

- AM572x IDK EVMs
- JTAG emulators
- EMAC: internal loopback with 10 packets, no loopback cable or Ethernet cable required
- USB: host example, a USB stick is required in USB1 port (J23), one can run "ls" command to list all the files on the USB stick
- PCIE: one IDK EVM as PCIE root complex, the other as PCIE EP point, connected via PCIEx1 crossover cable, using J22 PCIEx1 connector, see setup at: http://processors.wiki.ti.com/index.php/Processor_SDK_RTOS_PCIe

2.3 Load and test

Load the .out file and run, look at the UART output for the correct execution of the project.

3. Debug the application

As expected, the integrated project doesn't work as it is without further debug and code changes.

3.1 EMAC loopback failure

It is found out that the EMAC loopback doesn't work. One needs to isolate whether it is caused by USB or PCIE initialization code or USB or PCIE task functions, by commenting out the initialization code or changing task priorities. Then, it is clear that USB initialization code cause the EMAC Rx ISR failure.

In USB code:

```
CSL_xbarMpuIrqConfigure(CSL_XBAR_INST_MPU_IRQ_76,  CSL_XBAR_USB1_IRQ_INTR0);  /* main irq */
CSL_xbarMpuIrqConfigure(CSL_XBAR_INST_MPU_IRQ_77,  CSL_XBAR_USB1_IRQ_INTR1);  /* misc irq */

CSL_xbarMpuIrqConfigure(CSL_XBAR_INST_MPU_IRQ_78,  CSL_XBAR_USB2_IRQ_INTR0);  /* main irq */
CSL_xbarMpuIrqConfigure(CSL_XBAR_INST_MPU_IRQ_92,  CSL_XBAR_USB2_IRQ_INTR1);  /* misc irq */
```

In EMAC code:

```
CSL_xbarMpuIrqConfigure(CSL_XBAR_INST_MPU_IRQ_92, CSL_XBAR_GMAC_SW_IRQ_RX_PULSE);
CSL_xbarMpuIrqConfigure(CSL_XBAR_INST_MPU_IRQ_93, CSL_XBAR_GMAC_SW_IRQ_TX_PULSE);
```

It is seen that both USB2_IRQ_INTR1 and GMAC_SW_IRQ_RX_PULSE are routed to the same MPU interrupt 92, this is a conflict. The simplest fix is to remove the cross bar 92 configuration in USB code, as only USB 1 is used in test. To illustrate how to change the driver to resolve the issue, by looking at AM572x Technical Reference Manual (http://www.ti.com/lit/ug/spruhz6h/spruhz6h.pdf):

MPU_IRQ_92 (ID124)	87	CTRL_CORE_MPU_IRQ_92_93[8:0]	87	USB2_IRQ_INTR1	USB2 interrupt 1

The MPU_IRQ_92 by default comes from USB2_IRQ_INTR1. The EMAC borrows this interrupt for some reason. One can change this to an unused MPU IRQ#, e.g., 103 for IVA_IRQ_SYNC_1, in EMAC code:

```
    CSL_xbarMpuIrqConfigure(CSL_XBAR_INST_MPU_IRQ_103, CSL_XBAR_GMAC_SW_IRQ_RX_PULSE);
```

The EMAC driver in \ti\drv\emac\soc\am572x\emac_soc.c also needs to update, then rebuild the EMAC library:

```
EMAC_HwAttrs_V4 EMACInitCfg[1] =
{
{
……
#ifdef _TMS320C6X
    13,
    75, /* CSL_XBAR_INST_DSP1_IRQ_75 is available */
    12,
    76, /* CSL_XBAR_INST_DSP1_IRQ_76 is available */
#elif defined(__ARM_ARCH_7A__)
    103 + 32, /* corresponds to  CSL_XBAR_INST_MPU_IRQ_92 + 32 */
    0,
    93 + 32, /* corresponds to  CSL_XBAR_INST_MPU_IRQ_92 + 32 */
    0,
….
}
```

3.2 PCIE data exchange failure

The PCIE example is designed to test between two TI EVMs with data exchange and interrupt. When one side is loaded with the integrated project in PCIE RC mode, the other side is loaded with the original PCIE project in EP mode, the PCIE link comes up however the data exchange stuck. In debugging into this, it finds out the dstBuf is asymmetric on both sides as the code is asymmetric:

```
dstBuf_t dstBuf
#ifdef __ARM_ARCH_7A__
__attribute__((aligned(256), section(".bss:dstBufSec"))) // GCC way of aligning
#endif

  pcieBase = (char *)pcieBase +
                     PCIE_WINDOW_MEM_BASE +  /* data area doesn't start at low address */
                     (((uint32_t)&dstBuf) & 0xfff); /* dstBuf needs to be 4K aligned in addr tran
*/
```

To fix this: `__attribute__((aligned(4096), section(".bss:dstBufSec"))) // GCC way of aligning`

After the issues are addressed, the application runs as expected.

4. Boot from a SD card

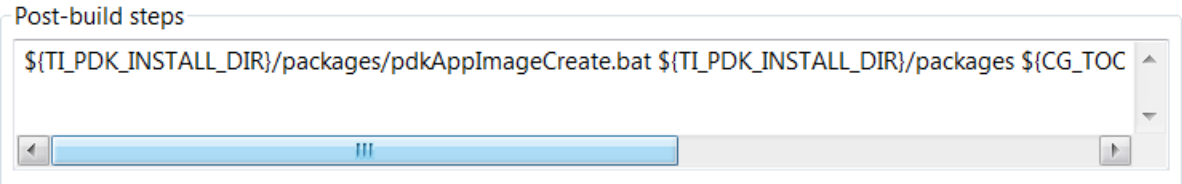
The application can run standalone with JTAG/CCS approach. Or, it can also boot from a SD card. The former uses the GEL file to initialize the SOC. The latter depends on the SBL for the same. The PINMUX setting can only be run once. As the SBL runs the PINMUX setting already, the same code needs to be commented out from standalone application, when run from the SD card. Like other Processor SDK RTOS diagnostic applications (ti/board/diag), this can be done by undefine PDK_RAW_BOOT:

```
#ifdef PDK_RAW_BOOT
       cfg = BOARD_INIT_UART_STDIO | BOARD_INIT_PINMUX_CONFIG | BOARD_INIT_MODULE_CLOCK;
#else
```

```
        cfg = BOARD_INIT_UART_STDIO;
#endif
```

So only BOARD_INIT_UART_STDIO flag is passed into Board_init(), then rebuild the application.

The CCS post-build step is needed to convert the .out file into multi-core bootable image format, the output is called “app”:




```
${TI_PDK_INSTALL_DIR}/packages/pdkAppImageCreate.bat ${TI_PDK_INSTALL_DIR}/packages ${CG_TOOL_ROOT}
${PROJECT_LOC}/${ConfigName} ${ProjName} AM572x arm
```

To boot from the SD card:

- Copy the Merge_BasicExample_idkAM572x_armBiosExampleProject\Debug\app to a SD card
- Copy pdk_am57xx_1_0_x\packages\ti\boot\sbl\binary\idkAM572x\mmcsd\bin\MLO to the same SD card
- Insert SD card into EVM and power cycle

The below is expected from UART console. Note: the USB output depends on the files on the USB stick. If the application runs standalone, the UART output is almost same, just without the first two lines indicating ARM A15 booting from the SD card.

```
COM6 - Tera Term VT
File  Edit  Setup  Control  Window  Help
 SD Boot - file open completed successfully
Jumping to MPU CPU0 Application...

Board_init success
EMAC loopback test application initialization
SetPhyMode:000001e1 Auto:1, FD10:64, HD10:32, FD100:256, HD100:128, FD1000:0 LPBK:0
SetPhyMode:000001e1 Auto:1, FD10:64, HD10:32, FD100:256, HD100:128, FD1000:0 LPBK:0
main: emac_open sucess
link status: Up    Link in loopback state
received packet: 1
received packet: 2
received packet: 3
received packet: 4
received packet: 5
received packet: 6
received packet: 7
received packet: 8
received packet: 9
received packet: 10
All tests have passed

RTOS USB Host MSC example!!
got intr. portNum = 0. ptrEvent=0x810f4000
-> PSC on port 1
xhciPortNum!=2. Using USB2.0 initialization sequence
about to do port reset! PORTSC1=0x6e1
about to do enable slot. PORTSC1=0xe03
about to set Address
about to get device descriptor
about to get Configuration descriptor
USBHostConfigureEndpoint. EP=1. pks=767. epType=6
USBHostConfigureEndpoint. EP=2. pks=767. epType=2
about to set Configuration
0:>D---- 2007/06/05 11:38         0  test_empty
D---- 2007/06/05 11:38         0  test
D---- 2013/03/18 14:06         0  Drivers and manuals
----A 2013/11/22 10:14    233984  Socket setup Complete.doc
----A 2013/11/22 10:16     75520  udp.pcapng
----A 2013/11/20 17:12     50456  cap.pcapng
----A 2013/11/20 17:12      1100  mylog.txt
----A 2013/10/15 13:45      1734  Readme.txt
----A 2013/11/22 12:14   4636678  helloWorld_evmc6670_customer.out
----A 2013/11/22 12:19   4636678  helloWorld_evmc6670_no_100.out
----A 2013/12/06 11:13      9064  pingtest_1206.pcapng
----- 2014/02/24 04:01    392720  u-boot-spi-keystone-evm.gph
----- 2014/02/24 04:01    327120  u-boot-keystone-evm.bin

  10 File(s),  10365054 bytes total
   3 Dir(s),    1926432K bytes free
0:>*********************************************
*              PCIe Test Start                *
*                 RC mode                     *
*********************************************

Version #: 0x02020006; string PCIE LLD Revision: 02.02.00.06:Oct  4 2016:15:22:26

PCIe Power Up.
PLL configured.
Successfully configured Inbound Translation!
Successfully configured Outbound Translation!
Starting link training...
Link is up.
Checking link speed and # of lanes
Expect 1 lanes, found 1 lanes (PASS)
Expect gen 2 speed, found gen 2 speed (PASS)
Root Complex received data.
RC waiting for 10 of each of 2 types of interrupts
RC got all 20 interrupts
Root Complex DMA received data.
EDMA Test passed.

=== PCIe results using DMA transfers (optimal)===
Passed 1 tokens round trip (read+write through PCIe) in 47676 cycles
=== this is not an optimized example ===
Test passed.
```

5. Appendix

5.1 Text file for creating CCS project

```
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/emac/main_idkAM572x.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/emac/test_utils.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/emac/am572x/test_osal.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/uart/am572x/UART_soc.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/i2c/am572x/I2C_soc.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/usb/fs_shell_app_utils.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/usb/fatfs_port_usbmsc.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/usb/host_msc_main.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/usb/biosTimer.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/usb/usb_osal.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/pcie/am572x/pcie_soc.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/pcie/pcie_sample.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/pcie/pcie_sample_board.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/pcie/EDMA/commonEDMA.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/pcie/EDMA/PCIeEDMA.h"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/pcie/EDMA/PCIeDMA.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/pcie/EDMA/PCIeEDMAselector.c"
-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/src/main.c"

-ccs.linkFile "PDK_INSTALL_PATH/ti/drv/merge/example/cfg/am572x/armv7/bios/merge_idk_AM572x.cfg"

-ccs.setCompilerOptions "-c -mcpu=cortex-a15 -mtune=cortex-a15 -marm -mfloat-abi=hard -DSOC_AM572x -
DIDK_AM572X -Dgcc -g -gstrict-dwarf -gdwarf-3 -finstrument-functions -Wall -MMD -MP -mno-unaligned-access -
I${PDK_INSTALL_PATH}/ti/drv/usb -I${PDK_INSTALL_PATH} -
I${PDK_INSTALL_PATH}/ti/drv/usb/example/usb_host/msc -I${PDK_INSTALL_PATH}/ti/drv/usb/src/usb_func/include
-I${PDK_INSTALL_PATH}/ti/drv/usb/src/include  -I${PDK_INSTALL_PATH}/ti/drv/usb/example/shell -
I${PDK_INSTALL_PATH}/ti/drv/pcie/example/sample/src -
I${PDK_INSTALL_PATH}/ti/drv/pcie/example/sample/am57x/src" -rtsc.enableRtsc
-ccs.setLinkerOptions "-lrdimon -lgcc -lm -lnosys -nostartfiles -static -Wl,--gc-sections -
L$(BIOS_INSTALL_PATH)/packages/gnu/targets/arm/libs/install-native/arm-none-eabi/lib/fpu"
```

5.2 SYSBIOS configuration file

```
/**
 *  \file  merge_ idk_AM572x.cfg
 *
 *  \brief  Sysbios config file for emac example project on AM572X IDK EVM.
 *
 */

/*
 * Copyright (C) 2015 Texas Instruments Incorporated - http://www.ti.com/
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 *
 * Redistributions in binary form 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.
 *
 * Neither the name of Texas Instruments Incorporated nor the names of
 * its contributors may be used to endorse or promote products derived
 * from this software without specific prior written permission.
```

```
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 * "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 THE COPYRIGHT
 * OWNER OR CONTRIBUTORS 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 SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 *
 */
/* ================ General configuration ================ */
var Edma = xdc.loadPackage ("ti.sdo.edma3.drv.sample");
var drv = xdc.loadPackage ("ti.sdo.edma3.drv");
var rm = xdc.loadPackage ("ti.sdo.edma3.rm");
var Defaults                    = xdc.useModule('xdc.runtime.Defaults');
var Diags                       = xdc.useModule('xdc.runtime.Diags');
var Error                       = xdc.useModule('xdc.runtime.Error');
var Log                         = xdc.useModule('xdc.runtime.Log');
var LoggerBuf                   = xdc.useModule('xdc.runtime.LoggerBuf');
var Main                        = xdc.useModule('xdc.runtime.Main');
var System                              = xdc.useModule('xdc.runtime.System');
var Text                        = xdc.useModule('xdc.runtime.Text');
var HeapMem = xdc.useModule('ti.sysbios.heaps.HeapMem');
var Hwi = xdc.useModule('ti.sysbios.hal.Hwi');
var InitXbar    = xdc.useModule("ti.sysbios.family.shared.vayu.IntXbar");
var IntXbar             = xdc.useModule('ti.sysbios.family.shared.vayu.IntXbar');
var Clock                       = xdc.useModule('ti.sysbios.knl.Clock');
var Timer                               = xdc.useModule('ti.sysbios.timers.dmtimer.Timer');
var Swi                         = xdc.useModule('ti.sysbios.knl.Swi');
var Task                        = xdc.useModule('ti.sysbios.knl.Task');
var Semaphore                   = xdc.useModule('ti.sysbios.knl.Semaphore');
var ti_sysbios_hal_Hwi = xdc.useModule('ti.sysbios.hal.Hwi');
var SysMin                              = xdc.useModule('xdc.runtime.SysMin');
var SemiHostSupport = xdc.useModule('ti.sysbios.rts.gnu.SemiHostSupport');
var Memory                              = xdc.useModule('xdc.runtime.Memory'); // from USB
var Program             = xdc.useModule("xdc.cfg.Program");    // from USB
/*
 * Program.argSize sets the size of the .args section.
 * The examples don't use command line args so argSize is set to 0.
 */
Program.argSize = 0x0;
/* System stack size (used by ISRs and Swis) */
Program.stack                   = 0x20000;
Program.heap        = 0x10000;
/*
 * Uncomment this line to globally disable Asserts.
 * All modules inherit the default from the 'Defaults' module.  You
 * can override these defaults on a per-module basis using Module.common$.
 * Disabling Asserts will save code space and improve runtime performance.
Defaults.common$.diags_ASSERT = Diags.ALWAYS_OFF;
 */

/*
 * Uncomment this line to keep module names from being loaded on the target.
```

```
 * The module name strings are placed in the .const section. Setting this
 * parameter to false will save space in the .const section.  Error and
 * Assert messages will contain an "unknown module" prefix instead
 * of the actual module name.
Defaults.common$.namedModule = false;
 */

/*
 * Minimize exit handler array in System.  The System module includes
 * an array of functions that are registered with System_atexit() to be
 * called by System_exit().
 */
System.maxAtexitHandlers = 4;

/*
 * Uncomment this line to disable the Error print function.
 * We lose error information when this is disabled since the errors are
 * not printed.  Disabling the raiseHook will save some code space if
 * your app is not using System_printf() since the Error_print() function
 * calls System_printf().
Error.raiseHook = null;
 */

/*
 * Uncomment this line to keep Error, Assert, and Log strings from being
 * loaded on the target.  These strings are placed in the .const section.
 * Setting this parameter to false will save space in the .const section.
 * Error, Assert and Log message will print raw ids and args instead of
 * a formatted message.
Text.isLoaded = false;
 */

/*
 * Uncomment this line to disable the output of characters by SysMin
 * when the program exits.  SysMin writes characters to a circular buffer.
 * This buffer can be viewed using the SysMin Output view in ROV.
SysMin.flushAtExit = false;
 */


/* Circular buffer size for System_printf() */
SysMin.bufSize = 0x400;
System.SupportProxy = SysMin;


/* ================ BIOS configuration ================ */

var BIOS = xdc.useModule('ti.sysbios.BIOS');
/*
 * Build a custom SYS/BIOS library from sources.
 */
BIOS.libType = BIOS.LibType_Custom;


/*
 * The BIOS module will create the default heap for the system.
 * Specify the size of this default heap.
 */
```

```
BIOS.heapSize = 0x10000;

/*
** Create the stack Thread Task for our application.
*/
BIOS.taskEnabled                    =   true;

/*
 * Create and install logger for the whole system
 */
var loggerBufParams = new LoggerBuf.Params();
loggerBufParams.numEntries = 16;
var logger0 = LoggerBuf.create(loggerBufParams);
Defaults.common$.logger = logger0;
Main.common$.diags_INFO = Diags.ALWAYS_ON;
/* ================ Task configuration ================ */
/* Define and add one Task Hook Set */
Task.addHookSet({
        registerFxn: '&TaskRegisterId',
        switchFxn: '&mySwitch',
});

Task.defaultStackSize = 4096;
Task.idleTaskStackSize = 4096;

/* ================ Driver configuration ================ */
var deviceType          = "am572x";
/*use CSL package*/
var Csl                 = xdc.loadPackage('ti.csl');
Csl.Settings.deviceType  = deviceType;

/* Load Profiling package */
var Utils = xdc.loadPackage('ti.utils.profiling');

/* Load the OSAL package */
var osType          = "tirtos";
var Osal = xdc.useModule('ti.osal.Settings');
Osal.osType = osType;
/* Load the uart package */
var socType          = "am572x";
var Uart            = xdc.loadPackage('ti.drv.uart');
Uart.Settings.socType = socType;

/* Load the I2C package - board package needs this to read EVM ID */
var I2c = xdc.loadPackage('ti.drv.i2c');

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

/* Load the EMAC packages */
var Emac = xdc.loadPackage('ti.drv.emac');
Emac.Settings.socType = socType;

var Usb = xdc.loadPackage('ti.drv.usb');
var usbSettings = xdc.useModule('ti.drv.usb.Settings');
usbSettings.socType = "am572x";
```

```
/* fat file system for USB host example */
var Fatfs = xdc.loadPackage('ti.fs.fatfs');

/* Load and use the PCIE packages */
var Pcie = xdc.loadPackage('ti.drv.pcie');
Pcie.Settings.enableProfiling = true;
/* Enable only if soc-specific library should be used */
/* Pcie.Settings.socType = socType; */ /* use soc/am572x/src/pcie_soc.c */

/* Load the gpio package */
var Gpio = xdc.loadPackage('ti.drv.gpio');
Gpio.Settings.socType = socType;

/* ================ Cache and MMU configuration ================ */

/*******************************************************************************
 *
 * SYS/BIOS assigns the following defaults to MAIR0 ATTR0, ATTR1 and ATTR2:
 *
 * ATTR0 -> 0x44 (mark memory region as non-cacheable normal memory)
 * ATTR1 -> 0x04 (mark memory region as device memory, i.e. strongly
 * ordered and non-cacheable)
 * ATTR2 -> 0xFF (mark memory region as normal memory, RW cacheable and
 * RW allocate)
 *******************************************************************************
 */
var Cache = xdc.useModule('ti.sysbios.family.arm.a15.Cache');
var Mmu = xdc.useModule('ti.sysbios.family.arm.a15.Mmu');

// Disable the cache
Cache.enableCache = false;

// Enable the MMU (Required for L1/L2 data caching)
Mmu.enableMMU = true;

// descriptor attribute structure
var attrs0 = new Mmu.DescriptorAttrs();
Mmu.initDescAttrsMeta(attrs0);

attrs0.type = Mmu.DescriptorType_BLOCK;  // BLOCK descriptor
attrs0.noExecute = true;                 // not executable
attrs0.accPerm = 0;                      // read/write at PL1
attrs0.attrIndx = 1;                     // MAIR0 Byte1 describes

// Define the base address of the 2 MB page
// the peripheral resides in.
var peripheralBaseAddrs = [
  { base: 0x4ae00000, size: 0x00100000 },  // PRM
  { base: 0x51000000, size: 0x00800000 },  // pcie_ss1 regs
  { base: 0x51800000, size: 0x01000000 },  // pcie_ss2 regs
  { base: 0x20000000, size: 0x10000000 },  // pcie_ss1 data
  { base: 0x30000000, size: 0x10000000 },  // pcie_ss2 data
  { base: 0x48880000, size: 0x00030000 },  // usb1 regs
  { base: 0x488C0000, size: 0x00030000 },  // usb2 regs
  /* Following registers for IODELAY/PINMUX */
  { base: 0x4844a000, size: 0x00001000 },
  { base: 0x48447000, size: 0x00001000 },
  { base: 0x4a002000, size: 0x00001000 },
```

```
  { base: 0x43300000, size: 0x00300000 }
];

// Configure the corresponding MMU page descriptor accordingly
for (var i =0; i < peripheralBaseAddrs.length; i++)
{
  for (var j = 0; j < peripheralBaseAddrs[i].size; j += 0x200000)
  {
     var addr = peripheralBaseAddrs[i].base + j;
     Mmu.setSecondLevelDescMeta(addr, addr, attrs0);
  }
}

var attrs1 = new Mmu.DescriptorAttrs();
Mmu.initDescAttrsMeta(attrs1);

attrs1.type = Mmu.DescriptorType_BLOCK;    // BLOCK descriptor
attrs1.shareable = 2;                      // sharerable
attrs1.attrIndx = 2;                       // Cached, normal memory, original value was 2

// Set the descriptor for each entry in the address range
for (var i=0x80000000; i < 0xA0000000; i = i + 0x00200000) {
   // Each 'BLOCK' descriptor entry spans a 2MB address range
   Mmu.setSecondLevelDescMeta(i, i, attrs1);
}

var attrs2 = new Mmu.DescriptorAttrs();
Mmu.initDescAttrsMeta(attrs2);

attrs2.type = Mmu.DescriptorType_BLOCK;
attrs2.noExecute = true;
attrs2.accPerm = 0;       // R/W at PL1
attrs2.attrIndx = 2;      // Use MAIR0 Byte2
Mmu.setMAIRMeta(2, 0x04);
Mmu.setSecondLevelDescMeta(0x43200000, 0x43200000, attrs2);


/* ================ Memory sections configuration ================ */
Program.sectMap["BOARD_IO_DELAY_DATA"] = "OCMC_RAM1";
Program.sectMap["BOARD_IO_DELAY_CODE"] = "OCMC_RAM1";
```