SYSLIB

Release Notes

Applies to Product Release: 4.00.01.00
Publication Date: October 29, 2015

Document License

Contents

Release Notes

SYSLIB 4.00.01.00

1 INTRODUCTION

1.1 Overview

This document provides the release information for the SYSLIB software package. The SYSLIB package includes the following:-

- SYSLIB Release Notes
- SYSLIB User's Guide
- Source code of all SYSLIB components
- Pre-built libraries (Little Endian) of all SYSLIB components
- API reference guide
- Software Manifest

This is an engineering tested alpha release package. Release notes from previous releases are also available in the release notes archive directory

2 RELEASE OVERVIEW

2.1 Hardware Device Support

The device and platforms tested for this release include:

- K2H
- K2K

Please review the Device section for more details.

2.2 Components and Tools

The SYSLIB package is verified/tested using the **MCSDK 3.01.04.07** package. Please refer to the MCSDK Release notes for a list of all the component information. The following is the list of additional packages which were used to test the release:

1. SNOW3G 1.0.0.2

2. CUIA 1.01.00.06 Custom
3. UIA 2_00_03_43
4. SA3GPP Enabler 3.0.0.0

The SYSLIB supports **only the RT kernel** from the MCSDK release. Please use the RT DEVKIT for the development of user space applications.

2.3 Licensing

Please refer to the software manifest

2.4 MCSDK Patches

The section documents the MCSDK Patches which need to be added to the base MCSDK release.

2.4.1 Memory Reserve Size

Please ensure that the following environment variable is defined and saved in the UBOOT environment:-

```
setenv mem_reserve 1536M
```

This will ensure that the kernel reserved the higher order 1.5GB of memory for the DSP. Failure to do so will result in the kernel overwriting DSP memory. Application developers can modify and customize the DSP & ARM memory map. The default DSP SYSLIB memory map which is released in the `SYSLIB_INSTALL_PATH/ti/platforms` assumes the above reservation.

2.4.2 Installing the SA 3GPP Enabler to Linux devkit

As mentioned above the SA3GPP enabler is a prerequisite. While installing the SA3GPP; the installer will request for the PDK Path. This will ensure that the SA3GPP Installer will be correctly found and the DSP applications will be built properly. However the installer does not update the RT Linux development kit and so the following manual steps need to be done:

- Create directory `sa3gppEnabler` under the `ARAGODIR/include/ti/drv/sa`
- Copy the `sa3gpp.h` from the `PDK_INSTALL_PATH/ti/drv/sa/sa3gppEnabler` to the `ARAGODIR/include/ti/drv/sa/sa3gppEnabler`
- Copy the `sa3gppver.h` from the `PDK_INSTALL_PATH/ti/drv/sa/sa3gppEnabler` to the `ARAGODIR/include/ti/drv/sa/sa3gppEnabler`

- Copy the library `libsa3gpp.a` from the `PDK_INSTALL_PATH/ti/drv/sa/sa3gppEnabler/lib/armv7` to the `ARAGODIR/lib` folder

NOTE: Due to licensing the SA3GPP enabler is **not** enabled in the default NETFP Server executable. For customers to have signed the 3GPP license the NETFP Server (`ti/apps/netfp_server/netfp_server.c`) needs to be patched as described below:-

```
gNetfpServerMCB.netfpServerHandle = Netfp_initServer (&serverConfig, &errCode);
if (gNetfpServerMCB.netfpServerHandle == NULL)
...

/* Enable the SA 3GPP Enabler: */
Sa_3gppEnabler();
```

This patch will allow customers to use the EEA1 and EIA1 services. Failure to apply the patch will cause the LTE channel creation to fail.

2.4.3 Installing the BCP header files to Linux devkit

The SOC Initialization application is now capable of initializing and configuring the BCP. This requires the BCP header files. The BCP header files are not located in the RT Linux development kit. It is thus required to copy all the header files from the `PDK_INSTALL_PATH/ti/drv/bcp` directory to the ARAGO directory.

- Create directory `bcp` under the `ARAGODIR/include/ti/drv`
- Copy all the header files from the `PDK_INSTALL_PATH/ti/drv/bcp` to the `ARAGODIR/usr/include/ti/drv/bcp`

This is **only** required if the SOC Initialization application is being built.

2.4.4 PA library patch

PA library need to be patch to support new feature from this SYSLIB release. To patch PA libraries, please follow the following manual steps:

- Copy libpa.a, libpa2.a, libpa.so.1.0.0 and libpa2.so.1.0.0 under mcsdk_patches directory to `ARAGODIR/lib directory`
- Copy ti.drv.pa.ae66, ti.drv.pa.ae66e, ti.drv.pa2.ae66 and ti.drv.pa2.ae66e under mcsdk_patches directory to `PDK_INSTALL_PATH/ti/drv/pa/lib/c66`

2.4.5 Netlink symbolic link

A symbolic link needs to be added under to `ARAGODIR/include` directory.

- `ln –s libnl3/netlink/ netlink`

2.4.6 DTS File Updates

NOTE: Please integrate the SYSLIB released DTS files for the specific device with your application and always update the kernel DTB files and SYSLIB RMv2 DTB files. Failure to do so will result in out of the box failures.

2.4.6.1 K2H/K2K

The kernel DTS files have been modified for the following features:-

- GIC Queues 8722 to 8735 were originally reserved for the Linux kernel. These queues are not used by the Linux kernel so these have been marked as unreserved and could not be used by the ARM applications
- Wiring of the GIC Queue and INTC_SET2 interrupt queues from using the UIO module.

Along with the kernel DTS file; the SYSLIB RMv2 files have also been modified for the following features:-

- GIC Queues 8722 onwards have been marked as usable
- INTC_SET2 queues have been allocated to ARM
- Wildcarding support
- Simplified L2 and L3 QoS shapers. This is for illustration only. Customers are recommended to modify the shapers as per their requirements.

2.4.6.2 K2L

The kernel DTS files have been modified for the following features:-

- GIC Queues 546 to 559 were originally reserved for the Linux kernel. These queues are not used by the Linux kernel so these have been marked as unreserved and could not be used by the ARM applications
- Wiring of the GIC Queue and SOC_SET_1 interrupt queues from using the UIO module.

Along with the kernel DTS file; the SYSLIB RMv2 files have also been modified for the following features:-

- GIC Queues 546 onwards have been marked as usable
- SOC-SET1 queues have been allocated to ARM
- Wildcarding support

- Simplified L2 and L3 QoS shapers. This is for illustration only. Customers are recommended to modify the shapers as per their requirements.

3 What's new

3.1 New Features

The section documents the new features which have been added to the SYSLIB Release:

3.1.1 Path MTU

NETFP Proxy is now listening to all ICMPv4 and ICMPv6 Error messages. On the reception of a Fragmentation needed message or ICMPv6 Packet Too Big error message. The NETFP Proxy will notify the NETFP Server about the newly received MTU. This information is then propagated to all the NETFP Clients and Sockets.

Sockets can use `Netfp_Option_DONT_FRAG` to ensure that all packets sent have the DF bit in the IPv4 header set. Sockets are notified about the Path MTU change through the registered call back function with the reason code set to `Netfp_Reason_PMTU_CHANGE`. Sockets can also read the MTU associated with the socket through the socket option `Netfp_Option_SOCK_MTU`

PMTU entries are aged in the NETFP Server (*Timeout:* 10 minutes)

The NETFP Proxy supports a new message "`NETFP_PROXY_IPC_MSGTYPE_SETUP_PMTU_REQ`" which allows an OAM application to manually configure the PMTU of a given path.

3.1.2 FAPI Tracing library

3.1.2.1 FAPI Memory Logging

Memory Logging for FAPI logs are added in rel10-d1 demo. The feature is disabled by default to allow FAPI messages to ship out through the Ethernet port. It can be enabled by setting the FAPI_MEMLOGGING to 1. FAPI Memory logging is done through a MEMLOG channel and Socket post-routing Hook. Post processing script is provided to convert the memory dump to Wireshark file format.

3.1.2.2 FAPI Embedded mode support

In this release, the FAPI tracing library is updated to take into account FAPI header version 2.3. The new FAPI header version includes the support of both embedded mode and reference mode.

FAPI reference mode can be used when transferring FAPI messages between DSP cores. There is only 1 TX.request message and 1 RX_ULSCH.message transferred per TTI, addressing all of the PDUs to be transmitted or received in that TTI, respectively. The TX.request messages and

RX_ULSCH.indication messages contain pointers to the payloads in global memory space. This mode is supported by earlier FAPI tracing library versions (FAPI version 2.2 and earlier).

FAPI embedded mode is newly introduced with FAPI header version 2.3 and is primarily used for transferring FAPI messages between ARM and DSP cores, as in the case where the LTE stack is located on the ARM and LTE PHY is located on the DSPs. In this mode, there is 1 TX.request message per PDU in the downlink and 1 RX_ULSCH.indication message per PDU in the uplink. Multiple TX.requests and multiple RX_ULSCH.indication messages may be transferred for each TTI. The term embedded refers to the encapsulation of the payload data within the FAPI message, immediately following the last field of the embedded structures of the TX.request and RX_ULSCH.indication messages. The TX.request messages and RX_ULSCH.indication messages may arrive at the receiver in any order with respect to the "*pduIndex*" field value of the FAPI message.

The distinction between reference and embedded mode, outside of the number of TX.request and/or RX_ULSCH.indication messages received in a particular TTI, is via the "*numPdus*" field. A value of "0" in this field indicates that the TX.request or RX_ULSCH.indication message is an embedded mode message; the embedded structure in the union *TiFapi_txReqPdu_u* or *TiFapi_rxUlschPdu_u* should be used, with the payload immediately following the last field of the embedded structure for the FAPI message.

Note that when FAPI embedded mode is employed, the functions *FapiTracing_compressTxRequest()* and *FapiTracing_compressRxULSCHInd()* should not be called to update the "msgLen" field of the TX.request and RX_ULSCH.indication message headers directly, as was done for FAPI reference mode messages. Instead, the value returned from calling these functions should be summed with the payload size for the PDU, and the resulting sum should be written to the "*msgLen*" field.

3.1.3 Multicast services

The NETFP module has been extended to support the reception of multicast packets. The NETCP allows pre-classification of multicast packets. By default these packets were pushed to the Linux Ethernet driver for processing. It is now possible to change the configuration on the fly.

In order to receive multicast packets there are two modes which can be specified while creating a fast path:-

1. Shared Mode

   In this mode the NETFP creates two shared entries in the LUT1-1 for all IPv4 and IPv6 multicast. The application configures the multicast pre-classification configuration to pass packets all packets to an application for further inspection. The application analyzes the packets to determine if the packets need to be forwarded to Linux or

should they be passed to the NETFP Fast Paths. The application uses the Raw Interface to send out the packets.

2. Not specified Mode

   In this mode the multicast fast paths are treated similarly to the Unicast fast paths. The LUT1-1 entries are programmed per IPv4/IPv6 Multicast address. Applications need to ensure that they either create an Ethernet rule for the LUT1-0 match.

3.1.4 SYSLIB Runtime Directory

In the older SYSLIB releases '*local*' file descriptors opened by the SYSLIB ARM core libraries were placed in the present working directory. The location of this directory can now be controlled by an environment variable '`SYSLIB_RUNTIME_DIR`'. The default value is `/var/run`.

3.1.5 Frame Protocol CRC

Syslib supports the creation of frame protocol channels and offloading the computation and verification of frame protocol CRC. To enable CRC verification in the receive path, use `setSocketOption - Netfp_Option_SOCK_3GPP_CFG` before the socket bind. To enable CRC computation in the transmit path, use `setSocketOption - Netfp_Option_FRAME_PROTOCOL_CRC_OFFLOAD`. Netfp master's netfp.conf has been modified to add a configuration option to enable Frame Protocol CRC services.

3.2 API Changes

3.2.1 Reason codes

In order to support the Path MTU feature there has been a change to the reason code names to clearly differentiate between the Path MTU and Interface MTU change.

This implies that the older definition `Netfp_Reason_MTU_CHANGE` has not been renamed to `Netfp_Reason_IF_MTU_CHANGE`.

3.2.2 UINTC Select

The function `Uintc_select` has been modified to include the additional argument of timeout. The new definition is as follows:-

```
int32_t Uintc_select
(
UintcHandle     uintcHandle,
struct timeval* ptrTimeout,
int32_t*        errCode
);
```

The argument `ptrTimeout` is an optional argument which can be set to NULL if the UINTC module select is to be blocked indefinitely.

The return codes for the function have also been modified. Please refer to the API Documentation for more information.

3.2.3 NETFP Proxy Plugin

The plugin configuration has been modified to accept a new call back function called “report PMTU”. This call back function is invoked on the reception of an ICMP Fragmentation needed messages or an ICMPv6 Packet Too Big message. This function allows developers to:-

- Specify that Path MTU Discovery not be done on a given path.
- Change the PMTU value associated with a given path.

3.2.4 Inbound Fast Path Configuration

There is a new field added to the inbound fast path configuration structure:

```
typedef struct Netfp_InboundFPCfg
{
 /**
  * @brief   Name of the fast path: This should be unique in the system
  */
 char                    name[NETFP_MAX_CHAR];
 ...

 /**
  * @brief   Multicast mode: This is applicable only if the fast path is to
  * receive multicast packets.
  */
 Netfp_MulticastMode     multicastMode;
...
} Netfp_InboundFPCfg;
```

For existing inbound fast path configurations please ensure that the multicast mode is set to 0 i.e. `Netfp_MulticastMode_NOT_SPECIFIED`.

3.2.5 NETFP Master Configuration

The NETFP Master configuration file has been modified to allow separate pre-classification configuration for multicast and broadcast. Please refer to the sample configuration files in the `ti/apps/netfp_master` directory.

3.2.6 SOC Initialization configuration

The SOC Initialization application has been enhanced to ensure the power up of other peripherals:-

- TCP3D

- VCP

The application can also be used to configure the DDR3 memory configuration. In order to do so the configuration file format has been modified. Please refer to the sample configuration files in the `ti/apps/soc_init` directory

3.2.7 Add Ethernet Rule configuration

Configuration to add "any" VLAN id is changed. In older releases, when vlanId is 0, it is treated as "any" VLAN Id, hence the vlan id field is not used to match LUT 1-0 rules. This behavior is corrected by using a separate parameter "anyVlanId". When this flag is set to 1, it is treated as "any" VLAN id; If this flag is set 0, vlanId field will be used to match LUT 1-0 rules.

3.2.8 User Statistics support

It is now possible to add user statistics counters to the Ethernet rules. To support this feature the structure '`Netfp_EthRuleCfg`' has been modified and the following fields have been added:-

```
typedef struct Netfp_EthRuleCfg
{
...
uint32_t   outPort;

/**
 * @brief   Number of User stats associated with this rule. Support both
 * bytes and packets stats
 */
uint32_t    numUserStats;

/**
 * @brief   User Statistics configuration for the Ethernet Rule
*/
Netfp_UserStatsCfg    userStatsCfg[NETFP_MAX_USR_STATS_PER_RULE];
}
```

Please ensure that the RMv2 DTS files have the resource `pa-num64bUserStats` defined and configured as per the system configuration. This is required to determine the number of 64 bit counters which need to be supported in the system. The resources `pa-32bUsrStats` and `pa-64bUsrStats` also need to be modified appropriately depending upon the configuration

Please refer to the sample RMv2 DTS files in the SYSLIB.

User statistics support can also be hooked up with the IP rules. This configuration is global for a NETFP Server and is controlled via the NETFP Server configuration file. The following line will ensure that all IP rules will have a 32 bit packet counter added to each IP rule (Fast Path and SA). If there are no counters available then the API will fail.

```
# Enable user statistics counter to be instantiated with IP rules
```

```
IP_USER_STAT_COUNTER_SUPPORT=1
```

Default value is to have no counter support

3.2.9 HPLIB OSAL

In order to support the HPLIB multiple process safe allocation and cleanup the HPLIB library exposes the OSAL API (`Osal_hplibCsEnter` and `Osal_hplibCsExit`). The implementation of these API should be the same across the entire system else the critical section operations will not work. Thus the implementation of these functions has been moved into the SYSLIB Resource Management library.

This implies that the applications which had previously defined these implementations in their OSAL need to remove them; else the builds will fail with a linking error for duplicate symbols.

3.2.10 FAPI Tracing Library

FAPI tracing library has API changes. Two parameters have been introduced for FapiTracing_init().

```
fapiTracingCfg.transportCfg.srcUDPPort  =  23003;
fapiTracingCfg.transportCfg.sin_family     = Netfp_SockFamily_AF_INET;
```

3.3 SYSLIB 4.0.1 Bug/Feature update list from JIRA:

Issue Type	Key	Summary
Story	SCLTE-1870	Support WCDMA Frame Protocol CRC calculations
Bug	SCLTE-1542	Some Syslib modules use printf instead of osals or System_printf
Bug	SCLTE-1924	Syslib4 uses dynamic local file handles
Bug	SCLTE-2330	High volume traffic generated by multiple fragmented flows using both IPv4 and IPv6 locks SA
Bug	SCLTE-2387	Route lookup failed under bridge interface
Bug	SCLTE-2408	Packet size not incremented after adding L2 Header in Netfp_sendSecureIPv4Pkt
Story	SCLTE-2414	PMTU Discovery for UP
Bug	SCLTE-2416	Startup fails with a netfpproxy error
Bug	SCLTE-2420	Child interface created with incorrect MTU for bridge interface
Task	SCLTE-2421	SOC init changes for powerup of LTE related peripherals and BCP config
Bug	SCLTE-2424	_Netfp_secureBind -function sets always NETFP_EINTERNAL if Netfp_addGTPTunnel fails
Bug	SCLTE-2425	Netfp_registerReassemblyService pointer variable "errCode" not always set

Bug	SCLTE-2426	NetFP proxy has same error print in multiple places and it does not display return value
Bug	SCLTE-2429	syslib4 cppcheck errors
Bug	SCLTE-2432	Josh_submitJob does not set errCode if arguments are not correct
Story	SCLTE-2436	NetFP function to send packet to linux ethernet driver
Story	SCLTE-2437	NetFP function to send packet to NetCP/PDSP1
Story	SCLTE-2438	NetFP function to send packet out via given switch egress port
Story	SCLTE-2439	NetFP master static preclassification configuration of separate multicast and broadcast flows.
Story	SCLTE-2440	Dynamic pre-classification configuration for multicast and broadcast flows.
Story	SCLTE-2449	PKTLIB API to copy metadata from one packet descriptor to another
Bug	SCLTE-2454	Incorrect pacing mode for msgcom accumulated channel configuration
Bug	SCLTE-2455	Reassembly channel configuration pacing timeout
Bug	SCLTE-2456	Extra system call in netfp master thread to enable interrupts
Story	SCLTE-2468	Multiprocess safe HPLIB initialization
Bug	SCLTE-2490	NetfpServer_parseClientList function failure
Bug	SCLTE-2491	Invalid/zero IPv4 header causes an infinite loop in re-assembly function
Story	SCLTE-1494	FAPI Tracing Buffering
Bug	SCLTE-2199	PA does not support NAT-T with previous IP link
Story	SCLTE-2334	Include LTE Demos in Release
Story	SCLTE-2349	Port to MCSDK 3.1.4
Bug	SCLTE-2427	Error in Netfp_delIPSecChannelLUTEntry
Bug	SCLTE-2433	Add FAPI Embedded mode support in rel10 D1 demo
Story	SCLTE-2471	User Defined NetCP Counters - LUT1-0 matches
Bug	SCLTE-2311	Coverity bug fixes
Story	SCLTE-2475	Syslib Demo working in FAPI embedded mode

3.4 Known Issues:

Issue Type	Key	Summary
Bug	SCLTE-2443	errCode values changed in josh jobs even without an error
Bug	SCLTE-2045	syslib4 netfp_server may crash during cell shutdown
Bug	SCLTE-2019	Fixed 1GHz clock used in DAT_TIME_ELAPSED
Bug	SCLTE-1612	while(1) loop in msgcom code needs to be removed.
Bug	SCLTE-2506	Software Frame Protocol CRC computations are not supported on ARM
Story	SCLTE-2240	Add DAT support for K2L in syslib4
Story	SCLTE-1377	Outer IP Fragmentation option
Task	SCLTE-2312	eQOS/Cascading has not been verified on K2L

RELEASE BUILDING

SYSLIB release build & environment configuration scripts which are located in the SYSLIB Install directory `scripts` folder. Please setup the following environment variables:-

```
export
ARMTOOLS_INSTALL_PATH=/home/share/tools/gcc-linaro-arm-linux-gnueabihf-4.7-2013.03-20130313_linux
export
ARAGO_INSTALL_PATH=/home/share/ti/mcsdk_linux_3_01_04_07_devkit_rt/sysroots/cortexa15t2hf-vfp-neon-linux-gnueabi
export CGT_INSTALL_PATH=/home/share/ti/cgt_7.4.12
export XDC_INSTALL_PATH=/home/share/ti/xdctools_3_31_02_38_core
export PDK_INSTALL_PATH=/home/share/ti/pdk_keystone2_3_01_04_07/packages
export SNOW3G_INSTALL_PATH=/home/share/ti/snow3g_1_0_0_2/packages
export UIA_INSTALL_PATH=/home/share/ti/uia_2_00_03_40_eng/packages
export INSTALL_JAMMER_INSTALL_PATH=/home/share/tools/installjammer-1.2.15
export BIOS_INSTALL_PATH=/home/share/ti/bios_6_41_04_54/packages
export IPC_INSTALL_PATH=/home/share/ti/ipc_3_36_02_13/packages
export CUIA_INSTALL_PATH=/home/share/tools/cuia_1_01_00_06Custom
export SYSLIB_INSTALL_PATH=/home/share/work/k2_dev/syslib
export SYSLIB_DEVICE=k2h
```

The environment variables are illustrative and should be modified by the customer as per their install paths. Once configured please setup the build environment by executing the following script:-

```
cd scripts
source setupenv.sh
```

This will setup the build environment and will also sanity check to make sure that all the required environment variables are configured.

3.5 Building the ARM Libraries, Servers & Unit Tests

Once the build environment is configured; please execute the following script to build the libraries for a specific device:-

```
cd scripts
source dev.sh <DEV_NAME> <ARM_BUILD> <DSP_BUILD> <DEMO_BUILD> <ARM_UNIT_TEST>
<DSP_UNIT_TEST>
```

Argument	Description
DEV_NAME	Name of the device for which the builds need to be done. Valid values are k2h, k2k and k2l
ARM_BUILD	Set to 1 to build the ARM libraries and standard SYSLIB Servers
DSP_BUILD	**Always set to 0**. To build the DSP Libraries please refer below
DEMO_BUILD	Set to 1 to build the DEMO for the specific device
ARM_UNIT_TEST	Set to 1 to build the ARM Unit Test for all the SYSLIB modules
DSP_UNIT_TEST	Set to 1 to build the DSP Unit Test for all the SYSLIB modules

Example: To rebuild the ARM Libraries/applications for K2H

```
source dev.sh k2h 1 0 0 0 0
```

Example: To build the ARM Libraries & Unit Tests for K2L

```
source dev.sh k2l 1 0 0 1 0
```

3.6 Building the DSP Libraries

Ensure that the SYSLIB_DEVICE is correctly configured in the environment variable. The example below selects the device as K2L

```
export SYSLIB_DEVICE=k2l
```

Modify the environment variable

```
export SYSLIB_INSTALL_PATH=~/ti/syslib_4_00_01_00
```

NOTE: There is no /packages at the end of the SYSLIB_INSTALL_PATH

Once configured please setup the build environment again by executing the following script:-

```
cd scripts
source setupenv.sh
```

To rebuild SYSLIB DSP Libraries; please do the following from the top level directory:-

```
xdc clean –PR .
xdc –PR .
```

3.7 Building the DSP Unit Tests

DSP Unit Tests are built using the script described above. **Example:** To build all the DSP Unit Tests for K2L

```
source dev.sh k2l 0 0 0 0 1
```

4 Device Support

Please read the following section which documents details about each SYSLIB supported device:

4.1 K2H

Kernel DTS Files	**ti/runtime/resmgr/dts/k2h**
RMv2 DTS Files	ti/runtime/resmgr/dts/k2h
DSP Memory Map	ti/runtime/platforms/tmdxevm6638lxe
ARM Compilation Flags	-D_LITTLE_ENDIAN -D__ARMv7 -DDEVICE_K2 –DDEVICE_K2H -D_GNU_SOURCE -D_VIRTUAL_ADDR_SUPPORT
DSP Compilation Flags	--define=DEVICE_K2 --define=DEVICE_K2H
PA Library on DSP	var Pa = xdc.useModule('ti.drv.pa.Settings'); Pa.deviceType = “k2h”
PA Library on ARM	-lpa
SOC Sample configuration file	ti/apps/soc_init/soc_k2h.conf
NETFP Master configuration file	ti/apps/netfp_master/netfp.conf
Library & Executable Suffix	_k2h

4.2 K2K

Kernel DTS Files	**ti/runtime/resmgr/dts/k2h**
RMv2 DTS Files	ti/runtime/resmgr/dts/k2h
DSP Memory Map	ti/runtime/platforms/tmdxevm6638lxe
ARM Compilation Flags	-D_LITTLE_ENDIAN -D__ARMv7 -DDEVICE_K2 –DDEVICE_K2K -D_GNU_SOURCE -D_VIRTUAL_ADDR_SUPPORT
DSP Compilation Flags	--define=DEVICE_K2 --define=DEVICE_K2K
PA Library on DSP	var Pa = xdc.useModule('ti.drv.pa.Settings'); Pa.deviceType = “k2k”
PA Library on ARM	-lpa
SOC Sample configuration file	ti/apps/soc_init/soc_k2k.conf
NETFP Master configuration file	ti/apps/netfp_master/netfp.conf
Library & Executable Suffix	_k2k

4.3 K2L

Kernel DTS Files	**ti/runtime/resmgr/dts/k2l**
RMv2 DTS Files	ti/runtime/resmgr/dts/k2l
DSP Memory Map	ti/runtime/platforms/k2l
ARM Compilation Flags	-D_LITTLE_ENDIAN -D__ARMv7 -DDEVICE_K2 –DDEVICE_K2L -D_GNU_SOURCE -D_VIRTUAL_ADDR_SUPPORT
DSP Compilation Flags	--define=DEVICE_K2 --define=DEVICE_K2L
PA Library on DSP	var Pa = xdc.useModule('ti.drv.pa.Settings'); Pa.deviceType = “k2l”
PA Library on ARM	-lpa2
SOC Sample configuration file	ti/apps/soc_init/soc_k2l.conf
NETFP Master configuration file	ti/apps/netfp_master/netfp_k2l.conf
Library & Executable Suffix	_k2l

NOTE: The PA library on K2L is different. Including the wrong library will result in run time failures.