SYSLIB

Release Notes

Applies to Product Release: 4.00.02.00
Publication Date: December 9, 2015

Document License

Content

Release Notes

SYSLIB 4.00.02.00

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`

The source repository for the patch can be found at:
`http://git.ti.com/cgit/cgit.cgi/keystone-rtos/pa-lld.git/`

```
GIT tag “EA.PA_LLD.03.00.01.06”,
commit id=d26b38c3a43e4d5c979b3f547291e730832768fc.
```

2.4.5 QMSS Library Patch

In order to support the insertion of descriptors into the appropriate location in the linking RAM (Internal or External) the QMSS LLD has to be patched.

- Copy `libqmss_k2h.so.1.0.0` and `libqmss_k2l.so.1.0.0` under `mcsdk_patches` directory to `ARAGODIR/lib directory`
- Copy `ti.drv.qmss.ae66,` and `ti.drv.qmss.ae66e` under `mcsdk_patches/k2h` directory to `PDK_INSTALL_PATH/ti/drv/qmss/lib/k2h/c66`
- Copy `ti.drv.qmss.ae66,` and `ti.drv.qmss.ae66e` under `mcsdk_patches/k2l` directory to `PDK_INSTALL_PATH/ti/drv/qmss/lib/k2l/c66`

2.4.6 Netlink symbolic link

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

- `ln –s libnl3/netlink/ netlink`

2.4.7 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.

2.4.7.1 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 now 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.

2.4.7.2 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

3.1.1 Information Only Fast Path

Information only Fast Path is defined as inbound fast path that are only used for provide artifacts like source address and port to the socket connection. It does not use NetCP resources, hence no LUT entries programming will be done for information only Fast Path. Such fast paths aren't supposed to receive any ingress packets.

To create information only fast path, please set the "fastpathMode" to Netfp_FastpathMode_INFOONLY in Netfp_InboundFPCfg.

3.2 API Changes

The section documents any API changes introduced in the SYSLIB Release

3.2.1 SA PDSP Halt on Error

The SA LLD exposes a new configuration which allows the PDSP to be halted on error. This feature is disabled by default in NETFP. In order to use this feature; application developers can turn on the flag `SA_DEBUG` in the file `runtime/netfp/src/netfp_sa.c`.

3.2.2 Allocation/Cleanup of custom resources

The allocation and cleanup of the custom resources has been extended to allow N resources to be allocated or freed up in a single API call.

```
int32_t Resmgr_allocCustomResource
(
Resmgr_SysCfgHandle sysRMHandle,
char*               customName,
uint32_t            numResources,
uint32_t*           value,
int32_t*            errCode
);

int32_t Resmgr_freeCustomResource
(
Resmgr_SysCfgHandle sysRMHandle,
char*               customName,
uint32_t            numResources,
uint32_t            value
int32_t*            errCode
);
```

The additional argument is highlighted. The functions will only allocate N contiguous resources. The base value is returned in the output argument. If the resources in the DTS files are not contiguous the allocations will fail.

3.2.3 Additional FAPI Message Compression APIs

The FAPI tracing library introduces new FAPI APIs that can be used by the application to truncate trailing zeroes for additional FAPI messages, as shown in the table below.

FAPI Message	Compression function name
TI_FAPI_UL_HARQ_INDICATION	FapiTracing_compressHarqInd
TI_FAPI_UL_CRC_INDICATION	FapiTracing_compressCrcInd
TI_FAPI_UL_RACH_INDICATION	FapiTracing_compressRachInd
TI_FAPI_UL_SRS_INDICATION	FapiTracing_compressSrsInd
TI_FAPI_UL_RX_SR_INDICATION	FapiTracing_compressRxSrInd
TI_FAPI_UL_RX_CQI_INDICATION	FapiTracing_compressRxCqiInd

The value returned from calling these functions should be used to update the "*msgLen*" field of the FAPI message headers and to update the packet length of the FAPI message descriptor prior to calling *FapiTracing_trace()*.

3.2.4 Inbound Fast Path Configuration Change

In inbound fast path configuration structure, “multicastMode” is replaced with “fastpathMode”. It includes the modes supported in multicastMode. And a new mode – “information only fast path” has also been added.

```
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_FastpathMode      fastpathMode;
…
}Netfp_InboundFPCfg;
```

For existing inbound fast path configurations please set fastpathMode to 0 i.e. Netfp_FastpathMode_NORMAL. For shared Multicast mode, please set fastpathMode to Netfp_FastpathMode_MULTICASTSHARED.

3.3 SYSLIB 4.0.2 Bug/Feature update list from JIRA:

Issue Type	Key	Summary
Bug	SCLTE-2543	Security accelerator lockup
Bug	SCLTE-2552	When IPSEC channel is deleted, SA channel is deleted before PA LUT entry is removed
Bug	SCLTE-2524	IPv6: Neighbor discovery done by netfp_proxy does not work correctly
Bug	SCLTE-2539	ResMgr DSP shared memory client assumes that only first mailbox is used
Story	SCLTE-2540	FAPI tracer embedded mode support
Story	SCLTE-2541	FAPI tracer packet compression
Story	SCLTE-2469	IPv6 based FAPI tracing
Story	SCLTE-2518	NetFP support of "information only" inbound fast paths
Bug	SCLTE-2542	: Kill -10 debug crashes in server when ptrServerSecurityChannelList is changed while printing
Task	SCLTE-2549	Configure K2L DTS/DTB for bring up
Bug	SCLTE-2528	Netfp_Option_FRAME_PROTOCOL_CRC_OFFLOAD is causing memory access violation on DSP

3.4 Known Issues:

Key	Summary
SCLTE-2443	errCode values changed in josh jobs even without an error
SCLTE-2019	Fixed 1GHz clock used in DAT_TIME_ELAPSED
SCLTE-1612	while(1) loop in msgcom code needs to be removed.
SCLTE-2506	Software Frame Protocol CRC computations are not supported on ARM
SCLTE-2240	Add DAT support for K2L in syslib4
SCLTE-1377	Outer IP Fragmentation option
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.