SYSLIB

Release Notes

Applies to Product Release: 4.00.05.00
Publication Date: December 21, 2016

Document License

Content

Release Notes

SYSLIB 4.00.05.00

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
- K2L

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 Installing the Custom UIA files

The cUIA package was modified to support 8 instances of logger streamer. It is now present in `SYSLIB_INSTALL_PATH/mcsdk_patches` directory. Untar the `cuia_1_01_00_06Custom.tar` file to `CUIA_INSTALL_PATH`.

2.4.5 Linux kernel update

If new IPSEC authentication or encryption features are to be used, a kernel update is required to enable appropriate Security Accelerator Linux support. Integrator would need to update kernel. Patch is not included with Syslib release.

2.4.6 ARM Cache Coherency (QMSS Bouncing Queue)

ARM Cache Coherency issue workaround is requiring PDSP accumulator firmware upgrades;

- QMSS PDSP Firmware
  - Copy QMSS firmware patch from `mcsdk_patches/qmss/firmware/` to `LINUX_KERNEL_INSTALL_PATH/firmware/keystone/`
    - acc48_le.bib → qmss_pdsp_acc48_k2_le_1_0_0_9.fw
- PA PDSP Firmware
  - Copy PA firmware patches from `mcsdk_patches/pa.zip (see chapter 2.4.7 for more information)` to `LINUX_KERNEL_INSTALL_PATH/firmware/keystone/`
    - Hawking
      - `fw/v0/classify1_0.bib` → `pa_pdsp0_classify1.fw`
      - `fw/v0/classify1_1.bib` → `pa_pdsp1_classify1.fw`
      - `fw/v0/classify1_2.bib` → `pa_pdsp2_classify1.fw`
      - `fw/v0/classify2.bib` → `pa_pdsp3_classify2.fw`
      - `fw/v0/pam.bib` → `pa_pdsp45_pam.fw`
    - Lamarr
      - Bouncing queue feature is valid only for K2H at the moment!

See chapter 3.1.1 for more information.

2.4.7 PDK library patches

Please use the table below and apply the PDK patches as described below. All the patches described below are present in the `SYSLIB_INSTALL_PATH/mcsdk_patches/` directory

File Name	Issue	How to patch
qmss/libqmss_k2h.so.1.0.0 **qmss/libqmss_k2h.a** **qmss/libqmss_k2l.so.1.0.0** **qmss/libqmss_k2h.a**	Support the insertion of descriptors into the appropriate location in the linking RAM (Internal or External) Adds the	TFTP the shared library object onto the EVM into the `/usr/lib` directory. And copy files also to `ARAGO_INSTALL_PATH/usr/lib/`

	QMSS barrier queue support. This is for **ARM** build.	
qmss/firmware/acc48_le_bin.h	Accumulator firmware. Support the QMSS barrier queue. **ARM** and **DSP** builds	Copy file to `PDK_INSTALL_PATH/ti/drv/qmss/firmware/` and `ARAGO_INSTALL_PATH/usr/include/ti/drv/qmss/firmware/`
qmss/qmss_acc.h	Adds the QMSS barrier queue support. **ARM** and **DSP** builds	Copy file to `PDK_INSTALL_PATH/ti/drv/qmss/` and `ARAGO_INSTALL_PATH/usr/include/ti/drv/qmss/`
qmss/k2h/ti.drv.qmss.ae66 **qmss/k2h/ti.drv.qmss.ae66e** **qmss/k2l/ti.drv.qmss.ae66** **qmss/k2l/ti.drv.qmss.ae66e** **qmss/k2k/ti.drv.qmss.ae66** **qmss/k2k/ti.drv.qmss.ae66e**	Support the insertion of descriptors into the appropriate location in the linking RAM (Internal or External) This is for **DSP** build.	Copy the files and overwrite the default library files which are present in the: `PDK_INSTALL_PATH/ti/drv/qmss/lib/k2h/c66` `PDK_INSTALL_PATH/ti/drv/qmss/lib/k2l/c66` `PDK_INSTALL_PATH/ti/drv/qmss/lib/k2k/c66` This is also provided in the QMSS LLD GIT Repository [Click here]. `Commit Id:` `3a14ab3db213c69b13a19a8af08531e58e16ef32` *[NOTE]: Pick up just this one commit*.

```
else {
for (i = 0; i < QMSS_MAX_MEM_REGIONS; i++)
{
  if
(gObjPtr->memRegInfo[qGroup][i].descNum
== 0)
   {
```

		index = i; break; } } } if (lObjPtr->qmRmServiceHandle) { startIndex = memRegCfg->startIndex; if (startIndex >= 0) { /* Let RM override user specified (memset(0)) region */ startIndex = QMSS_PARAM_NOT_SPECIFIED; } } else { if (index == 0) startIndex = 0; else startIndex = gObjPtr->memRegInfo[qGroup][index - 1].descNum + gObjPtr->memRegInfo[qGroup][index - 1].startIndex; }
qmss/device/k2[hkl]/ src/qmss_device.c	Adds the QMSS barrier queue support. **ARM** and **DSP** builds	Copy file to `PDK_INSTALL_PATH/ti/drv/qmss/device/ k2[hkl]/src/` and `ARAGO_INSTALL_PATH/ usr/include/ti/drv/qmss/ device/ k2[hkl]/src/`
libpa.so.1.0.0 **libpa2.so.1.0.0**	PA Bouncing queue support. This is for the **ARM** Builds.	TFTP the shared library object onto the EVM into the `/usr/lib` directory.
ti.drv.pa.ae66 **ti.drv.pa.ae66e** **ti.drv.pa2.ae66** **ti.drv.pa2.ae66e**	PA Bouncing queue support. This is for the **DSP** Builds.	Copy the files and overwrite the default library files which are present in the:- `PDK_INSTALL_PATH/ti/drv/pa/lib/c66` This is also provided in the PA LLD GIT Repository [Click here]. `GIT Tag:` `EA.PA_LLD.03.00.01.06` `Commit Id:` `d26b38c3a43e4d5c979b3f547291e730832768fc`

sa/fw/v0/*.c **sa/fw/v1/*.c**	SA firmware files which enable usage of GMAC algorithm	It is required to rebuild SA LLD to enable firmware version of 03.00.00.16. For rebuild, please use SA LLD GIT Repository [Click here]. `GIT Tag:` `DEV.SA_LLD.03.00.00.16` `Commit Id:` `8104974d908496bc9f7aca1bfa0f965de902bbcb` For convenience, precompiled libs are made available: On ARM; the firmware files are located in the DEVKIT and so these files need to be updated in the following directory:- `ARAGO_INSTALL_PATH/usr/include/ti/drv/sa` The NETFP Master is responsible for downloading the SA Firmware. The NETFP Master displays the SA Firmware version. This can be used to verify if the patch is applied or not. The patched firmware version should be as follows:- `Debug: SA PDSP0 Version: 3000010` `Debug: SA PDSP1 Version: 3000010`
sa/*.h	SA LLD header files	SA firmware update requires also header files to be updated to the following directory:- `ARAGO_INSTALL_PATH/usr/include/ti/drv/sa` and `PDK_INSTALL_PATH/ti/drv/sa`
sa/libsa.so.1.0.0	SA LLD ARM library files	TFTP the shared library object onto the EVM into the `/usr/lib` directory. Make sure that the NETFP Master reports correct LLD version:- `Debug: SA Version : 3000010`
sa/ti.drv.sa.ae66 **sa/ti.drv.sa.ae66e**	SA LLD DSP library files	Copy to `PDK_INSTALL_PATH/ti/drv/sa/lib/c66` Make sure that the DSP application reports correct LLD version:- `Debug: SA Version : 3000010`
pa.zip	PA Bouncing queue support.	`Replaces the existing PA driver folder` *`PDK_INSTALL_PATH/ti/drv/pa`* `Copy PA header and firmware files also into ARAGO_INSTALL_PATH/usr/include/ti/drv/pa/`

2.4.8 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.

2.4.8.1 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.
- QMSS barrier queue support

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.

2.4.8.2 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.
- QMSS barrier queue support

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 QMSS Barrier Queue

Fix to K2 ARM cache coherency issue.

This adds support to configure QMSS barrier queues (MSMC or DDR). Syslib ResMgr DTS files are used to define which general purpose queue is used for each barrier. DTS is also used to defining a PDSP ID which will be running the modified QMSS barrier accumulator firmware.

global-resource-list.dts:

```
    qmssBarrier {
        msmc_barrier_Q {
            resource-range = <896 1>;
        };
        msmc_barrier_PDSPID {
            resource-range = <0 1>;
        };
        //ddr_barrier_Q {
        //    resource-range = <897 1>;
        //};
        ddr_barrier_PDSPID {
            resource-range = <0 1>;
        };
    }; /* qmssBarrier */
```

policy_arm_dsp_syslib.dts:

```
  qmssBarrier {
        msmc_barrier_Q {
            assignments = <896 1>, "iu = (Rm_System)";
        };
        msmc_barrier_PDSPID {
            assignments = <0 1>, "iu = (Rm_System)";
        };
        ddr_barrier_Q {
            assignments = <897 1>, "iu = (Rm_System)";
        };
        ddr_barrier_PDSPID {
            assignments = <0 1>, "iu = (Rm_System)";
        };
    }; /* qmssBarrier */
```

[NOTE] Remember to remove barrier queues from available general purpose queues list!

Accumulator PDSP firmware upgrades (PA and QMSS) are required to support this feature.

[NOTE] Linux kernel needs to be recompiled and upgraded.

3.1.2 TCP MSS Clamping

The handling of IPSEC ESP packet header and trailer allocation is changed to accommodate the true size better, instead of hard coded margins. Now only the padding is calculated to be the maximum needed and is per the algorithm needs. Other variable sizes are true sizes.

3.2 API Changes

None

3.3 SYSLIB 4.0.5.0 Bug/Feature update list from JIRA:

Issue Type	Key	Summary
New feature	SCLTE-2875	Workaround for K2 ARM cache coherency issue
New feature	SCLTE-2834	Unnecessary IPv6 fragmentation header added for UE packets of certain size
Bug	SCLTE-2864	Msgcom_SDS missing from SYSLIB4.0.3.0 release
Bug	SCLTE-2867	Incoming UL LTE traffic with incorrect UDP checksum is passed to UE
Bug	SCLTE-2870	IPv6 Path MTU Discovery for UP does not work
Bug	SCLTE-2873	Deleted inbound parent FP is activated during rekey
Bug	SCLTE-2874	PMTUD for UP can set MTU below IPv6 minimum
Bug	SCLTE-2856	Leaked security context ID in Netfp_secContextAlloc on error branch

3.4 Known Issues:

Key	Summary
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-1377	Outer IP Fragmentation option

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_04_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.