Application Report
SPRAC37–April 2016

SERDES Link Commissioning on KeyStone I and II Devices

ABSTRACT

The serializer-deserializer (SerDes) performs serial-to-parallel conversions on data received from a peripheral device and parallel-to-serial conversion on data received from the CPU. This application report explains the SerDes transmit and receive parameters tuning, tools and some debug techniques for TI Keystone I and Keystone II devices.

Contents

1 Overview of SerDes Link Commissioning Using MCSDK/ProcSDK 1
2 SerDes Diagnostic Tools 5
3 10GbE SerDes Usage Tips in Linux 8
4 Hyperlink SerDes Usage Tips in Linux 12
5 References 15

List of Figures

1 BER Plot 6
2 Eye Diagram 8
3 10GbE PLL Schematic Diagram 10

1 Overview of SerDes Link Commissioning Using MCSDK/ProcSDK

The SerDes configuration is provided in MCSDK and Processor SDK packages. The last MCSDK for the Keystone I device is MCSDK 2.1.2.6 (http://software-dl.ti.com/sdoemb/sdoemb_public_sw/bios_mcsdk/latest/index_FDS.html) and the last MCSDK for the Keystone II device is MCSDK 3.1.4.7 (http://software-dl.ti.com/sdoemb/sdoemb_public_sw/mcsdk/latest/index_FDS.html). Those two MCSDK packages are discontinued and migrated into the Processor SDK. There are separate download links for each platform:

- K2HK: http://software-dl.ti.com/processor-sw/esd/PROCESSOR-SDK-RTOS-K2HK/latest/index_FDS.html
- K2E: http://software-dl.ti.com/processor-sw/esd/PROCESSOR-SDK-RTOS-K2E/latest/index_FDS.html
- K2L: http://software-dl.ti.com/processor-sw/esd/PROCESSOR-SDK-RTOS-K2L/latest/index_FDS.html
- C665x: http://software-dl.ti.com/processor-sw/esd/PROCESSOR-SDK-RTOS-C665x/latest/index_FDS.html
- C667x: http://software-dl.ti.com/processor-sw/esd/PROCESSOR-SDK-RTOS-C667x/latest/index_FDS.html

Customers are encouraged to adapt to the latest processor SDK to get all the new features and bug fixes, while this application report applies to both MCSDK and Processor SDK packages.

An application report or user's guide is available for the Keystone I or Keystone II device SerDes. There are also user's guides for individual peripherals:

- *SerDes Implementation Guide for KeyStone I Devices* (SPRABC1)

- *KeyStone II Architecture Serializer/Deserializer (SerDes) User's Guide* (SPRUHO3)
- *KeyStone Architecture HyperLink User's Guide* (SPRUGW8)
- *KeyStone Architecture Peripheral Component Interconnect Express (PCIe) User's Guide* (SPRUGS6)
- *KeyStone Architecture Serial Rapid IO (SRIO) User's Guide* (SPRUGW1)
- *KeyStone Architecture Gigabit Ethernet (GbE) Switch Subsystem User's Guide* (SPRUGV9)

1.1 Keystone I Device SerDes Configuration

In the Keystone I device, the SerDes configuration is relatively simple. Typically, it involves a PLL configuration and a SerDes Tx/Rx pair configuration, and those registers are System-on-Chip (SoC) device-level registers. For example, the C66x device memory map showed the registers listed in Table 1 for SerDes PLL, Tx/Rx CFG and status.

Table 1. Keystone I SerDes Register Map

Address	Size	Field
0x02620340	4B	SGMII_SERDES_CFGPLL
0x02620344	4B	SGMII_SERDES_CFGRX0
0x02620348	4B	SGMII_SERDES_CFGTX0
0x0262034C	4B	SGMII_SERDES_CFGRX1
0x02620350	4B	SGMII_SERDES_CFGTX1
0x02620358	4B	PCIE_SERDES_CFGPLL
0x02620360	4B	SRIO_SERDES_CFGPLL
0x02620364	4B	SRIO_SERDES_CFGRX0
0x02620368	4B	SRIO_SERDES_CFGTX0
0x0262036C	4B	SRIO_SERDES_CFGRX1
0x02620370	4B	SRIO_SERDES_CFGTX1
0x02620374	4B	SRIO_SERDES_CFGRX2
0x02620378	4B	SRIO_SERDES_CFGTX2
0x0262037C	4B	SRIO_SERDES_CFGRX3
0x02620380	4B	SRIO_SERDES_CFGTX3
0x026203B4	4B	HYPERLINK_SERDES_CFGPLL
0x026203B8	4B	HYPERLINK_SERDES_CFGRX0
0x026203BC	4B	HYPERLINK_SERDES_CFGTX0
0x026203C0	4B	HYPERLINK_SERDES_CFGRX1
0x026203C4	4B	HYPERLINK_SERDES_CFGTX1
0x026203C8	4B	HYPERLINK_SERDES_CFGRX2
0x026203CC	4B	HYPERLINK_SERDES_CFGTX2
0x026203D0	4B	HYPERLINK_SERDES_CFGRX3
0x026203D4	4B	HYPERLINK_SERDES_CFGTX3
0x02620154	4B	SRIO_SERDES_STS
0x02620158	4B	SMGII_SERDES_STS
0x0262015C	4B	PCIE_SERDES_STS
0x02620160	4B	HYPERLINK_SERDES_STS
0x21800390	4B	SERDES_CFG0
0x21800394	4B	SERDES_CFG1

Note for PCIE, the SerDes Tx/Rx configuration registers are inside PCIE peripheral configuration space.

The configuration of those SerDes is provided in the driver code, for example:

- Hyperlink: in packages\ti\drv\hyplnk\example\common\hyplnkLLDIFace.c through CSL_BootCfgSetVUSRConfigPLL() and hyplnkExampleSerdesCfg() routines
- PCIE: in CSL_BootCfgSetPCIEConfigPLL() with default SerDes configuration

1.2 Keystone I Device SerDes Tx Tuning

The SRIO and Hyperlink transmitters include a 3-tap FIR filter (TWPRE and TWPST1 fields) that can be adjusted independently for each lane, whereas, SGMII includes a 1-tap, de-emphasis (DEMPHASIS field) filter that can be adjusted independently for each lane. For PCIE, it also uses a 1-tap de-emphasis filter and it is configured by CFG_TX_SWING field in PCIE PL_GEN2 register.

These settings must be optimized together for each lane of the PCB system to maximize the signal eye width and height, and ensure that the signal is not over or under-equalized at the intended receiver. Attenuation of a signal through a PCB transmission line tends to increase as the frequency of the signal increases such that most channels act like a low-pass filter. Given these losses, the transmitter swing must be increased as the transmission line loses increase to provide uniform gain of the signal. However, increased swing can also result in an over-equalized signal at the receiver, increased cross-talk between adjacent signals, and increased power consumption in the system.

Simulation of the target PCB platform with appropriate 3D EM modeling and the KeyStone I IBIS-AMI SerDes models is the best method of creating initial starting values for the actual PCB.

Another method for creating valid transmitter settings is to implement an exhaustive search across the entire transmitter swing and FIR de-emphasis settings space for each link partner while leaving the link receivers in fully-adaptive equalizer mode (or an equivalent receiver equalizer mode for non-Keystone I link partners). A test program can be setup that sweeps through each transmitter setting while live traffic is exchanged between link partners. The test programs create and store error-rate maps based on the ratio of the number of bits received with no errors compared to those with errors over a given window of time for a given transmitter setting iteration. This error-rate map can then be analyzed offline in any convenient data analysis tool to determine the exact parameters that will result in the most operating margin of the SerDes link.

1.3 Keystone I Device SerDes Rx Tuning

The receiver incorporates an adaptive equalizer (EQ field), which can compensate for channel insertion loss by attenuating the low frequency components and amplifying the high frequency components of the signal, thereby, reducing inter-symbol interference. When enabled, the receiver equalization logic analyzes data patterns and transition times to determine whether the low frequency gain should be increased or decreased:

- No equalization (000) setting provides a flat response at the maximum gain. This setting may be appropriate if jitter at the receiver occurs predominantly as a result of crosstalk rather than frequency dependent loss.
- Fully adaptive equalization setting (001) could be appropriate for most applications.

Another area to configure is the Clock/Date Recovery Algorithms (CDR field), the clock recovery algorithms operate to adjust the clocks used to sample rxpi and rxni so that the data samples are taken midway between data transitions. Both first and second order algorithms are provided.

1.4 Keystone II Device SerDes Configuration

In the Keystone II device, the SerDes configuration is done by programing the registers in SerDes configuration space for individual peripherals as shown in Table 2.

Table 2. Keystone II SerDes Register Map

Start Address	End Address	Size	Field
0x0231A000	0x0231BFFF	8K	HyperLink0 SerDes Config
0x0231C000	0x0231DFFF	8K	HyperLink1 SerDes Config
0x0231E000	0x0231FFFF	8K	10GbE SerDes Config (66AK2H14 only)
0x02320000	0x02323FFF	16K	PCIE SerDes Config
0x0232A000	0x0232BFFF	8K	PA SerDes Config
0x0232C000	0x0232DFFF	8K	SRIO SerDes Config

The MCSDK/Processor SDK provides the example in each individual driver. The driver typically calls the CSL code, which in turn calls SerDes configuration file for the SerDes initialization. The SerDes configuration files are provided under packages\ti\csl\src\ip\serdes_sb\V0. The CSL code has the SerDes configuration sequence and patches. The major one is csl_serdes(2).h under the same folder.

As the Keystone II device has ARM® cores, there are also Linux drivers for those peripherals. The Hyperlink driver is provided in Linux® user space by MPM transport (http://git.ti.com/gitweb/?p=keystone-linux/mpm-transport.git;a=summary), the same CSL and SerDes configuration file as in SYSBIOS are cross-compiled by GCC Linaro tool chain for ARM Linux. All other drivers are provided inside Linux kernel and are re-written based on the CSL code and SerDes configuration file: http://processors.wiki.ti.com/index.php/MCSDK_UG_Chapter_Exploring#Linux.

The Keystone II SerDes interfaces consist of two main variants of the SerDes PHYs: PHY-A and PHY-B. Within PHY-A, there are two variants: PHY-A 2 Lane and PHY-A 4 Lane.

- PHY-A 2 Lane: JESD204A/B, AIL, AIF2 (lane 5-6), PCIE
- PHY-A 4 Lane: AIF2 (lanes 1-4), SGMII, SRIO, HyperLink
- PHY-B 2 Lane: 10GbE

1.5 Keystone II Device SerDes Tx Tuning

The TX driver output swing level is controlled by TXDRV_ATT_IN_OVR_O and PMA_LN_TX_VREG_LEV_O. The former controls the number of voltage mode slices dedicated to driving output. Increasing this value increases the number of slices, thus increasing the output voltage swing. The latter controls the regulator voltage output level that is fed to each slice.

Slew rate settings are set by loading the default SerDes configuration that TI provides as part of the MCSDK package.

The transmitters include a 4-tap FIR filter with coefficients CM, C0, C1, and C2. Those coefficients follow the rule that $|c_m| + |c_0| + |c_1| + |c_2| = 1$. As the values of any of Cm, C1, or C2 are increased, C0 is automatically decreased to maintain this equality.

Overall, there are five Tx parameters can be tuned: TX_ATT, VREG, Cm, C1 and C2. They can be adjusted independently for each lane.

1.6 Keystone II Device SerDes Rx Tuning

On the receive path of SerDes PHY, the first stage is an Attenuator (ATT). This stage is required to reduce signal amplitude received from channel before entering the BOOST stage in order to not saturate the BOOST input. The ATT stage is adapted automatically by an in-built PHY algorithm but may be overridden to a fixed value through register settings.

The next stage is BOOST. This stage is used to provide high-frequency gain to the received channel signal in order to mitigate the high frequency loss incurred in the channel. The BOOST stage is automatically adapted by an in-built PHY algorithm but may be overridden to a fixed value through register settings.

As discussed, the receiver can set the RX_ATT (to differentiate from TX_ATT) and BOOST to adaptation mode or forced mode. This is controlled by the CSL code:

```
typedef enum
{
    /** Force Attentuation Boost Values Flag Disable */
    CSL_SERDES_FORCE_ATT_BOOST_DISABLED  =  0,

    /** Force Attentuation Boost Values Flag Enable */
    CSL_SERDES_FORCE_ATT_BOOST_ENABLED   =  1
} CSL_SERDES_FORCE_ATT_BOOST;

    /* When RX auto adaptation is on, these are the starting values used for att, boost
adaptation */
      serdes_lane_enable_params.rx_coeff.att_start[lane] = 7;
      serdes_lane_enable_params.rx_coeff.boost_start[lane] = 5;

      /* For higher speeds PHY-A, force attenuation and boost values  */
```

```
serdes_lane_enable_params.rx_coeff.force_att_val[i] = 1;
serdes_lane_enable_params.rx_coeff.force_boost_val[i] = 1;
```

As Keystone II errata, use the *KeyStoneII.BTS_errata_usagenote.28*: *SerDes Fails to Adapt RX BOOST Equalization* in the *66AK2H06/12/14 Multicore DSP+ARM KeyStone II SOC Silicon Revisions 1.0, 1.1, 2.0 Silicon Errata* (SPRZ402). For higher speeds PHY-A, forcing RX_ATT and BOOST is recommended. For lower speeds PHY-A, those can be set to adaptation or forced. For PHY-B, those are set to adaptation. To obtain the optimal forced values, a test can be performed by sending a PRBS data pattern to the Rx and the ATT/BOOST are allowed to auto-adapt, then found the average. This method is discussed in Section 2.2.

2 SerDes Diagnostic Tools

There are three tools available under packages\ti\diag\serdes_diag: BER sweep, Eye diagram and PRBS test. Those tools demonstrate how to configure and use the SERDES Diagnostic (Diag) APIs on Keystone II devices. They are essentially DSP programs built by Code Composer Studio (CCS), loaded, run and controlled by DSS scripts on DSP cores on both ends.

2.1 BER Sweep Tool

BER sweep tool finds out the optimal Tx coefficients (CM, C1, C2) by performing Bit Error Rate (BER) sweeps on the SerDes, where both ends send a transmit pattern (for example PRBS 31) and both ends detects the sequence and performs BER calculations across the desired SerDes using the CPU. The BER plot can be viewed using the BER DiagramViewer.jar program. For detailed usage, see the user's guide under \ti\diag\serdes_diag\doc.

2.1.1 Raw Data Collection

There are several steps involved for collecting the BER data:

1. Create ccxml target connection file for CCS.
2. Import the CCS project, configure the test (interface, reference clock rate, lane rate, lane mask, sweep CM, C1, C2 range, duration, …) and build.
3. Update the DSS test script for ccxml, .out file, GEL file …
4. Run.

The tool only sweeps across CM, C1 and C2. There are parameters tx_att_start, tx_att_end, tx_vreg_start and tx_vreg_end defined in the code, but the tool does not sweep across the TX_ATT and TX_VREG. So the start and end values have to be same. The default settings are TX_ATT = 12 and TX_VREG = 4. You can try to use a few different TX_ATT and TX_VREG pairs to evaluate the impact, in this case, a separate DSP program needs to be built each time.

Note that the BER results are stored in arrays in the test program during the sweep. Then, those results are printed into a log file when the whole sweep finished. It is impossible to look at the partial results during the progress. Therefore, it is good to sweep a very small range for a quick trial, to make sure the collected results are reasonable.

Also note that when sweep ranges (CM x C1 x C2) are bigger (roughly > 252), the program gets stuck during the sweep loop and nothing can be printed out. This issue is observed several times with no further debug, probably due to the arrays growing bigger and some memory gets corrupted.

2.1.2 Data Analysis

When the sweep finishes, a .txt file is generated for each side. The BER Diagram Viewer tool (located at serdes_diag/tools/ber_diagram_tool) can be used to create the BER plot. This program was written in Java using the Eclipse IDE, a free download from https://www.eclipse.org/. This program was compiled with Java version 7 update 60 build 1.7.0_60-b19 using the runtime environment JavaSE-1.6 (jre6).You must have Java version 7 update 60 or above to run this program, which can be freely downloaded at https://java.com/en/download/manual.jsp. You are also required to have the Java runtime environment installed to run this program.

Figure 1 shows a typical BER plot, the white dot is the best pick.

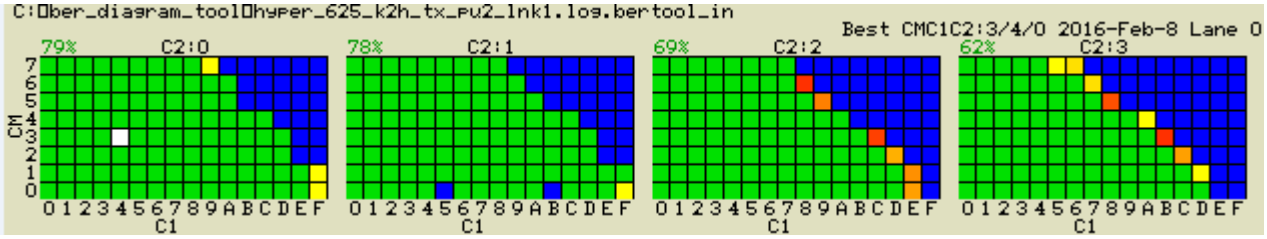

Figure 1. BER Plot

2.1.3 Long-Duration BER Link Testing

The time taken for a BER sweep is given by: C1 range x C2 range x CM range x (BER duration + overhead). It may take hours or days when doing a full range sweep. If TX_ATT and VREG are added into the sweep, the time spent will be even much longer.

It is good to do a BER sweep in two-steps:

- A full range sweep with short BER duration
- A small, concentrated area sweep with long BER duration

In the first step, a few seconds (say 4-5 seconds) can be chosen for BER test duration, the overhead (DSS script pulling, switching …) overweights the BER test duration if it is set too short. A rough idea of the error free region and error margin can be determined from the full range sweep.

The second step is a fine tuning with long duration BER test but only focusing around the best pick region. For example, the SRIO standard requires a BER of $1^{(-12)}$ or better. To validate this:

- Use the best pick from the full range sweep as the center point (for example, forming a 3x3x3 region)
- Calculate the minimum sweep duration time to achieve this BER. Assuming SRIO 5Gaud with 4 lanes, then the duration should be at least $1^{(12)}/4/5^{(9)}$ = 50 seconds.
- Configure the BER test with the new sweep range and duration to validate the link robustness.

2.1.4 Limitations and Workarounds With Keystone Device

There are customer board designs involved in interconnection between the Keystone II and Keystone I devices. SRIO, SGMII, 10GbE and PCIE are industry standards and typically there are switches available for inter-connection. However, Hyperlink is a TI standard and there have been some designs that Keystone II devices directly connected to Keystone I devices via separate Hyperlink ports. There is a need for Hyperlink SerDes tuning for this scenario.

For the Keystone I devices, the MCSDK/Processor SDK packages do not provide any SerDes tuning tools. The SerDes bit error counter is not chip-level MMR, thus it can not be accessed by a DSP program. TI does not support any tools to access the SerDes error counter in Keystone I device. In such a case, the Hyperlink block error counter can be used as an indicator to tune the SerDes parameters.

2.1.5 Limitations and Workarounds With Other Link Partners

When a TI Keystone I or II device is working with another link partner (for example, a switch, a FPGA, …), it creates a complex situation:

- May not have CCS/JTAG access to the link partner, the tools provided by TI use JTAG emulation and CCS to communicate with the SERDES configuration and control registers
- Limited or different ways of programing
- Limited or different ways to configure Tx/Rx parameters
- Limited or different ways to run PRBS tests and track error counters
- Limited or different ways to assess Rx Eye Quality

This requires tool customization which is beyond the support that TI can provide. If this type of automated data collection is needed then the customer must assess the capabilities of the remote link partner and to modify the existing tools to work with it. Alternately, this data can be collected manually by stepping through the sweep range and recording the BER error on the remote end after each test period. A similar manual tuning method can be used by Rx Eye quality measurements manually while stepping through the tunable parameters for each link.

If the link partner can perform a loopback, then the BER sweep test can still be run with automation but with two links in the path. Note, the BER sweep test is designed to run on two separate devices. If in the loopback case, the “devices” are actually just one device but different “cores”. The program will need to be updated to initialize the interface only once. In this setup, there are 2 separate links where each has tunable TX and RX parameters. It will be difficult to determine which link creates the bit errors. Tuning may have to be done multiple times, first on one link and then on the other, to obtain robust link quality in both directions.

On the other hand, the Keystone II device can perform the far end loopback if the user prefers to generating PRBS and measuring the BER at the link partner side. The loopback usage is described in the *Loopbacks* section in the *KeyStone II Architecture Serializer/Deserializer (SerDes) User's Guide* (SPRUHO3). However, this has the same disadvantages as discussed above.

2.2 PRBS Test

The PRBS test example finds out the optimal Rx coefficients (RX_ATT and BOOST) by sending a PRBS 31 pattern and training the receiver to get the best Rx coefficients. The output log displays these values that are used for Rx forced case.

To use the PRBS test correctly:

- Run the BER sweep tool to find out the best Tx side setting for C1, C2 and CM
- Use the best Tx setting, run the PRBS test to train the receiver
- Record the optimal RX_ATT and BOOST

2.3 Eye Mask Tool

Eye mask tool performs “on chip” eye measurement allowing the eye opening of the receive data to be seen after equalization. Both ends send the transmit pattern (for example, PRBS 31) and both ends detect the sequence and perform eye measurements across the desired SerDes using the CPU. The EYE plot can be viewed using the EYEDiagramViewer.jar program. For detailed usage, see the user's guide under \ti\diag\serdes_diag\doc.

2.3.1 Raw Data Collection

The same steps are involved for collecting the eye data for BER sweep test.

2.3.2 Data Analysis

When the test finishes, a .txt file is generated for each side. The Eye Diagram Viewer (EyeDiagramViewer.jar) is located at serdes_diag/tools/eye_diagram_tool. The purpose of this program is to create an eye diagram from an eye scan .txt file from Serdes Diag API Serdes Diag_EyeMonitor(). A typical eye diagram looks similar to the one shown in Figure 2.

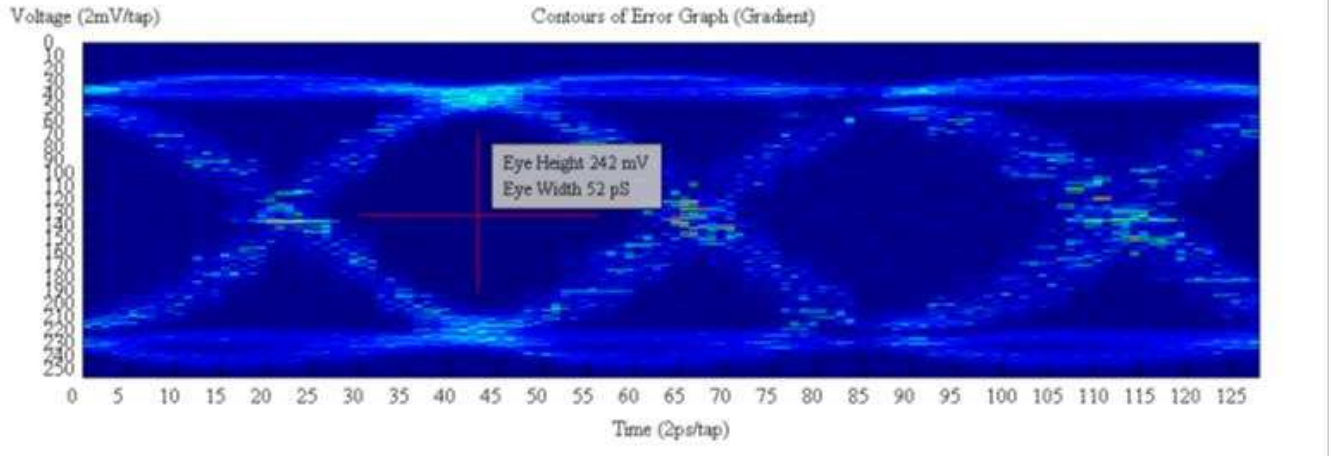

Figure 2. Eye Diagram

3 10GbE SerDes Usage Tips in Linux

This section discusses some debug tips on the Keystone II PHY-B. Keystone II devices support 10G Ethernet in XFI and KR modes via 2-lane PHY-B (one 10GbE port per lane). 10GBASE-KR, 10GBASE-R, 10GBASE-Xm are some of the encoding specs in the 10G PCS/PMA layer. XFI is an interface used in the transceiver when connected to a host or switch, XFI can use 10GBASE-R, 10GBASE-Xm (m = 2, 4, and so forth). The KR is commonly used in backplane. The KR protocol supports link auto negotiation and tuning, greatly reducing the SerDes Tx/Rx parameter tuning efforts between two link partners when using XFI mode.

3.1 Enable XFI Mode

XFI mode can be enabled in either u-boot script or Linux kernel run time via DTS binding. For more information, see the *10Gig Ethernet Driver* section in http://processors.wiki.ti.com/index.php/MCSDK_UG_Chapter_Exploring#10Gig_Ethernet_Driver.

3.2 Enable KR Mode

KR mode needs firmware support, it can be enabled in Linux kernel run time via DTS binding. For more information, see the *10Gig Ethernet Driver* section in http://processors.wiki.ti.com/index.php/MCSDK_UG_Chapter_Exploring#10Gig_Ethernet_Driver. Detailed example of DTSI: ./Documentation/devicetree/bindings/net/keystone-serdes.txt.

Example boot log using TI K2H EVM:

```
[ 3780.232377] Waiting for autonegotiated link up.
[ 3780.599868] Initialized KR firmware version: 323
[ 3780.604461] firmware restarted status:
[ 3780.608190]   pll_ctrl = 0xff000303
[ 3780.611503] LANE0:
[ 3780.613679]   pcsr_rx_sts = 0x40000000, lane_ctrl_sts = 0xe010e002
[ 3780.619836]   cm = 2, c1 = 2, c2 = 1
[ 3780.623393]   attn = 4, boost = 9, dlpf = 523, cdrcal = 102
[ 3780.628942] keystone-netcp 2f00000.netcp: cpswx serdes 0 config done lanes(mask) 0x0
[ 3780.637195] keystone-netcp 2f00000.netcp: Created interface "eth4"
[ 3780.643367] keystone-netcp 2f00000.netcp: dma_chan_name xgetx0
[ 3780.649173] serdes firmware already started
[ 3780.653346] keystone-netcp 2f00000.netcp: cpswx serdes 0 config done lanes(mask) 0x0
[ 3780.661584] keystone-netcp 2f00000.netcp: Created interface "eth5"
[ 3780.667744] keystone-netcp 2f00000.netcp: dma_chan_name xgetx1
[ 3780.673573] serdes firmware already started
[ 3780.677738] keystone-netcp 2f00000.netcp: cpswx serdes 0 config done lanes(mask) 0x0
```

3.3 Enable KR Mode With First 10GbE Port Only

In the KR set-up the second lane is always on. In order to disable the second interface, leave everything as-is and just set “num-interfaces” to 1 in the DTS file. This should just skip over trying to set up a second interface.

```
multi-interface;
        num-interfaces = <1>;
```

3.4 Set 10GbE SerDes Tx/Rx Parameters

This is done in the arch/arm/boot/dts/k2hk-net.dtsi file:

```
/*rx-force-enable;*/
lane0 {
            /*disable;*/
            /*loopback;*/
            rx-start    = <7 5>;
            rx-force    = <1 1>;
            tx-coeff    = <2 0 0 12 4>;
                  /* c1 c2 cm att vreg */
            control-rate    = <0>; /* full */
};
      lane1 {
            /*disable;*/
            /*loopback;*/
            rx-start    = <7 5>;
            rx-force    = <1 1>;
            tx-coeff    = <2 0 0 12 4>;
                  /* c1 c2 cm att vreg */
            control-rate    = <0>; /* full */
};
```

To force RX_ATT and BOOST, use “rx-force-enable” and update “rx-force = < 1 1>;" with the optimal values.

3.5 Debug 10GbE Failure

The 10GbE interface may fail with the following symptoms:

- Linux kernel XGE initialization stuck
- 10GbE link is up (seen in ifconfig), but ping failure with intermittent packet loss or even 100% packet loss; SSH failure.

The following sections discuss several things to check.

3.5.1 Are Both PLL and PLL2 locked?

The PLL lock is a fundamental requirement and should be obtained by:

- Power applied to SerDes
- SerDes input reference clock running
- SerDes removed from reset
- SerDes PLL configured

Check PLL_CTRL (0x0231_FFF4):

- Bit 28: should set, indicates PLL OK
- Bit 24: should set, indicates PLL2 OK

You should be able to easily verify SerDes power and clock with a scope. If PLL2 can not lock, this is more of HW issue. Some PHYs require one off-chip resistor, while others require two depending on the number of CMU macros used. 10GbE SerDes is a single macro with two lanes and two CMUs, and it has two REFRES resistor pins. So both resistors AN16 and AN19 should present.

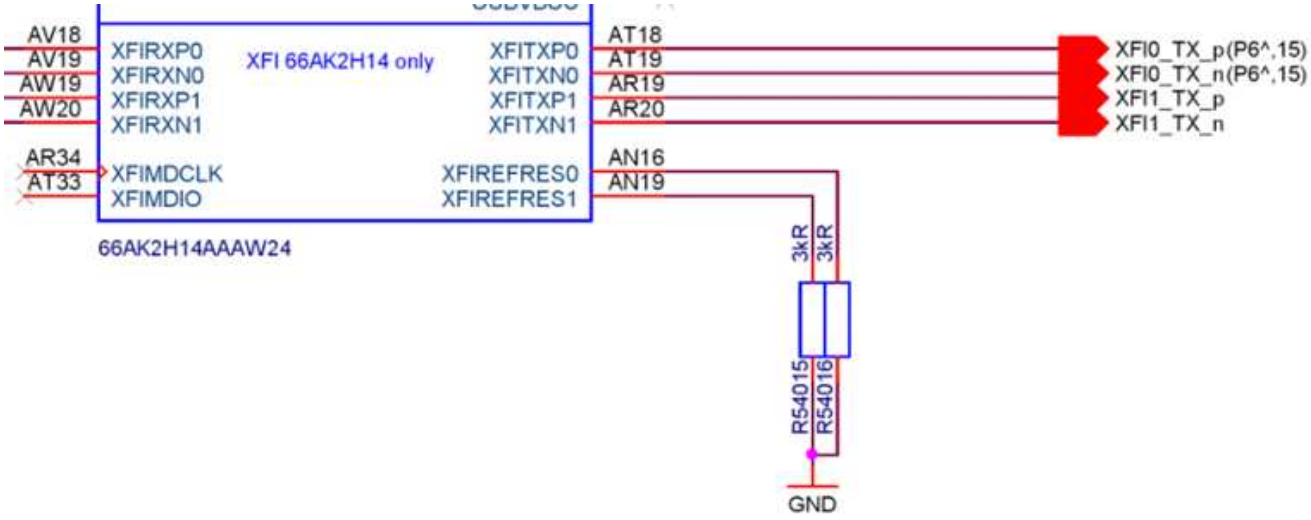

Figure 3. 10GbE PLL Schematic Diagram

3.5.2 Check PLL_CTRL and LANExCTL_STS for Lane Status

PLL_CTRL (0x0231_FFF4):

- Bit 0/1: signal detection should set (for lane 0/1, respectively)
- Bit 8/9: signal OK should set (for lane 0/1, respectively)

Lane0/1CTL_STS (0x0231_FFE0/0x0231_FFE4):

- Bit 0: Rx loss should be cleared
- Bit 1: Rx OK should set
- Bit 20: should set indicating 10G link is up

3.5.3 Check RX_ATT and BOOST

According to the following FAE Alert, SerDes RX adapts to a BOOST value of '0' and cannot move from a value of '0'.

When an incoming SerDes signal has too much high-frequency gain or has not experienced a significant amount of high-frequency loss on a channel, the BOOST block within the RX attempts to compensate by equalizing the signal during RX adaptation. Specifically, the high-frequency gain is compensated by reducing the adapted BOOST value. If the BOOST value adapts to the lower limit of the BOOST settings, it adapts to a value of '0'. There is a deficiency in the SerDes PHY where if the BOOST adapts to a value of '0', then any subsequent RX CDR resets, performed by toggling the RX signal detect, that are used in order to repeat RX adaptation will not be able to move the boost value away from a value of '0'. The value of '0' will stick even if this value of '0' is not optimal for the incoming RX signal. In other words, if the BOOST reaches a value of '0' during RX adaptation, it will remain at that value until the SerDes PHY is power cycled.

A workaround can be performed in order to manually bump the BOOST out of this '0' state and into a state in which the Rx can re-adapt to a new BOOST value. The workaround sequence should be performed immediately after Rx adaptation has occurred during initialization. The sequence is used in the CSL/MCSDK for all interfaces in diagnostic (diag) mode, and is used in 10GbE in functional mode. The sequence is captured in CSL/MCSDK v3.1.4.7 and later in the following function:

For PHY-A and PHY-B: *CSL_SerdesAttBoostPatch()*

The MCSDK 3.1.4.7/Processor SDK 2.0.1 Linux 10GbE driver was based on MCSDK 3.1.3.6 CSL code and did not implement this workaround. So, it is possible that BOOST adapt to 0 when the Rx adaptation is used in DTS. A patch on top of 3.1.4.7/2.0.1 is available to fix the issue.

In order check whether the RX_ATT and BOOST value to make sure BOOST has not adapted to 0 can be done via T-bus write then read operations. One approach is to write a small DSP code to run on a DSP core:

```
unsigned int att_read, boost_read, att_init, boost_init;
    unsigned int base_addr = 0x231e000;
    unsigned int lane_num;
    csl_serdes_phy_type phy_type = SERDES_10GE;

    for (lane_num = 0; lane_num < NUM_LANE; lane_num++) {
        att_read = (CSL_SerdesReadSelectedTbus(base_addr, lane_num + 1,
(phy_type==SERDES_10GE)?0x10:0x11)>>4)&0xF;
        boost_read = (CSL_SerdesReadSelectedTbus(base_addr, lane_num + 1,
(phy_type==SERDES_10GE)?0x10:0x11)>>8)&0xF;

        printf("Lane %d, att_read is 0x%x\n", lane_num, att_read);
        printf("Lane %d, boost_read is 0x%x\n", lane_num, boost_read);
}
```

Or, those values can be read out under Linux prompt using devmem2 (which is easier):

- devmem2 0x231e0fc w 0x02100000/0x03100000 (port 0/1)
- evmem2 0x231e0f8 (Bit 23-20 is ATT, Bit 27-24 is BOOST)

3.5.4 Check PCS-R Layer

The 10GBASE-R Physical Coding Sublayer (PCS-R) module provides the functionality of a physical coding sublayer (PCS) on data being transferred between a demuxed XGMII and 10 Gigabit SerDes supporting a 16- or 32-bit interface.

- PCSR_TX_CTL (0x02f0_0600/0x02f0_0680): should be 0 (Port 0/1, respectively)
- PCSR_RX_CTL (0x02f0_060C/0x02f0_068C):
  - Bit 30: should set to indicate Rx linked
  - Bit 23-16: this is Rx error block counter, expect to be 0

3.5.5 Use Linux Command to Trace Packet Loss

- ifconfig ethX: check Linux driver level Tx/Rx counters
- ethtool –S ethX: check 10GbE switch HW level Tx/Rx counters
- tcpdump –i ethX: sniff the interface by dumping out the Tx/Rx packets from driver

3.6 Recover a Failed 10GbE Link

If a link is up but showing packet loss, toggling signal detection (also known as reset CDR) may recover it from failure in XFI mode. For KR mode, a re-configuration of interface is required in addition to toggling signal detection. Check the *Enabling 10G-KR Firmware* section in http://processors.wiki.ti.com/index.php/MCSDK_UG_Chapter_Exploring#10Gig_Ethernet_Driver.

The signal detect is an analog block that compares the incoming signal envelope to a programmable threshold to determine whether valid signaling is present on the RX_P/N input pins. An optimal signal detect threshold is set by loading the SerDes configuration file provided by TI as part of the MCSDK. The signal detect can also be manually triggered or overridden as a means to reset the RX stage and force RX re-calibration (electrical idle exit calibration must be enabled).

This is done on lane_004 register, bit 2-1: SIGDET_OVR_O_1_0 Override signal detect output. Bit 1 is override enable, bit 0 is override value. This can be done under Linux prompt, (for example, set bit 2-1 to 2, then set the same bit back to 0):

- devmem2 0x0231e204 (Port 0)
- devmem2 0x0231e204 w 0xF0001084
- devmem2 0x0231e204 w 0xF0001080
- devmem2 0x0231e404 (Port 1)
- devmem2 0x0231e404 w 0xF0001084

- devmem2 0x0231e404 w 0xF0001080

3.7 Notes to Recover a Failed Link in XFI Mode and KR Mode

There are known issues documented in the *66AK2H06/12/14 Multicore DSP+ARM KeyStone II SOC Errata (Revs 1.0, 1.1, 2.0) Silicon Errata* (SPRZ402).

- KeyStoneII.BTS_errata_advisory.29 10GbE PCS Causes Data Corruption
- KeyStoneII.BTS_errata_usagenote.27 Device Hang if 10GbE PCS Registers are Accessed After Performing a 10G Lane Reset

"The workaround is currently implemented in the latest 10GbE Linux drivers (since MCSDK 3.0.4)" only applies to non-KR firmware, for example, the XFI mode.

- Advisory 29 applies to 10GbE in both XFI mode and KR mode
- When the toggle signal is detected on the XFI mode, it does not cause a 10G lane reset, so the PCS RX register for bit errors (note 27 does not apply here) is still accessible. In the 10GbE driver, there is code to check for the 0xFF condition and toggle signal detection, with 100 retries. It should not make it past this point with bit errors unless it retries that many times; it would also print a timeout error. That is, in XFI mode, you can read PCS RX register multiple times without issue.
- When toggle signal is detected with the 10G KR firmware, it triggers a 10G lane reset, so the PCS register is no longer accessible because of the details in usage note 27. In KR mode, PCS RX can only be read once. The same XFI code can not be used in the driver. Also, ifconfig can not be done inside the Linux kernel. This is why there is a script approach using ifconfig.
- Does Advisory 29 only apply at initialization stage or could it happen even after a link is established? For XFI mode, only at initialization. For KR mode, at all times whenever auto-negotiation occurs, which can happen again when the other side resets, causing the 10G lane reset.
- Continuous link monitoring at run-time on KR mode: Since no reliable way to figure out if the SerDes are in the problem state because of usage note 27 was found, just read the switch statistics using “ethtool -S eth0”. One of the rx_error stats should be incrementing if there is an issue.

4 Hyperlink SerDes Usage Tips in Linux

This section discusses some debug tips on Keystone II PHY-A. Hyperlink is used as an example as it has the highest data rate over other PHY-A peripherals, therefore, it is most challenging. The tips also apply to others when checking the SerDes and peripheral level registers.

Keystone I and II devices supports Hyperlink up to 10Gb/s. 12.5Gb/s is not supported or quantified due to silicon errata:

- KeyStoneII.BTS_errata_advisory.33 HyperLink Data Rate Limited to 40 Gbaud Issue
- Advisory 21 HyperLink Data Rate Limited to 40 Gbaud Issue

There are several customer systems running Hyperlink at 3.125Gb/s, 6.25Gb/s; some are trying to achieve the maximum 10.0Gb/s. The issue experienced and reported is an initialization stability issue across all lane rates and Keystone II/I devices. There is no data corruption issue during run time if channel is clean.

4.1 Enable Hyperlink

For the Keystone I device, the Hyperlink is enabled in DSP code. For the Keystone II device, Hyperlink can be enabled in DSP or ARM Linux.

As mentioned in Section 1.4, the Hyperlink driver is provided in the Linux user space by MPM transport (http://git.ti.com/gitweb/?p=keystone-linux/mpm-transport.git;a=summary). The driver file is a shared library called libmpmtransport.so.1.0.0. By default, the ARM Linux file system already includes this file under the /usr/lib folder. The driver file can be rebuilt then copied over, if necessary (for example, for debug purpose). It needs to be built along with the CSL package in a Linux host machine, as explained in: http://processors.wiki.ti.com/index.php/MCSDK_UG_Chapter_Developing_Transports#MPM_Transport. The CSL package is installed under the “sysroot”, for example, approximately /ti-processor-sdk-linux-k2hk-evm-02.00.01.07/linux-devkit/sysroots/cortexa15hf-vfp-neon-linux-gnueabi/usr/include/ti/csl.

To enable the Hyperlink, run command “mpmcl transport <slave name> open” under Linux prompt, where the slave name is given inside /etc/mpm/mpm_config.json This command uses a default timeout of 5 seconds, if the Hyperlink is not established within timeout, the UIO driver closes the Hyperlink by powering it off.

4.2 Set Hyperlink SerDes Tx/Rx Parameters

In ARM Linux, the Hyperlink setting (reference clock rate, lane rate, Tx parameters (TX_ATT, VREG, C1, C2, CM), Rx parameters (RX_ATT, BOOST starting or forced values)) are given in Linux DTS file arch/arm/boot/dts/k2x-evm.dts:

```
uio_hyperlink0: hyperlink0 {
     compatible = "ti,uio-module-drv";
     mem  = <0x21400000 0x00000100
          0x40000000 0x10000000
          0x0231a000 0x00002000>;
     clocks = <&clkhyperlink0>;
     interrupts    = <0 387 0x101>;
     label = "hyperlink0";
     cfg-params
     {
          ti,serdes_refclk_khz = <312500>;
          ti,serdes_maxserrate_khz = <6250000>;
          ti,serdes_lanerate = "half";
          ti,serdes_numlanes = <4>;
          ti,serdes_c1 = <4 4 4 4>;
          ti,serdes_c2 = <0 0 0 0>;
          ti,serdes_cm = <0 0 0 0>;
          ti,serdes_tx_att = <12 12 12 12>;
          ti,serdes_tx_vreg = <4 4 4 4>;
          ti,serdes_rx_att = <11 11 11 11>;
          ti,serdes_rx_boost = <3 3 3 3>;
     }
```

Note that the DTS file does not provide the Rx configuration selection either; it is adaptive or forced. This is hard-coded inside MPM transport: src/transport/hyplnk/mpm_transport_hyplnk_interface.c with adaptive:

```
serdes_init_params.forceattboost = CSL_SERDES_FORCE_ATT_BOOST_DISABLED;
for (i=0; i<num_lanes; i++)
{
     ……
     /* For higher speeds PHY-A, force attenuation and boost values  */
     serdes_init_params.rx_coeff.force_att_val[i] = 1;
     serdes_init_params.rx_coeff.force_boost_val[i] = 1;
```

The Errata usage note 28 advises to force attenuation and boost for Hyperlink data rate higher than 6.25Gbps. There are still chances that Rx adaptation fails, probably due to board design issues. Therefore, it is recommended to use forced mode for any lane rates, especially Hyperlink power on stability is a concern. The forced values needs to be obtained through PRBS testing discussed earlier. The above code needs to be updated and the library needs to be re-built:

```
serdes_init_params.forceattboost = CSL_SERDES_FORCE_ATT_BOOST_ENABLED;
for (i=0; i<num_lanes; i++)
{
     ……
     /* For higher speeds PHY-A, force attenuation and boost values  */
     serdes_init_params.rx_coeff.force_att_val[i] = rx_att[i];
     serdes_init_params.rx_coeff.force_boost_val[i] = rx_boost[i];
```

4.3 Enable Debug Trace

- To enable the debug trace in MPM transport library, check src/transport/mpm_transport_cfg.h, and change the debug trace level via #define VERBOSITY_LEVEL 0/1/2. Then, add *mpm_printf()* to print those out.
- To debug inside CSL code, the *printf()* can be added into the individual CSL functions under the sysroot of the host machine.

- The trace is dumped into /var/log/syslog, which is controlled by /etc/mpm/mpm_config.json.
- If the Hyperlink fails, the UIO driver powers the Hyperlink off so that all Hyperlink registers dumped out are zeros. The transport code needs further change to avoid Hyperlink powering off if those registers needs to be reserved for post analysis.
- Any code changes inside MPM transport or CSL need to rebuild the library, then copy it over to the ARM Linux file system

4.4 Debug Hyperlink Failure

The Hyperlink interface may fail with the following symptoms during initialization:

- Linux MPM transport driver open/enable timeout
- Link is up but with Hyperlink ECC errors
- Link is up but reading into remote cause the system hang

The debug can use the Linux command or CCS/JTAG. The following sections list several things to check.

4.4.1 Hyperlink Lane Power Management Control (offset 0x44)

Although Hyperlink can be set to zerolane or onelane mode for power saving purpose, it is observed that this can result in link failure when switching to quadlane mode. Therefore, it is recommended to set bit 2 to 1, while clear bit 1 and bit 0.

4.4.2 Hyperlink ECC Error Counters (offset 0x4C)

The register includes fields for single bit error and double bit error:

- The Single Error Correctable counter is incremented when a correctable error is detected by the Rx PLS layer. Writing any value to this register clears this counter.
- The Double Error Detectable counter is incremented when a detectable two-bit error is detected by the Rx PLS layer. This indicates that the receive channel signal is marginal. Writing any value to this register will clear it.

Although ECC is a bad thing, it doesn't necessarily lead to link failure. It is important to distinguish the ECC is one time or accumulating over time. For 10Gb/s interface, it is easily seen ECC error when the Hyperlink is initialized. One can do is to clear the ECC error, then let the link run for a few seconds to see if the ECC is accumulating. If there is no ECC any more, the link can be used normally or for further qualification using a stress test. If there is ECC accumulating over time, this is a problem link and a double bit error will quickly hangs the system up.

4.4.3 Hyperlink Link Status Register (offset 0x58)

This register is very important to determine the usefulness of the link.

The upper 16 bits indicates the Tx state of the link on this device. It should be B'1x1x11011111xxxx where x is any value. Tx is linked if (LinkStatus&0xadf00000) == 0xadf00000. Be careful, the last 4-bits indicate polarity and may be different on different links and boards.

The Lower 16 bits indicates the Rx state of the link on this device. It should be B'1x1x11011111xxxx where x is any value. Rx is linked if (LinkStatus&0x0000adf0)== 0x0000adf0. Again, the last 4-bits indicate polarity and may be different on different links and boards.

In most of the system design, 4 lanes are used and the Tx/Rx pair polarity are the same along the trace for all 4 lanes, so linkStatus = 0xfdf0bdf0.

In failed cases, one side is typically 0xfdf0_0cfx (Rx error) and the other side is 0xccf0_bdf0 (Tx error). The last 4-bit may toggling if monitored continuously, this is an indication the SerDes is inverting the individual lane polarity for signal detection.

4.4.4 SerDes PLL Control and LANExCTL_STS

PLL_CTRL (offset 0x1FF4) and LANExCTL_STS (offset 0x1FE0/4/8/C) provide PLL and lane status indication on SerDes level. A working case dump is given here for reference:

- 0231bfe0: f0c0f032 f0c0f032 f0c0f032 f0c0f032
- 0231bff0: 00000000 f0000f0f 00000000 000124f8

4.5 Recover a Failed Hyperlink Link

The Hyperlink failure may come from many reasons:

- Poor board design or non-optimal Tx/Rx setting, this can be checked by BER sweep tool and PRBS test
- Improper SerDes configuration file or SerDes sequence: always use the latest TI Processor SDK package

For trial/debug purpose, if the channel is proven clean via BER test, Hyperlink failure can be recovered by put both sides into reset then bring out of reset, it does not matter which sides execute the sequence first.

- If (linkStatus & 0xadf0adf0 != 0xadf0adf0)
  - Set Hyperlink control (offset 0x4) bit 0 on side A
  - Set Hyperlink control (offset 0x4) bit 0 on side B
  - Clear Hyperlink control (offset 0x4) bit 0 on side A
  - Clear Hyperlink control (offset 0x4) bit 0 on side B
  - For the Keystone II side, set SerDes PLL_CTRL (offset 0x1FF4) bit 7-4 to 0xF

This sequence should bring both sides back to link up status.

5 References

- *SerDes Implementation Guide for KeyStone I Devices* (SPRABC1)
- *KeyStone II Architecture Serializer/Deserializer (SerDes) User's Guide* (SPRUHO3)
- *KeyStone Architecture HyperLink User's Guide* (SPRUGW8)
- *KeyStone Architecture Peripheral Component Interconnect Express (PCIe) User's Guide* (SPRUGS6)
- *KeyStone Architecture Serial Rapid IO (SRIO) User's Guide* (SPRUGW1)
- *KeyStone Architecture Gigabit Ethernet (GbE) Switch Subsystem User's Guide* (SPRUGV9)
- *66AK2H06/12/14 Multicore DSP+ARM KeyStone II SOC Errata (Revs 1.0, 1.1, 2.0) Silicon Errata* (SPRZ402)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265