Application Report
SPRABX7–January 2015

Keystone II DDR3 Initialization

DavidLiu/Tom Johnson

ABSTRACT

This application report provides a step-to-step initialization guide for the Keystone II device DDR3 SDRAM controller.

Contents

1 Introduction

The initialization of the DDR3 SDRAM controller on Keystone II DSPs is straightforward as long as the proper steps are followed. However, if some steps are omitted or if some sequence-sensitive steps are implemented in the wrong order, DDR3 operation will be unpredictable.

Detailed explanation of the registers and their functionality are provided in the *KeyStone II Architecture DDR3 Memory Controller User's Guide* (SPRUHN7). Board layout guidance is provided in the *DDR3 Design Requirements for Keystone Devices* (SPRABI1).

All DDR3 initialization routines must contain the basic register writes to configure the memory controller within the DSP as well as register writes that configure the mode registers within the attached SDRAM devices. The device-specific data sheet for the implemented SDRAM must be referenced to optimize these values.

The physical DDR3 interface on the Keystone II DSPs is often called the DDR3 PHY. It includes the I/O buffers and all of the logic required to support the DDR3 interface technology. The DDR3 interface circuitry also includes registers and control logic to support the physical DDR3 interface as well as control for the SDRAM devices. The terms PHY and Controller are used interchangeably in this document when referring to this circuitry.

The spreadsheet discussed in this application report can be downloaded from the following URL: http://www.ti.com/lit/zip/sprabx7.

2 New DDR3 Feature: Leveling

This section provides an introduction to the new DDR3 physical interface concept called leveling. A basic understanding of this technology will help programmers understand the required configuration steps.

Write and read leveling are supported within the DDR3 PHY implemented within KeyStone II devices. The write and read leveling functionality are discussed more in the next subsection. The leveling circuitry in the DDR3 controller must be properly configured for robust operation.

2.1 Write and Read Leveling

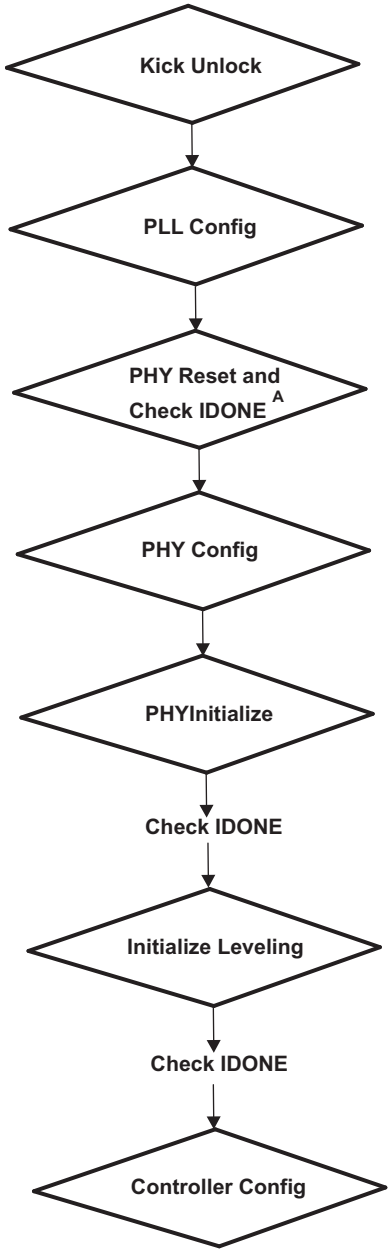
In DDR3 board layouts, the address, command, control and clock nets are routed in a fly-by topology. Data group nets are routed point-to-point. The fly-by topology can cause skew between the clock and the data strobe to the SDRAM, making it difficult for the controller to maintain t_{DQSS}, t_{DSS} and t_{DSH} specifications. Write and read leveling are used to compensate for this skew by aligning the clock with the data strobe at each SDRAM. The accuracy required for write leveling is equal to the SDRAM t_{DQSS} timing parameter. This is defined as the DQS, $\overline{DQS}$ rising edge to CK, $\overline{CK}$ rising edge and is equal to ±0.25 of tCK (SDRAM clock period). The required leveling intentionally sweeps the data strobe across the rising SDRAM clock edge and executes the algorithm to align the DQS of each byte for all ranks.

3 DDR3 Controller Configuration

The DDR3 controller and SDRAM must be initalized before use. The initialization routines normally have four basic pieces. This is true whether the routine is implemented within a Code Composer Studio™ (CCS) GEL file or embedded software such as C-code. The four sections are:

- Header section - contains '# define' macros that represent peripheral memory addresses and their register address offsets with useful names
- KICK unlock and DDR3 PLL configuration - this can reside elsewhere, but it must be completed prior to initializing the DDR3 controller and SDRAM
- PHY Configuration and Leveling - register writes to prepare and execute the write and read leveling operations
- Memory Controller Registers - register writes that configure the critical timing parameters and mode register parameters needed by the PHY and the controller

Figure 1 illustrates the configuration sequence.



A For TCI6634K2K PG1.0, TCI6638K2K PG1.0, TCI6636K2H PG1.1, 66AK2H70PG1.1, 66AK2H12 PG1.1 and 66AK2H06 PG1.1, checking IDONE between PLL configuration and PHY configuration is optional.

Figure 1. Configuration Sequence

3.1 Header Section

The header section contains '# define' macros that represent cryptic addresses with useful names. The register address offsets can be obtained from the *KeyStone II Architecture DDR3 Memory Controller User's Guide* (SPRUHN7). The base addresses for the register groups can be found in the device-specific data manual for the KeyStone II device being used. Example 1 shows the addresses for DDR3A as defined for the 66AK2H12 device. Note that many of the registers are located in the chip-level registers address region starting at 0x0262_0000, some of the registers are in the address region starting at 0x2101_0000 defined for the DDR3A Controller and some of the registers are in the address region starting at 0x0232_9000 for the DDR3A PHY. Chip-level registers are also referred to as boot configuration registers in other Keystone II documentation.

Example 1. Sample Header

```
#define CHIP_LEVEL_REG          0x02620000
#define KICK0                   *(unsigned int*)(CHIP_LEVEL_REG + 0x0038)
#define KICK1                   *(unsigned int*)(CHIP_LEVEL_REG + 0x003C)
#define KICK0_UNLOCK            (0x83E70B13)
#define KICK1_UNLOCK            (0x95A4F1E0)
#define KICK0_LOCK              0
#define KICK1_LOCK              0
#define MAINPLLCTL0             *(unsigned int*)(CHIP_LEVEL_REG + 0x0350)
#define MAINPLLCTL1             *(unsigned int*)(CHIP_LEVEL_REG + 0x0354)
#define OBSCLKCTL               *(unsigned int*)(CHIP_LEVEL_REG + 0x0C80)
#define CHIP_MISC1              *(unsigned int*)(CHIP_LEVEL_REG + 0x0C7C)
#define DDR3APLLCTL0            *(unsigned int*)(CHIP_LEVEL_REG + 0x0360)
#define DDR3APLLCTL1            *(unsigned int*)(CHIP_LEVEL_REG + 0x0364)
#define DDR3BPLLCTL0            *(unsigned int*)(CHIP_LEVEL_REG + 0x0368)
#define DDR3BPLLCTL1            *(unsigned int*)(CHIP_LEVEL_REG + 0x036C)
#define DDR3A_BASE_ADDR         (0x21010000)
#define DDR3A_STATUS            (*(int*)(DDR3A_BASE_ADDR + 0x00000004))
#define DDR3A_SDCFG             (*(int*)(DDR3A_BASE_ADDR + 0x00000008))
#define DDR3A_SDRFC             (*(int*)(DDR3A_BASE_ADDR + 0x00000010))
#define DDR3A_SDTIM1            (*(int*)(DDR3A_BASE_ADDR + 0x00000018))
#define DDR3A_SDTIM2            (*(int*)(DDR3A_BASE_ADDR + 0x0000001C))
#define DDR3A_SDTIM3            (*(int*)(DDR3A_BASE_ADDR + 0x00000020))
#define DDR3A_SDTIM4            (*(int*)(DDR3A_BASE_ADDR + 0x00000028))
#define DDR3A_ZQCFG             (*(int*)(DDR3A_BASE_ADDR + 0x000000C8))
#define DDR3A_TMPALRT           (*(int*)(DDR3A_BASE_ADDR + 0x000000CC))
#define DDR3A_DDRPHYC           (*(int*)(DDR3A_BASE_ADDR + 0x000000E4))
#define DDR3A_PHY_CFG_BASE      (0x02329000)
#define DDR3A_PIR               (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000004))
#define DDR3A_PGCR0             (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000008))
#define DDR3A_PGCR1             (*(int*)(DDR3A_PHY_CFG_BASE + 0x0000000C))
#define DDR3A_PGCR2             (*(int*)(DDR3A_PHY_CFG_BASE + 0x0000008C))
#define DDR3A_PGSR0             (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000010))
#define DDR3A_PGSR1             (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000014))
#define DDR3A_PLLCR             (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000018))
#define DDR3A_PTR0              (*(int*)(DDR3A_PHY_CFG_BASE + 0x0000001C))
#define DDR3A_PTR1              (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000020))
#define DDR3A_PTR2              (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000024))
#define DDR3A_PTR3              (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000028))
#define DDR3A_PTR4              (*(int*)(DDR3A_PHY_CFG_BASE + 0x0000002C))
#define DDR3A_DSGCR             (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000040))
#define DDR3A_DCR               (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000044))
#define DDR3A_MR0               (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000054))
#define DDR3A_MR1               (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000058))
#define DDR3A_MR2               (*(int*)(DDR3A_PHY_CFG_BASE + 0x0000005C))
#define DDR3A_DTCR              (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000068))
#define DDR3A_DTPR0             (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000048))
#define DDR3A_DTPR1             (*(int*)(DDR3A_PHY_CFG_BASE + 0x0000004C))
#define DDR3A_DTPR2             (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000050))
#define DDR3A_ZQ0CR1            (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000184))
#define DDR3A_ZQ1CR1            (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000194))
#define DDR3A_ZQ2CR1            (*(int*)(DDR3A_PHY_CFG_BASE + 0x000001A4))
```

Example 1. Sample Header (continued)

```
#define DDR3A_ZQ3CR1              (*(int*)(DDR3A_PHY_CFG_BASE + 0x000001B4))
#define DDR3A_DX4GCR              (*(int*)(DDR3A_PHY_CFG_BASE + 0x000002C0))
#define DDR3A_DX5GCR              (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000300))
#define DDR3A_DX6GCR              (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000340))
#define DDR3A_DX7GCR              (*(int*)(DDR3A_PHY_CFG_BASE + 0x00000380))
#define DDR3A_DX8GCR              (*(int*)(DDR3A_PHY_CFG_BASE + 0x000003C0))
```

3.2 KICK Unlock and DDR3 PLL Configuration

The KICK register unlock and DDR3 PLL configuration code may reside elsewhere in the initialization code but they must be completed prior to initializing the DDR3 controller and SDRAM.

Prior to writing to the DDR3 PLL configuration registers, the KICK registers need to be unlocked. Example 2 shows the unlocking of the KICK registers. Once the KICK registers have been unlocked, the DDR3 PLL configuration can be completed.

The DDR3 PLL within the Keystone II DSP receives the reference clock provided at the DDR3CLK input and multiples it up to the rate desired for the DDR3 interface depending on the DDR3 PLL multiplier, divider, and output divider configuration setting. Note that there is an internal x2 clock multiplier in the DDR PHY, so the clock rate to the DDR3 memory is half of the data rate (DDR3-1333 operates with a 666.666-MHz clock) while the clock frequency out of the DDR3 PLL is fourth of the data rate (333.333 MHz). Example 4 programs the DDR3 PLL to generate a 333.333-MHz clock (for DDR3-1333 operation) from an input clock of 100 MHz. For more details on PLL configuration, see the *Keystone Architecture Phase-Locked Loop (PLL) User's Guide* (SPRUGV2). Note that the DDR3 clock rate affects timing parameters throughout this application report.

Example 2. Unlocking KICK Registers

```
KICK0 = 0x83E70B13;
KICK1 = 0x95A4F1E0;
```

Example 3. DDR3 PLL Programming

```
unsigned int multiplier = 19;
unsigned int divider = 0;
unsigned int OD_val = 6;
int temp,i, delay = 1000;

   // DDR3A PLL setup
   GEL_TextOut ( "DDR3 PLL Setup ... \n");

   // Set ENSAT = 1 for optimal PLL operation
   DDR3APLLCTL1 |= 0x00000040;

   // Set BYPASS = 1 to put the PLL in bypass mode

   // Program the necessary multipliers/dividers and BW adjustments

   // Set the divider values
   DDR3APLLCTL0 &= ~(0x0000003F);
   DDR3APLLCTL0 |= (divider & 0x0000003F);

   // Program OD[3:0] in the SECCTL register

   // Clear the OD bit field
   DDR3APLLCTL0 &= 0xFF87FFFF;

   // Set the OD[3:0] bit field of PLLD to OD_val
   DDR3APLLCTL0 |= ~(0xFF87FFFF) & (OD_val - 1) << 19;
```

Example 3. DDR3 PLL Programming (continued)

```
   // Set the multiplier values
   DDR3APLLCTL0 &= ~(0x0007FFC0);
   DDR3APLLCTL0 |= ((multiplier << 6) & 0x0007FFC0 );

   // Set BWADJ
   temp = ((multiplier + 1) >> 1) - 1;
   DDR3APLLCTL0 &= ~(0xFF000000);
   DDR3APLLCTL0 |= ((temp << 24) & 0xFF000000);
   DDR3APLLCTL1 &= ~(0x0000000F);
   DDR3APLLCTL1 |= ((temp >> 8) & 0x0000000F);

   // In PLL controller, reset the PLL (bit 13 in DDR3APLLCTL1 register)
   DDR3APLLCTL1 |= 0x00004000;

      // Proper delay time is required
   for(i=0;i<delay;i++);

   // In DDR3PLLCTL1, write PLLRST = 0 to bring PLL out of reset
   DDR3APLLCTL1 &= ~(0x00004000);

      // Proper delay time is required
   for(i=0;i<delay;i++);

   // Put the PLL in PLL Mode
   DDR3APLLCTL0 &= ~(0x00800000); // Reset the Bit 23
   GEL_TextOut( "DDR3 PLL Setup complete, DDR3A clock now running at 666 MHz.\n" );
```

3.3 PHY Configuration Register

After the KICK registers have been unlocked and the DDR3 PLL completely initialized (these steps could have been done by other configuration routines previously), the PHY Reset needs to be asserted and released. Then, the DDR3 PHY executes an internal state machine that performs low-level initialization. The IDONE bit in the PHY General Status Register 0 (at address offset 0x010) is set high once this process completes. Therefore, the IDONE bit needs to be polled as shown in Example 4. Do not proceed until the IDONE bit is set.

Note that the steps in Example 4 are not required for TCI6634K2K PG1.0, TCI6638K2K PG1.0, TCI6636K2H PG1.1, 66AK2H70PG1.1, 66AK2H12 PG1.1 and 66AK2H06 PG1.1 devices but that they are required for all newer device versions.

Example 4. DDR3 PHY Initialization

```
//Assert DDR PHY reset after PLL enabled
DDR3APLLCTL1 = DDR3APLLCTL1 | 0x80000000;
for(i=0;i<delay;i++);
DDR3APLLCTL1 = DDR3APLLCTL1 & 0x7FFFFFFF;   //Release DDR PHY reset

// Poll IDONE after resetting PHY
do {
    read_val = DDR3A_PGSR0;
    } while ((read_val&0x00000001) != 0x00000001);
```

The PHY configuration registers that provide initialization values to the leveling circuitry need to be programmed next. These values will be used after the basic controller and SDRAM configuration but they must be written before that part of the initialization sequence. This is shown in Example 5.

Example 5. DDR3 PHY Programming

```
// Program FRQSEL in the PLL Control Register (address offset 0x018). The FREQSEL value is
// chosen based on the DDR3 PLL input reference clock frequency. In this example, FRQSEL is
// set to 0x11 for CTL_CLK frequency between 166-275MHz.
DDR3A_PLLCR = 0xDC000;

// Program WLSTEP=1, IODDRM=1, and ZCKSEL in the PHY General Configuration Register 1
// (address offset 0x00C). All other fields must be left at their default values by using a
// read-modify-write sequence to preserve the other bits. ZCKSEL is chosen from the register
// calculation spreadsheet based on the main PLL rate.
DDR3A_PGCR1 |= (1 << 2); //WLSTEP = 1
DDR3A_PGCR1 &= ~( 0x00000180);
DDR3A_PGCR1 |= (( 1 << 7) & 0x00000180);
DDR3A_PGCR1 &= ~( 0x01800000);
DDR3A_PGCR1 |= (( 1 << 23) & 0x01800000);

// Program PHY Timing Parameters Register 0-4 (address offset 0x01C - 0x02C) based on the
// value extracted from the register calculation spreadsheet.
DDR3A_PTR0 = 0x42C21590;
DDR3A_PTR1 = 0xD05612C0;

// Maintaining default values of PHY Timing Parameters Register 2 in PUB
DDR3A_PTR3 = 0x0B4515C2;
DDR3A_PTR4 = 0x0A6E08B4;

// Program PDQ, MPRDQ, and BYTEMASK in the SDRAM Configuration Register (address offset 0x044).
// All other fields must be left at their default values by using read-modify-write sequence.
DDR3A_DCR &= ~(0x00000070); //PDQ = 0
DDR3A_DCR &= ~(0x00000080); //MPRDQ = 0
DDR3A_DCR &= ~(0x0000FC00);
DDR3A_DCR |= (( 1 << 10) & 0x0000FC00);

// Program SDRAM Timing Parameters Register 0-2 (address offset 0x048 - 0x050) based on the
// timing parameters extracted from the register calculation spreadsheet.
DDR3A_DTPR0 = 0x8558AA75;
DDR3A_DTPR1 = 0x32857280;
DDR3A_DTPR2 = 0x5002C200;

// Program BL=0, CL, WR, and PD=1 in the Mode Register 0 (address offset 0x054) based on the
//timing parameters extracted from the register calculation spreadsheet.
//All other fields must be left at their default values.
DDR3A_MR0 = 0x00001A60;

// Program DIC, RTT, and TDQS in the Mode Register 1 (address offset 0x058). All other fields
// will programmed to their default values.
DDR3A_MR1 = 0x00000006;

// Program Mode Register 2 (address offset 0x05C). Maintaining default values of Program Mode
//Register 2
DDR3A_MR2 = 0x00000010;

// Program DTMPR=1, DTEXD, DTEXG, RANKEN=1 or 3, and RFSHDT=7 in the Data Training
// Configuration Register (address offset 0x068). All other fields must be left at their
//default values. See section 3.3.1 if configuring a dual-rank implementation.
DDR3A_DTCR = 0x710035C7;     //Single-Rank

// Program tREFPRD=(5*tREFI/ddr_clk_period) in the PHY General Configuration Register 2
//(address offset 0x08C). All other fields must be left at their default values.
DDR3A_PGCR2 = 0x00F065B8;
```

Example 5. DDR3 PHY Programming (continued)

```
//Set Impedance Register
DDR3A_ZQ0CR1 = 0x0001005D;
DDR3A_ZQ1CR1 = 0x0001005B;
DDR3A_ZQ2CR1 = 0x0001005B;

// Re-trigger PHY initialization in DDR PHY through the VBUSP interface.
// Program 0x00000033 to the PHY Initialization Register (address offset 0x004)
//to re-trigger PLL, ZCAL, and DCAL initialization.
DDR3A_PIR = 0x00000033;

// Poll for IDONE=1 in the PHY General Status Register 0 (address offset 0x010).
     do {
           read_val = DDR3A_PGSR0;
     } while ((read_val&0x00000001) != 0x00000001);
```

3.3.1 PHY Configuration for Dual Rank, Address Mirrored Considerations

Example 6 shows the register setting for dual rank, address mirrored DIMM configuration.

Example 6. Dual Rank, Address Mirrored Configuration

```
//Enable addressing mirroring on second rank
DDR3B_DCR &= ~(0x08000000);
DDR3B_DCR |= (( 1 << 27) & 0x08000000);
DDR3B_DCR &= ~(0x20000000);
DDR3B_DCR |= (( 1 << 29) & 0x20000000);
//Enable dual rank operation
DDR3A_DTCR = 0x730035C7;
```

3.4 PHY Leveling Execution

Example 7 shows the steps that actually trigger the write and read leveling steps and then waits for these to complete. The configuration steps in the previous PHY configuration section must be successfully completed before leveling.

Example 7. Leveling Programming

```
// If using different data bus bit width, refer to section 3.4.1 and add those lines at this point in
the sequence.

// Program 0x0000XF81 to the PHY Initialization Register to trigger DDR3 initialization and
leveling/training sequences
DDR3A_PIR = 0x0000FF81;

// Poll for IDONE=1 in the PHY General Status Register 0 (address offset 0x010).
     do {
               read_val = DDR3A_PGSR0;
     } while ((read_val&0x00000001) != 0x00000001);
/* End PHY Configuration */
```

3.4.1 Configuration for Different Data Bus Width

By default, setup supports 64-bit data bus and ECC enable. If using any data bus width other than 64 bits, then the DXEN bit needs to be cleared in the DATX8 General Configuration Registers for the unused byte lanes to disable the leveling and training for those upper byte lanes. If ECC is not required, then DXEN needs to be cleared in the DATX8 8 General Configuration Register (address offset 0x3C0) to disable the leveling and training for the ECC byte lane.

Example 8 shows the register settings for a 32-bit data bus width configuration. The register writes will need to be implemented as shown in Example 7 prior to triggering the leveling sequences.

Example 8. 32-Bit Data Bus Width Configuration

```
// Next 4 writes disable byte lanes of PHY for upper 32 bits
DDR3A_DX4GCR &=  ~(0x00000001);
DDR3A_DX5GCR &=  ~(0x00000001);
DDR3A_DX6GCR &=  ~(0x00000001);
DDR3A_DX7GCR &=  ~(0x00000001);
//This write disables the ECC byte
DDR3A_DX8GCR &=  ~(0x00000001);
```

3.5 DDR3 Memory Controller Registers and SDRAM Configuration

The basic controller and SDRAM configuration writes occur next. These register writes configure the controller critical timing parameters. *KeyStone II Architecture DDR3 Memory Controller User's Guide* (SPRUHN7) must be used to create these configuration values. The data sheet timing from the device-specific SDRAM also needs to be referenced to extract critical timing parameters. Formulas and look-up tables (LUTs) in the *KeyStone Architecture DDR3 Memory Controller User's Guide* (SPRUGV8) provide the translation from data sheet parameters to the bit-field values. A DDR3 Register Calculation spreadsheet is also available to streamline this process.

Each of the register writes is discussed individually in the recommended order.

At the end of this section, the DDR interface is programmed and ready for system use.

3.5.1 SDCFG Register Programming

The DDR_SDCFG register contains SDRAM and controller configuration parameters. DDR3 Register Calculation spreadsheet is used to calculate and assign the DDR_SDCFG value.

Example 9. SDRAM Configuration Register Programming

```
DDR3A_SDCFG        = 0x6200CA62; //Single-Rank
```

3.5.1.1 EBANK Configuration for Dual-Rank Considerations

For single rank, EBANK field in DDR_SDCFG register needs to be programmed to 0x0.

For dual rank, EBANK field in DDR_SDCFG register needs to be programmed to 0x1.

3.5.2 SDTIM[4:1] Register Programming

The DDR_SDTIM[4:1] registers contain most of the critical SDRAM and controller timing parameters. DDR3 Register Calculation spreadsheet is used to calculate and assign the DDR_SDTIM[4:1] values.

Example 10. DDR_SDTIM[4:1] Register Programming

```
DDR3A_SDTIM1   = 0x125C8464;
DDR3A_SDTIM2   = 0x00001D29;
DDR3A_SDTIM3   = 0x32CDFF43;
DDR3A_SDTIM4   = 0x543F0ADF;
```

3.5.3 ZQCFG Register Programming

The DDR_ZQCFG register configures the adaptive impedance calibration capability. Based on a 240-Ω resistor attached to each SDRAM, the SDRAM can re-calibrate its output impedance periodically as its temperature changes. This example causes the ZQ calibration to occur once every 100 ms. (This interval is under study.) For more details, see the *KeyStone II Architecture DDR3 Memory Controller User's Guide* (SPRUHN7).

Example 11. Dynamic Impedance Configuration Register Programming

```
DDR3A_ZQCFG    = 0x70073200; //Single-Rank
```

3.5.3.1 ZQCFG Configuration for Dual-Rank Considerations

For single rank, ZQCFG needs to be programmed to 0x70073200.

For dual rank, ZQCFG needs to be programmed to 0xF0073200.

3.5.4 SDRFC Register Programming

The DDR_SDRFC register controls the behavior of refresh. The 0x1457 portion of this register sets the refresh period to 7.81 µs with the current clock rate. The MSB of this register may be set or cleared to enable or disable SDRAM initialization and refreshes. At this time, it must be cleared to enable SDRAM initialization.

Example 12. Refresh Control Register Programming

```
// Program reg_initref_dis=0 in the SDRAM Refresh Control Register (address offset 0x10).
DDR3A_SDRFC = 0x00000A2C;
```

3.6 Lock Kick Registers

At the end of the DDR3 controller configuration, the KICK registers may be locked. This can be done immediately after the DDR3 controller configuration or later after other memory regions provided by this mechanism are completely configured.

Example 13. Lock Kick Registers

```
//lock kick registers
KICK0 = KICK0_LOCK;
KICK1 = KICK1_LOCK;
```

Note that the need to lock the KICK registers is application dependent. If the application will allow multiple cores to access the bootcfg registers, there may be race conditions (see the chip-specific Errata document). One workaround for this problem is not locking the KICK registers at the end of this process.

4 References

- *KeyStone II Architecture DDR3 Memory Controller User's Guide* (SPRUHN7)
- *DDR3 Design Requirements for Keystone Devices* (SPRABI1)
- *KeyStone Architecture DDR3 Memory Controller User's Guide* (SPRUGV8)

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