Application Report
SPRAC36E–April 2016–Revised January 2020

AM57x, DRA7x, and TDA2x EMIF Tools

ABSTRACT

At the center of every application is the need for memory. With limited on-chip processor memory, external memory serves as a solution for large software systems and data storage, and an unstable external memory interface can result in system failures or hinder software development. To prevent potential system level anomalies and ensure robust systems, hardware must be configured correctly and tested thoroughly.

The EMIF Tools application focuses on post layout activities, including configuring the Texas Instruments' processors for accessing external double data rate (DDR) memories and optimizing delay locked loop (DLL) ratios to compensate for skew. This application report provides a detailed description on how to use the associated application files. The document overview provides a complete list of processors and memory types supported by the EMIF Tools application.

The spreadsheet discussed in this application report can be downloaded from the following URL: http://www.ti.com/lit/zip/sprac36.

Contents

List of Figures

List of Tables

Trademarks

All trademarks are the property of their respective owners.

1 Overview

This document provides detailed steps outlining the procedure to initialize Texas Instruments' processors to access external DDR memories using the accompanying tools included in the EMIF Tools application.

1.1 Supported Features of EMIF Tools

The EMIF Tools application supports the following features:

- TI SOCs: AM57x, DRA7x, TDA2x, TDA3x
- DDR Types: DDR2, DDR3, DDR3L, LPDDR2
- Basic EMIF configuration topics including:
  - Initializing the EMIF and DDR for basic read/write functionality
  - Compensating for signal skew
  - Enabling the error correction code (ECC) feature of the EMIF interface

The source code included in the EMIF Tools application is supported by the CPU targets listed in Table 1.

Table 1. Supported CPU Targets

CPU	TI SoC
Cortex-A15	AM57x, DRA7x, TDA2x
Cortex-M4	TDA3x

2 EMIF Configuration

The following section describes how to use the supporting application files to configure the EMIF controller for DDR memory accesses.

2.1 Preliminary Requirements

Before using the supporting application files, ensure that you have access to the following system application information:

- The data sheet of the selected DDR memory
- PCB trace lengths of the DDR clock and strobe signals (required when not utilizing the hardware leveling features of the TI Soc and DDR memory)

2.2 Generating EMIF Register Values

To assist you in defining the EMIF configuration register values, the EMIF Tools application provides an EMIF register configuration workbook. The workbook is divided into six worksheets, and requires specific information pertaining to the system application environment. The first worksheet should be reviewed; it will allow you to save and load custom configurations. The next three worksheets require your input; their tabs are red. The fifth worksheet provides calculated register values that should be used to configure the EMIF controller; its tab is green. The last worksheet also provides register values, but in GEL format.

Table 2. EMIF Register Configuration Worksheets

Worksheet	Description
Title-README	Informative; load/save custom configurations
Step1-SystemDetails	User input: system information
Step2-BoardDetails	User input: trace lengths of clock / strobe
Step3-DDRTimings	User input: timing requirements from DDR datasheet
Register Values (U-Boot)	Output: register values (u-boot format)
Register Values (GEL)	Output: register values (GEL format)

The following sections outline the procedure to complete the various required input parameters of the EMIF configuration workbook.

2.2.1 Step1 – System Details

The first worksheet requires you to input both high level system application details, as well as specific I/O settings for the DDR pins of the TI application processor and DDR memory.

Step 1A seeks system level details and is shown in Table 3 with populated example values.

Table 3. System Details

Detail	Description	Value	Units
1	Company / Board Name / Revision (Ex: TI_EVM_revC)	TI_EVM_revG3	-
2	TI Soc Part Number	DRA75x	-
3	SYS_CLK1 Frequency	20	MHz
4	Required EMIF Interfaces	2	-
5	DDR Memory Type	DDR3/L	-
6	DDR Memory Frequency	532	MHz
7	DDR Data Bus Width Per EMIF	32	Bits
8	Leveling Technique: "S/W" or "H/W"	H/W	-
9	Max DRAM Operating Temperature	<=85	°C
10	Enable ECC (May not apply to all EMIFs)	Yes	-
11	ECC Region 1: System Start Address	80000000	Enabled
12	ECC Region 1: System End Address	8FFFFFFF	Enabled
13	ECC Region 2: System Start Address	90000000	Disabled
14	ECC Region 2: System End Address	90000000	Disabled

The table parameters are defined in detail in the bulleted list below:

- Detail 1 – Company/Board Name/Revision: This value is used as a unique identifier to name the structures of register values in the “Register Values” worksheet. Valid characters include: ‘a’-‘z’, ‘A’-‘Z’, ‘0’-‘9’, and ‘_’.
- Detail 2 – TI Soc Part Number: This value should be selected to match the TI application processor part number utilized in the system application. Pre-defined values of supported processors are provided in a drop-down menu list.
- Detail 3 – SYS_CLK1 Frequency: This value is used to determine the DDR PLL register settings. Pre-defined values of supported SYS_CLK1 frequencies are provided in a drop-down menu list.
- Detail 4 – Required EMIF Interfaces: This value should be selected based on the desired number of EMIF interfaces to configure. Pre-defined values are provided in a drop-down menu list.
- Detail 5 – DDR Memory Type: This value should be selected based on the DDR memory type connected to the TI processor. Pre-defined values are provided in a drop-down menu list based on the supported memory types.
- Detail 6 – DDR Memory Frequency: This value should be selected for the desired DDR clock frequency.
- Detail 7 – DDR Data Bus Width per EMIF: This value should be selected based on the bus width between the TI processor and the DDR memory. This value represents the bus width per EMIF channel. Pre-defined values are provided in a drop-down menu list.
- Detail 8 – Leveling Technique: This value should be set for the preferred leveling technique.

- Detail 9, Max DRAM Operating Temperature: This value should be set to match the maximum operating temperature that the DRAM will be subjected to in the end application environment. Pre-defined values are provided in a drop-down menu list.
- Detail 10 – Enable ECC: Setting this parameter to “Yes” enables the ECC functionality of the EMIF controller (may not apply to all EMIFs on a device). A physical DDR memory should be connected to the ECC pins of the application processor.
- Detail 11, ECC Region 1 System Start Address: This value should be set to a hexadecimal value between “80000000” and “FFFFFFFF”. Note that only numerical characters and letters ‘a'-'f’ are valid.
- Detail 12 ECC Region 1 System End Address: This value should be set to a hexadecimal value between the ECC region 1 system start address and “FFFFFFFF”. Note that only numerical characters and letters ‘a'-'f’ are valid. Setting this value less than or equal to the ECC region 1 system start address will disable the region.
- Detail 13, ECC Region 2 System Start Address: This value should be set to a hexadecimal value between “80000000” and “FFFFFFFF”, and outside of the ECC region 1 address range. Note that only numerical characters and letters ‘a'-'f’ are valid.
- Detail 14, ECC Region 2 System End Address: This value should be set to a hexadecimal value between the ECC region 2 system start address and “FFFFFFFF”, and outside of the ECC region 1 address range. Note that only numerical characters and letters ‘a'-'f’ are valid. Setting this value less than or equal to the ECC region 2 system start address, or within the ECC region 1 address range, will disable the region.

Care should be taken to provide details in order throughout the workbook. As the parameter values are selected, the drop-down menu lists of other parameters may change. In some instances, a previous selected value may no longer be available. In this case, the cell location turns red. Providing the details in order should avoid the necessity to re-select a parameter value. Table 4 shows an example in which the “TI Soc Part Number” parameter has been changed compared to Table 3. In this case, the fourth parameter is flagged because the new selected TI processor does not support two EMIF channels.

Table 4. System Details: Warnings

Detail	Description	Value	Units
1	Company / Board Name / Revision (Ex: TI_EVM_revC)	TI_EVM_revG3	-
2	TI Soc Part Number	DRA72x	-
3	SYS_CLK1 Frequency	20	MHz
4	Required EMIF Interfaces	2	-
5	DDR Memory Type	DDR3/L	-
6	DDR Memory Frequency	532	MHz
7	DDR Data Bus Width Per EMIF	32	Bits
8	Leveling Technique: "S/W" or "H/W"	H/W	-
9	Max DRAM Operating Temperature	<=85	°C
10	Enable ECC (May not apply to all EMIFs)	Yes	-
11	ECC Region 1: System Start Address	80000000	Enabled
12	ECC Region 1: System End Address	8FFFFFFF	Enabled
13	ECC Region 2: System Start Address	90000000	Disabled
14	ECC Region 2: System End Address	90000000	Disabled

Step 1B requests specific details pertaining to the DDR memory utilized in the system application. These details are required for the workbook to determine the size and speed bin of the DDR memory. Table 5 shows an example.

Table 5. DDR Details

Detail	Description	Value	Units
15	Speed Bin: Data Rate	1066	MHz
16	Density	2	Gb
17	Width	16	Bits
18	Speed Bin: CAS Latency	7	ntCK

The table parameters are defined in detail in the bulleted list below:

- Detail 15 – Speed Bin (Data Rate): This value should be selected to match the data rate of the DDR memory connected to the TI processor.
- Detail 16 – Density: This value should be selected to match the density of a single DDR memory connected to the TI processor.
- Detail 17 – Width: This value should be selected to match the width of a single DDR memory connected to the TI processor.
- Detail 18 – Speed Bin (CAS Latency): This value should be selected to match the required CAS latency to ensure functional operation of the DDR memory at the data rate specified by the “Speed Bin: Data Rate” parameter.

Step 1C requires the user to provide the desired I/O settings for the DDR memory termination and output driver impedance. Table 6 illustrates an example configuration for DDR3.

Table 6. DDR IO Configuration (Memory Side)

Detail	Description	Value	Units
19	ODT / Rtt_Nom	RZQ/4	Ω
20	Dynamic ODT / Rtt_Wr	Disabled	Ω
21	Output Driver Impedance	RZQ/7	Ω

The table parameters are defined in detail in the bulleted list below:

- Detail 19 – ODT/Rtt_Nom: This value applies to the on-die termination of the DDR memory I/O pins. For more information, see the data sheet of the DDR memory. (1)
- Detail 20 – Dynamic ODT / Rtt_Wr: This value applies to on-die termination during DDR writes when the dynamic ODT mode is enabled. For more information, see the data sheet of the DDR memory. (1)
- Detail 21 – Output Driver Impedance: This value applies to the output driver impedance of the DDR memory I/O pins. For more information, see the see the data sheet of the DDR memory. (1)

(1) A recommendation is provided; however, board level simulations and signal integrity analysis should be performed to ensure the appropriate settings.

Step 1D allows you to modify the I/O settings for the DDR pins of the TI application processor. Table 7 illustrates the required input details.

Table 7. DDR IO Configuration (SoC Side)

Detail	Description	Value	Units
22	ODT	60	Ω
23	Slew Rate: Addr/Ctrl/Clk	Fastest: SR[2:0] = 0b000	–
24	Slew Rate: Data/Strobe	Fastest: SR[2:0] = 0b000	–
25	Output Driver Impedance: Addr/Ctrl/Clk	34	Ω
26	Output Driver Impedance: Data/Strobe	48	Ω

The table parameters are defined in detail in the bulleted list below:

- Detail 22 – ODT: This value applies to the on-die termination of the EMIF I/O pins on the Texas Instruments application processor.
- Detail 23 – Slew Rate (Addr/Ctrl/Clk): This value applies to the slew rate of the EMIF address, control, and clock I/O pins on the Texas Instruments application processor. (1)
- Detail 24 – Slew Rate (Data/Strobe): This value applies to the slew rate of the EMIF data and strobe I/O pins on the Texas Instruments application processor.
- Detail 25 – Output Driver Impedance (Addr/Ctrl/Clk): This value applies to the output driver impedance of the EMIF address, control, and clock I/O pins on the Texas Instruments application processor. (1)
- Detail 26 – Output Driver Impedance (Data/Strobe): This value applies to the output driver impedance of the EMIF data and strobe I/O pins on the Texas Instruments application processor. (1)

2.2.2 Step 2 – Board Details

The second worksheet requests you to enter the PCB trace lengths of the DDR clock and strobe signals from the TI application processor to the DDR memories for each byte lane. This information is pertinent for systems utilizing a fly-by topology to interface to DDR3/L memories. The PCB trace lengths are required to determine an approximation of the PCB flight skew of the signals, and serves as a starting point to the leveling algorithm. Full optimization is achieved from leveling.

Table 8 and Figure 1 illustrates the format for which the PCB trace lengths should be provided. Because a particular trace may be routed on more than one layer of the PCB, you should identify which portions of the trace are routed on the outer layers of the PCB and which portions are routed on the inner PCB layers between two planes.

Table 8. PCB DDR Trace Lengths

DRAMs Connected to EMIF1, Rank 0										
	PCB Trace Length in MILs (1/1000 inch)									
	Byte 0		Byte 1		Byte 2		Byte 3		Byte 4	
Signal	Microstrip	Stripline	Microstrip	Stripline	Microstrip	Stripline	Microstrip	Stripline	Microstrip	Stripline
CLK	A1	A2	B1	B2	C1	C2	D1	D2	E1	E2
DQSn	F1	F2	G1	G2	H1	H2	I1	I2	J1	J2

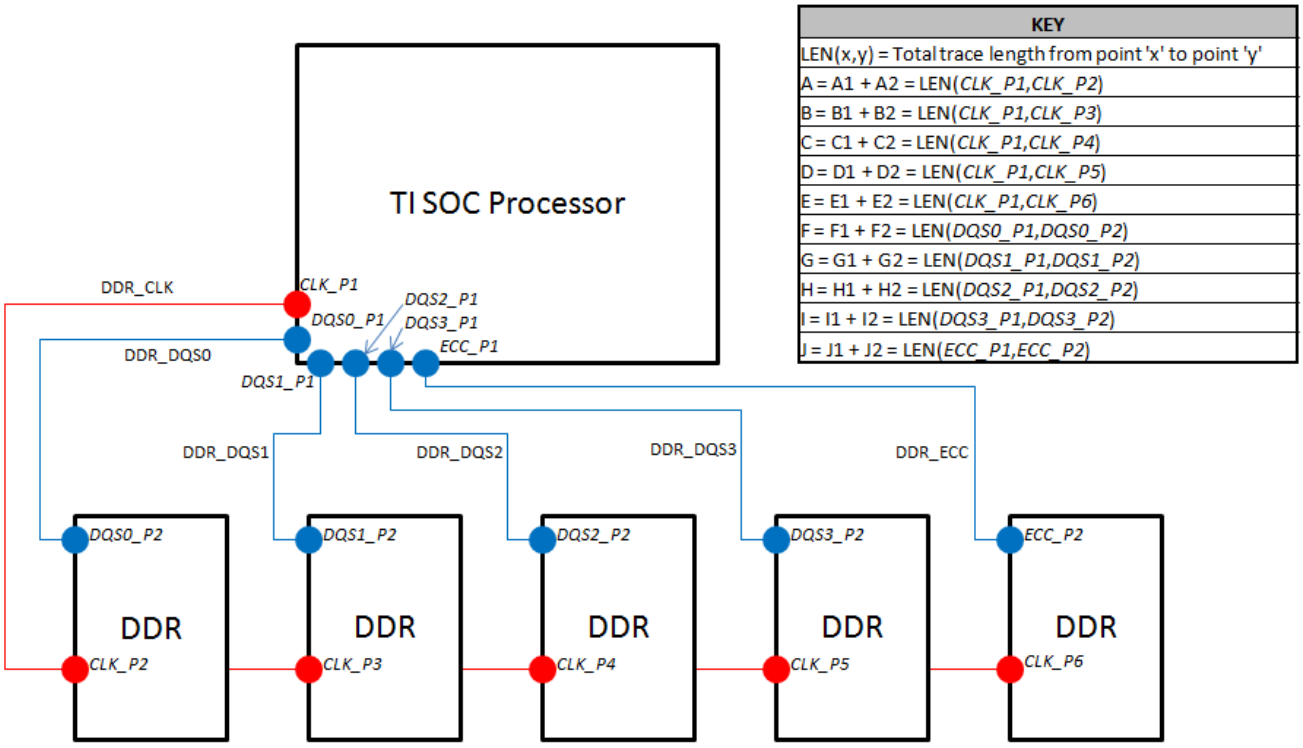

Figure 1. PCB DDR Trace Lengths

2.2.3 Step 3 – DDR Timings

The third worksheet requires you to input DDR timing values that can be found in the DDR memory data sheet. Figure 2 portrays the timing values required and is populated assuming a DDR3-1066F (2GB density, 2KB page size) memory operating at 532 MHz.

Parameter	Description	Datasheet Values		Final Bit Field Values		JEDEC Bit Field Values
		tCK	ns	Value	Units	(DDR3/L-1066 @ 532 MHz)
CAS Latency	Delay between internal READ command and data ready	7		7	tCK	7
CWL Latency	Delay between internal WRITE command and data ready	6		6	tCK	6
tRP	Precharge command period		13.125	6	tCK	6
tRCD	Active to read or write delay		13.125	6	tCK	6
tWR	Write recovery time		15	7	tCK	7
tRAS	Active to Precharge command period		37.5	19	tCK	19
tRC	Active to Active/Refresh command period		50.625	26	tCK	26
tRRD	Active Bank to Active Bank command period	4	10	6	tCK	6
tWTR	Internal Write to Read command delay	4	7.5	3	tCK	3
tXP	Exit power down mode to first valid command	3	7.5	3	tCK	3
tXSNR/tXS	Exit self refresh to commands not requiring a locked DLL	5	170	90	tCK	90
tXSRD/tXSDLL	Exit self refresh to commands requiring a locked DLL	512		511	tCK	511
tRTP	Internal Read to Precharge command delay	4	7.5	3	tCK	3
tCKE	CKE minimum pulse width	3	5.625	2	tCK	2
tCKESR	Minimum CKE low width for Self Refresh entry to exit	4		3	tCK	3
tZQCS	ZQ short calibration time	64		63	tCK	63
tRFC	Refresh to Active/Refresh command period		160	85	tCK	85
tRAS (max)	Active to Precharge command period (Max Value)		70200	8	tREFI intervals	8
tREFI	Average periodic refresh interval		7800	4149	tCK	4149
tFAW	Minimum Window for 4 Active Bank commands		50	See tRRD	-	6

Figure 2. DDR Timings

The following list describes the different columns in Figure 2:

- Parameter: The timing parameter name found in the DDR data sheet. All listed parameters require minimum timing values, except tRAS(max) and tREFI.
- Description: A description of the DDR timing parameter.
- Data Sheet Values: The corresponding DDR timing value found in the DDR data sheet. This value can either be defined in units tCK, ns, or the maximum of either a tCK or ns value. As illustrated in Figure 2, the worksheet calculates the bit field value based off of the maximum of either the tCK or ns value.
- Final Bit Field Value: The final bit field value programmed to the EMIF register. This value is typically in units of clock cycles.
- JEDEC Bit Field Values: This column's intended purpose is for reference only and is dynamically updated based off of the user's input from Section 2.2.1.

The workbook attempts to warn you if an input timing value is tighter than expected (based off of the JEDEC values populated for the memory device detailed in Section 2.2.1). In Figure 3, the timing parameters tWR and tREFI have been changed from that shown in Figure 2. However, a typical minimum write recovery time and average refresh interval for a DDR3 memory is 15 ns and 7.8 µs, respectively. The timing parameter tWR is flagged because the new tWR user input of 10 ns is less than a typical minimum value of 15 ns. In this case, the memory row may be pre-charged too early. The timing parameter tREFI is flagged because the new tREFI user input of 8 µs is larger than a typical average value of 7.8 µs. In this case, the memory may not be refreshed as often as necessary.

Parameter	Description	Datasheet Values tCK	Datasheet Values ns	Final Bit Field Values Value	Final Bit Field Values Units	JEDEC Bit Field Values (DDR3/L-1066 @ 532 MHz)
CAS Latency	Delay between internal READ command and data ready	7		7	tCK	7
CWL Latency	Delay between internal WRITE command and data ready	6		6	tCK	6
tRP	Precharge command period		13.125	6	tCK	6
tRCD	Active to read or write delay		13.125	6	tCK	6
tWR	Write recovery time		10	5	tCK	7
tRAS	Active to Precharge command period		37.5	19	tCK	19
tRC	Active to Active/Refresh command period		50.625	26	tCK	26
tRRD	Active Bank to Active Bank command period	4	10	6	tCK	6
tWTR	Internal Write to Read command delay	4	7.5	3	tCK	3
tXP	Exit power down mode to first valid command	3	7.5	3	tCK	3
tXSNR/tXS	Exit self refresh to commands not requiring a locked DLL	5	170	90	tCK	90
tXSRD/tXSDLL	Exit self refresh to commands requiring a locked DLL	512		511	tCK	511
tRTP	Internal Read to Precharge command delay	4	7.5	3	tCK	3
tCKE	CKE minimum pulse width	3	5.625	2	tCK	2
tCKESR	Minimum CKE low width for Self Refresh entry to exit	4		3	tCK	3
tZQCS	ZQ short calibration time	64		63	tCK	63
tRFC	Refresh to Active/Refresh command period		160	85	tCK	85
tRAS (max)	Active to Precharge command period (Max Value)		70200	7	tREFI intervals	8
tREFI	Average periodic refresh interval		8000	4256	tCK	4149
tFAW	Minimum Window for 4 Active Bank commands		50	See tRRD	-	6

Figure 3. DDR Timings (Warning)

Although the warnings may serve as a quick sanity check in the event that a timing parameter is accidentally input incorrectly, you should ultimately ensure the final bit field values comply with the timing values specified in their DDR data sheet.

2.2.4 Results – Register Values

After the system level, board level, and DDR timing details have been populated in their corresponding worksheets, you can access the fifth worksheet, “Register Values”, to obtain the calculated register settings based on the your input. The register values are output in u-boot format to allow for easy integration into a Linux environment. These values can also be utilized with the accompanying source files included in the EMIF Tools application.

In addition, a “Register Values (GEL)” worksheet is provided in the event that you wish to integrate the EMIF configuration values into a custom GEL file.

Note that the register values (GEL) and register values (u-boot) are output for a single EMIF channel in the results section. If utilizing multiple EMIF channels with the same DDR memory width and type, the same settings should be applied to the other EMIF channel.

In a scenario where multiple EMIF channels are used with different DDR memory widths and type - it is required to configure the tool per each EMIF channel to obtain the corresponding register settings.

2.2.5 Results – Saving/Loading Custom Configurations

As an additional feature of the Register Configuration workbook, you can save and load multiple custom configurations without having to create duplicate copies of the workbook.

From the 'README' worksheet, you can find a drop down box with existing configurations, as well as options to save or load a user configuration. This is illustrated in Figure 4.

Figure 4. User Defined Configurations

To save a custom user configuration, only click the 'Save User Config' push button. The configuration name is automatically generated based on your input, and is a combination of details 1, 2, 5, and 6 from worksheet discussed in Section 2.2.1. Invalid configurations may not successfully save, so it is important that you ensure that each worksheet requiring user input has been properly configured.

After the configuration has been successfully saved, the configuration name will appear from the drop down box, illustrated in Figure 5. Each time a unique configuration is saved, the EMIF Tools source code is automatically updated.

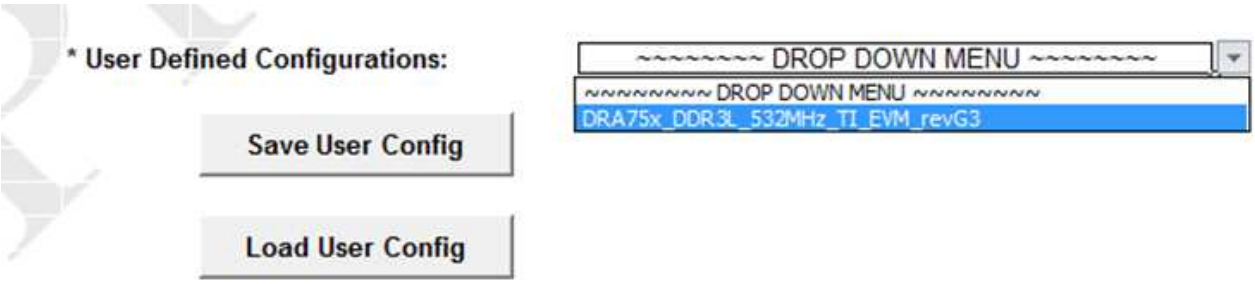

Figure 5. Saved User Configuration

NOTE: In addition to saving custom user configurations, you should also save the workbook!

If you create multiple configurations and later wish to recall the input from a previous saved configuration, you can select the configuration name from the drop down box and click the 'Load User Config' push button. This feature will only load your previous input to the Register Configuration workbook at the time in which the configuration was saved, and will NOT illustrate any changes that may have been made to the software manually.

3 Software Guidelines for EMIF Configuration

This section provides information on EMIF configuration of AM57x devices and guidelines on how to apply a specific configuration to SPL/u-boot source code. The procedure and code examples below are based on Processor SDK for AM57x Sitara Processors (v06.01.00.00), which uses the u-boot version 2019.01. The latest Processor SDK for AM57x can be obtained from the AM57x product page on www.ti.com. Ensure that you have downloaded the latest release and can successfully build u-boot.

3.1 Introduction

This scope of this section is to provide guidelines to configuration of the EMIF controller in SPL/U-boot for AM57x custom platform for DDR memory part.

- The section focuses on guidelines that must be performed in SPL/U-boot to configure the EMIF controller for DDR memory for a custom board.
- The section does not discuss how to obtain the timing parameters for the specific DDR part. This is covered in the previous sections of this document.

This section provides information on EMIF configuration of AM57x devices and guidelines on how to apply a specific configuration to SPL/u-boot source code. The procedure and code examples below are based on Processor SDK Linux for AM57x Sitara Processors (v06.01.00.00), which uses the u-boot version 2019.01. The latest Processor SDK for AM57x can be obtained from the AM57x product page on www.ti.com/processorsdk. Ensure that you have downloaded the latest release and can successfully build u-boot.

3.2 EMIF Configuration

The first step in properly configuring the AM57x EMIF controller and PHY for your custom board is to use the AM57x EMIF tool spreadsheet described earlier in this application note. The spreadsheet facilitates the calculation of various controller and PHY registers, based on the custom board design and DDR devices chosen. Follow all of the instructions in this document. When completed, the “Register Values (U-Boot)” tab contains the information needed in the next section.

NOTE: It is very important to enter the trace lengths in Section 2. Complete all steps or you may experience intermittent failures.

3.3 Updating U-boot Code

The next sections describe how to transfer the information in the “Register Values (U-Boot)” tab in the AM57x EMIF tools spreadsheet to SPL/U-boot code.

3.3.1 Device Type and Revision

In the subsequent sections, code changes must be made based on the device type and revision. To place the code changes properly, you must know the device identification. The function init_omap_revision() in arch/arm/mach-omap2/omap5/hwinit.c determines the device ID by reading the CTRL_WKUP_ID_CODE register. The following section in arch/arm/include/asm/arch-omap5/omap.h has a list of device IDs that correspond to the CTRL_WKUP_ID_CODE register in AM57x devices. Chapter 1 of the AM57x TRM has the valid device ID codes for your device.

```
#define OMAP5430_CONTROL_ID_CODE_ES1_0        0x0B94202F
#define OMAP5430_CONTROL_ID_CODE_ES2_0         0x1B94202F
#define OMAP5432_CONTROL_ID_CODE_ES1_0        0x0B99802F
#define OMAP5432_CONTROL_ID_CODE_ES2_0         0x1B99802F
#define DRA762_CONTROL_ID_CODE_ES1_0        0x0BB5002F
#define DRA752_CONTROL_ID_CODE_ES1_0        0x0B99002F
#define DRA752_CONTROL_ID_CODE_ES1_1        0x1B99002F
#define DRA752_CONTROL_ID_CODE_ES2_0        0x2B99002F
#define DRA722_CONTROL_ID_CODE_ES1_0        0x0B9BC02F
#define DRA722_CONTROL_ID_CODE_ES2_0        0x1B9BC02F
#define DRA722_CONTROL_ID_CODE_ES2_1        0x2B9BC02F
```

For example, if using an AM5728 SR2.0 device, look in the AM572x TRM in the ID_CODE Values table, where its ID code is 0x2B99002F. Checking init_omap_revision() function shows:

```
case DRA752_CONTROL_ID_CODE_ES2_0:
        *omap_si_rev = DRA752_ES2_0;
        printf("\n DRA752_CONTROL_ID_CODE_ES2_0\n");
        break;
```

Thus, DRA752_ES2_0 is used in other functions to distinguish the device and properly set its configuration.

3.3.2 PLL Parameters

The EMIF tool does not output any values for DDR PLL configuration. Instead, the PLLs are defined in arch/arm/mach-omap2/omap5/hw_data.c. This file contains several constant structure arrays defining different PLLs with different system clock frequencies. U-boot uses the structures listed in Table 9 by default, based on the device type.

Table 9. AM57x PLL Default Settings

Device	PLL Structure	DDR Speed
AM570x/AM571x	dra72x_dplls	DDR3-1333 (667 MHz)
AM572x	dra7xx_dplls	DDR3-1066 (532 MHz)
AM574x	dra76x_dplls	DDR3-1333 (667 Mhz)

If you do not need to change your DDR speed from these defaults, you do not need to make any modifications and can skip the rest of this section. If you are running the DDR PLL at a different frequency, the rest of this section describes how the DDR PLL is configured.

Specifically for DDR on AM57x, the two structures below define the frequency for DDR:

This is the structure that defines DDR3-1333 operation (666 MHz) for different system input clock frequencies:

```
static const struct dpll_params ddr_dpll_params_2664mhz[NUM_SYS_CLKS] = {
    {111, 0, 2, 1, 8, -1, -1, -1, -1, -1, -1, -1},        /* 12 MHz   */
    {333, 4, 2, 1, 8, -1, -1, -1, -1, -1, -1, -1},        /* 20 MHz   */
    {555, 6, 2, 1, 8, -1, -1, -1, -1, -1, -1, -1},        /* 16.8 MHz */
    {555, 7, 2, 1, 8, -1, -1, -1, -1, -1, -1, -1},        /* 19.2 MHz */
    {666, 12, 2, 1, 8, -1, -1, -1, -1, -1, -1, -1},    /* 26 MHz   */
    {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},    /* 27 MHz   */
    {555, 15, 2, 1, 8, -1, -1, -1, -1, -1, -1, -1},    /* 38.4 MHz */
};
```

This is the structure that defines DDR3-1066 operation (532 MHz) for different system input clock frequencies:

```
static const struct dpll_params ddr_dpll_params_2128mhz[NUM_SYS_CLKS] = {
    {266, 2, 2, 1, 8, -1, -1, -1, -1, -1, -1, -1},        /* 12 MHz   */
    {266, 4, 2, 1, 8, -1, -1, -1, -1, -1, -1, -1},        /* 20 MHz   */
    {190, 2, 2, 1, 8, -1, -1, -1, -1, -1, -1, -1},        /* 16.8 MHz */
    {665, 11, 2, 1, 8, -1, -1, -1, -1, -1, -1, -1},    /* 19.2 MHz */
    {532, 12, 2, 1, 8, -1, -1, -1, -1, -1, -1, -1},    /* 26 MHz   */
    {-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},    /* 27 MHz   */
    {665, 23, 2, 1, 8, -1, -1, -1, -1, -1, -1, -1},    /* 38.4 MHz */
};
```

These structures are used in the same file in the .ddr structure element (example below):

```
struct dplls dra7xx_dplls = {
    .mpu = mpu_dpll_params_1ghz,
    .core = core_dpll_params_2128mhz_dra7xx,
    .per = per_dpll_params_768mhz_dra7xx,
    .abe = abe_dpll_params_sysclk2_361267khz,
    .iva = iva_dpll_params_2330mhz_dra7xx,
    .usb = usb_dpll_params_1920mhz,
    .ddr = ddr_dpll_params_2128mhz,
    .gmac = gmac_dpll_params_2000mhz,
};
```

Which is used in the file to define the AM57xx board parameters board/ti/am57xx/board.c:

```
void hw_data_init(void)
{
    *prcm = &dra7xx_prcm;
    if (is_dra72x())
        *dplls_data = &dra72x_dplls;
    else if (is_dra76x())
        *dplls_data = &dra76x_dplls;
```

```
    else
        *dplls_data = &dra7xx_dplls;
    *ctrl = &dra7xx_ctrl;
}
```

Depending on your device ID, you can change the dplls data structure to the desired DDR frequency. For example, to run the AM574x at a slower speed for power reasons, for example DDR3-1066 (532MHz), change the pointer entry for .ddr in the dplls structure:

```
struct dplls dra76x_dplls = {
    .mpu = mpu_dpll_params_1ghz,
    .core = core_dpll_params_2128mhz_dra7xx,
    .per = per_dpll_params_768mhz_dra7xx,
    .abe = abe_dpll_params_sysclk2_361267khz,
    .iva = iva_dpll_params_2330mhz_dra7xx,
    .usb = usb_dpll_params_1920mhz,
    .ddr = ddr_dpll_params_2128mhz,
    .gmac = gmac_dpll_params_2000mhz,
};
```

3.3.3 I/O Settings

The EMIF tool outputs an I/O control register structure (example below):

```
const struct ctrl_ioregs AM572x_DDR3L_532MHz_TI_AM572x_EVM_ctrl_ioregs = {
    .ctrl_ddr3ch = 0x80808080,
    .ctrl_ddrch = 0x40404040,
    .ctrl_ddrio_0 = 0x00094A40,
    .ctrl_ddrio_1 = 0x04A52000,
    .ctrl_lpddr2ch = 0x00404000,
    .ctrl_emif_sdram_config_ext = 0x0001C123
};
```

If structure get_ioregs() in arch/arm/mach-omap2/omap5/hw_data.c does not have the “__weak” qualifier in your version of the SDK, then add this qualifier as shown in the following patch:

```
diff --git a/arch/arm/mach-omap2/omap5/hw_data.c b/arch/arm/mach-omap2/omap5/hw_data.c
index c4a41db92a..6eecc51b02 100644
--- a/arch/arm/mach-omap2/omap5/hw_data.c
+++ b/arch/arm/mach-omap2/omap5/hw_data.c
@@ -774,7 +774,7 @@ void __weak hw_data_init(void)
     }
 }

-void get_ioregs(const struct ctrl_ioregs **regs)
+void __weak get_ioregs(const struct ctrl_ioregs **regs)
 {
     u32 omap_rev = omap_revision();
```

The added qualifier allows the ioregs() function to be defined in board/ti/am57xx/board.c. Place the I/O Control Register structure in board/ti/am57xx/board.c, and create a get_ioregs() function. An example of the additions to board.c is below:

```
const struct ctrl_ioregs AM572x_DDR3L_532MHz_TI_AM572x_EVM_ctrl_ioregs = {
    .ctrl_ddr3ch = 0x80808080,
    .ctrl_ddrch = 0x40404040,
    .ctrl_ddrio_0 = 0x00094A40,
    .ctrl_ddrio_1 = 0x04A52000,
    .ctrl_lpddr2ch = 0x00404000,
    .ctrl_emif_sdram_config_ext = 0x0001C123
};

void get_ioregs(const struct ctrl_ioregs **regs)
{
    *regs = &AM572x_DDR3L_532MHz_TI_AM572x_EVM_ctrl_ioregs;
}
```

3.3.4 LISA Mapping

The EMIF tool outputs a series of registers to facilitate the mapping of the dynamic memory manager (example below):

```
const struct dmm_lisa_map_regs AM572x_DDR3L_532MHz_TI_AM572x_EVM_dmm_regs = {
    .dmm_lisa_map_0 = 0x00000000,
    .dmm_lisa_map_1 = 0x00000000,
    .dmm_lisa_map_2 = 0x80600100,
    .dmm_lisa_map_3 = 0xFF020100,
    .is_ma_present = 0x1
};
```

The structure should be added to board/ti/am57xx/board.c. There are similar structures towards the beginning of this file. This structure is used in emif_get_dmm_regs(). Ensure that *dmm_lisa_regs in this function gets set to the structure that you added (example given below):

```
void emif_get_dmm_regs(const struct dmm_lisa_map_regs **dmm_lisa_regs)
{
    if (board_is_am571x_idk())
        *dmm_lisa_regs = &am571x_idk_lisa_regs;
    else if (board_is_am574x_idk())
        *dmm_lisa_regs = &am574x_idk_lisa_regs;
    else
        *dmm_lisa_regs = &AM572x_DDR3L_532MHz_TI_AM572x_EVM_dmm_regs;
}
```

3.3.4.1 Adjustments to LISA Mapping

The EMIF tool shows a default LISA map configuration which may or may not be suitable for your application. Because of the many possibilities of LISA map configuration, you may need to manually adjust the LISA map registers.

Here are some tips to properly configure the LISA map registers (refer to the register description of the DMM_LISA_MAP_i registers in the TRM):

- When not using TILER, the LISA mapping associated with SDRC_ADDRSPC=2 can be removed. For example, the following LISA map register can be set to 0x0 or replaced by the previous LISA map entry.

```
.dmm_lisa_map_3 = 0xFF020100
```

- Ensure the size of the region in SYS_SIZE agrees with the size of the memory attached to the EMIF.
- If using ECC, see Section 3.3.7 for additional details on LISA mapping and ECC configuration registers.

3.3.4.2 LISA Mapping Examples

Single EMIF1 example with ECC, 2GB without TILER:

```
const struct dmm_lisa_map_regs Example_dmm_regs = {
    .dmm_lisa_map_0 = 0x00000000,
    .dmm_lisa_map_1 = 0x00000000,
    .dmm_lisa_map_2 = 0x00000000,
    .dmm_lisa_map_3 = 0x80700100,
    .is_ma_present = 0x1
};
```

Dual EMIF interleaved, 1GB each EMIF,no ECC, with TILER:

```
const struct dmm_lisa_map_regs Example_dmm_regs = {
    .dmm_lisa_map_0 = 0x00000000,
    .dmm_lisa_map_1 = 0x00000000,
    .dmm_lisa_map_2 = 0x80740300,
    .dmm_lisa_map_3 = 0xFF020100,
    .is_ma_present = 0x1
};
```

Dual EMIF interleaved, 512MB each EMIF, no ECC, without TILER:

```
const struct dmm_lisa_map_regs Example_dmm_regs = {
```

```
    .dmm_lisa_map_0 = 0x00000000,
    .dmm_lisa_map_1 = 0x00000000,
    .dmm_lisa_map_2 = 0x00000000,
    .dmm_lisa_map_3 = 0x80640300,
    .is_ma_present = 0x1
};
```

One EMIF1 (1GB)with ECC, EMIF2 (1GB) without ECC:

```
const struct dmm_lisa_map_regs Example_dmm_regs = {
    .dmm_lisa_map_0 = 0x00000000,
    .dmm_lisa_map_1 = 0x00000000,
    .dmm_lisa_map_2 = 0x80600000,
    .dmm_lisa_map_3 = 0xC0600000,
    .is_ma_present = 0x1
};
```

3.3.4.3 Supporting 4GB Memory

To support a total of 4GB interleaved memory (2GB each on EMIF1 and EMIF2), the LISA map would be:

```
const struct dmm_lisa_map_regs Example_dmm_regs = {
    .dmm_lisa_map_0 = 0x00000000,
    .dmm_lisa_map_1 = 0x00000000,
    .dmm_lisa_map_2 = 0x80740300,
    .dmm_lisa_map_3 = 0xFF020100,
    .is_ma_present = 0x1
};
```

Add the following lines to your defconfig file (include/configs/am57xx_evm.h):

```
#define CONFIG_VERY_BIG_RAM
#define CONFIG_MAX_MEM_MAPPED 0x80000000
CONFIG_PHYS_64BIT=y
```

Also include the following function in board/ti/am57xx/board.c:

```
int dram_init_banksize(void)
{
u64 ram_size;
ram_size = board_ti_get_emif_size();
gd->bd->bi_dram[0].start = CONFIG_SYS_SDRAM_BASE;
gd->bd->bi_dram[0].size = get_effective_memsize();
if (ram_size > CONFIG_MAX_MEM_MAPPED) {
gd->bd->bi_dram[1].start = 0x200000000;
gd->bd->bi_dram[1].size = ram_size - CONFIG_MAX_MEM_MAPPED;
}
return 0;
}
```

3.3.5 EMIF Registers

The EMIF tool outputs a structure which contains the EMIF configuration (example below):

```
const struct emif_regs AM572x_DDR3L_532MHz_TI_AM572x_EVM_emif_regs = {
    .sdram_config_init = 0x61862B32,
    .sdram_config = 0x61862B32,
    .sdram_config2 = 0x00000000,
    .ref_ctrl = 0x000040F1,
    .ref_ctrl_final = 0x00001035,
    .sdram_tim1 = 0xCEEF2663,
    .sdram_tim2 = 0x308F7FDA,
    .sdram_tim3 = 0x409F88A8,
    .read_idle_ctrl = 0x00050000,
    .zq_config = 0x5007190B,
    .temp_alert_config = 0x00000000,
    .emif_rd_wr_lvl_rmp_ctl = 0x80000000,
    .emif_rd_wr_lvl_ctl = 0x00000000,
    .emif_ddr_phy_ctlr_1_init = 0x0024400D,
    .emif_ddr_phy_ctlr_1 = 0x0E24400D,
```

```
    .emif_rd_wr_exec_thresh = 0x00000305,

    .emif_ecc_ctrl_reg = 0xC0000001,
    .emif_ecc_address_range_1 = 0x3FFF0000,
    .emif_ecc_address_range_2 = 0x00000000,

};
```

This structure should be added to board/ti/am57xx/board.c. There are similar structures in this file which describe other boards. This structure should be used from emif_get_reg_dump(). This function sets the registers for each of the EMIFs, so depending on how many EMIFs you selected in the EMIF tools spreadsheet, you must put an entry for each EMIF. The example below shows how to set the registers if using two EMIFs with the same configuration.

```
void emif_get_reg_dump(u32 emif_nr, const struct emif_regs **regs)
{
    switch (emif_nr) {
    case 1:
        if (board_is_am571x_idk())
            *regs = &am571x_emif1_ddr3_666mhz_emif_regs;
        else if (board_is_am574x_idk())
            *regs = &am574x_emif1_ddr3_666mhz_emif_ecc_regs;
        else
            *regs = &AM572x_DDR3L_532MHz_TI_AM572x_EVM_emif_regs;
        break;
    case 2:
        if (board_is_am574x_idk())
            *regs = &am571x_emif1_ddr3_666mhz_emif_regs;
        else
            *regs = &AM572x_DDR3L_532MHz_TI_AM572x_EVM_emif_regs;
        break;
    }
}
```

If using 2 EMIFs with the same devices on each (that is, same timings and configuration), you can use the same structure for both cases of the switch statement. If you have different devices, you must input the information in two instances of the EMIF tool to generate the proper register structure for each EMIF.

3.3.6 PHY Control Registers

The EMIF tool outputs a structure to configure the external PHY control registers (example below) for each EMIF.

```
const unsigned int AM572x_DDR3L_532MHz_TI_AM572x_EVM_emif1_ext_phy_regs [] = {
    0x04040100,    // EMIF1_EXT_PHY_CTRL_1
    0x006B006B,    // EMIF1_EXT_PHY_CTRL_2
    0x006B006B,    // EMIF1_EXT_PHY_CTRL_3
    0x006B006B,    // EMIF1_EXT_PHY_CTRL_4
    0x006B006B,    // EMIF1_EXT_PHY_CTRL_5
    0x006B006B,    // EMIF1_EXT_PHY_CTRL_6
    0x00320032,    // EMIF1_EXT_PHY_CTRL_7
    0x00320032,    // EMIF1_EXT_PHY_CTRL_8
    0x00320032,    // EMIF1_EXT_PHY_CTRL_9
    0x00320032,    // EMIF1_EXT_PHY_CTRL_10
    0x00320032,    // EMIF1_EXT_PHY_CTRL_11
    0x00600060,    // EMIF1_EXT_PHY_CTRL_12
    0x00600060,    // EMIF1_EXT_PHY_CTRL_13
    0x00600060,    // EMIF1_EXT_PHY_CTRL_14
    0x00600060,    // EMIF1_EXT_PHY_CTRL_15
    0x00600060,    // EMIF1_EXT_PHY_CTRL_16
    0x00400040,    // EMIF1_EXT_PHY_CTRL_17
    0x00400040,    // EMIF1_EXT_PHY_CTRL_18
    0x00400040,    // EMIF1_EXT_PHY_CTRL_19
    0x00400040,    // EMIF1_EXT_PHY_CTRL_20
    0x00400040,    // EMIF1_EXT_PHY_CTRL_21
    0x00800080,    // EMIF1_EXT_PHY_CTRL_22
```

```
    0x00800080,    // EMIF1_EXT_PHY_CTRL_23
    0x40010080,    // EMIF1_EXT_PHY_CTRL_24
    0x08102040,    // EMIF1_EXT_PHY_CTRL_25
    0x00000000,    // EMIF1_EXT_PHY_CTRL_26
    0x00000000,    // EMIF1_EXT_PHY_CTRL_27
    0x00000000,    // EMIF1_EXT_PHY_CTRL_28
    0x00000000,    // EMIF1_EXT_PHY_CTRL_29
    0x00000000,    // EMIF1_EXT_PHY_CTRL_30
    0x00000000,    // EMIF1_EXT_PHY_CTRL_31
    0x00000000,    // EMIF1_EXT_PHY_CTRL_32
    0x00000000,    // EMIF1_EXT_PHY_CTRL_33
    0x00000000,    // EMIF1_EXT_PHY_CTRL_34
    0x00000000,    // EMIF1_EXT_PHY_CTRL_35
    0x00000077    // EMIF1_EXT_PHY_CTRL_36
};
```

This structure should be placed in board/ti/am57xx/board.c and is used in function emif_get_ext_phy_ctrl_const_regs() in the same file. There is a case statement to define different structure for each EMIF. Continuing the previous example of an AM5728 device, point to the structure in the case statement for EMIF1 as shown below:

```
void emif_get_ext_phy_ctrl_const_regs(u32 emif_nr, const u32 **regs, u32 *size)
{
switch (emif_nr) {
    case 1:
        *regs = AM572x_DDR3L_532MHz_TI_AM572x_EVM_emif1_ext_phy_regs;
        *size = ARRAY_SIZE(AM572x_DDR3L_532MHz_TI_AM572x_EVM_emif1_ext_phy_regs);
        break;
    case 2:
        *regs = AM572x_DDR3L_532MHz_TI_AM572x_EVM_emif2_ext_phy_regs;
        *size = ARRAY_SIZE(AM572x_DDR3L_532MHz_TI_AM572x_EVM_emif2_ext_phy_regs);
        break;
}
}
```

If you have enabled two EMIFs, the tool outputs 2 different structures for each EMIF, which must be used in each of the case statements above.

3.3.7 ECC Configuration

If you are enabling ECC in your design, you must properly configure this in your code. Not all AM57x devices support ECC on DDR. Refer to your device specific data sheet to determine if your device supports ECC. The sections below describe the add configuration for ECC-enabled designs.

3.3.7.1 LISA Mapping for ECC

The LISA Mapping output from the EMIF tool spreadsheet configures the LISA map properly when ECC is set to enabled in the tool.

Interleaving is not supported when ECC is used. Thus, DMM_LISA_MAP_i.SDRC_MAP should be 1 or 2 (for EMIF1 or EMIF2, respectively), and DMM_LISA_MAP_i.SDRC_INTL=0. If only using EMIF1, the EMIF tool generally defines just one dmm_lisa_map_x register for the whole EMIF1 region. If using both EMIFs (EMIF1 with ECC, and EMIF2 without ECC), the EMIF tool defines two regions for each EMIF. An example for two 1GB EMIFs without TILER is below:

```
const struct dmm_lisa_map_regs AM574x_dmm_regs = {
    .dmm_lisa_map_0 = 0x00000000,
    .dmm_lisa_map_1 = 0x00000000,
    .dmm_lisa_map_2 = 0x80600100,
    .dmm_lisa_map_3 = 0xC0600200,
    .is_ma_present = 0x1
};
```

3.3.7.2 ECC Register Configuration

The EMIF tool spreadsheet configures 3 registers for ECC: emif_ecc_ctrl_reg, emif_ecc_address_range_1, and emif_ecc_address_range_2. Emif_ecc_ctrl_reg sets various bits for the ECC controller. The ECC address range registers is based on your inputs into the spreadsheet in the System Details tab. If you are enabling ECC on the full 2GB address range of EMIF1, you must define two 1 GB ranges as below:

11	ECC Region 1: System Start Address	80000000	Enabled
12	ECC Region 1: System End Address	BFFFFFFF	Enabled
13	ECC Region 2: System Start Address	C0000000	Enabled
14	ECC Region 2: System End Address	FFFFFFFF	Enabled

Otherwise, any configuration less than 2GB for ECC should be configured with one ECC region:

11	ECC Region 1: System Start Address	80000000	Enabled
12	ECC Region 1: System End Address	BFFFFFFF	Enabled
13	ECC Region 2: System Start Address	80000000	Disabled
14	ECC Region 2: System End Address	80000000	Disabled

Below is an example of the configuration output from the EMIF tool:

```
.emif_ecc_ctrl_reg = 0xD0000001,
     .emif_ecc_address_range_1 = 0x3FFF0000,
     .emif_ecc_address_range_2 = 0x00000000,
```

3.3.7.3 ECC Testing

When the configuration is correct, SBL code automatically primes the memory to get the processor ready for operation with ECC enabled. The SDK u-boot for AM57x includes a u-boot command line utility to test the ECC when configured. This is an example of its usage:

```
=> ddr ecc_err 0x80000000 0x1
Testing DDR ECC:
        ECC test: Disabling DDR ECC ...
        ECC test: addr 0x80000000, read data 0x0, written data 0x1, err pattern: 0x1, read after
wri1
        ECC test: Enabling DDR ECC ...
        ECC test: addr 0x80000000, read data 0x0
ECC test Status:
        ECC test: 1-bit ECC err count: 0x1
=>
```

In this example, the “ecc_err” test disables ECC, goes to memory location 0x80000000, and changes the value from 0x0 to 0x1. Next, ECC is enabled and the controller fixes the bit flip, increasing err_count by one.

Consult your u-boot documentation for more information.

3.4 Debug Resources

Here are some more debug resources:

- DSS scripts: https://git.ti.com/cgit/sitara-dss-files/am57xx-dss-files/tree/ The link points to some Debug Server Scripts that can be run in Code Composer to help debug DDR configuration issues. See the README file at this location for instructions on how to run the DSS scripts.
- http://e2e.ti.com: Public forum to ask questions to TI support engineers.

4 References

DDR3 SDRAM Standard, JESD79-3F, 2012 (http://www.jedec.org)

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from D Revision (November 2019) to E Revision	Page
• Added Software Guidelines for EMIF Configuration section.	10

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

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