



Technical notes on using Analog Devices products and development tools  
 Visit our Web resources <http://www.analog.com/ee-notes> and <http://www.analog.com/processors> or  
 e-mail [processor.support@analog.com](mailto:processor.support@analog.com) or [processor.tools.support@analog.com](mailto:processor.tools.support@analog.com) for technical support.

## Interfacing DDR3/DDR2/LPDDR Memory to ADSP-SC5xx/ADSP-215xx Processors

Contributed by Balaji Kannan, Mitesh Moonat, Gaurav Singh, and Sachin-V Kumar

Rev 12 – July 7, 2016

### Introduction

The ADSP-SC5xx/215xx family of processors (hereafter referred to collectively as “ADSP-SC5xx processors”) incorporate a Dynamic Memory Controller (DMC), which provides a glue-less interface between off-chip DDR3/DDR2/LPDDR (hereafter referred to as “DDR”) memory devices and the rest of the processor infrastructure. The block diagram in [Figure 1](#) shows at a high level how the DMC module connects various system bus masters (cores, peripheral DMAs, MDMA, hardware accelerators, etc.) to the DDR memory device via the system fabric. For further technical details on the DMC module, please refer to the appropriate *processor hardware reference manual*<sup>[2][13]</sup> and *datasheet*<sup>[1][14]</sup>.



Not all of the ADSP-SC5xx/ADSP-215xx processors support two DMC blocks. For more details on the exact number of DMC blocks supported on a particular part, please refer to the corresponding *datasheet*<sup>[1][14]</sup>.

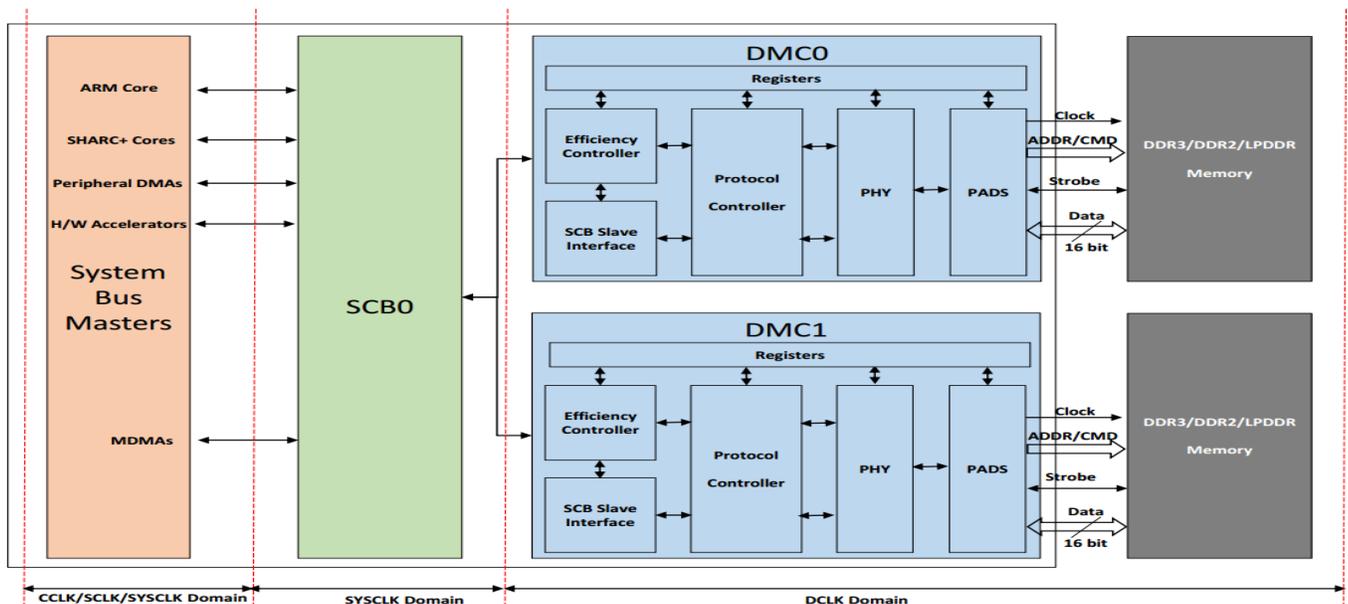


Figure 1 DMC Interface Block Diagram

This EE-note discusses some of the important software and hardware guidelines that need to be followed when interfacing ADSP-SC5xx processors with DDR memory devices and provides code examples that can be used for basic DMC initialization and for a few specific application scenarios. The code examples also

include a subroutine which can be used to validate the DMC interface for different types of accesses (e.g., core, DMA, 8-/16-/32-/64-bit) and different data patterns (e.g., all 0x0, all 0xF, all 0x5, all 0xA, incremental, random, and all bits toggling).

## DDR/LPDDR/DDR2/DDR3 Feature Comparison

Since ADSP-SC5xx processors support three DDR standards (LPDDR, DDR2, and DDR3), it is important to understand the major differences among them (Table 1). Even though DDR itself is not supported on ADSP-SC5xx processors, it is worthwhile to include in the overall comparison. Along with these details, the features not supported on ADSP-SC5xx processors for DDR are shown in red.

Item	DDR	LPDDR	DDR2	DDR3
Data rate/pin CLK frequency	200/266/333/400 MT/s (100/133/166/200 MHz)	200/266/333/360/400 MT/s (100/133/166/180/200 MHz)	400/533/667/800 MT/s (200/266/333/400 MHz)	800/1066/1333/1600 MT/s (400/533/667/800 MHz) *1
Power supply (VDD/VDDQ)	2.5 V	1.8 V	1.8 V	1.5 V
Interface	SSTL_2	LV-CMOS	SSTL_18	SSTL_15
Number of banks	4	4	4 or 8	8
Prefetch	2 bits	2 bits	4 bits	8 bits
Burst length	2/4/8	2, 4 or 8 (16 is optional)	4/8	4 (burst chop)/8
Posted CAS, Additive latency	No	No	Yes (AL = 0/1/2/3/4/5)	Yes (AL = 0/CL - 1/CL - 2)
RL, WL	RL = CL (no AL), WL = 1	RL = CL (no AL), WL = 1	RL = AL + CL, WL = RL - 1 = AL + CL - 1	RL = AL + CL, WL = AL + CWL
ZQ pin	NA	NA	NA	Available. For ZQ calibration*2
/Reset pin	NA	NA	NA	Available*4
Driver impedance (Ron)	Programmable	Programmable	Programmable	Programmable
Driver impedance calibration	NA	NA	For OCD calibration*3,6	For ZQ calibration*2
ODT function	NA	NA	Available	Available
ODT calibration	NA	NA	NA*6	For ZQ calibration*2
CLK-DQS de-skew mechanism	NA	NA	NA	Available (Write leveling, Read leveling) *5
Package	TSOP II	FBGA	FBGA	FBGA

Table 1. DDR/LPDDR/DDR2/DDR3 Feature Comparison

1. With clock frequency of 450 MHz, ADSP-SC5xx processors support data rate only up to 900 MT/s (Mega-Transfers/second).

2. ZQ Calibration (memory end): Calibrates DRAM ODT and Ron fluctuations with PVT (process, voltage, and temperature). External resistor (240Ω±1%) is inserted between DRAM ZQ pin and GND for reference. To perform ZQ calibration, ZQCL or ZQCS command is used. (This is a self-calibration in which DDR3 performs all the measurement and adjustment automatically.)

3. OCD (Off Chip Driver Calibration): Calibrates DRAM Ron fluctuation with PVT. The external device connected to DRAM performs impedance measurement and adjustment (not self-calibration). OCD is an optional feature in DDR2. This feature is not supported on ADSP-SC5xx processors.

4. /RESET pin is introduced in DDR3 for system stability. /RESET is an active-low signal.

5. DDR3 DIMM uses fly-by topology for CMD/ADD/CLK signals to improve signal quality, causing flight time differences between DQ/DM/DQS and CMD/ADD/CLK. DDR3 de-skew mechanism to compensate for this is not supported on ADSP-SC5xx processors, which supports only a single DRAM chip.

6. Driver Impedance/ODT calibration (processor end): Similar to ZQ calibration on the memory end, ADSP-SC5xx processors support driver impedance and ODT calibration for DDR3/DDR2 with an external resistor DMC\_RZQ (34 ohms). See "Driver Impedance and ODT Configuration" for more details.

## Software Considerations – DMC Programming Model

Figure 2 shows the programming steps which need to be followed when initializing the DMC module for various DDR modes.

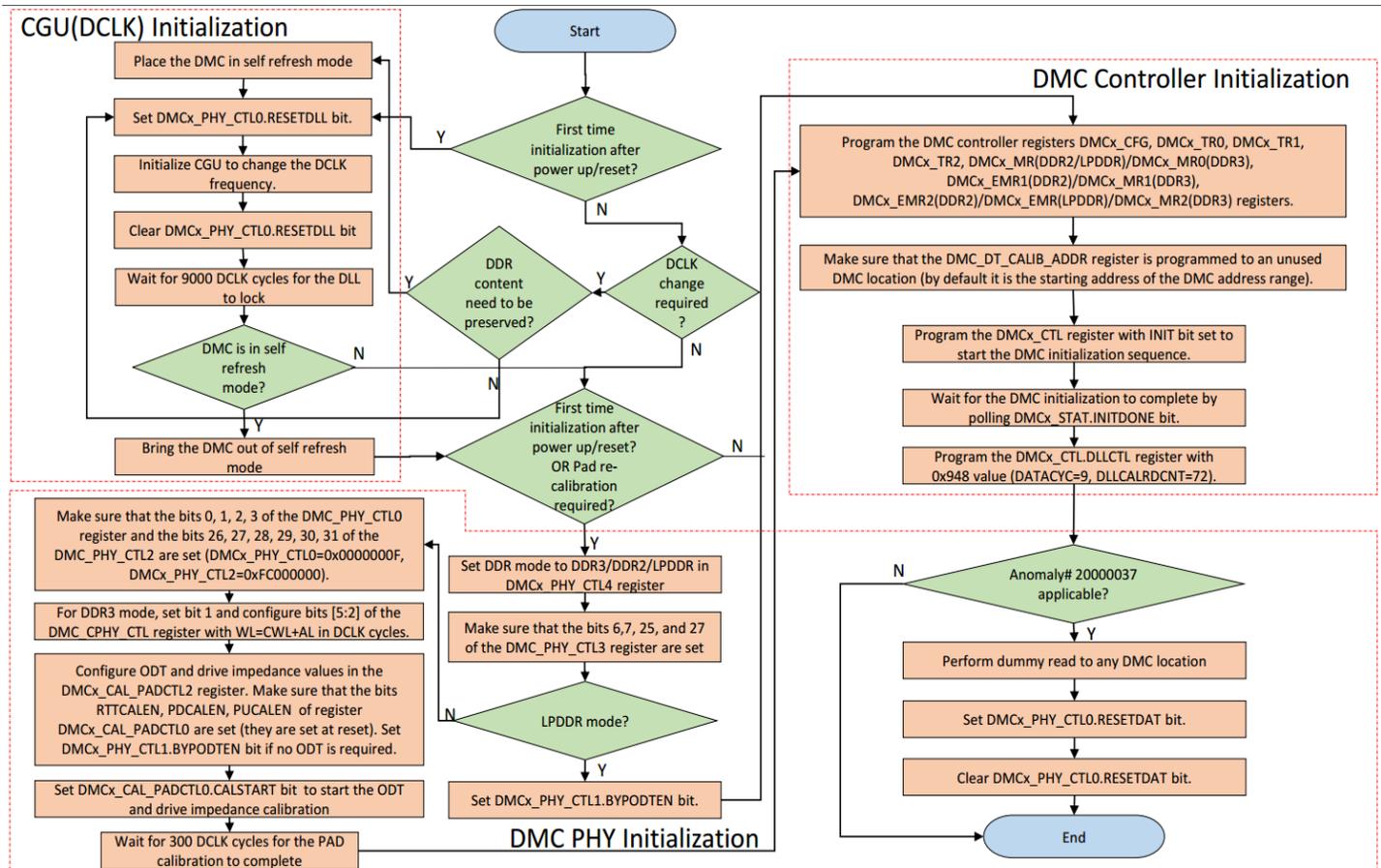


Figure 2. DMC Programming Model Flow Chart

As can be seen, DMC initialization consists of three major steps:

1. Clock Generation Unit (CGU) Initialization
2. DMC PHY Initialization
3. DMC Controller Initialization

### CGU Initialization

The first step of DMC initialization is to make sure that the DDR Clock (DCLK) is set to the required frequency. On ADSP-SC5xx processors, DCLK clocks both DMC0 and DMC1; however, DCLK can come from either CGU0 (default) or CGU1 by programming the Clock Distribution Unit (CDU). Routing DCLK from CGU1 is useful for cases where the required DCLK frequency is asynchronous to the CCLK and SYSCLK frequencies. For example, assume a case where the required CCLK frequency is 450 MHz, the SYSCLK frequency is 225 MHz, and the DCLK frequency is 400 MHz. Achieving this frequency combination may not be possible with a single CGU. One way to realize this configuration is to generate

CCLK and SYSCLK using CGU0 and DCLK using CGU1. For more details on how to program the CGU and CDU, please refer to the hardware reference manual<sup>[2][13]</sup>, and keep these important notes in mind:

- Ensure that the DCLK frequency does not violate the unique maximum and minimum JEDEC-specified frequencies for all three DDR modes (DDR3/DDR2/LPDDR). For example, the minimum DCLK frequency for DDR2 is 125 MHz.
- Additional latencies may be added by the fabric for an asynchronous SYSCLK::DCLK ratio.
- The SYSCLK to DCLK clock domain crossing may need to be reprogrammed based on the SYSCLK::DCLK ratio. Please refer to the System Crossbars (SCB) chapter of the hardware reference manual<sup>[2][13]</sup> for more details.

#### *DMC Initialization after System Reset*

After reset, the DCLK generated from CGU0 might be set to the default frequency, which will require that the CGU be reinitialized to set the DCLK to the required new frequency. As shown in [Figure 2](#), the following steps need to be followed when initializing the CGU to generate the required DCLK for the first time after reset:

1. Set the `DMCx_PHY_CTL0.RESETDLL` bit.
2. Initialize the CGU to change the DCLK frequency.
3. Clear the `DMCx_PHY_CTL0.RESETDLL` bit.
4. Wait 9000 DCLK cycles for the DLL to lock. Polling status bits is not necessary, a simple delay loop suffices.



Typically, the CGU is first initialized in either “*preload code*” (when the application is loaded via emulator) or by “*initialization code*” (when the application is loaded by the boot process, in the init block), which may need to be modified to meet system requirements. Refer to the [Modifying Default Preload and Initialization Code for Customized CGU/DMC Settings](#) section for more details.

#### *On-the-Fly DMC Re-Initialization*

If the DCLK frequency is not being changed as part of the re-initialization process, no CGU re-initialization is necessary.

If the DCLK frequency is being changed as part of the re-initialization process, but the DDR content needn't be preserved, use the same steps as described in [DMC Initialization after System Reset](#) to re-initialize the CGU.

However, if the DCLK frequency is being changed *and* code/data already resident in DDR memory must be preserved, these steps must be followed when re-initializing the CGU:

1. If anomaly 20-00-0067 applies, ensure that bit 22 of the `DMCx_PHY_CTL1` register is set. See the anomaly lists<sup>[5][15]</sup> for details.
2. Ensure that the DMC is in the idle state by waiting for the `DMCx_STAT.IDLE` bit to be set.
3. Place the DMC into self-refresh mode by setting the `DMCx_CTL.SRREQ` bit.
4. Poll for the setting of the `DMCx_STAT.SRREQ` bit to wait for the self-refresh mode transition to complete.

5. Set the `DMCx_PHY_CTL0.RESETDLL` bit.
6. Initialize the CGU to change the DCLK frequency.
7. Clear the `DMCx_PHY_CTL0.RESETDLL` bit.
8. Wait 9000 DCLK cycles for the DLL to lock.
9. Bring the DMC out of self-refresh mode by clearing the `DMCx_CTL.SRREQ` bit.
10. Poll for the clearing of the `DMCx_STAT.SRREQ` bit to wait for the self-refresh exit to complete.



When re-initializing the DMC, make sure that the CGU/DMC initialization code is not executed from the DDR memory corresponding to the same DMC. For more details, refer to [Question 2](#) in the [Frequently Asked Questions](#) section of this EE-note.

### DMC PHY Initialization

The DMC PHY provides the interface between the DMC controller and the DMC pads. There are a few registers in the DMC PHY which need to be programmed based on the DDR mode. As shown in [Figure 2](#), the PHY programming includes the following steps:

1. Set the appropriate DDR mode (DDR3/DDR2/LPDDR) in the `DMCx_PHY_CTL4` register.
2. Make sure that bits 6, 7, 25, and 27 of the `DMC_PHY_CTL3` register are set.
3. For DDR2/DDR3 mode, make sure that the lower four bits of the `DMC_PHY_CTL0` register (`DMCx_PHY_CTL0=0x0000000F`) and the upper six bits of the `DMC_PHY_CTL2` register (`DMCx_PHY_CTL2=0xFC000000`) are set.
4. For DDR3 mode, write the `DMC_CPHY_CTL` register with bit 1 set and the `DMC_CPHY_CTL[5:2]` bits configured such that  $WL = CWL + AL$  (in DCLK cycles).
5. Configure the Driver Impedance and On-Die Termination (ODT), as discussed in the [Driver Impedance and On-Die Termination \(ODT\) Configuration](#) section of this EE-Note.



DMC PHY programming typically needs to be done only once after power-up or reset, unless driver impedance and/or ODT recalibration is required.

#### *Driver Impedance and On-Die Termination (ODT) Configuration*

Driver impedance (strength) and on-die termination (ODT) apply to both the memory side and the processor side of the interface, each with its own considerations.

#### **Memory Side Driver Impedance and ODT**

Driver impedance is programmable for all DDR modes:

- LPDDR: `DMC_EMR`
- DDR2: `DMC_EMR1`
- DDR3: `DMC_MR1`

However, drive impedance calibration is supported only for DDR3 mode.

ODT is programmable only for DDR2 and DDR3 modes:

- DDR2: DMC\_EMR1
- DDR3: DMC\_MR1

However, ODT calibration is supported only for DDR3 mode. For more details, please refer to the corresponding DDR memory device data sheet.

Both the driver impedance and ODT for the memory can be programmed as part of the DMC controller initialization by programming the corresponding mode registers accordingly.



In DDR3 mode, during DMC initialization, the DDR3 controller sends a *ZQ Cal Long* command (coarse calibration) to the memory device. After every self-refresh exit command, the controller also issues a *ZQ Cal Long* command to the device. A *ZQ Cal Short* command for fine calibration must be issued by the application (e.g., a GP timer) in software. For more details, refer to the hardware reference manual <sup>[2]</sup>.

### **Processor Side Driver Impedance and ODT**

Driver impedance is programmable only for the DDR2 and DDR3 modes, and driver impedance calibration is supported for both. The DMC pads can be auto-calibrated to the required driver impedance using an external resistance DMC\_RZQ by programming the DMC<sub>x</sub>\_CAL\_PADCTL2 register:

- The DMC<sub>x</sub>\_CAL\_PADCTL2.IMPWRAD bit field for the address (DMC\_A[nn], DMC\_BA[n]) and command (DMC\_CKE, DMC\_CS[n], DMC\_ODT, DMC\_RAS, DMC\_RESET, DMC\_WE) pads should be programmed to the value for 60 Ω (0x3C) for both DDR2 and DDR3.
- The DMC<sub>x</sub>\_CAL\_PADCTL2.IMPWRDQ bit field for the data (DMC\_DQ[nn]), DQS (DMC\_LDQS, /DMC\_LDQS, DMC\_UDQS, /DMC\_UDQS), clock (DMC\_CK, /DMC\_CK), and DM (DMC\_UDM, DMC\_LDM) pads should be programmed to the value for 40 Ω (0x28) for both DDR2 and DDR3.

ODT is programmable only for the DDR2 and DDR3 modes, and ODT calibration is supported for both. The DMC PHY can be programmed to auto-calibrate the ODT value (parallel termination) on the data and DQS pads by programming the DMC<sub>x</sub>\_CAL\_PADCTL2.IMPRTT bit field to 1.6 times the required ODT value in Ohms. For example, for 75 Ω termination, the DMC<sub>x</sub>\_CAL\_PADCTL2.IMPRTT should be programmed to 1.6 \* 75 = 120 (0x78). ODT programming can optionally be bypassed by setting the DMC<sub>x</sub>\_PHY\_CTL1.BYPODTEN bit.

The following steps must be followed for ODT and driver impedance calibration on the ADSP-SC5xx processor:

1. Configure ODT and drive impedance values in the DMC<sub>x</sub>\_CAL\_PADCTL2 register, as discussed above. Ensure that the DMC<sub>x</sub>\_CAL\_PADCTL0 register bits RTTCALEN, PDCALEN, and PUCALEN are set (they are set at reset).
2. Set the DMC<sub>x</sub>\_CAL\_PADCTL0.CALSTART bit to start the ODT and drive impedance calibration.
3. Wait for 300 DCLK cycles for the pad driver impedance and ODT calibration to complete. Polling status bits is unnecessary, as a simple delay loop suffices.

## DMC Controller Initialization

The final step for the DMC initialization is the DMC controller initialization. The controller has a set of registers which has various bit fields that can be categorized as follows:

- Hard-Wired Settings
- Mandatory Settings
- Optional Settings

### Hard-Wired Settings

There are a few bits which are hard-coded in the DDR controller that software cannot adjust. [Table 2](#) through [Table 7](#) shade these bit fields in **ORANGE**.

### Mandatory Settings

Many bits in the configuration, timing, and mode registers need to be programmed based on the system to ensure proper DMC operation in the application. [Table 2](#) through [Table 7](#) shade these bit fields in **GREEN**. For more details on how to program these bit fields, please refer to the hardware reference manual <sup>[2]</sup>.

### Optional Settings

There are a few bit fields which are not required to be modified for standard DMC operation; however, deeper knowledge of these bits helps to save power and improve throughput in certain application configurations. For example, the `DMC_CTL.SRREQ` bit can be used to operate the DMC in a low-power (self-refresh) mode, and the `DMC_EMR2.PASR` bit (in LPDDR only) may help for intelligent power savings. Bits such as the `DMC_CTL.PREC` bit can help to enable automatic precharge after each access, and the `DMC_CTL.ADDRMODE` bit can help to improve throughput by switching between page and bank interleaving addressing modes. Users are expected to understand the functionality of these bits clearly by going through the hardware reference manual <sup>[2][13]</sup> and the corresponding memory device data sheet (especially for mode registers). These bits are shaded in **RED** in [Table 2](#) through [Table 7](#).

Referring back to [Figure 2](#), the following steps should be followed to initialize the DMC controller:

1. Program the DMC configuration register (`DMCx_CFG`), as described in [Table 2](#).

Register	Bit Field		Bit field (C=Controller, M=JEDEC)	Value	Comment
DMC_CFG	<b>IFWID</b>	Interface Width	3:0-C	<b>Mandatory</b>	Always program to <b>2</b> (16-bit). All other values are reserved.
	<b>SDRWID</b>	SDRAM Width	7:4-C		Always program to <b>2</b> (16-bit). All other values are reserved.
	<b>SDRSIZE</b>	SDRAM Size	11:8-C		Obtain from memory device data sheet.
	<b>EXTBANK</b>	External Banks	15:12-C		Always program to <b>zero</b> (16-bit). All other values are reserved.
	<b>Reserved</b>	Reserved	31:16-C		Always write these bits with zero.

Table 2. DMC Controller Configuration Register Bit Fields

2. Program the DMC timing registers (DMC<sub>x</sub>\_TR0, DMC<sub>x</sub>\_TR1, and DMC<sub>x</sub>\_TR2), as described in [Table 3](#).

Register	Bit Field		Bit field (C=Controller, M=JEDEC)	Value	Comment		
DMC_TR0	TRCD	RAS# to CAS# delay time	3:0-C	<b>Mandatory</b>	Obtain from memory device data sheet.		
	TWTR	Write-to-Read delay	7:4-C				
	TRP	Precharge-to-Active time	11:8-C				
	TRAS	Active-to-Precharge time	16:12-C				
	Reserved	Reserved	19:17-C			Always write these bits with <b>zero</b> .	
	TRC	Active-to-Active time	25:20-C			Obtain from memory device data sheet.	
	Reserved	Reserved	27:26-C			Always write these bits with <b>zero</b> .	
	TMRD	Mode register set-to-active	31:28-C		Obtain from memory device data sheet.		
DMC_TR1	TREF	Refresh Interval	13:0-C		<b>Mandatory</b>	Obtain from memory device data sheet.	
	Reserved	Reserved	15:14-C				Always write these bits with <b>zero</b> .
	TRFC	Refresh-to-Active command delay	23:16-C				Obtain from memory device data sheet.
	Reserved	Reserved	27:24-C				Always write these bits with <b>zero</b> .
	TRRD	Active-to-Active time	30:28-C				Obtain from memory device data sheet.
	Reserved	Reserved	31				Always write this bit with <b>zero</b> .
DMC_TR2	TFAW	Four Activate Window	4:0-C	<b>Mandatory</b>		Obtain from memory device data sheet. tFAW is not applicable for LPDDR mode and should be kept <b>zero</b> .	
	Reserved	Reserved	7:5-C				Always write these bits with <b>zero</b> .
	TRTP	Internal Read to Precharge time	11:8-C				Obtain from memory device data sheet. tRTP is not applicable for LPDDR mode and should be kept <b>zero</b> .
	TWR (LPDDR only)	Write recovery time	15:12-C				
	TXP	Exit power down to next valid command	19:16-C				Obtain from memory device data sheet.
	tCKE	CKE min pulse width	23:20-C				
	Reserved	Reserved	31:24-C				Always write these bits with <b>zero</b> .

Table 3. DMC Controller Timing Register Bit Fields

3. Program the DMC controller mode registers, depending on the mode being used:

- a. DDR2: Program the DMC<sub>x</sub>\_MR, DMC<sub>x</sub>\_EMR1, and DMC<sub>x</sub>\_EMR2 registers ([Table 4](#)).

Register	Bit Field		Bit field (C=Controller, M=JEDEC)	Value	Comment
DMC_MR	BL	Burst Length	2:0-C, A2:A0-M	Mandatory	Program burst length to 4/8, as per requirements. Refer to the hardware reference manual <sup>[2][13]</sup> for more details.
	BT	Burst Type	3-C, A3-M		Only sequential burst type is supported. Keep this bit <b>zero</b> .
	CL	CAS Latency	6:4-C, A6:A4-M		Program these bits with the required CAS latency.
	TM	Test Mode	7-C, A7-M		Always program this bit to <b>zero</b> , as test mode is not supported.
	DLLRST	DLL Reset	8-C, A8-M		Clear this bit for DDR2 mode.
	WRRECOV	Write recovery	11:9-C, A11:A9-M		Program these bits with tWR value from the memory device data sheet. Refer to the hardware reference manual <sup>[2]</sup> for more details.
	PD	Active Power Down Mode	12-C, A12-M	Optional	Can be left unchanged for standard DMC operation. Refer to the hardware reference manual <sup>[2][13]</sup> for more details.
	Reserved	Reserved	15:13-C, A15:A13-M	Hard Wired	These bits are hard-wired to <b>zero</b> .
	Reserved	Reserved	31:16-C		
DMC_EMR1	DLEN	DLL Enable	0-C, A0-M	Mandatory	Keep this bit set to <b>zero</b> .
	DIC	Output Driver Impedance Control	1-C, A1-M		Select the driver impedance using these bits from the memory side.
	RTT	On Die Termination (ODT)	6,2-C, A6,A2-M		Select ODT value using these bits from the memory side.
	AL	Additive Latency	5,3-C, A5,A3-M		Can be cleared for basic DMC initialization. Refer to the memory device data sheet for more details on this bit.
	OCD	Off Chip Driver (OCD) Calibration	9:7-C, A9:A7-M		OCD not supported. Keep this bit set to <b>zero</b> .
	DQS#	DQS# Enable	10-C, A10-M		Keep this bit set to <b>zero</b> .
	RDQS	RDQS Enable	11-C, A11-M		RDQS is not supported. Keep this bit set to <b>zero</b> .
	QOFF	Output Buffer Enable	12-C, A12-M		Keep this bit set to <b>zero</b> .
	Reserved	Reserved	15:13-C, A15:A13-M	Hard Wired	These bits are hard-wired to <b>zero</b> .
	Reserved	Reserved	31:16-C		
DMC_EMR2	PASR	Partial Array Self Refresh	2:0-C, A2:A0-M	Optional	This bit is unchanged for standard DMC operation. Refer to the hardware reference manual <sup>[2][13]</sup> for more details.
	DCC	DCC Enable	3-C, A3-M	Mandatory	DCC is not supported. Keep this bit set to <b>zero</b> .
	Reserved	Reserved	6:4-C, A6:A4-M	Mandatory	Program these bits to <b>zero</b> .
	SRF	High Temperature Self Refresh Rate Enable	7-C, A7-M	Optional	These bits are unchanged for standard DMC operation. Refer to the hardware reference manual <sup>[2][13]</sup> for more details.
	Reserved	Reserved	15:8-C, A15:A8-M	Hard Wired	These bits are hard-wired to <b>zero</b> .
	Reserved	Reserved	31:16-C		

Table 4. DMC Controller DDR2 Mode Register Bit Fields

b. DDR3: Program the DMC<sub>x</sub>\_MR0, DMC<sub>x</sub>\_MR1, and DMC<sub>x</sub>\_MR2 registers (Table 5).

Register	Bit Field		Bit field (C=Controller, M=JEDEC)	Value	Comment
DMC_MR0	BL	Burst Length	1:0-C, A1:A0-M	Mandatory	Only BL=8 is supported for DDR3. Always program these bits with <b>zero</b> .
	CL	CAS Latency	6:4,2-C, A6:A4, A2- M		Program these bits with the required CAS latency.
	BT	Burst Type	3-C, A3-M		Only sequential burst type supported. Keep this bit <b>zero</b> .
	TM	Test Mode	7-C, A7-M		Always program this bit to <b>zero</b> , as test mode is not supported.
	DLLRST	DLL Reset	8-C, A8-M		Set this bit for DDR3 mode.
	WRRECOV	Write recovery	11:9-C, A11:A9-M		Program these bits with tWR value from the memory device data sheet. Refer to the hardware reference manual <sup>[2][13]</sup> for more details.
	PD	Active Power Down Mode	12-C, A12-M	Optional	Can be left unchanged for standard DMC operation. Refer to the hardware reference manual <sup>[2][13]</sup> for more details on these bits.
	Reserved	Reserved	15:13-C, A15:A13-M	Hard Wired	These bits are hard-wired to <b>zero</b> .
	Reserved	Reserved	31:16-C		
DMC_MR1	DLEN	DLL Enable	0-C, A0-M	Mandatory	Keep this bit set to <b>zero</b> .
	DIC0, DIC1	Output Driver Impedance Control	5,1-C, A5,A1-M		Select the driver impedance using these bits from the memory side.
	RTT0, RTT1, RTT2	On Die Termination (ODT)	9,6,2-C, A9,A6,A2-M		Select ODT value using these bits from the memory side.
	AL	Additive Latency	4,3-C, A4,A3-M		Can be cleared for basic DMC initialization. Refer to the memory device data sheet for more details.
	WL	Write Levelization	7-C, A7-M		Write levelling is not supported. Keep this bit <b>zero</b> .
	Reserved	Reserved	8, 10-C, A8, A10 –M		These bits are reserved for future use (must be programmed to <b>zero</b> ).
	TDQS	Termination Data Strobe	11-C, A11-M		Should be <b>zero</b> , as it is not applicable for 16-bit devices.
	QOFF	Output Buffer Enable	12-C, A12-M		Should be <b>zero</b> .
	Reserved	Reserved	15:13-C, A15:A13-M	Hard Wired	These bits are hard-wired to <b>zero</b> .
	Reserved	Reserved	31:16-C		
DMC_MR2	PASR	Partial Array Self Refresh	2:0-C, A2:A0-M	Optional	This bit is unchanged for standard DMC operation. Refer to the hardware reference manual <sup>[2][13]</sup> for more details.
	CWL	CAS Write Latency	5:3-C, A5:A3-M	Mandatory	Obtain from memory device data sheet.
	ASR	Auto Self Refresh	6-C, A6-M	Optional	These bits are unchanged for standard DMC operation. Refer to the hardware reference manual <sup>[2][13]</sup> for more details.
	SRT	Self-Refresh Temperature Range	7-C, A7-M		
	Reserved	Reserved	8-C, A8-M		This bit is hard-wired to <b>zero</b> .
	RTTWR	Dynamic ODT Write	10:9-C, A10:A9-M	Hard Wired	Rtt_WR is not supported. These bits are hard-wired to <b>zero</b> .
	Reserved	Reserved	15:11-C, A15:A11-M		These bits are hard-wired to <b>zero</b> .
Reserved	Reserved	31:16-C			

Table 5. DMC Controller DDR3 Mode Register Bit Fields

c. LPDDR: Program the DMC<sub>x</sub>\_MR and DMC<sub>x</sub>\_EMR registers (Table 6).

Register	Bit Field		Bit field (C=Controller, M=JEDEC)	Value	Comment
DMC_MR	BL	Burst Length	2:0-C, A2:A0-M	Mandatory	Program burst length to 4/8 as per requirements (burst length of 2 is not supported). Refer to the hardware reference manual <sup>[2]</sup> for details.
	BT	Burst Type	3-C, A3-M		Only sequential burst type is supported. Keep this bit set to <b>zero</b> .
	CL	CAS Latency	6:4-C, A6:A4-M		Only CL=3 is supported for LPDDR mode.
	Reserved	Reserved	15:13-C, A15:A13-M	Hard Wired	These bits are hard-wired to <b>zero</b> .
	Reserved	Reserved	31:16-C		
DMC_EMR	PASR	Partial Array Self Refresh	2:0-C, A2:A0-M	Optional	This bit is unchanged for standard DMC operation. Refer to the hardware reference manual <sup>[2][13]</sup> for more details.
	TSCR	Temperature Compensated Self Refresh	4:3-C, A4-A3-M		
	DS	Drive Strength	7:5-C, A7-A5-M	Mandatory	Select the driver impedance (strength) using these bits from the memory side.
	Reserved	Reserved	15:8-C, A15:A8-M	Hard Wired	These bits are hard-wired to <b>zero</b> .
	Reserved	Reserved	31:16-C		

Table 6. DMC Controller LPDDR Mode Register Bit Fields

- Make sure that the `DMC_DT_CALIB_ADDR` register (Table 7) is programmed to an unused DMC location (by default, it is the starting address of the DMC address range). Please refer to the hardware reference manual<sup>[2]</sup> for more details.
- Program the `DMCx_CTL` register (Table 7) with `DMCx_CTL.INIT` set to start the DMC initialization sequence.
- Wait for the DMC initialization to complete by polling for the `DMCx_STAT.INITDONE` bit to set.
- Program the `DMCx_CTL.DLLCTL` register (Table 7) with the value `0x948` (`DATA CYC = 9`, `DLLCALRDCNT = 72`).

Register	Bit Field		Bit field (C=Controller, M=JEDEC)	Value	Comment
DMC_CTL	DDR3EN	Enable DDR3 mode	0-C	Mandatory	Set this bit for DDR3 mode, clear for DDR2 and LPDDR modes.
	LPDDR	Enable LPDDR mode	1-C		Set this bit for LPDDR mode, clear for DDR3 and DDR2 modes.
	INIT	Start DMC initialization	2-C		Set this bit to start the DMC initialization sequence.
	SRREQ	Self-Refresh Request	3-C	Optional	These bits are unchanged for standard DMC operation. Refer to the hardware reference manual <sup>[2][13]</sup> for more details.
	PDREQ	Power Down Request	4-C		
	DPDREQ	Deep Power Down Request	5-C		
	PREC	Precharge	6-C		
	RESET	Reset SDRAM	7-C		
	ADDRMODE	Addressing Mode	8-C		
	RDTOWR	Read-to-Write Cycle	11:9-C	Mandatory	Always program this bit field with <b>2</b> .
	PPREF	Postpone Refresh	12-C	Optional	These bits are unchanged for standard DMC operation. Refer to the hardware reference manual <sup>[2][13]</sup> for more details.
	DLLCAL	DLL Calibration Start	13-C		
	Reserved	Reserved	23:14-C	Mandatory	Always write these bits with <b>zero</b> .
	ZQCS	ZQ Calibration Short	24-C	Optional	These bits are unchanged for standard DMC operation. Refer to the hardware reference manual <sup>[2][13]</sup> for more details.
	ZQCL	ZQ Calibration Long	25-C		
Reserved	Reserved	31:26-C	Mandatory	Always write these bits with <b>zero</b> .	
DLLCTL	DLLCALRDCNT	DLL Calibration RD Count	7:0-C	Mandatory	Program this bit field to <b>72</b> .
	DATA CYC	Data Cycles	11:8-C		Program this field to <b>9</b> .
	Reserved	Reserved	31:9-C		Always write these bits with <b>zero</b> .
DMC_DT_CALIB_ADDR		Data Calibration Address Register	31:0-C		Program this value with an unused DMC address. Refer to the hardware reference manual <sup>[2]</sup> for more details.

Table 7. DMC Controller Control/Calibration Bit Fields



In addition to the above steps, also ensure that the workaround to anomaly 20-00-0037 (if applicable to the processor silicon revision being used) is also implemented *after* the DMC initialization is complete. See the processor silicon anomaly lists <sup>[5][15]</sup> for details.



Before the DDR controller enters the initialization procedure, it will automatically provide the appropriate power-up/reset/first command timing according to the JEDEC standards.

## DMC Initialization Code

The ZIP file<sup>[6]</sup> associated with this EE-note provides the CGU and DMC initialization subroutines along with code examples which can be used to initialize the CGU and DMC controller for any custom settings.

### CGU Initialization

The `CGUInit.c` and `CGUInit.h` files can be used to initialize the CGU to the required clock settings. For example, the `main.c` file in the `ADSP-SC58x/DDR3_Init_Core1` folder illustrates how to initialize the CGU to generate the following clock combination on the ADSP-SC589 EZ-Board®, which features a 25 MHz CLKIN, as shown in [Listing 1](#):

- PLLCLK = 450 MHz
- CCLK = 450 MHz
- SYCLK = 225 MHz
- DCLK = 450 MHz
- SCLK0 = SCLK1 = 225 MHz

```
void CGUInit(void)
{
    int i;

    //PLLCLK = 450 MHz, CCLK = 450 MHz, SYCLK = 225 MHz, DCLK = 450 MHz, SCLK0 = SCLK1 = 225 MHz
    CGU0_PARAM_LIST_DEFAULT.ctl_MSEL=18;
    CGU0_PARAM_LIST_DEFAULT.ctl_DF=0;
    CGU0_PARAM_LIST_DEFAULT.div_CSEL=1;
    CGU0_PARAM_LIST_DEFAULT.div_SYSEL=2;
    CGU0_PARAM_LIST_DEFAULT.div_S0SEL=2;
    CGU0_PARAM_LIST_DEFAULT.div_S1SEL=2;
    CGU0_PARAM_LIST_DEFAULT.div_DSEL=1;

    //Set DMCx_PHY_CTL0.RESETDLL bit.
    *pREG_DMC0_PHY_CTL0=0x800;
    *pREG_DMC1_PHY_CTL0=0x800;

    //Initialize CGU to change the DCLK frequency
    CGU_Init(0, 0, &CGU0_PARAM_LIST_DEFAULT, 0);

    //Clear DMCx_PHY_CTL0.RESETDLL bit
    *pREG_DMC0_PHY_CTL0=0x000;
    *pREG_DMC1_PHY_CTL0=0x000;

    //Wait for 9000 DCLK cycles for the DLL to lock
    for(i=0;i<300*DMC_Param_List.cclk_dclk_ratio;i++);
}
```

Listing 1. CGU Initialization

For custom clock settings, the values of MSEL, DF, CSEL, SYSEL, S0SEL, S1SEL, and DSEL can be changed accordingly.

### DMC Initialization

The `DMCInit.c` and `DMCInit.h` files can be used to initialize the DMC to the required settings. For example, the `main.c` file in the `ADSP-SC58x/DDR3_Init_Core1` folder illustrates how to initialize the DMC for the Micron MT41K128M16-125 DDR3 memory device<sup>[7]</sup> with a DCLK frequency of 450 MHz, as shown in [Listing 2](#).

```
//Initialize the DMC parameters list
DMC_Param_List.dmc_no=0;
DMC_Param_List.ddd_mode=DDR3_MODE;
DMC_Param_List.phy_init_required=true;
DMC_Param_List.cclk_dclk_ratio=1;
DMC_Param_List.anomaly_20000037_applicable=true;
DMC_Param_List.dmc_ctl_value=DMC_CTL_VALUE;
DMC_Param_List.dmc_cfg_value=DMC_CFG_VALUE;
DMC_Param_List.dmc_tr0_value=DMC_TR0_VALUE;
DMC_Param_List.dmc_tr1_value=DMC_TR1_VALUE;
DMC_Param_List.dmc_tr2_value=DMC_TR2_VALUE;
DMC_Param_List.dmc_mr0_value=DMC_MR0_VALUE;
DMC_Param_List.dmc_mr1_value=DMC_MR1_VALUE;
DMC_Param_List.dmc_mr2_value=DMC_MR2_VALUE;
DMC_Param_List.padctl2_value=DMC_PADCTL2_VALUE;
DMC_Param_List.dmc_cphyctl_value=DMC_CPHYCTL_VALUE;
DMC_Param_List.dmc_data_calib_add_value=DMC0_DATA_CALIB_ADD;

//Initialize CGU - PLLCLK = 450 MHz, CCLK = 450 MHz, SYCLK = 225 MHz, DCLK = 450 MHz, SCLK0 = SCLK1 = 112.5 MHz
CGUInit();

//Initialize DMC0 now
DMCInit(DMC_Param_List);

DMC_Param_List.dmc_no=1;
DMC_Param_List.dmc_data_calib_add_value=DMC1_DATA_CALIB_ADD;

//Initialize DMC1 now
DMCInit(DMC_Param_List);
```

### Listing 2. DMC Initialization



Though the ADSP-SC5xx processors do not support the DDR3L (1.35 V) standard, the MT41K128M16 is backward compatible with the 1.5 V standard, thus it can be used without issue. When interfacing a DDR3L device with ADSP-SC5xx processors, contact the memory vendor to make sure that it is compatible with the DDR3 standard.

For custom DMC settings, the `DMC_Param_List` structure must be updated according to the system requirements. The DMC register values can be derived using the `DMC_Registers_List.xlsx` spreadsheet in the associated ZIP file<sup>[6]</sup> by entering various DMC-specific processor and DDR memory-specific parameters (from the device data sheet), as shown in [Figure 3](#).

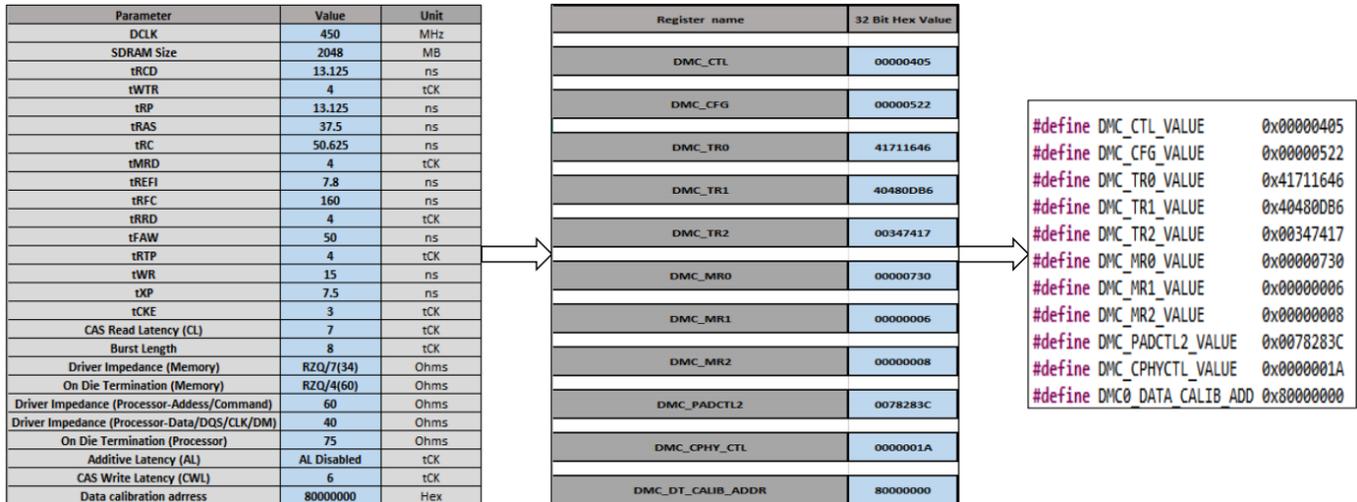


Figure 3. Deriving DMC Register Settings

### Validating the DMC Interface

Once the DMC is initialized, it is important to validate it. It is recommended to check whether or not all the DMC registers have been initialized to the correct values, if there are any basic issues with the DMC hardware interface, and whether or not the DMC has indeed been correctly initialized by the software.

The `DMCInit_Debug.c` and `DMCInit_Debug.h` files in the associated ZIP file provide some assistance for performing this validation. The `Print_DMC_Register_Contents()` function can be used to print the DMC controller and PHY register values, and the `Memory_Sweep_Test()` function can be used to check whether or not both core and DMA (MDMA0) accesses to the DMC are working for different data word sizes (8-/16-/32-/64-bit and 32-byte DMA) and for different data patterns (0x0, 0xF, 0x5, 0xA, incremental, random, and all bits toggling). [Figure 4](#) shows how the `main.c` file in the `DDR3_Init_Core1` folder uses these functions to validate the DMC interface. The memory sweep size used in this code is 0x100000 (1 MB), which can be changed to validate the full DMC memory range (e.g., 2 Gb = 256 MB).

<pre> //Print DMC register contents for both the DMCs Print_DMC_Register_Contents(0); Print_DMC_Register_Contents(1);  printf("\nTesting for DMC0...\n"); failure=Memory_Sweep_Test(DMC0_START_ADDRESS, 0x100000);  printf("\nTesting for DMC1...\n"); failure=Memory_Sweep_Test(DMC1_START_ADDRESS, 0x100000);  if(failure==0)     printf("\nTest passed...\n"); else     printf("\nTest failed...\n"); </pre>	<pre> DMC0_CTL = 0x401 DMC0_CFG = 0x522 DMC0_TR0 = 0x4150f646 DMC0_TR1 = 0x40400c30 DMC0_TR2 = 0x336414 DMC0_MR = 0x530 DMC0_MR1 = 0x6 DMC0_MR2 = 0x8 DMC0_DCLLCTL = 0x948 DMC0_PHY_CTL0 = 0xf DMC0_PHY_CTL1 = 0x0 DMC0_PHY_CTL2 = 0xfc000000 DMC0_PHY_CTL3 = 0xa0000c0 DMC0_PHY_CTL4 = 0x0 DMC0_CAL_PADCTL0 = 0xf0000000 DMC0_CAL_PADCTL2 = 0x78283c DMC1_CTL = 0x401 DMC1_CFG = 0x522 DMC1_TR0 = 0x4150f646 DMC1_TR1 = 0x40400c30 DMC1_TR2 = 0x336414 DMC1_MR = 0x530 DMC1_MR1 = 0x6 DMC1_MR2 = 0x8 DMC1_DCLLCTL = 0x948 DMC1_PHY_CTL0 = 0xf DMC1_PHY_CTL1 = 0x0 DMC1_PHY_CTL2 = 0xfc000000 DMC1_PHY_CTL3 = 0xa0000c0 DMC1_PHY_CTL4 = 0x0 DMC1_CAL_PADCTL0 = 0xf0000000 DMC1_CAL_PADCTL2 = 0x78283c Testing for DMC0... </pre>
<p>main.c</p>	<p>Output console</p> <pre> Word size = 8 bits  Core Test  Data pattern = All ZEROS Test Passed...  Data pattern = All ONES Test Passed...  Data pattern = All A Test Passed...  Data pattern = All 5 Test Passed...  Data pattern = INCREMENTAL Test Passed...  Data pattern = RANDOM Test Passed...  Data pattern = ALL_BITS_TOGGING Test Passed...  DMA Test  Data pattern = All ZEROS Test Passed...  Data pattern = All ONES Test Passed...  Data pattern = All A Test Passed...  Data pattern = All 5 Test Passed... </pre>

Figure 4. Validating the DMC Interface

## Creating Preload and Initialization Code with Customized CGU and DMC Settings

Preload and initialization code are two concepts that are related to configuring the CGU and DMC prior to the application code running, depending on whether performing active debug via the emulator or controlling the boot stream for a stand-alone application.

### Preload Code

When performing active debug on a target platform, an emulator is used. To make working with the board as transparent as possible for the user, the CrossCore® Embedded Studio (CCES) tools automate initialization of the CGU/DMC hardware such that applications can be built and loaded to off-chip memory for use in a debug session on the targeted board. This is done via *Preload Code*, and there are examples in the Custom Preload Codes folder for each of DDR2, DDR3, and LPDDR in the associated ZIP file<sup>[6]</sup>. When the project is built, the associated executable file (DXE) is used by CCES (as shown in [Figure 5](#)) to initialize the CGU and DMC before loading the actual application using the emulator.

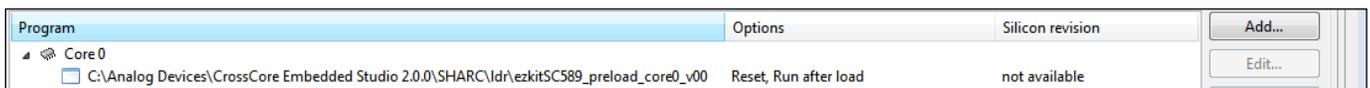


Figure 5. Preload Code Being Used for the ADSP-SC589 EZ-Board

### Initialization Code

Unlike preload code, initialization code is actually a part of the application. It is separate from the application itself, and its DXE output is pre-pended to the application's DXE file when CCES assembles the loader stream (LDR) that the processor parses during the boot process. This separate DXE is called the *Initialization Block* in the LDR file, which is booted first into on-chip memory and is then executed before any attempts are made to resolve anything to the external DDR space, making it the ideal place for configuring the CGU and DMC in advance of trying to boot to DDR memory. In the `Custom Init Codes` folder in the associated ZIP file<sup>[6]</sup>, there are projects for each of DDR2, DDR3, and LPDDR, and the DXE output corresponding to the project can be used as the default initialization code when generating an LDR file by pointing to the DXE in the Loader Options page of the Project Properties, as shown in [Figure 6](#).



Figure 6. Initialization Code Selection in the Loader Options

Typically, for applications requiring a one-time CGU and DMC initialization after reset, the preload (when loading the application via emulator) or initialization code (when booting the application standalone) should be sufficient. Thus, it is important to understand how to use and modify the default preload and initialization code for customized CGU/DMC settings.

### Default Preload and Initialization Code

The CCES installation provides the DXE files produced by the default preload and initialization code for each of the ADSP-SC589, ADSP-SC584, and ADSP-SC573 evaluation platforms in the directory:

```
<CCES Root>\Analog Devices\CrossCore Embedded Studio 2.2.0\SHARC\ldr
```

The corresponding source code and project files can be found in the `\init_code` subdirectory.

The default CGU configuration which the DMC initialization code is designed for is as follows:

1. ADSP-SC589 EZ-Board – Micron MT41K128M16-125 DDR3 device<sup>[7]</sup> with:
  - PLLCLK = CCLK = DCLK = 450 MHz
  - SYSCLK = 225 MHz
  - SCLK0 = SCLK1 = 112.5 MHz
2. ADSP-SC584 EZ-Board – Micron MT47H128M16RT-25E XIT:C DDR2 device<sup>[8]</sup> with:
  - PLLCLK = CCLK = DCLK = 400 MHz
  - SYSCLK = 200 MHz
  - SCLK0 = SCLK1 = 100 MHz
3. ADSP-SC573 EZ-Board – Micron MT41K128M16-125 DDR3 device<sup>[7]</sup> with:
  - PLLCLK = CCLK = DCLK = 450 MHz
  - SYSCLK = 225 MHz
  - SCLK0 = SCLK1 = 112.5 MHz

### Modifying Default Preload and Initialization Code for Customized CGU/DMC Settings

The CGU and DMC settings in the default preload and initialization source code may need to be modified for the following conditions:

1. When using the EZ-Board (or a custom board with the same DDR memory device as that populated on the EZ-Board) with non-default CGU settings.
2. When using a custom board with a different memory device.

For example, the following steps can be followed to modify the default preload code in the `sc589_preload_Core0` directory in CCES to interface the ADSP-SC589 processor to a Micron MT46H128M16LFDD-48 LPDDR memory device<sup>[9]</sup> with a DCLK of 200 MHz:

1. Copy the `sc589_preload_Core0` and `src` folders from `\ldr\init_code\SC58x_Init` and paste them to a new folder called `SC589_LPDDR`.
2. Create a workspace in the new `SC589_LPDDR` folder and import the `sc589_preload_Core0` project into the workspace.
3. Replace the `DMCInit.c` and `DMCInit.h` files in the `init_src` folder with the ones provided in the associated ZIP file<sup>[6]</sup>.
4. In the project properties window, remove the `DMCInit.c` and `DMCInit.h` files from the filter, as shown in [Figure 7](#):

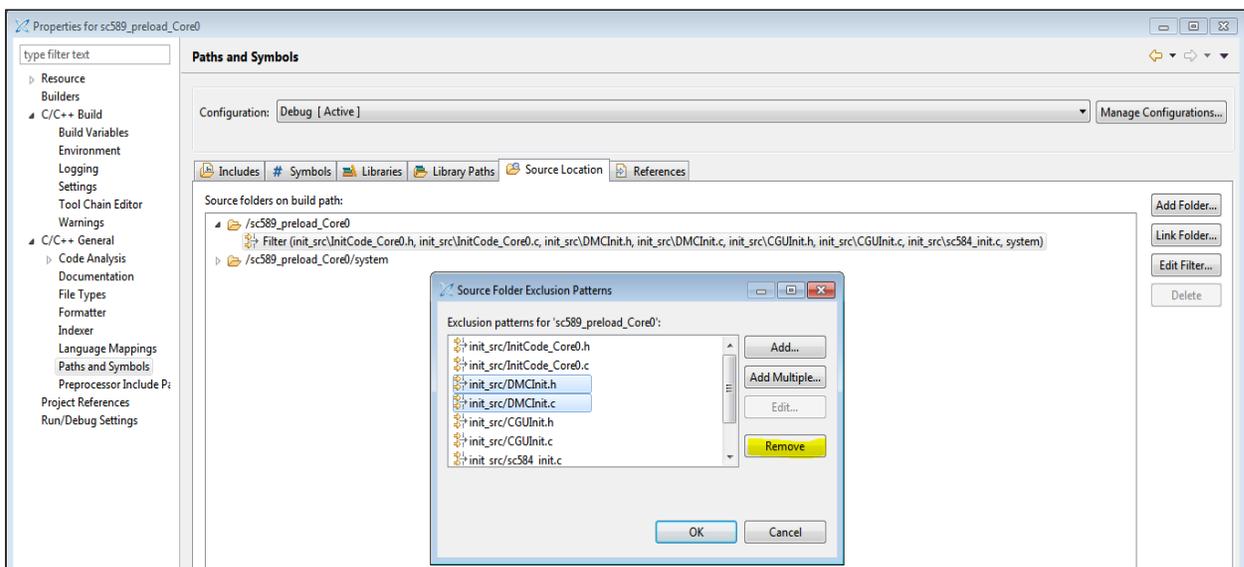


Figure 7. Modifying Project Options for Building Custom ADSP-SC589 Preload Code

5. Modify the CGU settings in the `sc58x_init.h` file to generate `CCLK = 400 MHz`, `SYSCLK = 200 MHz`, `SCLK = 100 MHz`, and `DCLK = 200 MHz`. As shown in [Figure 8](#), `MSEL` is modified from 18 to 16, and `DSEL` is modified from 1 to 2 to achieve these settings.
6. Add the customized DMC register settings in the `sc58x_init.h` file, and replace the DMC initialization portion of the code for both `DMC0` and `DMC1` under the conditional build `__CONFIG_DMC0__` and `__CONFIG_DMC1__` macros, respectively. Be sure to also verify that the `DMC_PARAM DMC_Param_List` variable is defined, as shown in [Figure 8](#).

<pre> /* default power settings */ #define DF_DEFAULT (0x0u) #if defined (__ADSPSC584__) #define MSEL_DEFAULT (0x10u) #else #define MSEL_DEFAULT (0x12u) #endif #define SYSSSEL_DEFAULT (0x2u) #define S0SEL_DEFAULT (0x2u) #define S1SEL_DEFAULT (0x2u) #define CSEL_DEFAULT (0x1u) #define DSEL_DEFAULT (0x1u) </pre> <p style="text-align: center;"><b>Default CGU Settings</b></p> <pre> #define DMC_CTL_VALUE 0x00002406 #define DMC_CFG_VALUE 0x00000522 #define DMC_TR0_VALUE 0x20B08323 #define DMC_TR1_VALUE 0x200F0618 #define DMC_TR2_VALUE 0x00123000 #define DMC_MR0_VALUE 0x00000032 #define DMC_MR1_VALUE 0 #define DMC_MR2_VALUE 0 #define DMC_PADCTL2_VALUE 0 #define DMC_CPHYCTL_VALUE 0x00000000 #define DMC0_DATA_CALIB_ADD 0x80000000 #define DMC1_DATA_CALIB_ADD 0xC0000000 </pre> <p style="text-align: center;"><b>"sc58x_init.h" modifications</b></p>	<pre> DMC_PARAM DMC_Param_List;  #if defined(__CONFIG_DMC0__) //Initialize the DMC parameters list DMC_Param_List.dmc_no=0; DMC_Param_List.ldr_mode=LPDDR_MODE; DMC_Param_List.phy_init_required=true; DMC_Param_List.cclk_dclk_ratio=2; DMC_Param_List.anomaly_2000037_applicable=true; DMC_Param_List.dmc_ctl_value=DMC_CTL_VALUE; DMC_Param_List.dmc_cfg_value=DMC_CFG_VALUE; DMC_Param_List.dmc_tr0_value=DMC_TR0_VALUE; DMC_Param_List.dmc_tr1_value=DMC_TR1_VALUE; DMC_Param_List.dmc_tr2_value=DMC_TR2_VALUE; DMC_Param_List.dmc_mr0_value=DMC_MR0_VALUE; DMC_Param_List.dmc_mr1_value=DMC_MR1_VALUE; DMC_Param_List.dmc_mr2_value=DMC_MR2_VALUE; DMC_Param_List.padctl2_value=DMC_PADCTL2_VALUE; DMC_Param_List.dmc_cphyctl_value=DMC_CPHYCTL_VALUE; DMC_Param_List.dmc_data_calib_add_value=DMC0_DATA_CALIB_ADD;  //Initialize DMC0 now DMCInit(DMC_Param_List);  #endif // __CONFIG_DMC0__  #if defined(__CONFIG_DMC1__) //Initialize the DMC parameters list DMC_Param_List.dmc_no=1; DMC_Param_List.ldr_mode=LPDDR_MODE; DMC_Param_List.phy_init_required=true; DMC_Param_List.cclk_dclk_ratio=2; DMC_Param_List.anomaly_2000037_applicable=true; DMC_Param_List.dmc_ctl_value=DMC_CTL_VALUE; DMC_Param_List.dmc_cfg_value=DMC_CFG_VALUE; DMC_Param_List.dmc_tr0_value=DMC_TR0_VALUE; DMC_Param_List.dmc_tr1_value=DMC_TR1_VALUE; DMC_Param_List.dmc_tr2_value=DMC_TR2_VALUE; DMC_Param_List.dmc_mr0_value=DMC_MR0_VALUE; DMC_Param_List.dmc_mr1_value=DMC_MR1_VALUE; DMC_Param_List.dmc_mr2_value=DMC_MR2_VALUE; DMC_Param_List.padctl2_value=DMC_PADCTL2_VALUE; DMC_Param_List.dmc_cphyctl_value=DMC_CPHYCTL_VALUE; DMC_Param_List.dmc_data_calib_add_value=DMC1_DATA_CALIB_ADD;  //Initialize DMC01 now DMCInit(DMC_Param_List);  #endif // __CONFIG_DMC1__ </pre> <p style="text-align: center;"><b>"sc58x_init.c" modifications</b></p>
---	--

Figure 8. Modifying Source Code Files for Customized CGU/DMC Settings

7. Now build the project, and the customized preload DXE file is available in the \Debug folder.

Similarly, the preload code for the ADSP-SC584 and ADSP-SC573 evaluation platforms, as well as the initialization code for the ADSP-SC589, ADSP-SC584, and ADSP-SC573 evaluation platforms, can be modified for any customized CGU and DMC settings.



Even though this EE-note provides guidance to customize the preload and initialization code for the ADSP-SC589 (DDR3 and LPDDR modes), ADSP-SC584 (DDR2 mode), and ADSP-SC573 (DDR3, DDR2, and LPDDR modes) processors, the examples should be used only as a reference to understand the procedure. It is the user's responsibility to ensure that the latest preload and initialization code from the CCES installation is the starting point and careful modification is performed when making the required customization to support the desired CGU/DMC settings of the end application.

## Frequently Asked Questions

**Question 1:** I am using the same DDR device as used on the EZ-Board, but I want to run the DMC at a different DCLK value than the EZ-Board. What do I need to take care of when initializing the DMC?

**Answer:** Just change the DCLK parameter value in the `DMC_Registers_List.xlsx` spreadsheet and copy the generated DMC register values into the `main.h` file of the corresponding example DMC initialization code provided in the associated ZIP file<sup>[6]</sup>. For example, [Figure 9](#) shows how to generate and use the DMC register values for the Micron MT41K128M16-125 DDR3 device with a DCLK frequency of 400 MHz instead of 450 MHz.

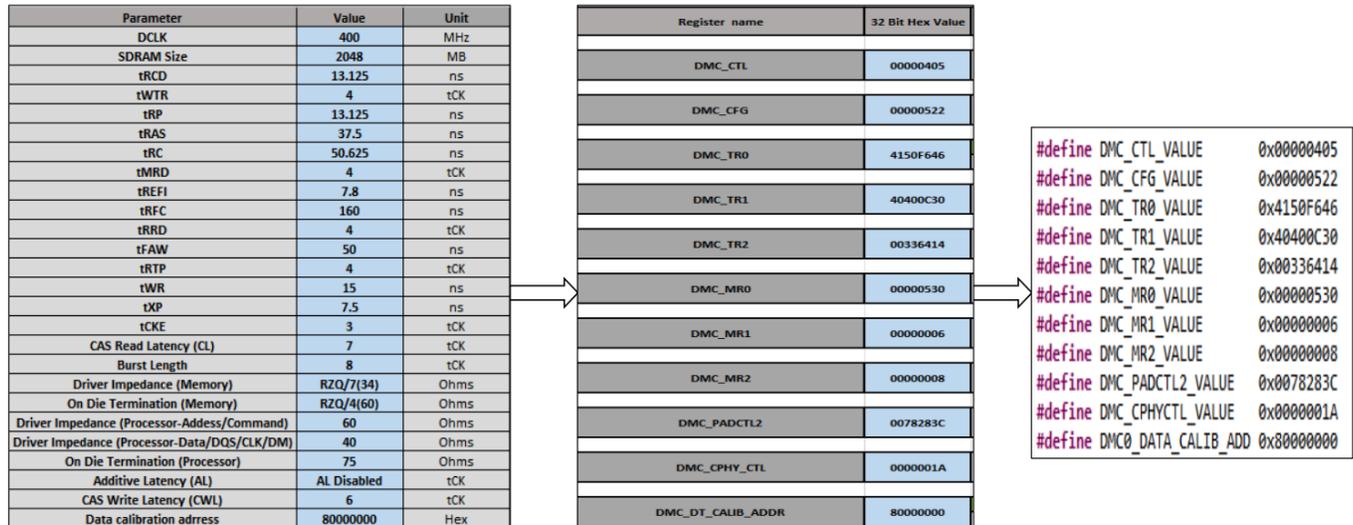


Figure 9. Generating DMC Register Values for the MT41K128M16-125 DDR3 Memory Device w/ DCLK = 400 MHz

**Question 2:** I am using the ARM core, and all my data/code is in the external DDR memory. How can I re-initialize the DMC with a non-default DCLK value without affecting the memory contents?

**Answer:** To make sure that the code/data in the DMC is retained, the CGU should be initialized as per the steps mentioned in the [On-the-Fly DMC Re-Initialization](#) section of this EE-note. By default, the linker file for the ARM core on the ADSP-SC5xx processors puts all the code and data in the L3 (DMC) space. Thus, a custom linker file would be required to place the complete CGU and DMC re-initialization code into non-L3 (DDR) space (e.g., L2). The `DDR3_Re_Initialization_ARM_Core0` code example in the associated ZIP file<sup>[6]</sup> illustrates how to reinitialize the DMC via the ARM core to run the DCLK at 400 MHz instead of the default DCLK frequency of 450 MHz.

**Question 3:** Do I need to re-initialize PHY registers each time I reinitialize the DMC dynamically in my code?

**Answer:** PHY registers need not be initialized every time the DMC is reinitialized, unless the pads need to be calibrated to a different driver impedance and/or ODT.

**Question 4:** Can I run the DDR device at a DCLK of 400 MHz and still run the core at 450 MHz?

Yes, this is possible using two CGUs. One CGU can generate a CCLK of 450 MHz (e.g., with  $\text{SYSCLK} = 225 \text{ MHz}$  and  $\text{SCLK} = 112.5 \text{ MHz}$ ), and the other CGU can generate a DCLK at 400 MHz. The `DDR2_Init_Two_CGU_Core1` example in the associated ZIP file<sup>[6]</sup> illustrates how to configure the DDR2 device on the ADSP-SC584 EZ-Board with a DCLK of 400 MHz while maintaining a CCLK of 450 MHz.

**Question 5:** Does the controller perform only a one-time DLL calibration during DMC initialization?

**Answer:** No, during initialization, the controller does a coarse DLL calibration. During runtime, the controller continues with fine calibration during auto-refresh commands and continues to update the corresponding calibration registers automatically to accommodate for temperature/PCB changes during runtime.

## Hardware Considerations

Standard DDR interface board design guidelines are presented in this section to have optimal ADSP-SC5xx DMC interface signal and power integrity.

### Placement and General Routing Guidelines

- Place the ADSP-SC5xx and the DDR3/DDR2/LPDDR memory as close as possible to each other, as this reduces routing length.
- The ADSP-SC5xx DMC interface supports only point-to-point design and does not support fly-by topology.
- Plan the stack-up such that all the DDR3/DDR2/LPDDR signals have continuous and immediate reference planes, and ensure that they do not cross splits.
- The DDR3/DDR2/LPDDR address, command, and control signals can use an even power plane as a reference. Ensure that there are no splits or discontinuities in the referenced power plane.
- The PCB trace characteristic impedance must be  $50 \Omega$  for single-ended signals and  $100 \Omega$  for differential signals.
- Route all the DDR signals as a group in each and every layer to avoid a mismatch in trace impedance and propagation delay.
- Ensure that the DDR signals are routed as groups and that signals within groups do not change layers.
  - E.g., the `DMC_DQ00-07`, `DMC_LDQS`, and `DMC_LDQM` signals should be routed as a group, which should be routed in the same layer and should have the same reference. Changing the reference plane can change the trace impedance.
- To avoid crosstalk, ensure that all DDR signals have sufficient center-to-center spacing of at least  $3W$  between DDR signals and  $4W$  to other signals.
- Maintain perpendicularity between DMC signal routing if the DMC signals are routed in the adjacent layers. This reduces crosstalk, as signals on adjacent layers will not be parallel to each other.
- Avoid test points on the DDR signals, as they create stubs which can act as an EMI source. Instead, use vias for probing or JEDEC-recommended methods that DDR memory vendors offer for diagnostics.

## Trace Length Matching Criteria



Routing of all the DDR interface signals should be length-matched to avoid set-up and hold time violations due to propagation delay.

The length-matching criteria are as follows:

- Match all address ( $DMC\_A[n]$ ,  $DMC\_BA[n]$ ) and command ( $DMC\_CKE$ ,  $DMC\_CS[n]$ ,  $DMC\_ODT$ ,  $DMC\_RAS$ ,  $DMC\_RESET$ ,  $DMC\_WE$ ) signals' trace length within +/- 40 mils relative to the  $DMC\_CK$  signal.
- Match all data ( $DMC\_DQ[n]$ ) and data mask ( $DMC\_UDM$ ,  $DMC\_LDM$ ) signals' trace lengths within +/- 40 mils relative to their corresponding  $DQS$  signal.
  - E.g., the lower order data byte ( $DMC\_DQ00 - DMC\_DQ07$ ) and the corresponding data mask ( $DMC\_LDM$ ) signals should be length-matched with the lower order strobe ( $DMC\_LDQS$ ) signal.
- The differential signals such as clock ( $DMC\_CK$  and  $/DMC\_CK$ ) and  $DQS$  pairs ( $DMC\_LDQS$  and  $/DMC\_LDQS$ ,  $DMC\_UDQS$  and  $/DMC\_UDQS$ ) should be length-matched within 10 mils.
  - E.g., the trace length of the  $DMC\_CK$  and  $/DMC\_CK$  signals should be matched within 10 mils relative to each other.

## Interface Termination



All the DDR interface signals without ODT require external termination for improved signal integrity.

The termination guidelines are as follows:

- Install a 100  $\Omega$  termination resistor on the differential clock between  $DMC\_CK$  and  $/DMC\_CK$ , and place it close to the memory.
- Install external series termination resistors on all address and control signals, and place them closer to the processor.
- The value of the series termination resistors on the ADSP-SC5xx EZ-Board is 33  $\Omega$ . This can vary based on the routing in the board, and appropriate series termination resistor values can be obtained based on simulation results of the board. The IBIS models for the ADSP-SC5xx processor can be downloaded from [www.analog.com](http://www.analog.com).
- The DDR2/3 data group signals do not require external termination, as they have ODT.
- The ODT value can be obtained based on simulation results of the board. IBIS models for the ADSP-SC5xx processor can be downloaded from [www.analog.com](http://www.analog.com).
- For the LPDDR interface, it is advisable to also install termination resistors for data signals, as ODT is not supported for LPDDR mode.

## DMC Power (VDD\_DMC) Decoupling



The DMC interface should have sufficient decoupling on the VDD\_DMC rail and the memory power rail to avoid data corruption.

- Use the schematics in the appropriate *evaluation system manual*<sup>[3][4][16]</sup> as a reference to identify the decoupling capacitor quantity and values for the ADSP-SC5xx VDD\_DMC rail.
- Refer to the DDR memory datasheet or consult the DDR memory vendor to identify the decoupling capacitor requirement for the DDR memory power rails. A critical parameter is  $I_{dd7}$ , which defines the peak current during multi-bank operation, depending on the speed grade and ambient temperature of the DDR memory.
- All the decoupling capacitors must be placed very near to the VDD\_DMC power rail and should use a solid power/ground plane.
- Ensure that the power and ground planes are adjacent to each other to provide the shortest return path, as well as higher capacitance.
- Have individual power and ground vias for each and every power and ground pin of the ADSP-SC5xx processor and the memory device to its associated plane.
- The decoupling capacitors for the VDD\_DMC power rail should be placed as close as possible to the pin. This is very important, as DDR signal slew rates are really aggressive, so having these capacitors closer to the pin is a must ([Figure 10](#)).

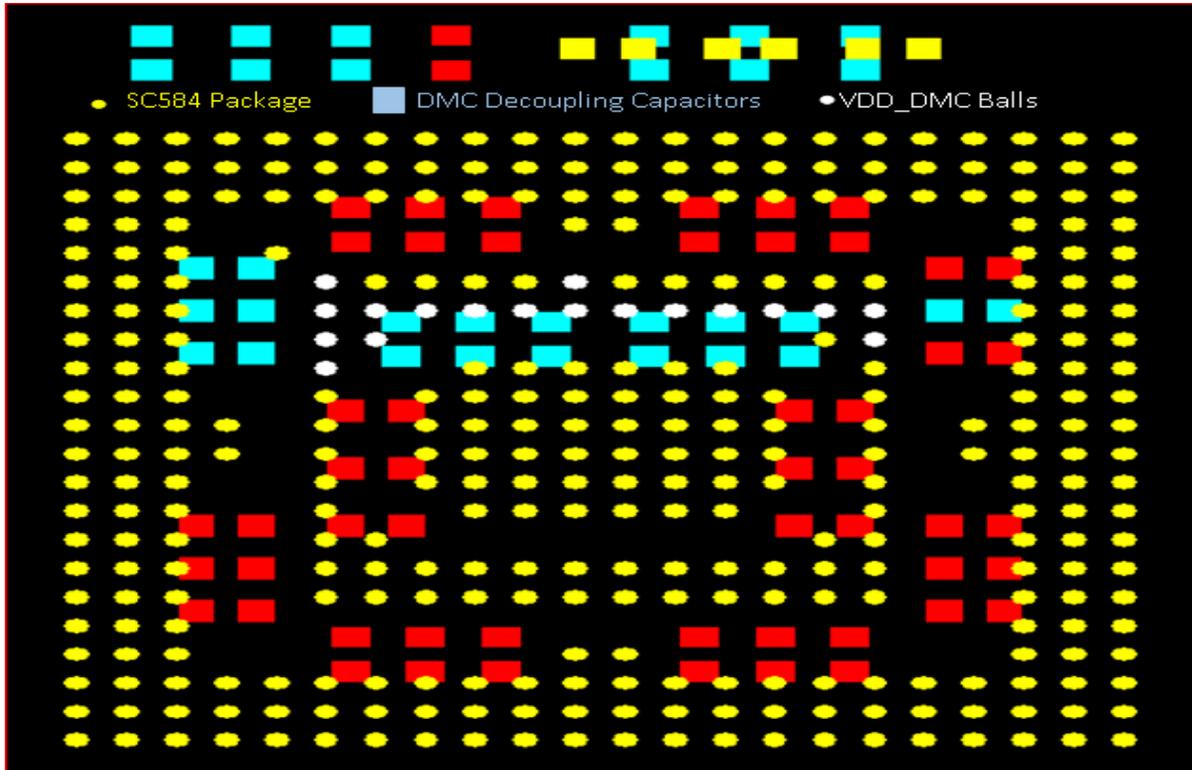


Figure 10. Sample VDD\_DMC Decoupling Capacitor Placement for the ADSP-SC584 Device

- Have dedicated power and ground vias for each and every decoupling capacitor pin, and sharing of vias should be not allowed.
- Ideally, the trace length from the power via to the device pad should not exceed 30 mils. The maximum trace length from power via to bypass capacitor is 60 mils. The maximum trace length from power via to power ball pad is 35 mils.
- Placement of the mid-bulk bypass capacitors (10  $\mu$ F) is not that critical, and they can be placed to accommodate other circuitry with more constrained placement and routing requirements.

### Clean DMC\_VREF Voltage Supply



DMC\_VREF acts as a voltage reference for DDR2/3 data signals and compares the difference between a steady reference voltage (VREF) and the signal received for identifying the logic. Hence, it is recommended to have them clean and noise-free.

- Route the DMC\_VREF trace away from high-speed signals and noisy power supplies with a distance of at least 40 mil.
- Guard traces can be provided around the DMC\_VREF trace, if required. Ensure that the guard traces have sufficient ground vias stitched to the main ground plane.
- Provide adequate decoupling near the DMC\_VREF pins of the ADSP-SC5xx processor, as well as the memory device.
- Keep the DMC\_VREF trace as short as possible with a width of at least 20 mils.

#### Recommended ADSP-SC5xx DMC\_VREF Filtering Scheme

- The RCR filtering scheme on DMC\_VREF is recommended for ADSP-SC5xx products, as shown in [Figure 11](#).

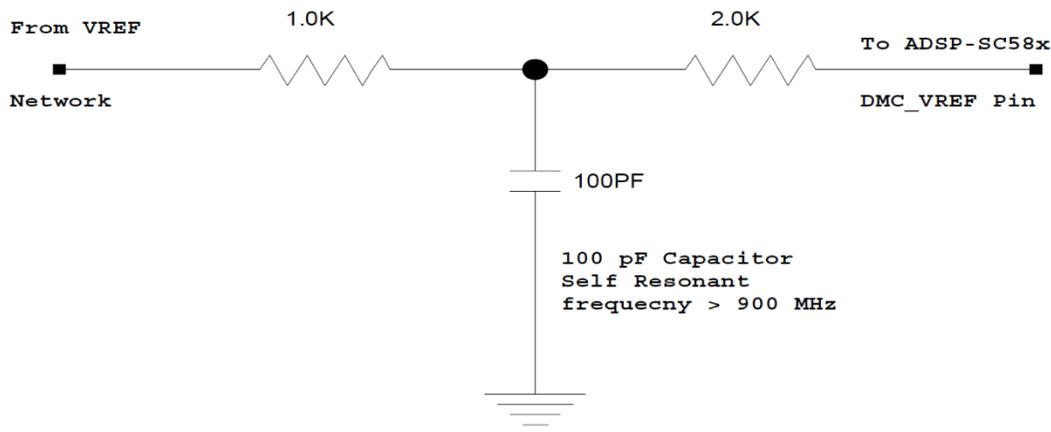


Figure 11. Recommended DMC\_VREF Filtering Network

### PCB Placement and Routing Guidelines for the DMC\_VREF Filter Network

- Place the recommended RCR network between the DMC\_VREF supply paths from the memory to the processor, as shown in [Figure 12](#). For more details, please refer to the ADSP-SC589 EZ-Board schematics<sup>[3]</sup>.

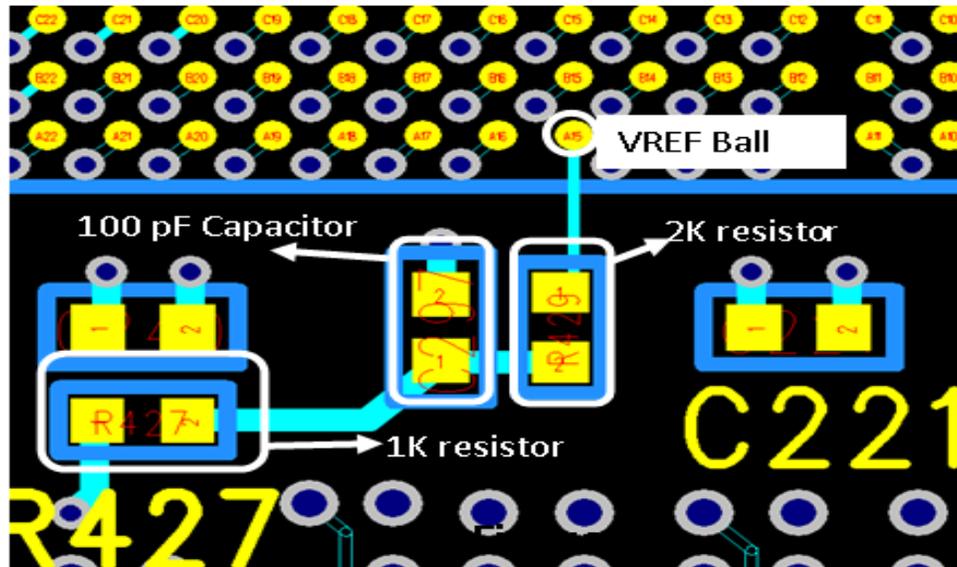


Figure 12. Recommended Placement and Routing for DMC\_VREF Filtering Network

- The C (100 pF) – R (2 kΩ) portion of the RCR network should be placed as close as possible to the ADSP-SC5xx DMC\_VREF pin.
- Use a small package, preferably 0402 size, for the 100pF capacitor to guarantee a high self-resonant frequency.
- The entire RCR network should be placed in one layer, with no vias allowed here in the trace.

### Memory-Side Recommendations for VREF Supply

There are unique requirements for the VREF supply, depending on whether the design is for DDR2 or DDR3 memory.

### VREF Supply Recommendations for DDR3 Mode

- DDR3 memory has two VREF pins, VREFCA and VREFDQ. VREFCA serves as the reference for clock, address, command, and control signals, and VREFDQ serves as the reference for strobe, data, and data mask signals.
- VREFCA and VREFDQ can have a common supply source but should be “STAR” routed and decoupled at the dedicated DRAM pins.
- Place two decoupling capacitors, 0.1 μF and 0.01 μF, for each VREF pin (VREFCA and VREFDQ).
- Place the 0.01 μF capacitor closer to the DRAM pin, followed by the 0.1 μF capacitor.
- Keep the length from the decoupling capacitor to the DRAM pin short with a wide trace.

[Figure 13](#) shows the recommended VREF supply circuit for a DDR3 memory device.

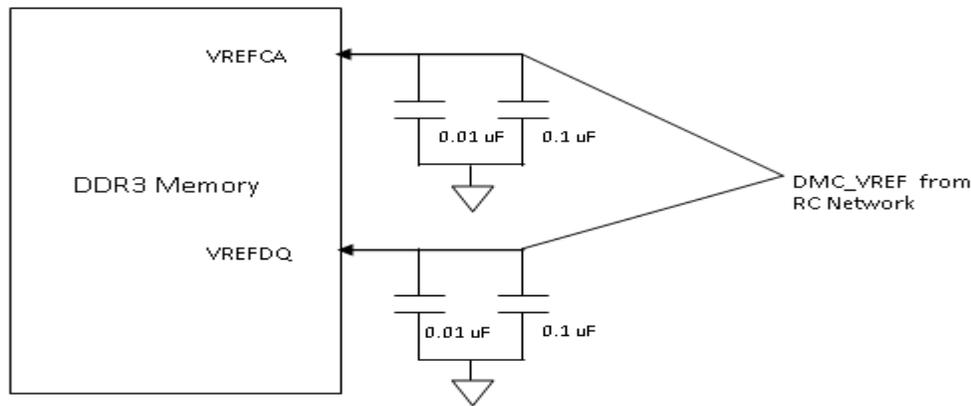


Figure 13. Recommended VREF Supply Circuitry for DDR3 Memory Device

### VREF Supply Recommendations for DDR2 Mode

- DDR2 memory has only one VREF pin. Add 0.01  $\mu\text{F}$  and 0.1  $\mu\text{F}$  decoupling capacitors on VREF close to the DDR2 SDRAM.
- Place the 0.01  $\mu\text{F}$  capacitor closer to the DRAM pin, followed by the 0.1  $\mu\text{F}$  capacitor.

Figure 14 shows the VREF supply circuitry used on the ADSP-SC589 EZ-Board for DDR3 mode, and Figure 15 shows the same for the ADSP-SC584 EZ-Board for DDR2 mode.

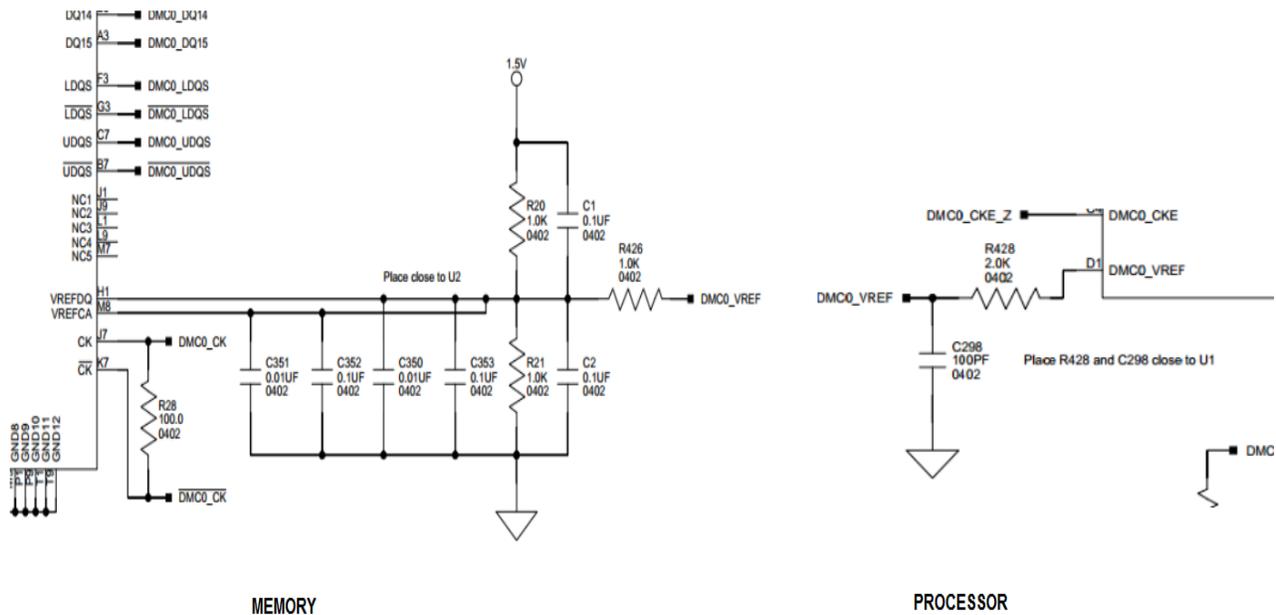


Figure 14. DMC\_VREF Circuit for DDR3 Mode on the ADSP-SC589 EZ-Board

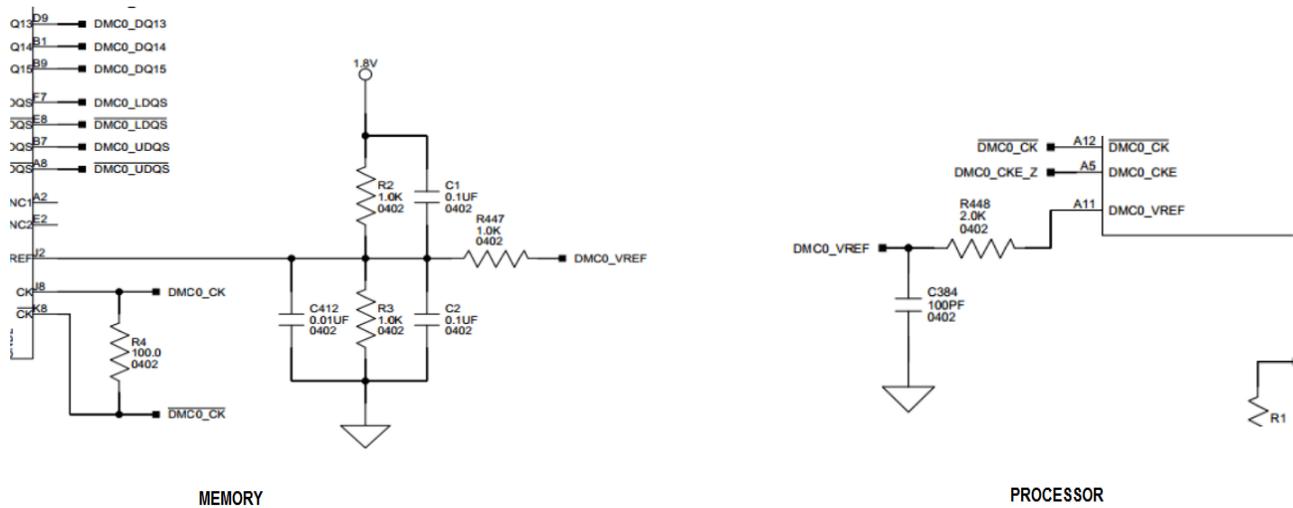


Figure 15. DMC\_VREF Circuit for DDR2 Mode on the ADSP-SC584 EZ-Board

## References

- [1] *ADSP-SC582/583/584/587/589/ADSP-21583/584/587 Dual-Core SHARC+ and ARM Cortex-A5 SOC Data Sheet*. Rev PrG, June 2016. Analog Devices, Inc.
- [2] *ADSP-SC58x SHARC Processor Hardware Reference Manual*, Rev 0.3, April 2016. Analog Devices, Inc.
- [3] *ADSP-SC589 EZ-Board® Evaluation System Manual*, Rev 1.0.0, May 2015. Analog Devices, Inc.
- [4] *ADSP-SC584 EZ-Board® Evaluation System Manual*, Rev 1.0.0, May 2015. Analog Devices, Inc.
- [5] *ADSP-SC582/583/584/587/589/ADSP-21583/584/587 SHARC+ Dual-Core DSP with ARM Cortex-A5 Silicon Anomaly List*. Rev A, June 2015. Analog Devices, Inc.
- [6] *Associated ZIP File (EE387v02.zip) for Interfacing DDR3/DDR2/LPDDR Memory to ADSP-SC5xx/215xx Processors (EE-387)*. July 2016. Analog Devices, Inc.
- [7] *MT41K128M16 DDR3L SDRAM Data Sheet*. Rev K, September 2013. Micron Technology, Inc.
- [8] *MT47H128M16 DDR2 SDRAM Data Sheet*. Rev H, November 2010. Micron Technology, Inc.
- [9] *MT46H128M16LF LPDDR SDRAM Data Sheet*. Rev F, September 2014. Micron Technology, Inc.
- [10] *ADSP-2146x Board Design Guidelines for DDR2 Memory (EE-349)*. Rev 3, November 2012. Analog Devices, Inc.
- [11] *Hardware Tips for Point-to-Point System Design (TN-46-14)*. Rev B, June 2008. Micron Technology, Inc.
- [12] *USER'S MANUAL: New Features of DDR3 SDRAM*. Ver.1.0, 2009. © Elpida Memory, Inc.
- [13] *ADSP-SC57x SHARC Processor Hardware Reference Manual*, Rev 0.1, June 2016. Analog Devices, Inc.
- [14] *ADSP-SC570/571/572/573 ADSP-21571/573 SHARC+ Dual-Core SHARC+ and ARM Cortex-A5 SOC Data Sheet*. Rev PrA, June 2016. Analog Devices, Inc.
- [15] *ADSP-SC570/571/572/573/ADSP-21571/573 SHARC+ Dual-Core DSP with ARM Cortex-A5 Silicon Anomaly List*. Rev A, June 2016. Analog Devices, Inc.
- [16] *ADSP-SC573 EZ-Board® Evaluation System Manual*, Rev 1.0.0, June 2016. Analog Devices, Inc.

## Document History

Revision	Description
<i>Rev 2 – July 7<sup>th</sup>, 2016 by Sachin-V Kumar</i>	Title, content, and code examples updated to include ADSP-SC57x/ADSP-2157x processors
<i>Rev 1 – December 17th, 2015 by Balaji Kannan, Mitesh Moonat, and Gaurav Singh</i>	Initial Release