AN12026

SDRAM interface to LPC546xx external memory controller

Rev. 1.0 — 18 August 2017 Application note

Document information

Info	Content
Keywords	LPC546xx, LPC5460x, EMC, SDRAM
Abstract	This application note describes how to interface SDRAM memory to the LPC546xx External Memory Controller.

Revision history

Rev	Date	Description
1.0	20170818	Initial version

Contact information

For more information, please visit: http://www.nxp.com

1. Introduction

The LPC546xx 32-bit microcontroller families are designed for embedded applications requiring high performance and low power consumption. The LPC43xx and LPC18xx use the same External Memory Controller (EMC), an ARM PrimeCell MultiPort Memory Controller peripheral. It supports the asynchronous static memory devices such as RAM, ROM and flash, as well as the dynamic memories such as Single Data Rate (SDR) SDRAM. For microcontrollers with a requirement of external memory more than 2 MB, SDRAM is a practical solution.

This application note focuses on the interface connectivity and board layout guidelines while using the SDR SDRAM memories with the LPC546xx microcontrollers.

The LPC546xx user manuals will be referenced in this application note, so keep the corresponding user manual at hand.

2. External Memory Controller(EMC)

2.1 EMC key features

The LPC546xx EMC is an ARM PrimeCell MultiPort Memory Controller peripheral supporting asynchronous static memory devices such as RAM, ROM and flash, and dynamic memories such as SDR SDRAM. The EMC does not support double data rate (DDR) SDRAMs and synchronous static memory devices (synchronous burst mode). EMC SDRAM supports the following features:

- Dynamic chip selects: nDYCS0 to nDYCS3
- Dynamic chip selects each supporting up to 256 MB of SDRAM per DYCS
- 16-bit and 32-bit wide chip select SDRAM memory support
- Power saving modes dynamically control CKE and CLKOUT to SDRAMs
- Dynamic memory self-refresh mode controlled by software
- 2 kbit, 4 kbit, and 8 kbit row address synchronous memory parts
  - Typically, 512 Mb, 256 Mb, and 128 Mb parts, with 4, 8, 16, or 32 data bits per device
- If required, separate reset domains allow auto-refresh through a chip reset
- Programmable command delay and feedback delay elements allow fine-tuning EMC timing

Table 1. EMC memory map

Chip select pin	Address range	Memory type	Size of range
EMC_CS0	0x8000 000 – 0x83FF FFFF	Static	64 MB
EMC_$\overline{CS1}$	0x9000 0000 – 0x93FF FFFF	Static	64 MB
EMC_$\overline{CS2}$	0x9800 0000 – 0x9BFF FFFF	Static	64 MB
EMC_$\overline{CS3}$	0x9C00 0000 – 0x9FFF FFFF	Static	64 MB
EMC_$\overline{DYCS0}$	0xA000 0000 – 0xAFFF FFFF	Dynamic	256 MB
EMC_$\overline{DYCS1}$	0xB000 0000 – 0xBFFF FFFF	Dynamic	256 MB
EMC_$\overline{DYCS2}$	0xC000 0000 – 0xCFFF FFFF	Dynamic	256 MB
EMC_$\overline{DYCS3}$	0xD000 0000 – 0xDFFF FFFF	Dynamic	256 MB

2.2 EMC pins

The EMC supports 8/16/32-bit data bus on BGA180 and LQFP208 and 8/16-bit data bus on BGA100 package. The EMC supports a 16-bit or 32-bit wide data interface built from 8, 16 or 32-bit SDRAM chips. When mixed with SDRAM, the static memory data bus may be 8-bit, 16-bit or 32-bit. The EMC signals are multiplexed at the chip port pins with other functions. Every EMC signal is available on only one fixed pin. So, allocate pins for required EMC signals first before allocating pins for other peripherals. Refer to LPC546xx User Manual or datasheet for pin MUX details. The EMC functions required for interfacing with SDRAM and SRAM are listed in Fig 1 and Table 2.

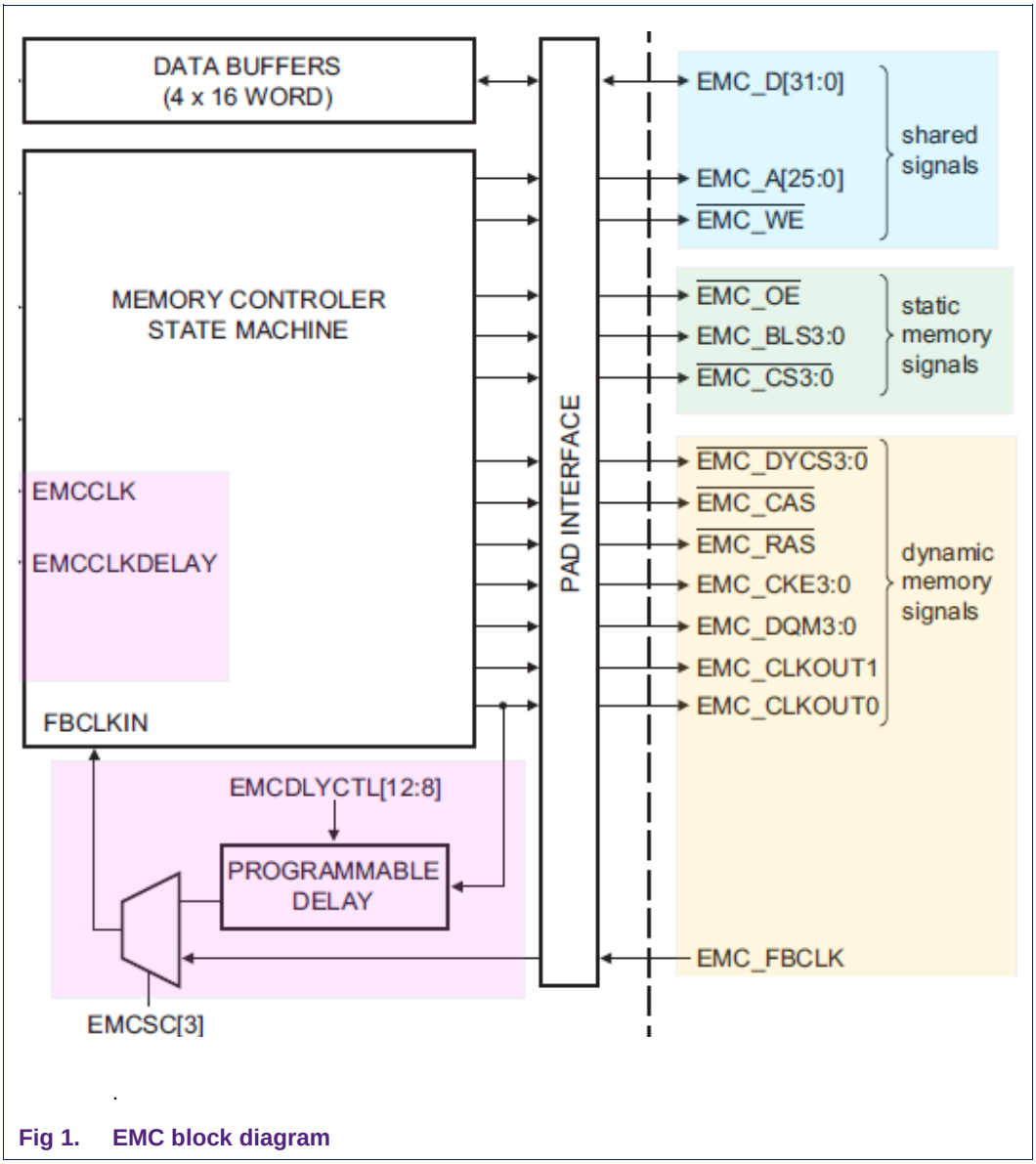


Fig 1. EMC block diagram

Table 2 lists the EMC pins

Table 2. EMC signals

Signal	SRAM function	SDRAM function	Notes
EMC_A12:0	memory address [1]	row/column address	-
EMC_A14:13	memory address [1]	bank	EMC_A13 to Bank0; EMC_A14 to Bank1 If only 1 bank pin is required, 8-bit SDRAM chips uses EMC_13 is used for bank if there is one
EMC_A23:15	memory address [1]	-	-
EMC_D31:0 [1]	data	data	-
EMC_OE [1]	OE	-	active LOW
EMC_BLS3:0	byte lane/8b: byte WE [1]	-	active LOW; specific behavior is controlled by STATICCONFIG[0:3] register PB bit
EMC_CS3:0	chip select [1]	-	active low
EMC_WE	write enable [1]	write enable	active LOW; defines command to SDRAM. SRAM Specific behavior is controlled by STATICCONFIG[0:3] register PB bit. [1]
EMC_DYCS3:0	-	chip select	active LOW
EMC_CKE3:0	-	clock enable	active LOW, one for each DYCS.
EMC_CAS	-	column address strobe	defines command to SDRAM along with RAS, WE and DYCS.
EMC_RAS	-	row address strobe	defines command to SDRAM along with CAS, WE and DYCS
EMC_DQM3:0	-	data mask	active HIGH; one mask bit for each data byte lane
EMC_CLK1:0	-	clock	SDRAM clock input; EMC_CLK0 has loopback signal for signal flight compensation
EMC_FBCLK	-	feedback clock pin	EMC can use either EMC_FBCLK pin or EMC_CLK0 loopback signal as its feedback input; for EMC_CLK0 loopback signal, programmable delays can be added to clock feedback path, to compensate signal flight time

[1] not used by SDRAM.

Note: The 100-pin package of LPC546xx supports SDRAM. But, as the package has only one bank select signal (A13, **no** A14), it can access only two banks.

2.2.1 SDRAM clock enable when using multiple DYCSs

For systems using multiple chip selects, the CKE3:0 should be connected to all SDRAMs on EMC_DYCS3:0. It enables SDRAMs on all chip selects to be placed in self-refresh mode and allows use of the dynamic memory clock enable bit in the

EMCDynamicControl register to negate clock enable of an idle dynamic chip select to reduce overall system power.

2.3 Supported SDRAM cell layouts

Though EMC_DYCS3:0 each chip select addresses 256 MB, not all the layout of SDRAM cells are supported. The maximum supported single SDRAM size is 64 MB (512 Mb). Table 3 summarizes 23 supported SDRAM cell layouts.

Table 3. EMC SDRAM cell layout summary

DYNAMICCONFIG register bit-field				SDRAM geometry			SDRAM capacity		
14	12	11:9	8:7	Banks	Rows	Columns	Cells	cell width	Capacity (Mbits)
16-bit bus									
0	X	000	00	2	11	9	2M	8	16
0	X	000	01	2	11	8	1M	16	16
0	X	001	00	4	12	9	8M	8	64
0	X	001	01	4	12	8	4M	16	64
0	X	010	00	4	12	10	16M	8	128
0	X	010	01	4	12	9	8M	16	128
0	X	011	00	4	13	10	32M	8	256
0	X	011	01	4	13	9	16M	16	256
0	X	100	00	4	13	11	64M	8	512
0	X	100	01	4	13	10	32M	16	512
32-bit bus									
1	X	000	00	2	11	9	2M	8	16
1	X	000	01	2	11	8	1M	16	16
1	X	001	00	4	12	9	8M	8	64
1	X	001	01	4	12	8	4M	16	64
1	X	001	10	4	11	8	2M	32	64
1	X	010	00	4	12	10	16M	8	128
1	X	010	01	4	12	9	8M	16	128
1	X	010	10	4	12	8	4M	32	128
1	X	011	00	4	13	10	32M	8	256
1	X	011	01	4	13	9	16M	16	256
1	X	011	10	4	13	8	8M	32	256
1	X	100	00	4	13	11	64M	8	512
1	X	100	01	4	13	10	64M	16	512

Note: Bit 12 of DYNAMICCONFIG selects addressing mode:

- 0 = bank-row-column (BRC), AHB address bits from MSB to LSB maps to bank, row, column.
- 1 = row-bank-column (RBC), AHB address bits from MSB to LSB maps to row, bank, column.

Note: For SDRAM chips with lower data width than bus width, multiple chips form the data bus. For example, for 32-bit bus with 8-bit SDRAM chips, four chips form the 32-bit bus, and total SDRAM capacity is 32 / 8 = 4 times the single SDRAM chip capacity.

2.3.1 Using 32-bit SDRAMs with cell layout not supported by EMC

There are 256 Mbit (8M x 32), 512 Mbit (16M x 32) and 1 Gbit (32M x 32) SDR SDRAMs that have a row and column configuration that is not listed in the Table 3. These devices can be used by setting the EMCDynamicConfig Address Mapping (AM) field as if two 16-bit devices with the same bank/row/column mapping are being used.

2.4 EMC clocking to SDRAM

In LPC5460x, the clock source of the EMC block is a branch from system clock which supplies CPU, AHB and Sync APB. The EMC clock branch has its dedicated clock divider to divide system clock by 1, 2, 3...256.

At the SDRAM, all input signals are sampled on the positive edge of the EMC_CLK1:0 (usually EMC_CLK0 is used) clock pin. The clock pin also increments the SDRAM internal burst counter and controls the output registers.

3. SDRAM connections

The LPC546xx EMC supports either a 16-bit wide or 32-bit wide system SDR SDRAM bus. The system SDR SDRAM bus may be constructed with:

- one 32-bit wide SDR SDRAM
- one or two 16-bit wide SDR SDRAMs
- two or four 8-bit wide SDR SDRAMs

3.1 Pin connections for using single SDRAM chip

Connections to a single x32 SDR SDRAM is shown in Fig 2.

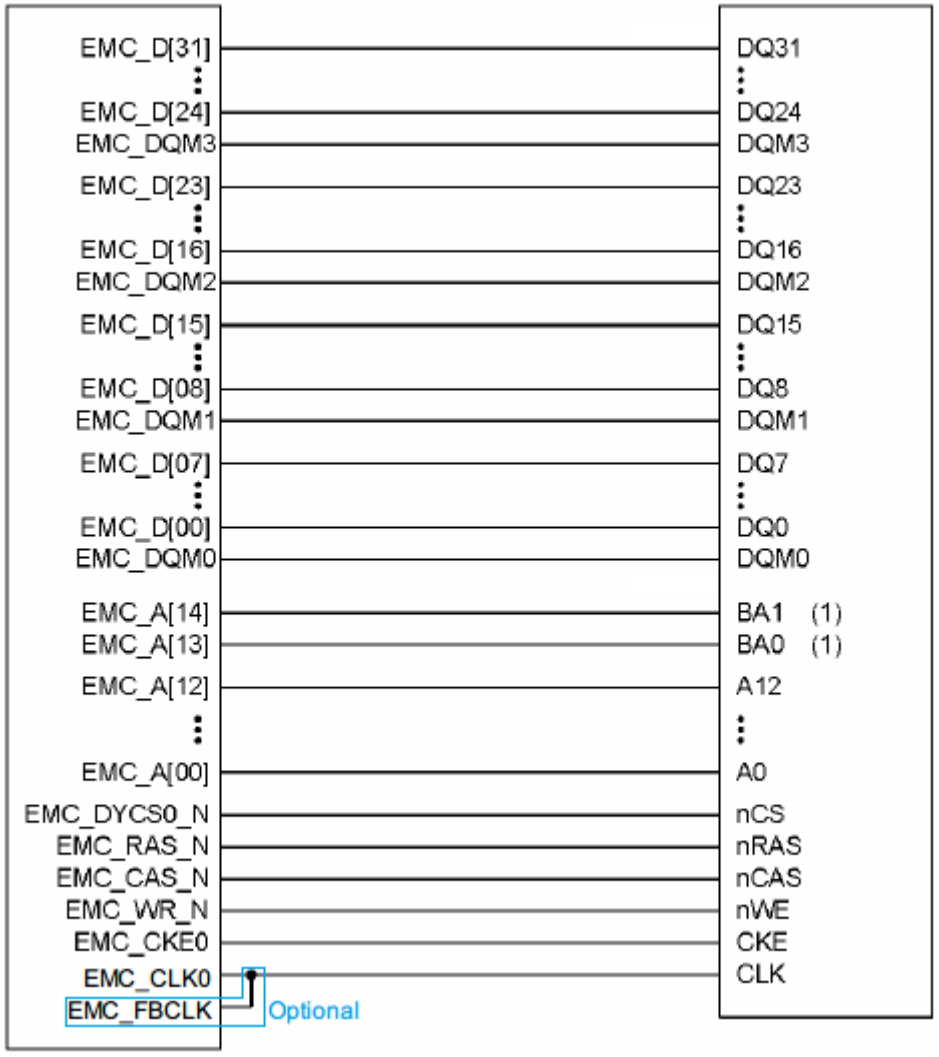


Fig 2. Block diagram for connecting a single SDRAM chip

Note that the "EMC_FBCLK" is optional. EMC can alternatively use loopback signal of EMC_CLK0 with programmable delay (from 0.25 ns to 8 ns) as feedback clock. The arrangement saves one pin. However, the user should carefully determine the delay value and ensure that it can compensate the signal flight time which is determined by the PCB trace length of CLK signal.

Fig 3 shows an example of single chip 16-bit data bus.

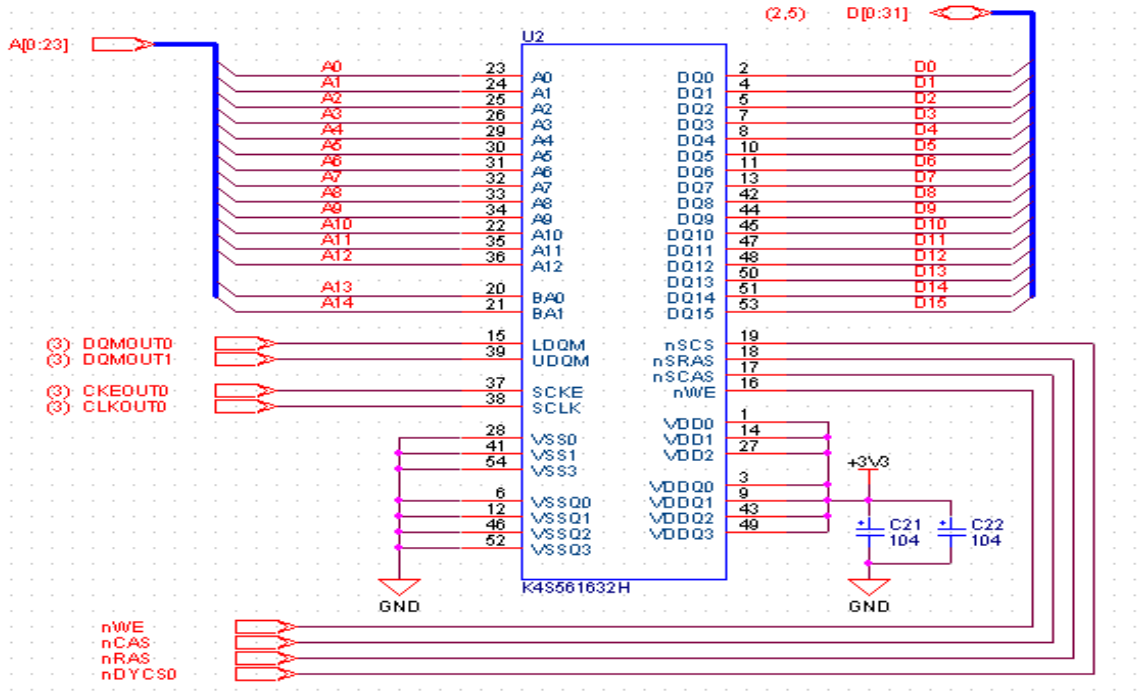


Fig 3. Example of connecting a single 16-bit SDRAM chip

Fig 4 shows an example of single chip 32-bit data bus.

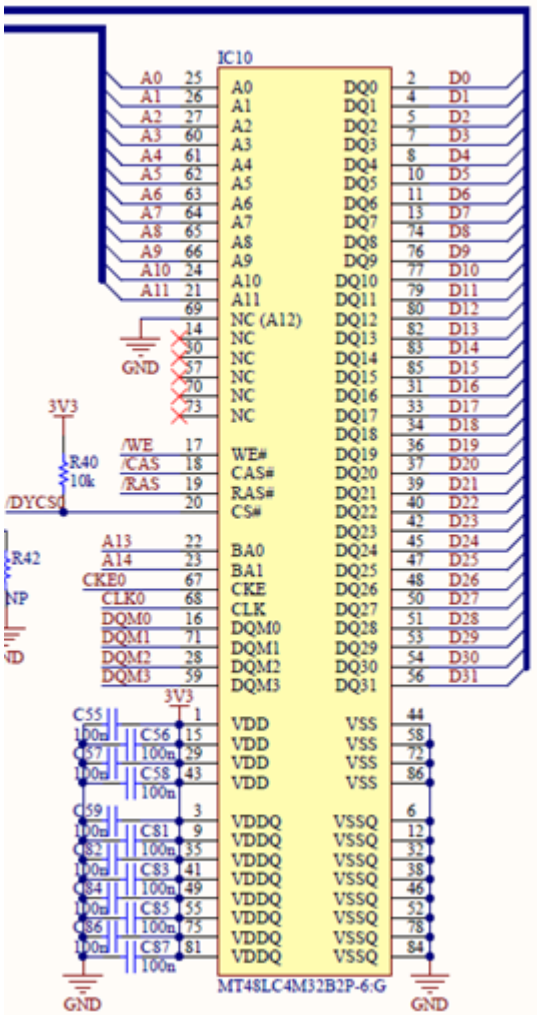


Fig 4. Example of connecting a single 32-bit SDRAM chip

3.2 Pin connections for using multiple SDRAM chips

For a single x32 SDR, the maximum memory for each EMC_DYCS[3:0] is 64 MB (16 M × 32). For two x16 SDRs, the maximum memory for each EMC_DYCS[3:0] is 128 MB (two 32 M × 16). For four x8 SDRs the maximum memory for each EMC_DYCS[3:0] is 256 MB (four 64M × 8). Using x8 SDRAMs is **not** recommended, as it is an expensive configuration. Also, a high capacitive loading on the EMC address and control signals to all SDRAMs reduces the maximum operational speed of the interface.

To extend total SDRAM capacity, users can use two 8-bit SDRAM chips to form 16-bit data bus, two 16-bit SDRAM chips to form 32-bit data bus, or four 8-bit SDRAM chips to form 32-bit data bus. By *splitting* the data bus, most applications require only a single

dynamic chip select (DYNCSx). By data bus splitting, the data bus is broken into byte lanes and only goes to a single SDRAM. It ensures data integrity and lower capacitive loads. EMC performance may be compromised when PCB traces are longer than 6” or capacitive loads exceed those listed in the datasheet under dynamic characteristics, dynamic external interface. Even if a single DYCS is used, a single SDRAM with either an x16 or x32 data bus provides minimum loading to the EMC. It results in maximum EMC performance. Table 4 shows signal connection when more than one SDRAM chips are used.

Table 4. Signals of connecting to multiple SDRAM chips, summary

Two 16-bit SDRAM chips form 32-bit bus				
	Chip #0	Chip #1	-	-
Data	15:0	31:16	-	-
Address and bank [14:0]	shared			-
Data mask (DQM)	1:0	3:2	-	-
CLK#, CKE#, RAS, CAS, WE	shared			-
FBCLK (optional)	shared, if used, short to CLK0		-	-
Two 8-bit SDRAM chips form 16-bit bus.				
	Chip #0	Chip #1	-	-
Data	7:0	15:8	-	-
address & bank [14:0]	shared			-
Data mask (DQM)	0	1	-	-
CLK#, CKE#, RAS, CAS, WE	shared		-	-
FBCLK (optional)	shared, if used, short to CLK0		-	-
Four 8-bit SDRAM chips form 32-bit bus.				
	Chip #0	Chip #1	Chip #2	Chip #3
Data	7:0	15:8	23:16	31:24
address & bank [14:0]	shared			
Data mask (DQM)	0	1	2	3
CLK#, CKE#, RAS, CAS, WE	shared			

Note: For one DYCS, regardless of how many chips form the data bus, only one CLK and CKE is used. For example, for DYCS0, usually CLK0 and CKE0 are connected to all SDRAM chips.

Fig 5 is an example of using two 16-bit SDRAM chips to form 32-bit data bus:

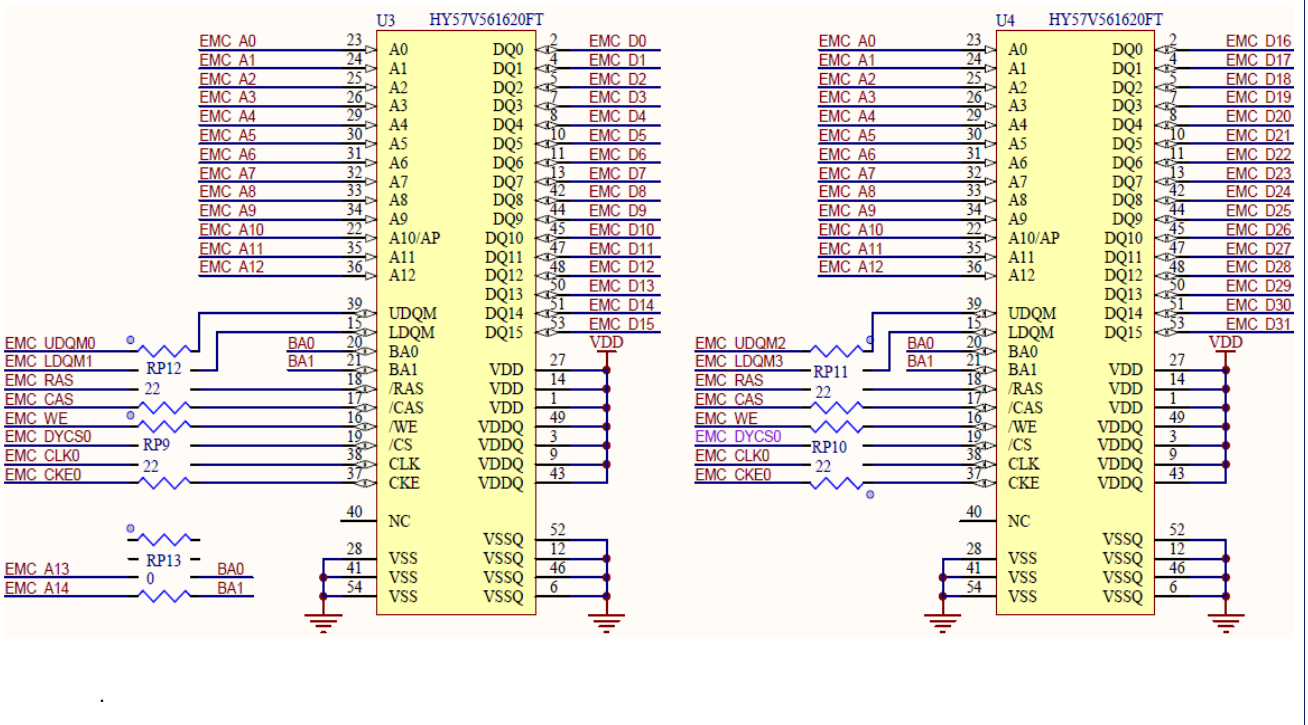


Fig 5. Example of connecting 2 16-bit SDRAM chips to form 32-bit bus

Note: The CLK pins usage for LPC18xx/43xx is different. Do not mix LPC5460x with LPC18xx/43xx.

4. SDRAM controller programming and operations

SDRAM devices are more complex to setup than standard asynchronous memories. In addition to setting up the interface signal timings, factors such as RAS and CAS latency (also referred to as RAS and CAS delay), burst size, and refresh timing are considered.

SDRAM setup is procedural and requires a series of steps to initialize and configure SDRAM.

4.1 Typical SDRAM programming sequence

To program the EMC controller for a specific SDRAM device, the following information about the system using the SDRAM devices is required:

- Desired clock speed for the SDRAM access speed. It can be up to the CPU clock rate (branched from system clock) with the divider of EMC
- SDRAM types (voltage): Low power (mobile) or standard SDRAM. The EMC supports JEDEC low-power SDRAM partial array refresh
- SDRAM geometry: Number of columns, rows, and banks of the device

- Expected RAS and CAS cycles (may be limited by memory speed and board design)
- SDRAM access (or address) strategy: low-power or high-performance mode
- SDRAM data bus width
- SDRAM interface timings (from the SDRAM datasheet)
- Dynamic refresh period timing

4.1.1 Standard and low-power SDRAM programming sequence

Prior to these steps, it is assumed that the main system clocking is already setup and the system is running at the desired clock rate.

Note: The programming sequence discussed below is based on the JDEC programming sequence and includes the steps to setup the SDRAM and the controller. The low power and standard SDR SDRAM programming sequences are similar, except for an extra extended mode register programming cycle on low-power devices.

The programming sequences are only meant as guidelines.

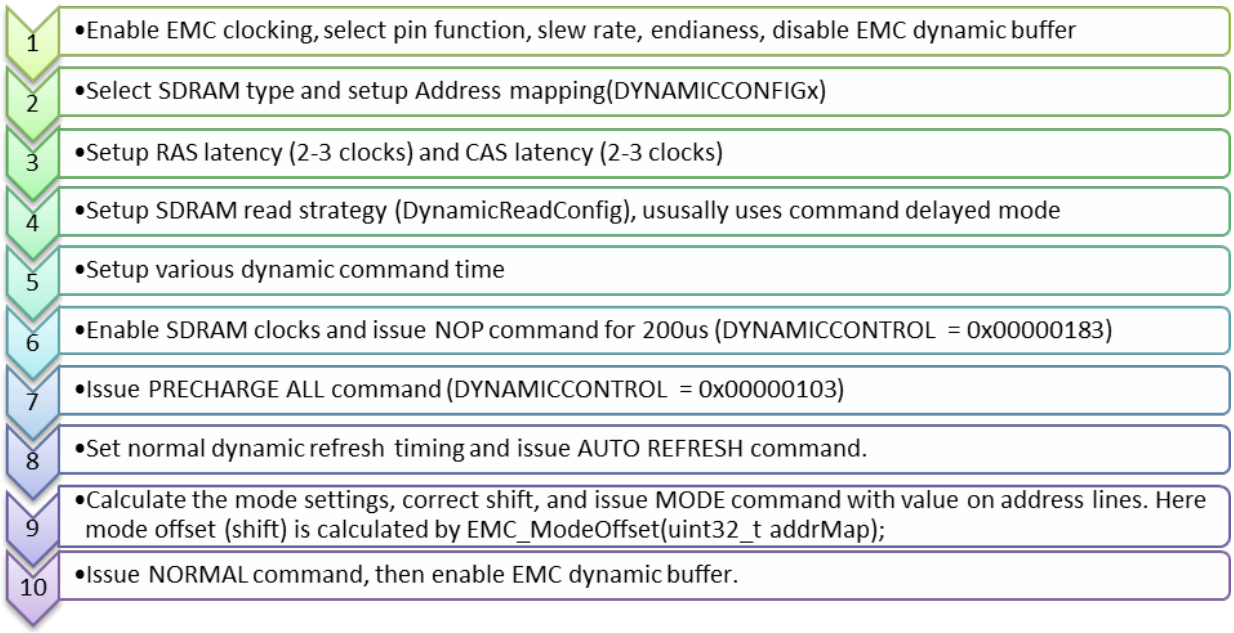


Fig 6. SDRAM initialization sequence

Step 1: Enable EMC clocking, select pin function, slew rate, endian format, and disable EMC dynamic buffer

Before any access to the EMC, the EMC clock should be enabled. The EMC interface is enabled by setting bit 0 of the EMC control register. The endian format can be selected by the EMC configuration register. Usually, little-endian format is used.

SDK 2.0 provides an useful function *EMC_Init( )* to handle EMC initialization.

Note: To download SDK 2.0 for LPC546xx, go to https://mcuxpresso.nxp.com/en/welcome, and click *SDK Builder*. Login is required to setup SDK configurations and download generated SDK package.

Step 2: Select SDRAM type and setup address mapping (DYNAMICCONFIG#)

Setup the row/column/bank address mapping and DRAM type used by the interface. It can be selected by configuring the EMC Dynamic Memory Configuration 0 register.

For the Address Mapping (AM) field (bits 14, 12...7), select a device that matches the SDRAM data bus width (16-bit or 32-bit) and has the same row/column/bank numbers as the SDRAM device(s). Note that the listed size for a mapping is not important. Only the row/column/bank mapping must match for the selected data bus width. Multiple configurations may exist with the same mapping but with different listed sizes. Any one of the mapping can work. If a device is also connected to the second SDRAM chip select, it also requires address mapping to be setup in the EMC Dynamic Memory Configuration 1 register.

Step 3: Setup the RAS latency (2-3 clocks) and CAS latency (2-3 clocks)

Setup the RAS and CAS latencies in the EMC Dynamic Memory RAS and CAS Delay 0 register. The CAS latency should be setup in full clock cycles and bit 7 in the CAS latency field should always be 0. If a device is also connected to the second SDRAM chip select, the CAS and RAS latencies should be setup in the EMC Dynamic Memory RAS and CAS Delay 1 register.

Step 4: Setup SDRAM read strategy (DynamicReadConfig); uses command delayed mode

The SDRAM read strategy is setup in the EMC Dynamic Memory Read Configuration register. The positive capture edge should be used for the SDR_SRP and the command delayed by COMMAND_DELAY time should be used for the SDR_SRD fields.

Step 5: Setup various dynamic command time

Setup the interface timing parameters for the SDRAM device in the EMC Dynamic Memory timing registers. These registers control the number of bus clocks cycles for which a specific timing value is delayed. The value for these registers is obtained from the User's Guide or datasheet for the SDRAM device. A list of the registers to be setup is shown in Table 4.

Table 5. EMC Dynamic Memory timing registers

Timing Registers[0]	Description
tRCD(DynamicRASCAS.[1:0])	Bank and Row (active) to column delay (also known as RAS latency)
tRP(DynamicRP)	precharge time
tRAS(DynamicRAS)	last active to precharge time
tSREX(DynamicSREX)	self-refresh exit time
tAPR(DynamicAPR)	last data-read to active time (auto precharge, same bank), set it to 0 (1 clock)
tDAL(DynamicDAL)	last data-write to active delay; tDAL = tWR + tRP
tWR(DynamicWR)	last data-write to precharge delay (also called as tDPL, tRWL, tRDL)
tRC(DynamicRC)	last active to active time; tRC = tRAS + tRP

tCL(DynamicRASCAS.[9:8])	CAS latency (from CAS to data out)
tRFC(DynamicRFC)	auto-refresh period (duration of one refresh operation)
tXSR(DynamicXSR)	exit self-refresh to active time, simply set it to maximum
tRRD(DynamicRRD)	bank switch delay
tMRD(DynamicMRD)	mode register set to active delay

[0] Values in these registers are based on the bus (EMCCLK) period.

Step 6: Enable SDRAM clocks and issue NOP command for 200 µs (DYNAMICCONTROL = 0x00000183)

While issuing the NOP command (bits 8...7) for 200 µs, setup the clock enables (bit 0) and clocks (bit 1) to be always active. The inverted memory clock (bit 4) used for DDR mode should be disabled. They are setup in the EMC Dynamic Memory Control register.

Step 7: Issue PRECHARGE-ALL command (DYNAMICCONTROL = 0x00000103)

While issuing the PRECHARGE-ALL command (bits 8...7) for 10 µs, setup the clock enables (bit 0) and clocks (bit 1) to be always active.

Note: The dynamic refresh must be set to a fast rate during this command so that at least a few dynamic refresh cycles occur during the 10 µs period.

Step 8: Set normal dynamic refresh timing and issue AUTO REFRESH command

Set the EMC Dynamic Memory Refresh Timer register to the correct value for an optimal refresh period.

Step 9: Issue MODE command

Calculate the mode settings, correct shift, and issue MODE command with value on the address lines. Here, mode offset (shift) is calculated by EMC_ModeOffset(uint32_t addrMap).

While issuing the mode command (bits 8...7) in the EMC Dynamic Memory Control register, setup the clock enables (bit 0) and clocks (bit 1) to be always active. Issue the mode word to the device by writing a value in the mode word address; see Fig 7. Also, see Section 2.3 Supported SDRAM cell layouts for supported cell layouts.

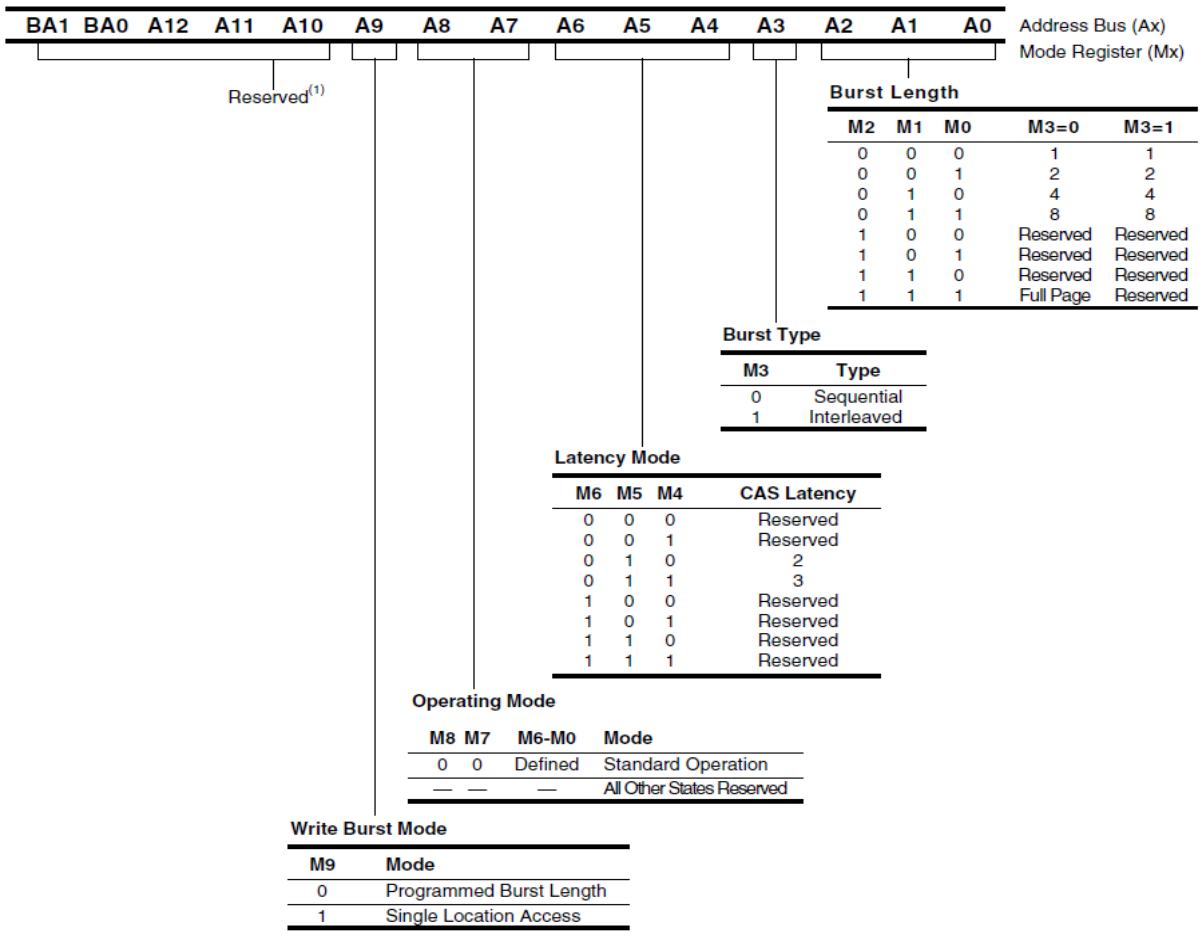


Fig 7. SDRAM "MODE" register layout

- Address for the mode word operation

address = SDRAM_BASE + (mode_word << (offset))

- Note that the mode value is represented on address lines with offset. The <offset> value is calculated as below; see SDK *`EMC_ModeOffset()`* API.

BRC mapping/16-bit device: offset = (column bits) + 1

RBC mapping/16-bit device: offset = (column bits + bank bits) + 1

BRC mapping/32-bit device: offset = (column bits) + 2

RBC mapping/32-bit device: offset = (column bits + bank bits) + 2

The value written to the address is not important. After the write, provide a small delay of about 1 µs before performing SDRAM operations.

The mode word is used to specify the burst length and CAS latency. For 32-bit interfaces, the burst length is 1. For 16-bit interfaces, the burst length is 2. The CAS latency must match the latency programmed into the EMC Dynamic Memory RAS and CAS Delay 0 registers.

If a device is also connected to the second SDRAM chip select, it also needs mode word configuration.

Step 10: Issue NORMAL command, then enable EMC dynamic buffer

After the one-shot but long initialization sequence, SDRAM can be accessed normally.

4.1.2 SDRAM configuration example (Board_InitSDRAM())

Let us see an example of SDRAM configuration. It is based on the SDRAM part MT48LC8M16A2B4-6A on LPCXpresso54608 board. The related code is in board.c.

Through the SDRAM data sheet, we define the dynamic parameters in the following macros.

```
#define SDRAM_REFRESHPERIOD_NS (64 * 1000000 / 4096) /* 4096 rows/ 64ms */
#define SDRAM_TRP_NS (18u)
#define SDRAM_TRAS_NS (42u)
#define SDRAM_TSREX_NS (67u)
#define SDRAM_TAPR_NS (18u)
#define SDRAM_TWRDELT_NS (6u)
#define SDRAM_TRC_NS (60u)
#define SDRAM_RFC_NS (60u)
#define SDRAM_XSR_NS (67u)
#define SDRAM_RRD_NS (12u)
#define SDRAM_MRD_NCLK (2u)
#define SDRAM_RAS_NCLK (2u)
#define SDRAM_MODEREG_VALUE (0x23u)
#define SDRAM_DEV_MEMORYMAP (0x09u) /* 128Mbits (8M*16, 4banks, 12 rows, 9
columns)*/
```

SDK has an API named *EMC_DynamicMemInit( )* that accepts these parameters through a structure emc_dynamic_timing_config_t. In *BOARD_InitSDRAM( )*, we define an instance as emc_dynamic_chip_config_t dynChipConfig and fill it with our defined timing parameters.

```
dynTiming.readConfig = kEMC_Cmddelay;
dynTiming.refreshPeriod_Nanosec = SDRAM_REFRESHPERIOD_NS;
dynTiming.tRp_Ns = SDRAM_TRP_NS;
dynTiming.tRas_Ns = SDRAM_TRAS_NS;
dynTiming.tSrex_Ns = SDRAM_TSREX_NS;
dynTiming.tApr_Ns = SDRAM_TAPR_NS;
dynTiming.tWr_Ns = (1000000000 / CLOCK_GetFreq(kCLOCK_EMC) + SDRAM_TWRDELT_NS);
/* one clk + 6ns */
dynTiming.tDal_Ns = dynTiming.tWr_Ns + dynTiming.tRp_Ns;
dynTiming.tRc_Ns = SDRAM_TRC_NS;
dynTiming.tRfc_Ns = SDRAM_RFC_NS;
dynTiming.tXsr_Ns = SDRAM_XSR_NS;
dynTiming.tRrd_Ns = SDRAM_RRD_NS;
dynTiming.tMrd_Nclk = SDRAM_MRD_NCLK;
```

To use other SDRAM, user should check the parameters and update the macros.

Besides dynamic parameter configuration, there are also some important EMC basic configurations, including endian format, feedback clock mode, and EMC clock divider. As an example, for CPU at 180 MHz, if the SDRAM operates at a maximum frequency of 133 MHz, the EMC should divide clock by two. If CPU is 96 MHz from FRO, then EMC clock does not need to divide.

```
28    emc_basic_config_t basicConfig;
29    basicConfig.endian = kEMC_LittleEndian;
30    basicConfig.fbClkSrc = kEMC_IntloopbackEmcclk;
31    basicConfig.emcClkDiv = 2 - 1;  // EMC clock = system clock / 2 when CPU is
      180MHz, divider = emcClkDiv + 1.
```

5. Schematics, PCB design tips and suggestions

EMC CLKx signals should be routed using point-to-point routing topology. When multiple SDRAMs are connected, use one CLKx pin to drive one SDRAM. For a design that requires more than four SDRAMs, each CLKx pin may have two SDRAM clocks connected in a daisy-chain, placing one SDRAM at the end of the daisy-chain and the other no more than 1 inch from the end of the daisy-chain, keeping any branch stub to 0.5 inch. EMC signals that go to multiple devices excluding CLKx pins, should be routed in a daisy-chain with branch stubs no longer than 2 inches.

It is possible to route the EMC CLK and other signals in a star topology. The length of each PCB trace in the star should match with each other. Each PCB trace characteristic impedance should be a little more than twice the source impedance of the EMC I/O buffer. It is strongly recommended to use star topology routing only after using LPC18xx/43xx and SDRAM Ibis models along with signal integrity software to simulate your design.

5.1 Shared EMC configuration

The EMC bus can be shared by multiple devices, such as SDRAM, NOR flash, SRAM or devices with an SRAM-type interface.

When the design requires EMC bus to be shared, the following steps should be followed:

- Route the EMC bus in a daisy-chain topology
- In daisy-chain topology, place the MCU at one extreme end of the daisy-chain and the SDRAM at the other extreme end. Other devices can be distributed along the daisy-chain, see Fig 8:

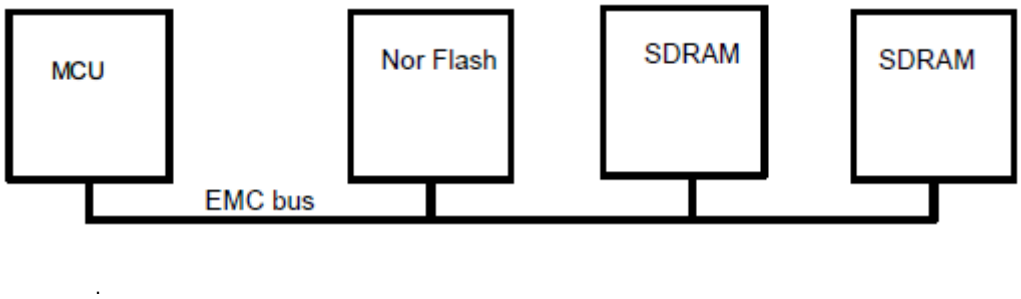


Fig 8. SDRAM signals without buffer

- For more than four devices (including the SDRAM and MCU) connected to the EMC bus, connect only the SDRAM and the bidirectional bus buffers to the MCU EMC directly. Connect other devices to the buffered side of the EMC bus; see Fig 9.

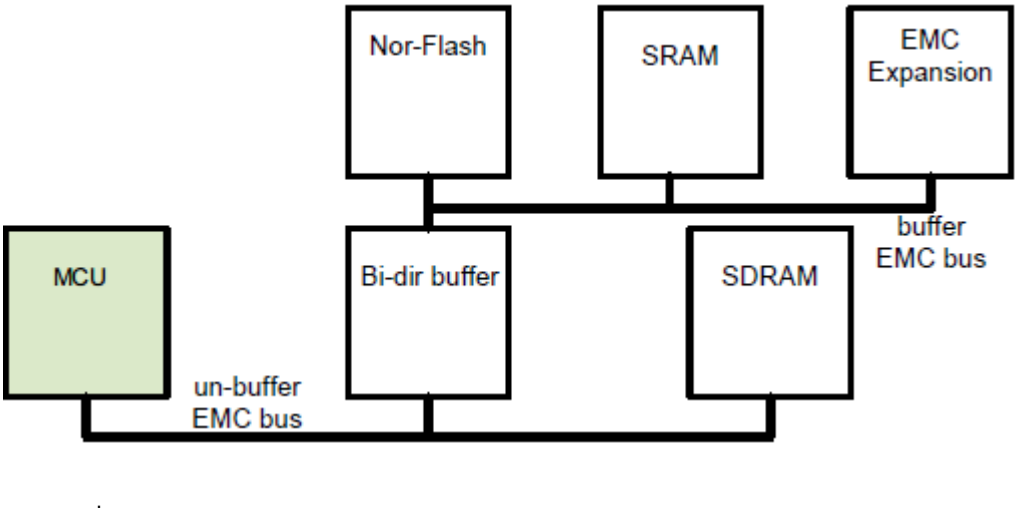


Fig 9. SDRAM signals with buffer

Fig 10 is an example of buffered schematic.

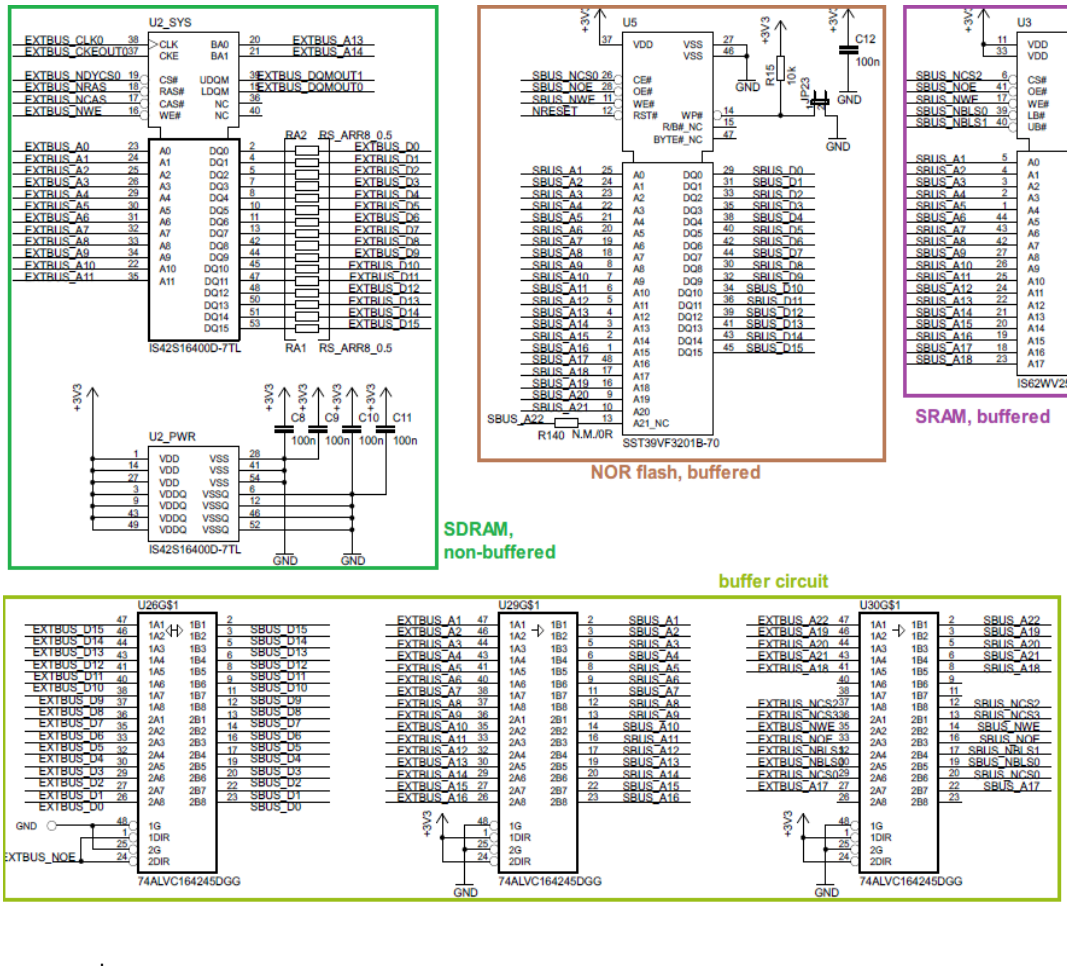


Fig 10. An example of SDRAM signals with buffer

5.2 EMC PCB trace impedance recommendation

The EMC CLKx pins are driven by a high-speed I/O buffer. To run the EMC above 75 MHz, the CLKx pins must be configured for high-speed slew rate (corresponding IOCON.PIO[port][pin]. bit10 (SLEW) set). The configuration generates rise or fall times of 0.4 ns to 0.7 ns. The CLKx pin source impedance is typically 50 Ω. It matches well to a PCB trace with characteristic impedance of 60 Ω to 80 Ω, without the need for any additional series termination near the MCU. All other EMC pins have a normal drive strength buffer output and must have the slew rate set to fast speed (corresponding IOCON.PIO[port][pin]. bit10 (SLEW) set). The configuration generates rise or fall times of 0.9 ns to 2.5 ns. For all EMC pins, other than CLKx, use a PCB trace with a characteristic impedance of 80 Ω to 100 Ω with no external series termination near the EMC pin.

5.3 EMC bus terminations

The internal source impedance of the EMC I/O buffer matches well to the 60 Ω to 100 Ω PCB trace characteristic impedance. In other words, the series source termination is built-in to the I/O buffer so that no termination resistor is required near the EMC pin.

All single data rate 3 V SDRAM data I/O buffers have very strong, low impedance drive strength and no internal drive strength reduction control. It leads to very fast rise and fall times. When the total EMC data bus trace length for any individual data signal is longer than 1.5 inches, the series source termination should be near to the data pins of all SDRAM chips in the design. It minimizes the overshoot and undershoot associated with the data driven by the SDRAM. It also reduces the ElectroMagnetic Interference (EMI) when the SDRAM is driving the EMC data bus.

5.4 EMC SDRAM PCB routing guides

5.4.1 Wire distance control

The EMC data bus signals should be short and the capacitive loading should be minimum. The following rules are followed to achieve proper SDRAM timing.

Rule 1: At each SDRAM, match EMC_D [31:0], EMC_DQM[3:0], EMC_A[14:0], EMC_RAS, EMC_CAS, EMC_DYCS3:0, EMC_WR, EMC_CKE[3:0] to within 2 inches of each other. As the signals require the reflected wave to reach full signal swing, the length is calculated from the MCU to the end of the daisy-chain trace (incident wave) plus the distance from the end of the daisy-chain back to the SDRAM pin (reflected wave).

An example for calculating effective trace length for matching is shown in Fig 11.

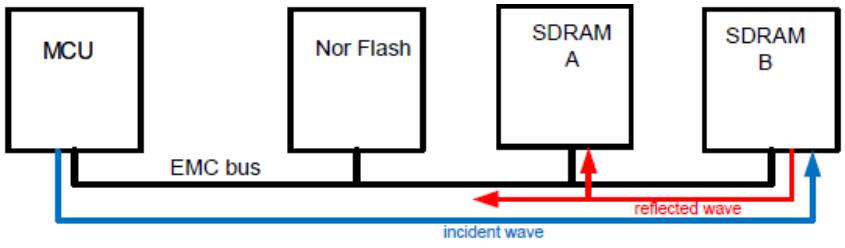


Fig 11. Signal flight model

Assume Signal 1 length between LPC546xx and SDRAM “A” = 3 inches

Assume Signal 1 length between SDRAM “A” and SDRAM “B” = 2 inches

Assume Signal 2 length between LPC546xx and SDRAM “A” = 2 inches

Assume Signal 2 length between SDRAM “A” and SDRAM “B” = 2 inches

The calculation for each signal trace length matching is as follows:

Signal 1 at SDRAM “B” = 3 inches + 2 inches = 5 inches; overall length of the daisy-chain; incident wave

Signal 2 at SDRAM “B” = 2 inches + 2 inches = 4 inches; overall length of the daisy-chain; incident wave

Signal 1 at SDRAM “A” = 3 inches + 2 inches + 2 inches = 7 inches; incident wave + reflected signal back to SDRAM

Signal 2 at SDRAM “A” = 2 inches + 2 inches + 2 inches = 6 inches; incident wave + reflected signal back to SDRAM

Rule 2: When EMC CLKx is also used to provide the feedback clock for SDRAM read data capture, it should be connected to an SDRAM clock less than 6 inches apart. The CLKx trace can be kept short and does not require to match length to other EMC signals. The EMC_CLKx can be delayed as required to provide proper SDRAM setup or hold timing.

5.4.2 Distance control for multiple SDRAM chips

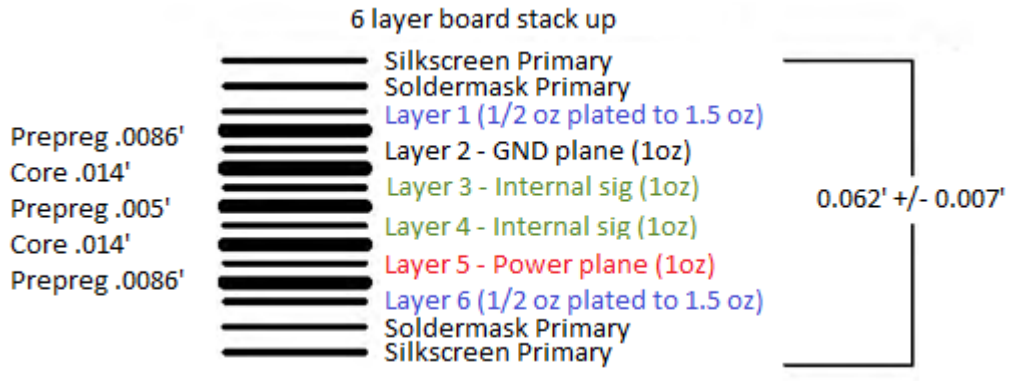
When using multiple SDR SDRAM configurations, place all memories near each other and route address, control and clock signals in a Y topology. Keep as much of the overall trace length in the trunk of the Y as possible, and the length to each device after the split as short as possible. Match the trace length to each device to within 0.3 inches.

Following is the list of points to be noted:

- match EMC_Data[7:0], EMC_DQM0 to within 0.3 inches of each other
- match EMC_Data[15:8], EMC_DQM1 to within 0.3 inches of each other
- match EMC_Data[23:16], EMC_DQM2 to within 0.3 inches of each other
- match EMC_Data[31:24], EMC_DQM3 to within 0.3 inches of each other
- match signal groups from rule 1 through rule 4 to within 1.0 inches of each other
- match EMC_CLK to within ±0.30 inches of signal groups rule 1 through rule 4
- match EMC_A[14:0], EMC_RAS, EMC_CAS, EMC_DYCS3:0, EMC_WR, EMC_CKE3:0 to within ±0.500 inches of EMC_CLK

5.4.3 Example PCB layer stack up

Stack up is a term used to describe the physical arrangement of the layers in a PCB. For any high-speed design, the two key elements determined by the stack up are the transmission line impedance and the inter-plane capacitance. Any board design that uses SDRAM should use a PCB stack up that includes power and ground planes. Pairing each signal layer with an adjacent plane layer (either power or ground) improves the impedance control for each signal layer. The two key dimensional characteristics of the trace affecting impedance value are the height to the nearest plane layer and the trace width. There are sources for trace impedance calculators available on the internet which can be used to help you get your design near the target trace impedance. However, it is always best to work closely with your board fabrication supplier and indicate on your supplied board fabrication drawing about the nominal impedance value desired for each trace width which needs to be impedance-controlled. Trace impedance and velocity of propagation are also controlled by the dielectric constant of the insulating material used for the PCB. There are a number of materials used for PCB dielectrics. However, FR4 is the most common low-cost dielectric material used for PCBs. An example of a six flayered PCB stack up is shown in Fig 12.



Ground on layer 2 and power on layer 5 are assumed to occupy the full layer.

Fig 12. Example of a six layered PCB with SDRAM

Fig 12 is an example of a six layered PCB stack up with 66 Ω and 80 Ω transmission lines. It assumes FR4 (high-tg), having a dielectric constant (er) of 4.7, where signal layers one, three, four and six with 0.007-inch trace width have a nominal characteristic impedance of 65 Ω to 67 Ω. A 0.004-inch trace width has a nominal characteristic impedance of 80 Ω. One nice thing about this stack up is for any given trace width, no matter which layer it is routed on, the characteristic impedance stays within 2 Ω. Using the example stack up, all CLKx signals would be routed with 0.007-inch trace width and all other EMC signals routed with 0.004-inch width traces. This example is just one of a number of possible six-layer stack up. Again, we recommend you to involve your PCB fabrication supplier early in the board design process to assist in selecting dielectric materials, a board stack up and trace widths to meet the desired trace characteristic impedance goals.

5.5 Software timing fine tuning: command delay and feedback delay

EMC outputs clock signal and the clock signal feedback to EMC with a programmable delay block. The EMC outputs other command signals based on the feedback clock signal. The EMC can delay driving SDRAM command signals for a short while after the next feedback CLK rising edge (command delay).

Both feedback delay and command delay are programmed via the EMCSysCtrl register within SYSCON module.

EMC has an internal ~50 MHz ring oscillator to act as the time base of both delay blocks. EMC uses FRO of 12 MHz output for calibrating to get 0.25 ns tick. FRO is the Free Running Oscillator inside LPC5460x. The calibration is controlled via the EMCCAL register in the SYSCON module. It is started by writing a 1 to the START bit of EMCCAL register. When EMC completes the calibration, it automatically stores the calibrated value in EMCCAL.[7:0].

5.5.1 Feedback delay: Compensating for signal flight time

EMC samples data from the feedback of clock pin. The programmable delay can be added to the feedback path to compensate for poor PCB layout. The feedback delay time is set in the multiples of ~0.25 ns in EMCDlyCtrl.[12:8]. So, the maximum delay that can be set for signal travelling is 32 × 0.25 = 8 ns. The programmable EMC feedback delay block is shown in Fig 13.

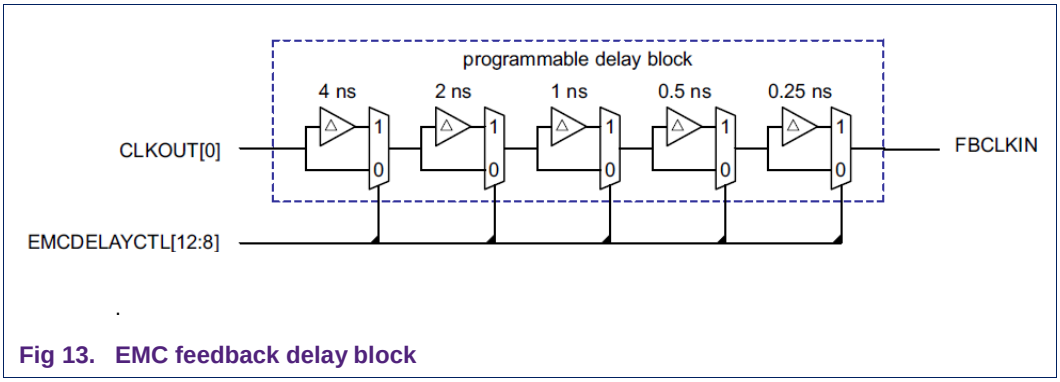


Fig 13. EMC feedback delay block

The delay block makes EMC sample data later. The feedback delay is recommended when PCB trace of CLOCK is longer than 6 inches and SDRAM clock reaches 100 MHz or when there are timing issues. It is because CLK signal itself is delayed due to signal flight time between the EMC and the SDRAM, depending on the PCB. For a 50 Ω PCB trace, the delay is 1 ns per 6-inch trace length.

EMC samples SDRAM data after command delayed time plus feedback delay time.

Note: Maximum delay value does **not** mean safe value. It should be comparable with travel time.

5.5.2 Command delay: Adapt to SDRAM set-up or holding timing constraints

With a feedback delay, the clock flight time is compensated. However, for higher bus frequency, the SDRAM set-up or holding timing could also be a problem. *Command delay* can be used to address this issue.

At the SDRAM, all input signals are sampled on the positive edge of the clock pin. To give SDRAM longer set-up time before EMC samples data, the data can be sampled after rising edge on EMC_CLK pin. It can be achieved by using the *command delayed mode* EMCDynamicReadConfig.RD = 1. It delays all the EMC signals except CLK and CKE. The block diagram of the command delay unit is shown in Fig 14.

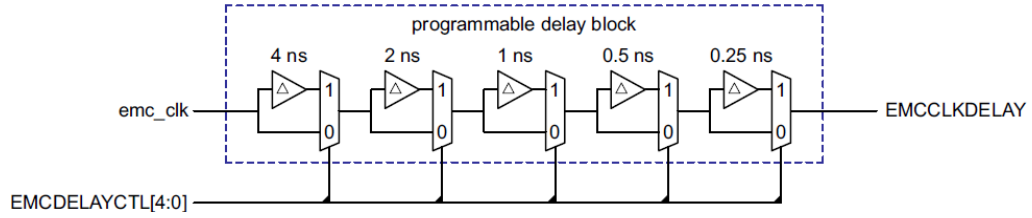


Fig 14. EMC command delay block

The command delay time is set in the multiples of ~0.25 ns in EMCDlyCtrl.[4:0]. So, the maximum delay that can be set is 32 × 0.25 = 8 ns. The EMC samples SDRAM data after a delay between 0.25 ns to 8 ns.

Note: Maximum delay value does **not** mean safe value. As an example, for a 90 MHz SDRAM clock, 0.25 ns is about 0.25 / (1000 / 90) = 2.25% clock cycle with a maximum delay of 32.

6. Appendix A: EMC pin MUX setup of 16-bit SDRAM bus example code (based on LPC5460x SDK 2.0)

```
32      /* EMC SDRAM Pins setting. */
33      IOCON_PinMuxSet(IOCON, 0, 18, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[0] */
34      IOCON_PinMuxSet(IOCON, 0, 19, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[1] */
35      IOCON_PinMuxSet(IOCON, 0, 20, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[2] */
36      IOCON_PinMuxSet(IOCON, 0, 21, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[3] */
37      IOCON_PinMuxSet(IOCON, 1, 5, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[4] */
38      IOCON_PinMuxSet(IOCON, 1, 6, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[5] */
39      IOCON_PinMuxSet(IOCON, 1, 7, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[6] */
40      IOCON_PinMuxSet(IOCON, 1, 8, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[7] */
41      IOCON_PinMuxSet(IOCON, 1, 26, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[8] */
42      IOCON_PinMuxSet(IOCON, 1, 27, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[9] */
43      IOCON_PinMuxSet(IOCON, 1, 16, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[10] */
```

```
44      IOCON_PinMuxSet(IOCON, 1, 23, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[11] */
45      IOCON_PinMuxSet(IOCON, 1, 24, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[12] */
46      IOCON_PinMuxSet(IOCON, 1, 25, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[13] */
47      IOCON_PinMuxSet(IOCON, 3, 25, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_A[14] */
48      IOCON_PinMuxSet(IOCON, 0, 2, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[0] */
49      IOCON_PinMuxSet(IOCON, 0, 3, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[1] */
50      IOCON_PinMuxSet(IOCON, 0, 4, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[2] */
51      IOCON_PinMuxSet(IOCON, 0, 5, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[3] */
52      IOCON_PinMuxSet(IOCON, 0, 6, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[4] */
53      IOCON_PinMuxSet(IOCON, 0, 7, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[5] */
54      IOCON_PinMuxSet(IOCON, 0, 8, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[6] */
55      IOCON_PinMuxSet(IOCON, 0, 9, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[7] */
56      IOCON_PinMuxSet(IOCON, 1, 19, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[8] */
57      IOCON_PinMuxSet(IOCON, 1, 20, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[9] */
58      IOCON_PinMuxSet(IOCON, 1, 21, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[10] */
59      IOCON_PinMuxSet(IOCON, 1, 4, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[11] */
60      IOCON_PinMuxSet(IOCON, 1, 28, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[12] */
61      IOCON_PinMuxSet(IOCON, 1, 29, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[13] */
62      IOCON_PinMuxSet(IOCON, 1, 30, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[14] */
63      IOCON_PinMuxSet(IOCON, 1, 31, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_D[15] */
64      IOCON_PinMuxSet(IOCON, 1, 9, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_CASN */
65      IOCON_PinMuxSet(IOCON, 1, 10, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_RASN */
66      IOCON_PinMuxSet(IOCON, 1, 11, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_CLK[0] */
67      IOCON_PinMuxSet(IOCON, 1, 12, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_DYCSN[0] */
68      IOCON_PinMuxSet(IOCON, 1, 13, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_DQM[0] */
```

```
69      IOCON_PinMuxSet(IOCON, 1, 14, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_DQM[1] */
70      IOCON_PinMuxSet(IOCON, 1, 15, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_CKE[0] */
71      IOCON_PinMuxSet(IOCON, 0, 15, IOCON_MODE_INACT | IOCON_FASTI2C_EN |
    IOCON_FUNC6 | IOCON_DIGITAL_EN); /*EMC_WEN */
```

7. Appendix B: Standard SDRAM initialization sequence example code (provided by LPC5460x SDK 2.0)

```
72  void EMC_DynamicMemInit(EMC_Type *base, emc_dynamic_timing_config_t *timing,
73          emc_dynamic_chip_config_t *config, uint32_t totalChips)
74  {
75      assert(config);
76      assert(timing);
77      assert(totalChips <= EMC_DYNAMIC_MEMDEV_NUM);
78      uint32_t count;
79      uint8_t casLatency;
80      uint32_t addr;
81      uint32_t offset;
82      uint32_t data;
83      emc_dynamic_chip_config_t *dynamicConfig = config;
84      /* Setting for dynamic memory controller chip independent configuration. */
85      for (count = 0; (count < totalChips) && (dynamicConfig != NULL); count ++)
86      {
87          base->DYNAMIC[dynamicConfig->chipIndex].DYNAMICCONFIG  =
    EMC_DYNAMIC_DYNAMICCONFIG_MD(dynamicConfig->dynamicDevice) |
88              EMC_ADDRMAP(dynamicConfig->devAddrMap);
89          /* Abstract CAS latency from the sdram mode reigster setting values. */
90          casLatency = (dynamicConfig->sdramModeReg & EMC_SDRAM_MODE_CL_MASK) >>
    EMC_SDRAM_MODE_CL_SHIFT;
91          base->DYNAMIC[dynamicConfig->chipIndex].DYNAMICRASCAS  =
    EMC_DYNAMIC_DYNAMICRASCAS_RAS(dynamicConfig->rAS_Nclk) |
92          EMC_DYNAMIC_DYNAMICRASCAS_CAS(casLatency);
93          dynamicConfig ++;
94      }
95      /* Configure the Dynamic Memory controller timing/latency for all chips. */
96      base->DYNAMICREADCONFIG = EMC_DYNAMICREADCONFIG_RD(timing->readConfig);
97      base->DYNAMICRP = EMC_CalculateTimerCycles(base, timing->tRp_Ns, 1) &
    EMC_DYNAMICRP_TRP_MASK;
98      base->DYNAMICRAS = EMC_CalculateTimerCycles(base, timing->tRas_Ns, 1) &
    EMC_DYNAMICRAS_TRAS_MASK;
99      base->DYNAMICSREX = EMC_CalculateTimerCycles(base, timing->tSrex_Ns, 1) &
    EMC_DYNAMICSREX_TSREX_MASK;
100     base->DYNAMICAPR = EMC_CalculateTimerCycles(base, timing->tApr_Ns, 1) &
    EMC_DYNAMICAPR_TAPR_MASK;
101     base->DYNAMICDAL = EMC_CalculateTimerCycles(base, timing->tDal_Ns, 0) &
    EMC_DYNAMICDAL_TDAL_MASK;
102     base->DYNAMICWR = EMC_CalculateTimerCycles(base, timing->tWr_Ns, 1) &
    EMC_DYNAMICWR_TWR_MASK;
103     base->DYNAMICRC = EMC_CalculateTimerCycles(base, timing->tRc_Ns, 1) &
    EMC_DYNAMICRC_TRC_MASK;
104     base->DYNAMICRFC = EMC_CalculateTimerCycles(base, timing->tRfc_Ns, 1)
    &EMC_DYNAMICRFC_TRFC_MASK;
105     base->DYNAMICXSR = EMC_CalculateTimerCycles(base, timing->tXsr_Ns, 1) &
    EMC_DYNAMICXSR_TXSR_MASK;
```

```
106     base->DYNAMICRRD = EMC_CalculateTimerCycles(base, timing->tRrd_Ns, 1) &
    EMC_DYNAMICRRD_TRRD_MASK;
107     base->DYNAMICMRD = EMC_DYNAMICMRD_TMRD((timing->tMrd_Nclk > 0)?timing-
    >tMrd_Nclk - 1:0);
108     /* Initialize the SDRAM.*/
109     for (count = 0; count < EMC_SDRAM_WAIT_CYCLES;  count ++)
110     {
111     }
112     /* Step 2. issue nop command. */
113     base->DYNAMICCONTROL  = 0x00000183;
114     for (count = 0; count < EMC_SDRAM_WAIT_CYCLES;  count ++)
115     {
116     }
117     /* Step 3. issue precharge all command. */
118     base->DYNAMICCONTROL  = 0x00000103;
119
120     /* Step 4. issue two auto-refresh command. */
121     base->DYNAMICREFRESH = 2;
122     for (count = 0; count < EMC_SDRAM_WAIT_CYCLES/2; count ++)
123     {
124     }
125     base->DYNAMICREFRESH = EMC_CalculateTimerCycles(base, timing-
    >refreshPeriod_Nanosec, 0)/EMC_REFRESH_CLOCK_PARAM;
126     /* Step 5. issue a mode command and set the mode value. */
127     base->DYNAMICCONTROL  = 0x00000083;
128     /* Calculate the mode settings here and to reach the 8 auto-refresh time
    requirement. */
129     dynamicConfig = config;
130     for (count = 0; (count < totalChips) && (dynamicConfig != NULL); count ++)
131     {
132         /* Get the shift value first. */
133         offset = EMC_ModeOffset(dynamicConfig->devAddrMap);
134         addr = (s_EMCDYCSBases[dynamicConfig->chipIndex] |
135             ((uint32_t)(dynamicConfig->sdramModeReg & ~EMC_SDRAM_BANKCS_BA_MASK )
    << offset));
136         /* Set the right mode setting value. */
137         data = *(volatile uint32_t *)addr;
138         data = data;
139         dynamicConfig ++;
140     }
141     if (config->dynamicDevice)
142     {
143         /* Add extended mode register if the low-power sdram is used. */
144         base->DYNAMICCONTROL  = 0x00000083;
145         /* Calculate the mode settings for extended mode register. */
146         dynamicConfig = config;
147         for (count = 0; (count < totalChips) && (dynamicConfig != NULL); count++)
148         {
149             /* Get the shift value first. */
150             offset = EMC_ModeOffset(dynamicConfig->devAddrMap);
```

```
151             addr = (s_EMCDYCSBases[dynamicConfig->chipIndex] |
    (((uint32_t)(dynamicConfig->sdramExtModeReg & ~EMC_SDRAM_BANKCS_BA_MASK) |
152                 EMC_SDRAM_BANKCS_BA1_MASK) << offset));
153             /* Set the right mode setting value. */
154             data = *(volatile uint32_t *)addr;
155             data = data;
156             dynamicConfig ++;
157         }
158     }
159     /* Step 6. issue normal operation command. */
160     base->DYNAMICCONTROL  = 0x00000000; /* Issue NORMAL command */
161
162     /* The buffer shall be disabled when do the sdram initialization and
163      * enabled after the initialization during normal opeation.
164      */
165     dynamicConfig = config;
166     for (count = 0; (count < totalChips) && (dynamicConfig != NULL); count ++)
167     {
168         base->DYNAMIC[dynamicConfig->chipIndex].DYNAMICCONFIG |=
    EMC_DYNAMIC_DYNAMICCONFIG_B_MASK;
169         dynamicConfig ++;
170     }
171 }
```

8. Legal information

8.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

8.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

8.3 Licenses

Purchase of NXP <xxx> components
<License statement text>

8.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — owned by <Company name>

8.5 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

<Name> — is a trademark of NXP Semiconductors N.V.

9. List of figures

Fig 1. EMC block diagram....4
Fig 2. Block diagram for connecting a single SDRAM chip....8
Fig 3. Example of connecting a single 16-bit SDRAM chip....9
Fig 4. Example of connecting a single 32-bit SDRAM chip....10
Fig 5. Example of connecting 2 16-bit SDRAM chips to form 32-bit bus....12
Fig 6. SDRAM initialization sequence....13
Fig 7. SDRAM "MODE" register layout....16
Fig 8. SDRAM signals without buffer....19
Fig 9. SDRAM signals with buffer....19
Fig 10. An example of SDRAM signals with buffer....20
Fig 11. Signal flight model....21
Fig 12. Example of a six layered PCB with SDRAM....23
Fig 13. EMC feedback delay block....24
Fig 14. EMC command delay block....25

10. List of tables

Table 1. EMC memory map 3
Table 2. EMC signals 5
Table 3. EMC SDRAM cell layout summary 6
Table 4. Signals of connecting to multiple SDRAM chips, summary 11
Table 5. EMC Dynamic Memory timing registers 14

11. Contents

1. Introduction 3
2. External Memory Controller(EMC) 3
2.1 EMC key features 3
2.2 EMC pins 4
2.2.1 SDRAM clock enable when using multiple DYCSs 5
2.3 Supported SDRAM cell layouts 6
2.3.1 Using 32-bit SDRAMs with cell layout not supported by EMC 7
2.4 EMC clocking to SDRAM 7
3. SDRAM connections 7
3.1 Pin connections for using single SDRAM chip 8
3.2 Pin connections for using multiple SDRAM chips 10
4. SDRAM controller programming and operations 12
4.1 Typical SDRAM programming sequence 12
4.1.1 Standard and low-power SDRAM programming sequence 13
4.1.2 SDRAM configuration example (Board_InitSDRAM()) 17
5. Schematics, PCB design tips and suggestions 18
5.1 Shared EMC configuration 18
5.2 EMC PCB trace impedance recommendation 20
5.3 EMC bus terminations 21
5.4 EMC SDRAM PCB routing guides 21
5.4.1 Wire distance control 21
5.4.2 Distance control for multiple SDRAM chips 22
5.4.3 Example PCB layer stack up 22
5.5 Software timing fine tuning: command delay and feedback delay 23
5.5.1 Feedback delay: Compensating for signal flight time 24
5.5.2 Command delay: Adapt to SDRAM set-up or holding timing constraints 24
6. Appendix A: EMC pin MUX setup of 16-bit SDRAM bus example code (based on LPC5460x SDK 2.0) 25
7. Appendix B: Standard SDRAM initialization sequence example code (provided by LPC5460x SDK 2.0) 28
8. Legal information 31
8.1 Definitions 31
8.2 Disclaimers 31
8.3 Licenses 31
8.4 Patents 31
8.5 Trademarks 31
9. List of figures 32
10. List of tables 33
11. Contents 34

For more information, visit: http://www.nxp.com

Date of release: 18 August 2017
Document identifier: AN12026