Engineer To Engineer Note EE-127

Technical Notes on using Analog Devices' DSP components and development tools
Phone: (800) ANALOG-D, FAX: (781) 461-3010, EMAIL: dsp.support@analog.com, FTP: ftp.analog.com, WEB: www.analog.com/dsp

Contributed by R. Hoffmann, European DSP Applications Rev 1 (20-March-02)

The ADSP-21065L On-chip SDRAM Controller

If you use a DSP to address SDRAM, you will need additional hardware or software to handle the multiplexed row and column addressing and the refresh and precharge requirements. The ADSP-21065L uses a hardware intensive solution, an on-chip SDRAM controller.

Introduction

The application note introduces the On-chip SDRAM controller's characteristics. Basically, the internal signal chain is shown with the necessary address-mapping scheme. The command truth table gives detailed information about execution in the SDRAM. The important power up sequence summarizes deep information to start successful designs. Code execution is described to get optimized performance. A timing overview demonstrates the performance for different access modes. Refer to *"EE-126: ABC of SDRAMs"*

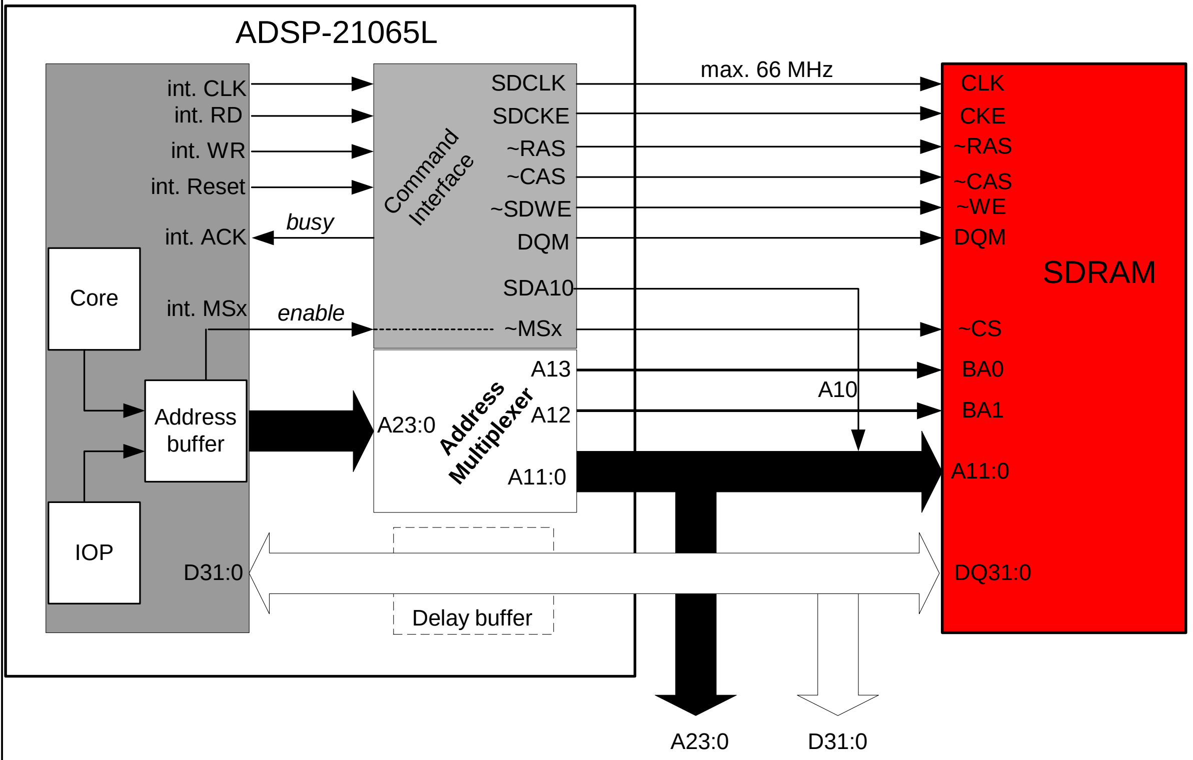
a

1 – Signal Chain of SDRAM

Figure 1 illustrates the signal chain between the ADSP-21065L, the controller and the SDRAM itself. The 3 parts for the signal flow to be considered:

- ADSP-21065L (core, IOP, address buffer)
- SDRAM Controller (control interface, delay buffer, address multiplexer)
- SDRAMemory

Figure 1: Signal chain: ADSP-21065L to SDRAM



2 – On-Chip Controller Architecture

The synchronous interface between the ADSP-21065L and the on-chip controller can be described in 3 basic parts:

2.1 – Controller Command Interface

Because of the 2 different timing protocols, the internal SHARC commands are converted to comply with the JEDEC standard for SDRAMs. The SDCLK clock, maximum 66 MHz, is used for synchronous operation. The SHARC's internal request lines or strobes are used to access the SDRAM with pulsed commands. The controller's internal ACK line inserts variable wait states to the DSP during overhead cycles, caused by DRAM technology.

2.2 – SHARC Address Buffer

The SHARC's address buffer enables the controller activation depending on bank assignment and external address. The SHARC's address pipeline depth is 1, therefore address pipelining is not supported.

2.3 – Controller Address Multiplexer

Every first read or write action is issued in multiplexed mode. A maximum of 4096 Rows within 1024 columns can be addressed. Furthermore, the A13 and A12 lines are used to select the current SDRAM bank.

2.4 – Controller Delay Buffers

If systems incorporate a heavy busload, additional buffers are used to decouple the input from the capacitive load. The internal delay buffer in conjunction with an external buffer reduces additional logic to a minimum.

3 – Command Coding

3.1 - Pin Description of Controller

Controller Pin	*Type*	*Signal*	*Description*
SDCLK0	(I/O/T/S)	pulse	master clock
SDCLK1	(O/T/S)	pulse	clock 2
SDCKE	(I/O/T)	level	command
~MSx	(I/O/T)	pulse	command
~RAS	(I/O/T)	pulse	command
~CAS	(I/O/T)	pulse	command
~SDWE	(I/O/T)	pulse	command
SDA10	(O/T)	level/pulse	address/command
DQM	(O/T)	pulse	DQ buffer control
A[9:0], A[11]	(I/O/T)	level	address

A[12]	(I/O/T)	level	bank select
A[13]	(I/O/T)	level	bank select
D[31:0]	(I/O/T)	level	data

I=input, O=output, T=Hi-Z, S=synchronous

3.2 - Controller Command Truth Table

This section provides a table to get an overview of all commands provided by the SDRAM controller. These commands are handled automatically by the interface.

Commands with SDCKE = high

Command	*SDCKE(n-1)*	*SDCKE(n)*	*~CS*	*~RAS*	*~CAS*	*~SDWE*	*SDA10*	*ADDR*
MRS	1	1	0	0	0	0	v	v
ACT	1	1	0	0	1	1	v	v
RD	1	1	0	1	0	1	0	v
WR	1	1	0	1	0	0	0	v
Command without validity of address								
DESL	1	1	1	x	x	x	x	x
NOP	1	1	0	1	1	1	x	x
BST	1	1	0	1	1	0	x	x
PREA	1	1	0	0	1	0	1	x
REF	1	1	0	0	0	1	1	x

Commands with Transition of SDCKE

While the SDCKE line toggles in asynchronous manner, the commands are sampled synchronous to the CLK signal.

Command	*SDCKE(n-1)*	*SDCKE(n)*	*~CS*	*~RAS*	*~CAS*	*~SDWE*	*SDA10*	*ADDR*
SREF En	1	0	0	0	0	1	x	x
SREF Ma	0	0	x	x	x	x	x	x
SREF Ex	0	1	1	x	x	x	x	x

x=don’t care, v=valid data input, 0=logic 0, 1=logic 1, En=entry, Ma=maintain, Ex=exit

Note: Power-down and Suspend mode are not supported and auto precharge is not allowed.

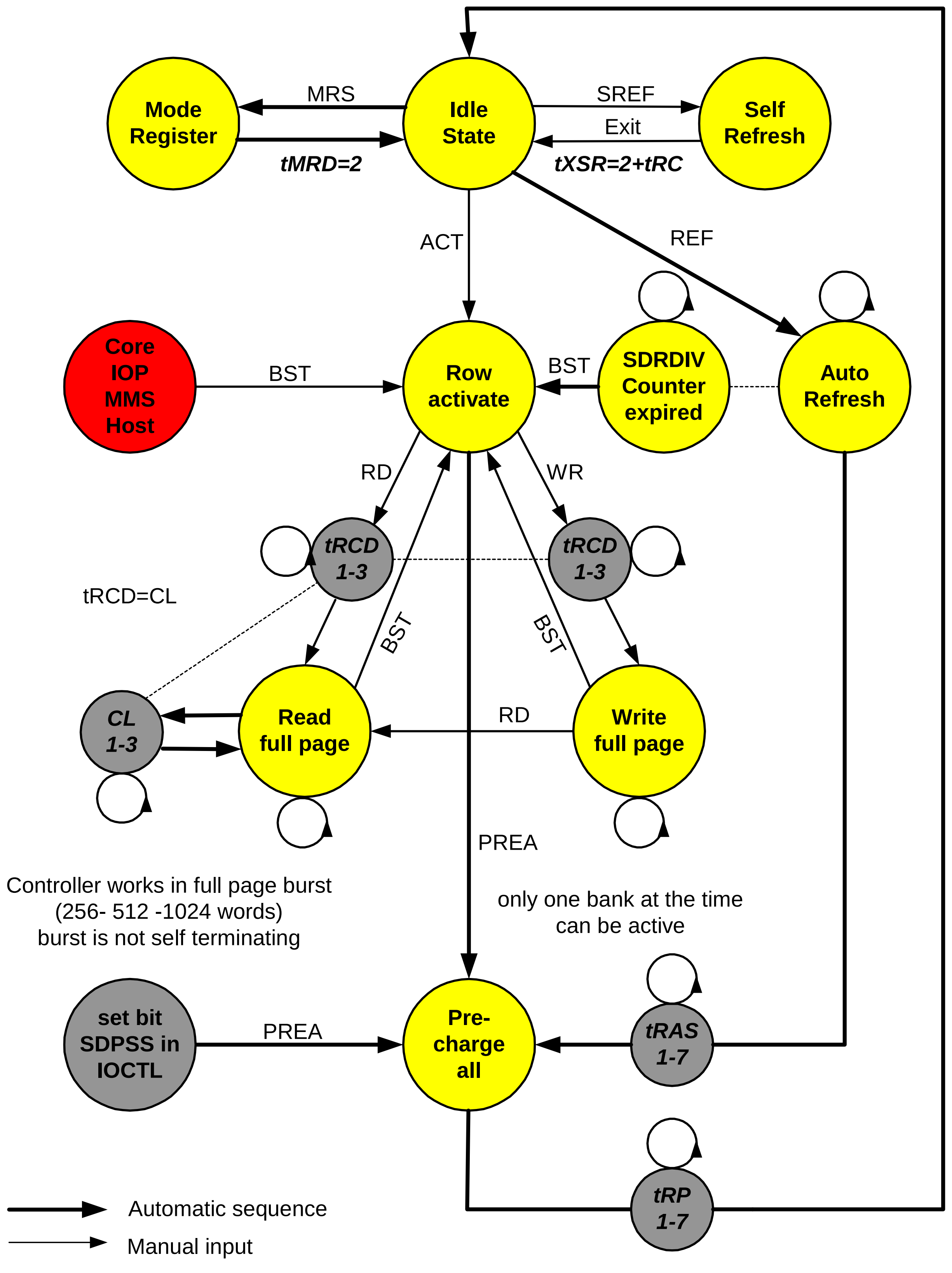
3.3 – Setup and Hold Times

The synchronous operation uses the SDCLK as reference. Commands, addresses and data are latched at the rising edge of SDCLK. The valid time margin around the rising edge is defined as setup time (time before rising edge) and hold time (time after rising edge) to guarantee that both the controller and the SDRAM are working reliably together. Signal’s- slew rates, propagation delays (PCB) and capacitive loads (devices) influence these parameters and should be taken into consideration. The controller’s timing characteristics are available in the ADSP-21065L datasheet.

3.4 - Simplified State Diagram

The state diagram is useful to analyze all deterministic sequences. Figure 4 shows all the possible states.

Figure 4: Simplified State Diagram: ADSP-21065L SDRAM Controller



4 – Controller specific Properties

The next properties applies especially for the ADSP-21065L:

4.1 – Address Mapping Scheme

Basically, there are various possibilities accessing the SDRAM, for instance you access all rows in a bank sequentially or you access all banks in a row sequentially. PC DIMM modules are accessed different by its controller compared to a typical DSP application. The ADSP-21065L controller uses a hardware map scheme optimized for digital signal processing.
Because different sizes can be switched to the controller, different map schemes should be enabled decoded by the number of banks (bit SDBN in IOCTL) and page size (bit SDPGS in IOCTL):

Page x Bank	*Column Address*		*Bank Select*		*Row Address*	
256 x 2	IA[7:0] =>	**EA[7:0]**	IA[8] =>	**EA[13]**	IA[19:9] =>	**EA[10:0]**
512 x 2	IA[8:0]	**EA[8:0]**	IA[9]	**EA[13]**	IA[20:10]	**EA[10:0]**
1024 x 2	IA[9:0]	**EA[9:0]**	IA[10]	**EA[13]**	IA[21:11]	**EA[10:0]**
256 x 4	IA[7:0]	**EA[7:0]**	IA[9:8]	**EA[13:12]**	IA[21:10]	**EA[11:0]**
512 x 4	IA[8:0]	**EA[8:0]**	IA[10:9]	**EA[13:12]**	IA[22:11]	**EA[11:0]**
1024 x 4	IA[9:0]	**EA[9:0]**	IA[11:10]	**EA[13:12]**	IA[23:12]	**EA[11:0]**

IA= controller input address, EA= controller output address
Note: The address map scheme allows you to understand the system's performance.

Figure 2 reproduces an example of the controller's address mapping for 32bit data. In page 0, the SDRAM's banks A to D are sequentially selected. As bank interleaving is not supported, every off bank access has the same overhead as an off page access. The address region of a full page for instance 0x20000 to 0x203FF (1024 words) can be accessed with 1 cycle/word.
Note: Only one bank at a time can be active. Off-bank and off-page access have the same overhead.

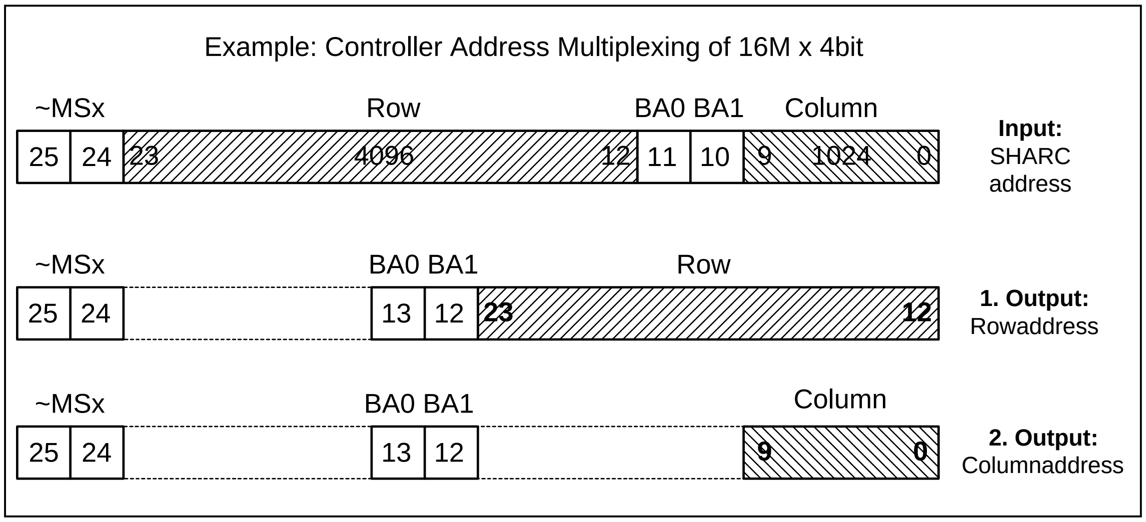
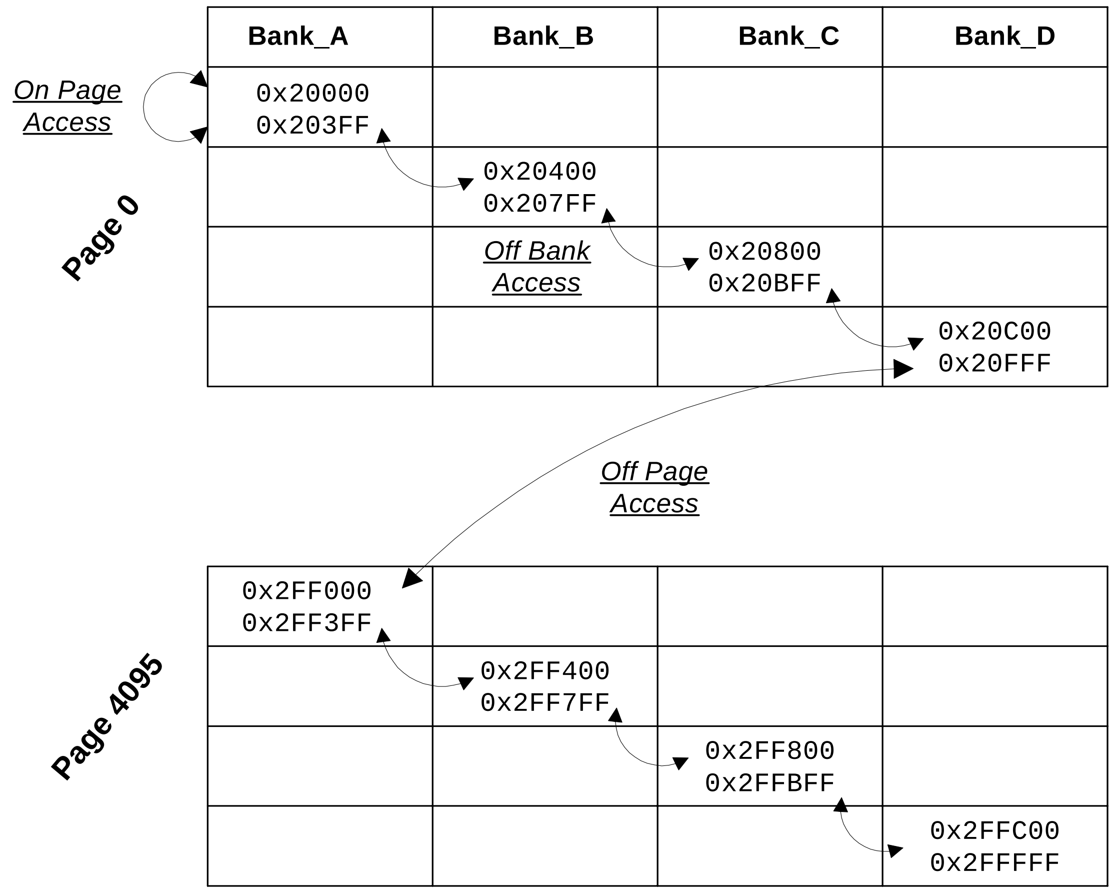


Figure 2: Controller Address Mapping to bank 0 of ADSP-21065L
4M x 4bit x 4, Page size 1024 words



4.2 – SHARC Bank Select (~MSx)

Each SHARC bank (~MS0-3) can be mapped in the memory region of the SDRAM. Only one SHARC bank can be connected (bit SDBS in IOCTL) to the interface at a time.

4.3 – Burst Stop (BST)

Although the controller works in full page burst only, there is one way to interrupt the full page burst with the burst stop command. The next table lists the different situations in which the BST occurs:

BST issued if next access is:
• Non external SDRAM access (access to another SHARC bank) • Core access (e.g. computation unit, interrupt, cache, DAG)

- DMA operation (higher priority request of IOP)
- SDRDIV counter expired (refresh period counter)
- SDRAM read to write transition
- SDRAM off page and off bank access
- ~HBR asserted (host interface)
- ~BRx asserted (multiprocessing)

Note: The BST is an indicator for a SHARC core accesses.

4.4 – Mask Function (DQM)

The DQM pin is used by the controller to mask write and precharge operations directly, the affect latency is zero cycles. It does not apply to read operations. It's used to disconnect the SDRAM's DQ buffer to avoid data contention.

Command(n-1)	*Command(n)*	*DQM(n)*
x	PREA	1
WR	BST	1

x=don't cares
Note: The controller does not support partial reads or writes. All DQM lines should be tied together.

4.5 – SDRAM Bank Select A[13:12]

The next tables show how address lines A12 and A13 select the different banks:
2 banked access:

Banks	*A13*	*A12*	*SDA10*
Bank_A	0	-	0
Bank_B	1	-	0
Both banks	x	-	1

Note: A12 is not used in 2 banked memories.

4 banked access:

Banks	*A13*	*A12*	*SDA10*
Bank_A	0	0	0
Bank_B	0	1	0
Bank_C	1	0	0
Bank_D	1	1	0
All_banks	x	x	1

x=don't care, 0=logic 0, 1=logic 1
Note: It doesn't matter if you connect A[13:12] to BA[1:0] or A[13:12] to BA[0:1].

4.6 - Controller Address 10 (SDA 10)

This pin provides a special solution to gain control about the SDRAM, even by host accesses. It allows accessing all banks simultaneously (without using addresses 12 and 13) during a refresh in slave mode. It must be connected with the A10 pin of the SDRAM.
Note: This pin has multifunctional character: depending on command, it acts as an address- or as a command pin.

4.7 – Burst Mode

The controller uses a full page burst only. Only the first read or write command is accompanied with an external address, which is driven by the controller until the burst is interrupted by another address. The SDRAM's burst counter depending on the used page size internally addresses only sequential locations.
Note: The SDRAM must support full page burst.

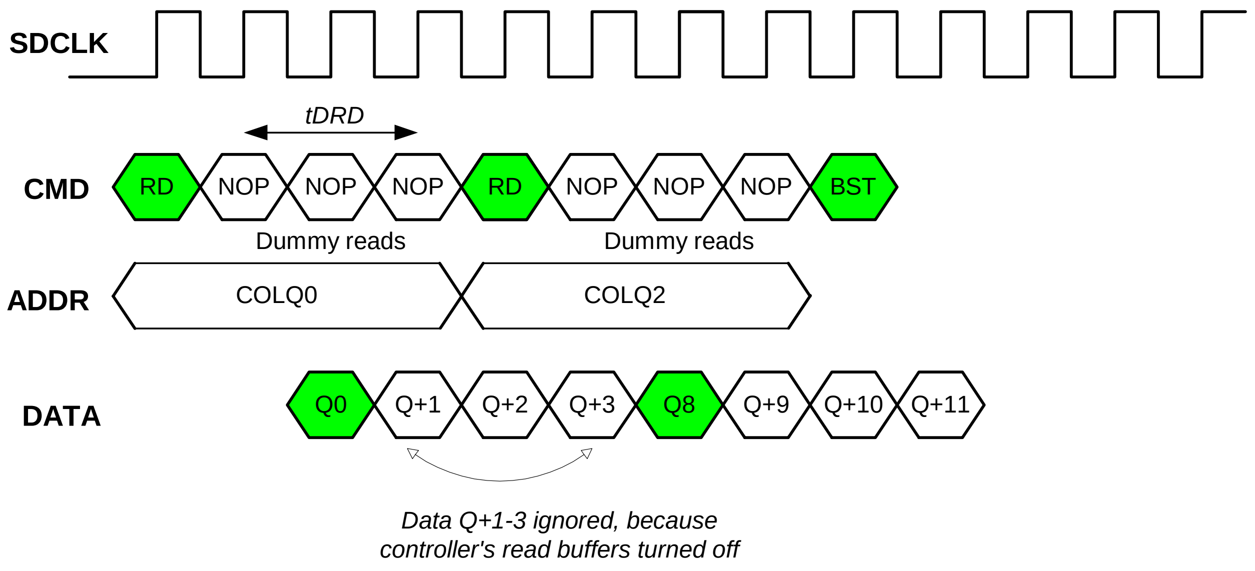
4.8 – Read Interruption

The SHARC architecture doesn't support address pipelining. The next address will only be latched until the previous data are off-chip. For sequential reads, the controller simply applies the command and address assuming that it will be sequential. For non-sequential reads, it inserts additional dummy reads (NOPs) in order to reject the data given out by SDRAM (Figure 3), if the next address from the core is non-sequential. It applies to:

- Non sequential reads
- Sequential reads interruption
- Read to write transition

Note: The burst (sequential reads) assumes address pipelining to benefit from its throughput.

Figure 3: Non sequential Reads, CL=2



The controller inserts dummy reads depending from the used read latency:

Command (n-x)	*tDRD cycles (n-1)*	*Command (n)*
RD	CL+1	RD/BST

4.9 – Write Interruption

While the controller is working in full page burst only, there is a need to interrupt the current burst with the BST command. Additionally, the write buffer must be masked to block the data immediately. It's realized by asserting the DQM pin during the BST command.
Note: Every write burst interruption is masked with the DQM pin during the BST command. It doesn't apply for non-sequential writes.

4.10 – Precharge All (PREA)

This command precharges all banks of the SDRAM simultaneously, (SDA 10 high to select all banks) which returns the banks in idle state.
Note: The controller doesn't support precharge of single bank while only one bank at the time is active.

4.11 – Circular Access

The controller supports the circular access during sequential read or writes in a page, performing a fixed throughput of 1 cycle/word. At the end of the page (defined in the IOCTL register), for instance 1024 words, the instructions *r0=dm(0x203FF)* followed by *r1=dm(0x20000*) are also executed with 1 cycle/word.

Note: This functionality is similar to the DAG's circular buffering mode.

4.12 – Auto Refresh (REF)

The auto- or CAS before RAS- refresh requires only a command, no addresses. After the SDRAM registers the command, it asserts internally CAS and delayed RAS to execute a row's refresh. The row interval is typically tRC=15,625 µs, it's a good compromise between data access time and the refresh reliability. The limit of refresh period is given through the *tREFmax* spec.
Note: The controller doesn't support burst refresh.

5.13 – Self Refresh (SREF)

The self-refresh is a very effective way to reduce the application's power consumption to a minimum. The device can run in this mode during long SHARC core operations. This mode is used to disable the SDRAM controller. The device starts refreshing itself triggered by an internal timer.

4.14 – Mode Register Set (MRS)

With the MRS, the controller initializes the SDRAM with:
Burst length is fixed to full page burst
Burst type is fixed to sequential burst

5 – Programming the SDRAM Interface

The Interface is programmed through the mapped registers in following order:

1. WAIT Register (wait states and wait states mode)
2. SDRDIV Register (setting the refresh period value)
3. IOCTL Register (programs the controller and the SDRAM)

5.1 – Wait Register

For proper operation, make sure that the dedicated WAIT register of SHARC bank setting is

- Wait state mode: SDRAM is not stalled or suspended by assertion of ACK
- Wait states: zero wait states allowed only

Note: Zero wait state is required to enable the handshake between SHARC and SDRAM controller.

5.2 – Refresh Register

This counter enables applications to coordinate the clock rate with the SDRAM's required refresh rate. The SDRDIV register is used to enter the period of refresh commands in number of cycles calculated in the following equation:

$$SDRDIV = \left(SDCLK \cdot \frac{tREF}{Rows} \right) - tRP - CL - 4$$

Note: This formula is not linear, because the specs tRP and CL vary through the different devices.

The controller sets SDRDIV=0 during the power up mode to initialize the refresh counter with 8 REF cycles. Hereby, the controller issues permanent refresh commands within a *tRCmin* period.
Note: If you configure SDRDIV to 0, the controller issues permanent REF commands.

5.3 - SDRAM Controller

In order to support timing requirements and power up modes for different SDRAM vendors, the ADSP-21065L provides programmability of *tRAS, tRP* and *CL* in the IOCTL register.

Timing Specs

The **SDTRP bit** defines the precharge time *tRPmin* related to the vendor's device.
The **SDTRAS bit** is used to define the row active time *tRASmin* related to the vendor's device.
Note: The spec tRAS is used for the refresh cycle with tRC=tRAS+tRP.
Note: The specs tRP and tRAS depend on the different devices' speed grades only.

The **SDCL bit** is used to define the Read latency CL*min* related to the vendor's device.
Note: This value depends on the application's speed and different speed grades of devices. CL is the most critical parameter to be set.
Note: The specification requires defining tRASmin, tRPmin and CLmin as a fraction of the SDCLK period.

Power up Option

The **SDPM bit** controls 2 different software power-up options:
SDPM=0

• PREA command	(brings the SDRAM in the defined idle state)
• 8 REF commands	(charges SDRAM's internal nodes)
• MRS command	(initializes the SDRAM's working mode)

SDPM=1

- PREA command
- MRS command
- 8 REF commands

Note: Some vendors don't stick to the power up sequence.

5.4 – SDRAM

With the MRS (Mode register set), the Read latency (SDCL bit) is also used to program the state machine of the SDRAM in order to get the best performance depending from the application's speed. Addresses [11:0] are used during the MRS to program the following specs:

Address bit	*Mode of operation*
• A[2:0]	Burst length is hardwired to full page burst
• A[3]	Burst type is hardwired to sequential burst
• A[6:4]	Read Latency: the user defines *CLmin* in 1-3 SDCLK cycles
• A[11:7]	Operation mode is hardwired to 0

Note: The operation mode is reserved for test and future.

5.5 – Memory Organization

The SDBN- and SDPGS bits in IOCTL register are used to program the address map scheme:

256 x 2

Refresh	*Type*	*Size*	*Parallel*	*Whole Size*
2k	1M x 16bit	16 Mbit	2	32 Mbit

512 x 2

Refresh	*Type*	*Size*	*Parallel*	*Whole Size*
2k	2M x 8bit	16 Mbit	4	64 Mbit

1024 x 2

Refresh	*Type*	*Size*	*Parallel*	*Whole Size*
2k	4M x 4bit	16 Mbit	8	128 Mbit

256 x 4

Refresh	*Type*	*Size*	*Parallel*	*Whole Size*
2k	2M x 32bit	64 Mbit		
4k	4M x 16bit	64 Mbit	2	128 Mbit
4k	4M x 32bit	128 Mbit		

512 x 4

Refresh	*Type*	*Size*	*Parallel*	*Whole Size*
4k	8M x 8bit	64 Mbit	4	256 Mbit
4k	8M x 16bit	128 Mbit	2	256 Mbit

1024 x 4

Refresh	*Type*	*Size*	*Parallel*	*Whole Size*	
4k	16M x 4bit	64 Mbit	8	512 Mbit	(maximum size)
4k	16M x 8bit	128 Mbit	4	512 Mbit	(maximum size)

Note: Each bank supports up to 16M x 32bit.

5.6 – Setup Overview

1. Configuring the controller's state machine

Timing spec	*Description*	*Controller Configuration*
tCK	clock cycle time, hardware	20–66 MHz

tREFmax	refresh period	0–xxxx cycles
CLmin	read latency	1-3 cycles
tRASmin	activate to precharge	1-7 cycles
tRPmin	precharge period	1-7 cycles
tRCD	RAS to CAS delay	tRCD = CL (fixed)
tDRD	dummy read to burst stop	tDRD=CL+1 (fixed)
tRCmin	activate period	tRC = tRAS+tRP (fixed)
tRRD	activate A to activate B	tRRD = tRC (fixed)
tMRD	mode register to command	2 cycles (fixed)
tXSR	self refresh to auto refresh	2 + tRC (fixed)

2. Configuring the SDRAM

Power up Mode	PREA-MRS-REF or PREA-REF-MRS
Burst Mode	sequential (fixed)
Burst Length	full page (fixed)
CLmin	1-3 cycles

Note: All cycles in SDCLK cycles, the bolded terms are only valid during the mode register set.

5.7 – ADSP-21065L EZ-Kit Lite

On the starter kit 2 x (1M x 16bit) SDRAMs are connected in parallel, which adds up to 1M x 32bit. Following settings can be used depending on the different SDRAM vendors on the board:

“NEC µPD4516161A”, 1M x 16bit

Speed grade=10, Power up mode: PREA-MRS-REF						
tRCD=20 ns, tREF= 32ms, tRASmin=50 ns, tRPmin=20 ns						
SDCLK	**20**	**33**	**40**	**50**	**60**	**MHz**
SDTRAS	1	2	2	3	3	*tRASmin*
SDTRP	1	1	1	1	2	*tRPmin*
SDCL	2	2	2	2	2	*CLmin*
-	2	3	3	4	5	*tRCmin, fixed by controller*
-	2	2	2	2	2	*tRCDmin, fixed by controller*
SDRDIV	305	513	618	774	929	*tREF*

Note: Values are fractions of the speed

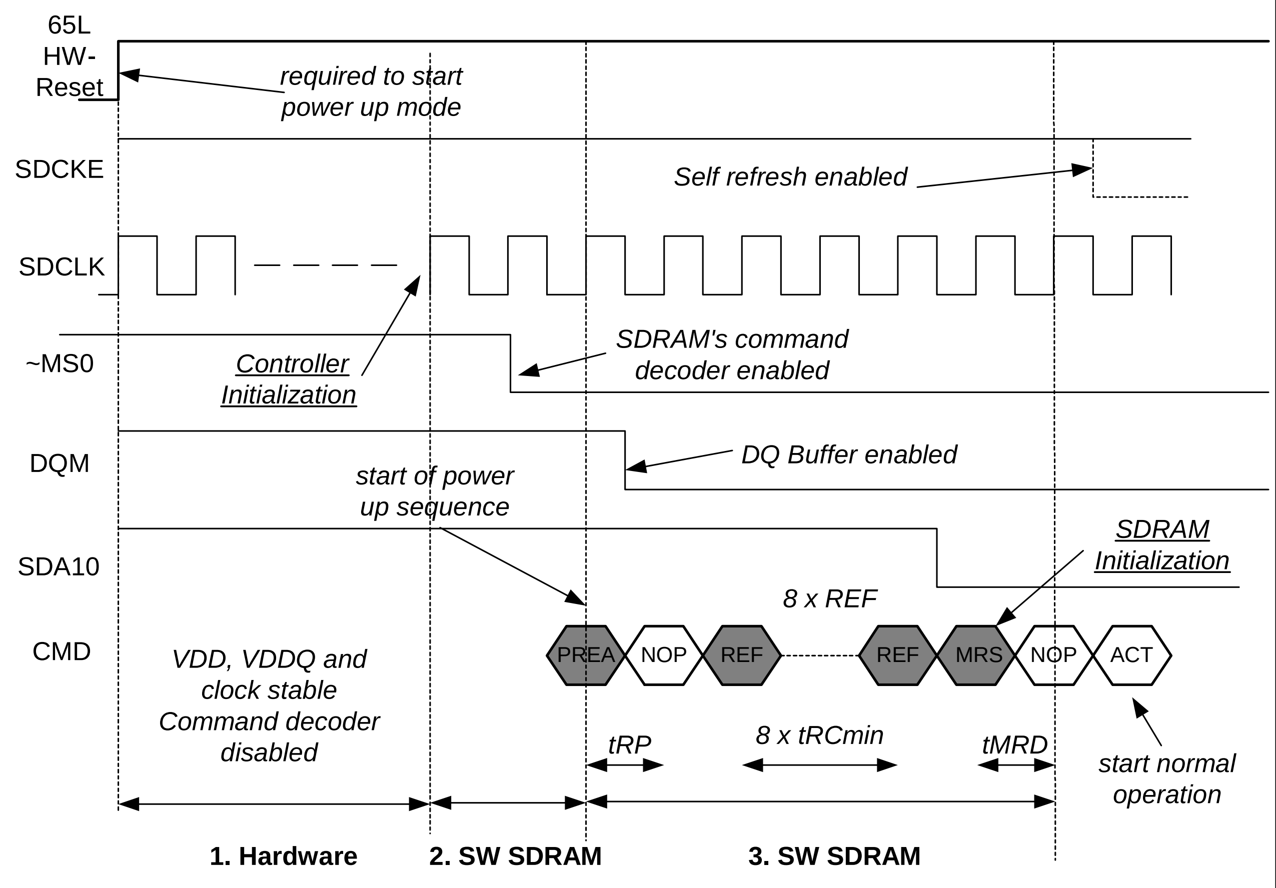
“Micron MT48LC1M16A1”, 1M x 16bit

Speed grade=12, Power up mode: PREA-REF-MRS
tRCD=30 ns, tREF= 32ms, tRASmin=72 ns, tRPmin=36 ns

SDCLK	**20**	**33**	**40**	**50**	**60**	**MHz**
SDTRAS	2	3	3	4	5	*tRASmin*
SDTRP	1	2	2	2	3	*tRPmin*
SDCL	1	1	2	2	2	*CLmin*
-	3	5	5	6	8	*tRCmin, fixed by controller*
-	1	1	2	2	2	*tRCDmin, fixed by controller*
SDRDIV	306	513	617	773	930	*tREF*

Note: As the SDRAM is designed for common PC cache technology, some vendors have a read latency setting between 2 or 3 SDCLK cycles only.

Figure 5: The Initialization and power up mode (PREA-REF-MRS)



6 – Timing of Power up Sequence

The whole procedure (figure 5) is executed in 3 steps:

6.1 – Hardware

After a hardware reset of the ADSP-21065L, the SDCLK clock and the SDRAM's power supply pins VDD and VDDQ must provide a stable signal for a typical minimum time of 200µs. After this time is elapsed, the IOCTL register can be accessed.
Note: If you don't meet this requirement, the SDRAM will not work properly.

6.2 – Software Controller

The ADSP-21065L core executes the SDRDIV and IOCTL settings, which take for each 2-core cycles. It enables first the command decoder as soon as the dedicated ~MSx line is asserted.

6.3 – Software Uniprocessor SDRAM

The bit SDPSS in IOCTL starts the power up sequence. Depending on the settings, the power up sequence starts with its first command PREA. According to this, the DQM line is deasserted while enabling the I/O buffer of the SDRAM. Finally after the MRS, the device is ready for normal operation.
The first access can occur after:

$taccess \approx tRP + 8(tRAS + tRP) + tMRD$ (SDCLK cycles)

Note: In order to initialize properly SDRAM, the first access to it is delayed with the int. ACK until the power up sequence has finished.
Note: The controller keeps the ~MSx line low in order to perform further refresh commands.

6.4 – Example

Data sheet of a vendors 1M x 16 bit:
• 2 banks, page 256 words
• SDCLK = 33,3 MHz
• Speed grade = 10
• Refresh cycles = 2k/32ms
• Power up mode: PRE - MRS - REF
• CLmin=2 @33,3MHz, tRASmin=50 ns, tRPmin=20 ns
• No self refresh
• No buffering
• Mapped to bank 0 of SHARC

```
SDRAM:    ustat1=0x21AD6B41;
          dm(WAIT)=ustat1;         /*int. WS only, 0 WS*/
          ustat1=513;
          dm(SDRDIV)=ustat1;        /*Refresh Counter @ 33,3 MHz*/
          ustat1=0x892A2800;        /* tRAS=2, tRP=1, CL=2   */
          dm(IOCTL)= ustat1;        /*Control, Power-up sequence*/
```

Note: The minimum specs (tRAS, tRP, CL) must be guaranteed. Higher settings cause a degradation of performance only; tREF is a maximum spec to meet.
Note: You can start the power up sequence in the same cycle as the configuration bits

6.5 – Interface after Reset

When the Reset pin from the ADSP-21065L is deasserted after power-up, the SDRAM-lines are in following states:

Pin	*State after Reset*	*Description*
SDCLK0	driven	master clock 0 driven
SDCLK1	driven	master clock 1 driven
SDCKE:	1	both master clocks enabled
~MSx:	1	command decoder disabled
~RAS:	1	deselected
~CAS:	1	deselected
~SDWE:	1	deselected
DQM	1	data buffer disabled
SDA10	1	provide to access all banks simultaneously

6.6 – Disabling the Interface

Writing to the DSDCTL bit in IOCTL register puts the interface in Hi-Z with:

SDCLK0	(I/O/T/S)	
SDCKE	(I/O/T)	
~RAS	(I/O/T)	
~CAS	(I/O/T)	
~SDWE	(I/O/T)	
SDA10	(O/T)	
DQM	(O/T)	
~MSx	(I/O/T)	(not Hi-Z if SDBS bit field in IOCTL is set to 000=no SDRAM)

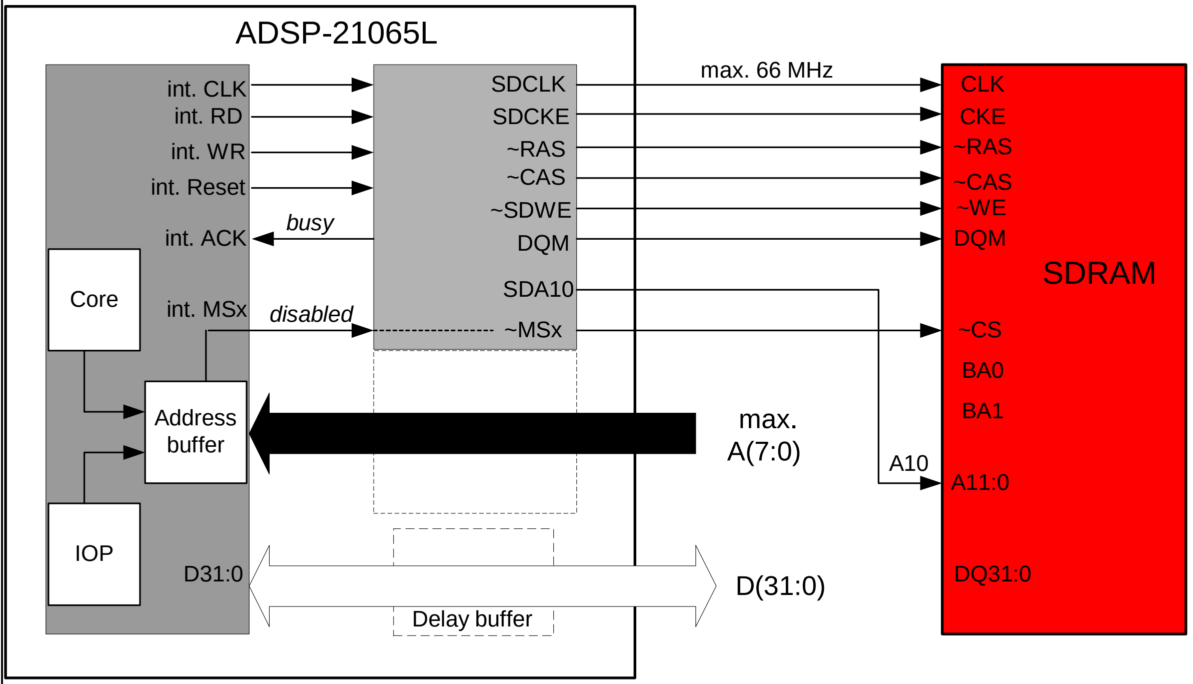
The second clock source can be put in Hi-Z by writing to the bit DSDCK1 in IOCTL register.

SDCLK1	(O/T/S)

Note: If your system does not use SDRAM, set both bits DSDCTL and DSDCK1 to 1, the bit SDBS=000, no SDRAM. The interface pins should be left unconnected.

6.7 – Setting the I/O Flags

Figure 6: Signal chain: Host to ADSP-21065L



The IOCTL register shares the SDRAM settings and the input/output configuration for the memory mapped I/O flags4-11. Changing the direction of these flags during runtime requires that all SDRAM settings be rewritten as well. Otherwise the ~MSx line to the SDRAM is deasserted, which results in blocking further commands including the refresh.
Note: If you change the flags, you must rewrite all SDRAM settings, otherwise the SDRAM interface stops working.

7 – Interface in Host Mode

Note: In host mode, the SDRAM is not directly accessible. Only the auto refresh command can be issued with the help of SDA10.

If the SHARC has answered with ~HBG to a host request ~HBR, it will start working in slave mode. In slave mode, the host can only directly access the IOP registers A[7:0] of the SHARC (figure 6). The direct SDRAM access via DMA is not possible, while the SDRAM's bank select lines are not accessible by the controller. But to keep the SDRAM's information during host access, the controller issues REF commands with the help of SDA10, which is independent from the host address line 10.
This status for the interface changes as following:

Pin	*Slave Mode*	*Description*
~MSx:	0	command decoder enabled
~RAS:	driven	command input
~CAS:	driven	command input
~SDWE:	driven	command input
SDCKE:	1	enabled
SDCLK0	driven	clock 0 driven
SDCLK1	driven	clock 1 driven
DQM	driven	data mask function

SDA10	driven	simultaneous access to all banks
A[7:0]	*driven by host*	*address lines*

8 – DMA Transfers

8.1 – Internal Memory and SDRAM

Since the SDRAM is mapped to any one of SHARC memory banks, it can be externally accessed as source or destination for a DMA transfer. Only the DMA master mode can transfer data between SDRAM and internal memory. In this mode, the I/O Processor initiates transfers up to 264 Mbyte/s.

Parameter	*IIx*	*IMx*	*ICx*	*EIx*	*EMx*	*ECx*
Master Mode	yes	yes	yes	yes	yes	yes

8.2 – Host and SDRAM

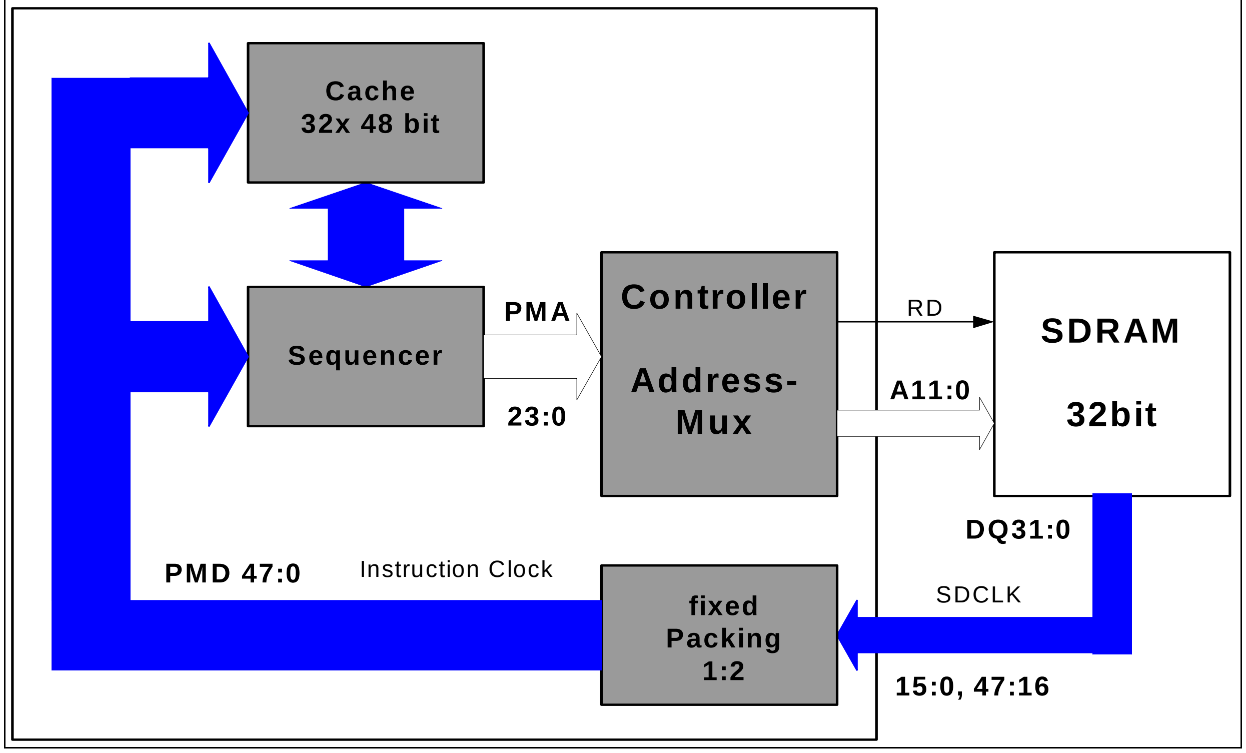
There is no possibility to transfer data directly between a host and the SDRAM.
The Host can only access asynchronously the IOP Registers of the SHARC. Therefore, the user should start 2 DMA transfers via the SHARC's internal memory:

- Slave DMA, Host to internal memory
- Master DMA, Internal memory to SDRAM

Note: The external handshake (fly by DMA) and paced master mode are not supported for the SDRAM.

9 – Code Execution from SDRAM

Figure 7: Signal flow for code execution from SDRAM



The figure 7 demonstrates the signal flow for code execution from SDRAM. The Program sequencer issues a 24 bit PM address to the controllers's input. The controller's multiplexed addressing operation reads a dedicated location in the SDRAM. Since data packing and latency cycles slow down the dataflow, the sequencer receives the 48-bit opcode of the issued address some cycles delayed.
Note: The sequencer's 24-bit address allows program execution in external bank 0 only.

9.1 – Requirements

As the external port of the low cost SHARC is reduced to 32-bits, every instruction execution in SDRAM requires 2 read accesses from the controller. This packing mode is supported by hardware in order to generate a large instruction with 48-bit. The linker's memory command in the VisualDSP++ linker description file (.LDF) is used to activate the packing mode for code execution when the SDRAM's 64k segment is declared with PM width of 48-bit:

```
Memory
{
seg_rth   { TYPE(PM RAM) START(0x00008000) END(0x000080FF) WIDTH(48) }
     .
     .
bk0_pmco  { TYPE(PM RAM) START(0x00020000) END(0x0002FFFF) WIDTH(48) }
}
```

Note: The maximum performance is 50 % compared to executing the same code in internal memory.

Next table demonstrates the hardware-packing scheme:

24bit PM address	*24bit SDRAM address*	*32bit PM data*	
0x20001	0x20002	[15:0]	first part of instruction
	0x20003	[47:16]	second part of instruction

Due to the fixed packing mode of 1:2, the scheme limits the size of the internal contiguous program segments to 64k. The sequencer's 24 bit addresses (e.g. 0x20001) differ from 32bit SDRAM (e.g. 0x20002, 0x20003) addresses (see table below). Using multiple segments, the program can incorporate jump instructions towards the end of individual 64k segments. The next formula calculates the PMA into SDRAM address:

Fixed packing 32:48	2 SDRAM addresses = PMA*2 – Startaddress PMA

e.g. PMA = 0x25880 SDRAM address = 0x25880 * 2 – 0x20000 = 0x2B100 and 0x2B101

Note: The PM width must be declared as 48 bits in the linker description file, if declared to 32 bits, the packing scheme will be inactive.

24 bit PM address		*24 bit SDRAM address*	
0x20000	start of 64k for sequencer	**0x20000**	start of the 128k SDRAM segment
		0x20001	
0x20001		0x20002	
		0x20003	
0x2FFFF	end of 64k for sequencer	**0x3FFFE**	
		0x3FFFF	end of the 128k SDRAM segment

Note: The SDRAM address space requires twice as much space as the program sequencers PMA space. Due to this, some memory segments are not accessible for the application.

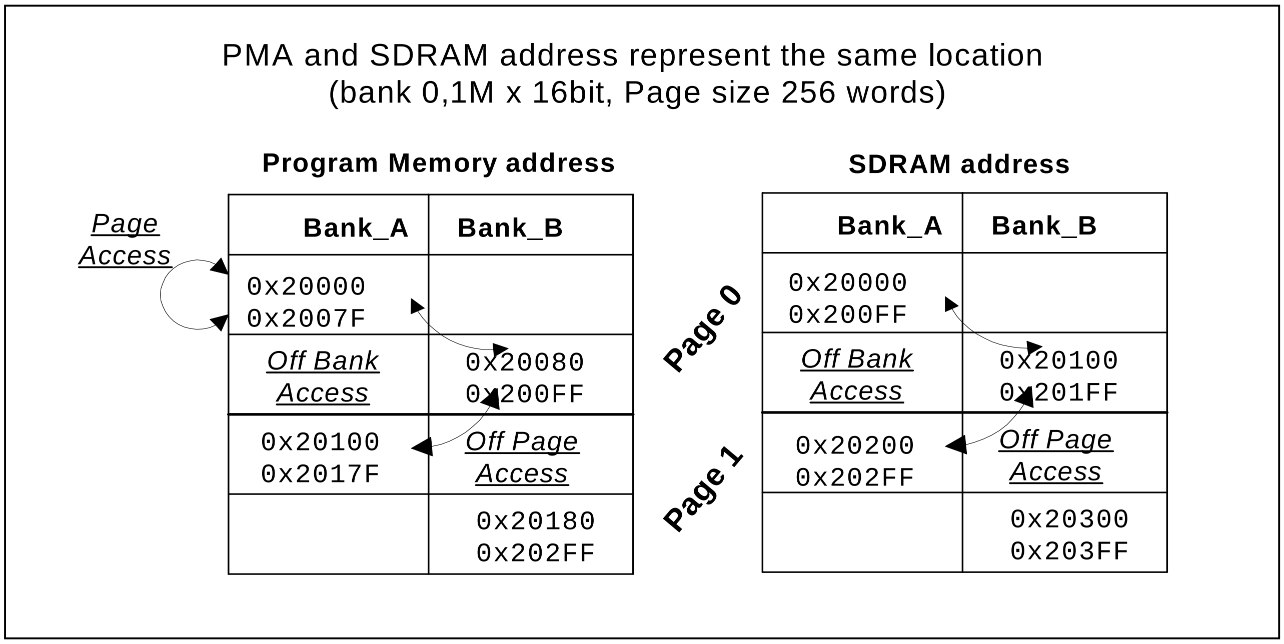
9.2 – Instruction Pipeline

The next code extract will be analyzed:

```
PMA       SDRAM     Instruction
20000:    20000     bit set ASTAT FLAG0;          */ instr. 1 */
          20001
20001:    20002     bit clr ASTAT FLAG0;          */ instr. 2 */
          20003
20002:    20004     next instr;                   */ instr. 3 */
          20005
```

The first instruction (figure 8) gets executed after roughly 11 cycles. To understand this number, the following sequence has to be considered:

1st step: the program sequencer sends a row activation command to the SDRAM.
2nd step: the column is opened with the period of time tRCD, e.g. 2 cycles.
3rd step: the data are valid within the CAS latency, e.g. 2 cycles.
4th step: in order to execute an instruction, the 3 level pipeline has to be filled. The 1st instruction gets fetched, decoded (meanwhile the 2nd Instruction gets fetched) and then executed (same time the 2 instruction gets decoded), within 2 cycles each.
This adds up to the complete delay of the first instruction: tRCD+CL+6 cycles. From here on every instruction of straight line code (SDRAM in sequential address order) gets executed every 2 cycles, since the pipeline is filled.



Note: For debug, the "tree column memory" of VDSP++ displays the PMA address of the sequencer; the "two column memory" displays the SDRAM address.

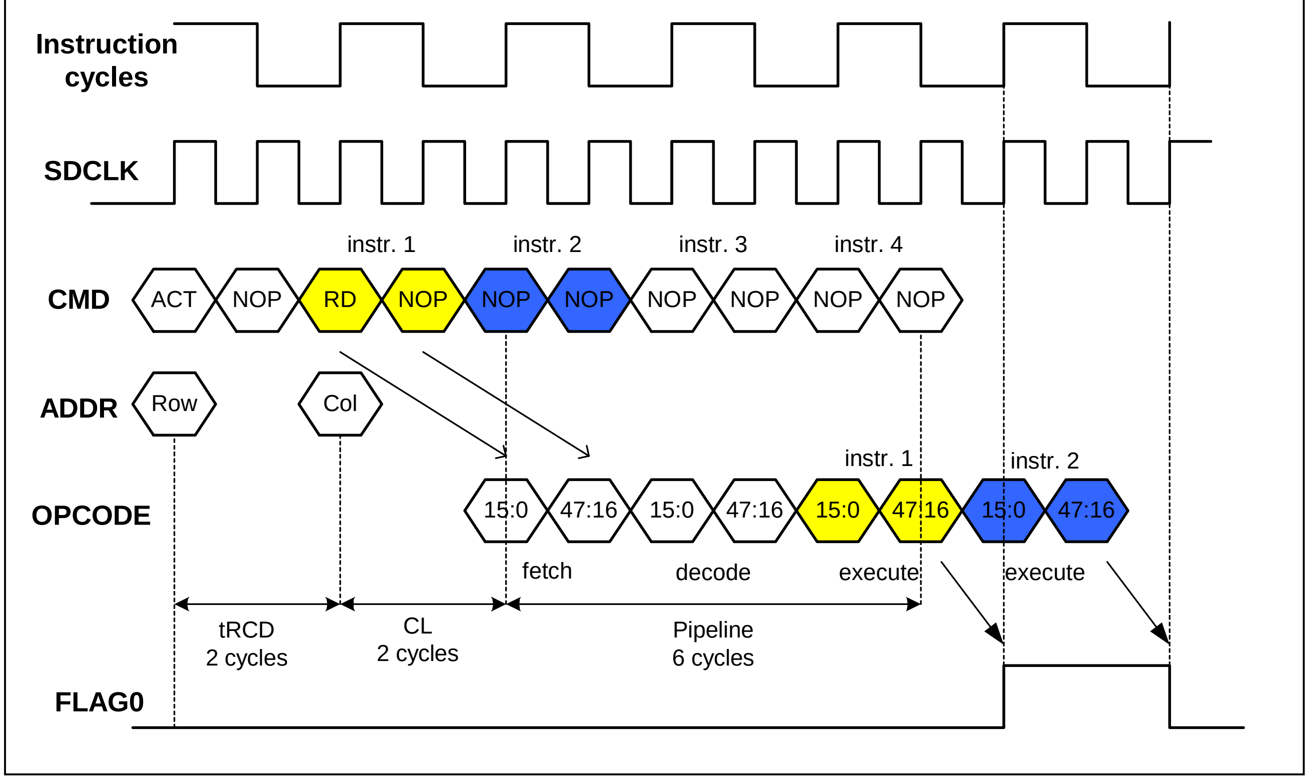
9.3 – Sequential Code crossing Page/Bank

This mode describes a sequential code, which exceeds the page or the bank. As the code is sequential, the pipeline must is not flushed in order to lose performance. Stall cycles are inserted for the following:

- Precharging the current page or bank
- Activation of the new page or bank

Note: Take the overhead into consideration while crossing between pages or banks of SDRAM.

Figure 8: the instruction pipeline for sequential code



9.4 – Non Sequential Code same Page/Bank

Interrupt service routines or subroutines require non-sequential addressing. With the help of delayed branch instruction (db), there is a possibility to reduce the bottleneck as much as possible using the *jump(db) or call(db)* instructions. The pipeline doesn't have to be flushed; because the two delayed instructions are executed in 4 SDCLK cycles before the jump or call. This fact improves the performance by 4 SDCLK cycles compared to non-delayed branches. For non delayed branches:

- Flush the pipeline
- Fill the pipeline

9.5 – Non Sequential Code different Page/Bank

When the code jumps between different pages or banks, additional terms should be taken into consideration: the pipeline must be flushed which adds extra cycles to the procedure:

- Flush the pipeline
- Precharge the current page or bank
- Activate the new page or bank
- Fill the pipeline

Note: This sequence builds the worst case for stall cycles.

10 – Loop Execution from SDRAM

Basically, single instruction loops are analyzed to figure out the controller's loop handling

10.1 – Single Loop with DM Data Access

In this mode, the burst sequence must be interrupted, since the addressing mode is not sequentially built. For each loop iteration, the controller performs the sequence listed in figure 9: The first two accesses fetch the instruction in the first iteration, 3 dummy reads (NOP's) are inserted to interrupt the burst. The next iteration fetches the instruction again. Thus every iteration requires 5 SDCLK cycles per loop.

10.2 – Single Loop with PM Data Access, Cache enabled

Just like DM data access, (figure 9): but since the sequencer checks for the fetched instruction in the cache (core access), the BST must be issued and every iteration requires 6 SDCLK cycles per loop iteration. In case of cache hits, the instructions are executed with 1 cycle/word in the core during another BST.

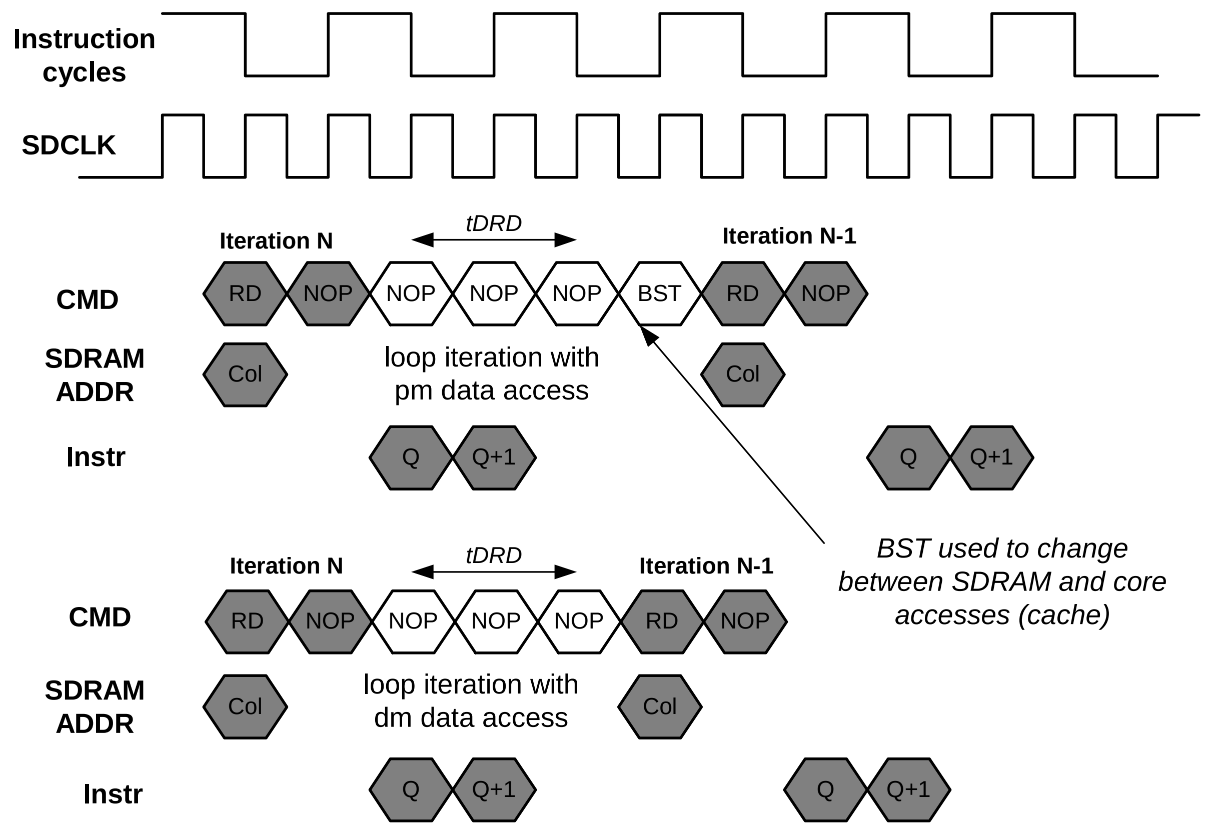
10.3 – Single Loop with PM Data Access, Cache disabled

This mode is basically same to the PM data access with one difference that every instruction is executed from SDRAM and takes 5+1=6 SDCLK cycles per iteration.

Example: single pm data loop

PMA	SDRAM	Instruction
20000:	20000	bit set ASTAT FLAG1;
	20001	
20001:	20002	lcntr=4, do (pc,1) until lce;
	20003	
20002:	20004	f12=f12+f8,pm(i8,m8)=f12; /* cached */
	20005	

Figure 9: Loop Iteration Handling: pm-access vs. dm-access (CL=2 cycles)



```
20003:    20006    bit clr ASTAT FLAG1;         /* cached */
          20007
20004:    20008    f10=f4*f6;                   /* cached */
          20009
20005:    2000A    next instr;
          2000B
```

In the code example, the controller fetches the instructions at PM address 0x20000 and 0x20001 (SDRAM addresses 0x20000-0x20003). Address 0x20002 (SDRAM addresses 0x20004, 0x20005) with pm access is loaded into the cache, therefore a BST is issued. The sequencer restarts with another read at the same address and detects a cache hit. After the BST, each instruction is executed at a rate 1 cycle/word. The next-to-last iteration caches 0x20003 (SDRAM addresses 0x20006, 0x20007) with BST again. The last iteration fetches 0x20004 (SDRAM addresses 0x20008, 0x20009) and the loop execution has finished. This explains, why a single pm data access loop with 3 iterations gives no advantage since the instructions are only cached and not executed. From four iterations on, all additional iterations are executed from the cache, thus increasing the cache efficiency a lot.

10.4 – Code Execution Performance Overview

Sequential Code	***on page***	***off page/bank***	***stall cycles***	***testcase***
fill pipeline	x		*tRCD+CL+6*	*10*
sequential crossing		x	*tDRD+tRP+tRCD+1*	*7*
Non sequential Code	***on page***	***off page/bank***	***stall cycles***	***testcase***

branch	x		*tDRD+6*	*9*
branch (db)	x		*tDRD+2*	*5*
branch		x	*tDRD+tRP+tRCD+1+6*	*13*
branch (db)		x	*tDRD+tRP+tRCD+1+2*	*9*
Single Instr. Loop	***cache***	***no cache***	***cycles / iteration***	
PM data	x		*1 upon 4th iteration*	
PM data		x	*6*	
DM data	-	-	*5*	

Note: Testcase: tRP=1, CL=2, the controller sets tRCD=CL, tDRD=CL+1, SDCLK cycles

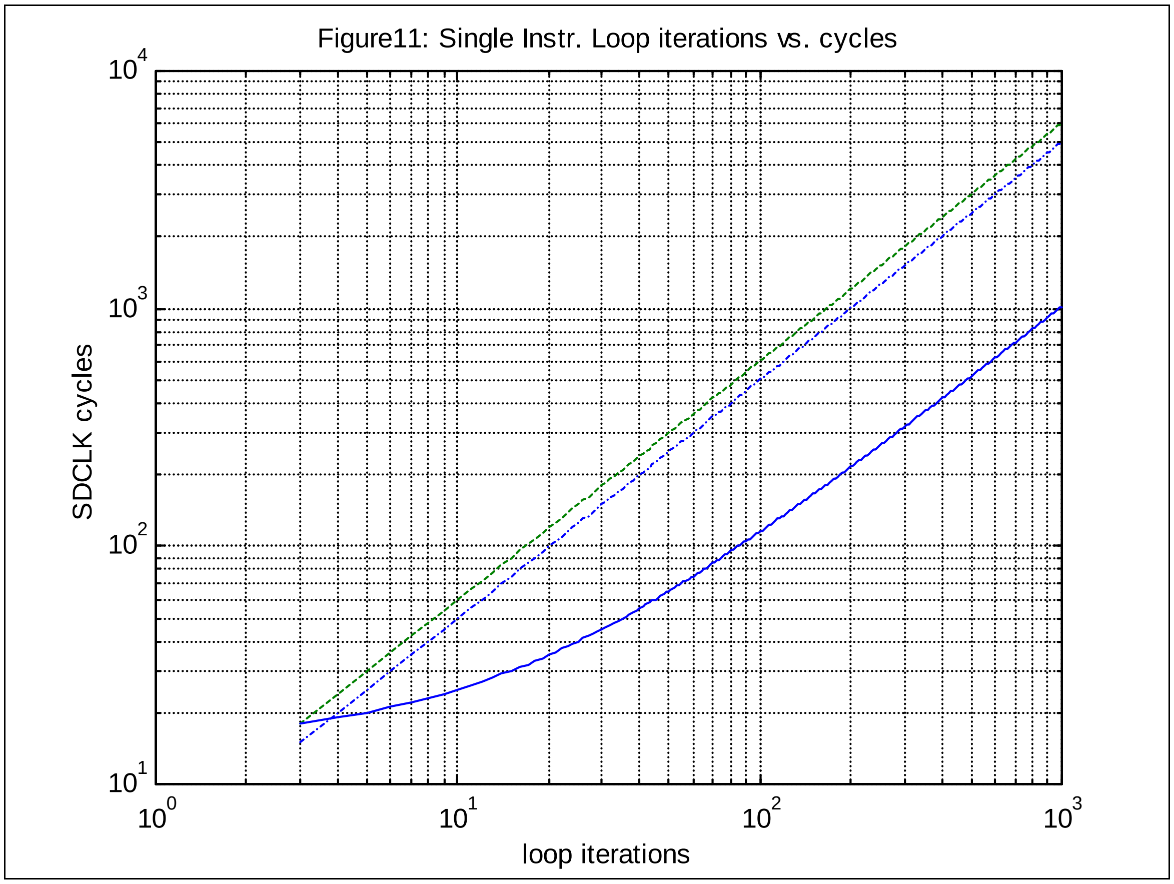


Figure 11 illustrates the following:
For high loop iterations for instance 1000, the dashed line (pm, no cache) takes about 6000 cycles to execute. The dash dot line (dm) takes about 5000 cycles. The best case gives of course the use of cache. This results to 1000 cycles only.

11 – Multiprocessing

This section covers the arbitration logic used to guarantee multiprocessing systems.

11.1 – Command Decoding

The ADSP-21065L can be connected to a multiprocessor cluster of two. Only one SHARC can drive the bus at the time. To build a glueless hardware, the interface works in slave mode as well to detect commands. These commands are MRS, REF and SREF. Following pins are necessary for detection:

Pin	***State***	***Description***
SDCLK0	(I/O/T/S)	master clock input
SDCKE	(I/O/T)	clock enable
~MSx:	(I/O/T)	command input
~RAS:	(I/O/T)	command input
~CAS:	(I/O/T)	command input
~SDWE:	(I/O/T)	command input

I=input, O=output, T=Hi-Z, S=synchronous
Note: SDCLK1 and SDA10 are not required to detect a command.

11.2 – MRS Decoding

Power up sequence can be done by any of two DSPs. If all two DSPs set the bit SDPSS in IOCTL, the one which has busmasterchip will do power-up. The slave recognizes power-up with the MRS and clears its read-only SDPSS bit.
Note: For multiprocessing, the control settings must be identical for each DSP.
Note: For multiprocessing, the MRS is issued once.

11.3 – REF Decoding

This detection helps to synchronize all six refresh counters. The slave’s SDRDIV counter will be decremented each time the interface detects a refresh. This feature guarantees a periodic refresh and requires the exact same settings of SDRDIV and IOCTL registers.

11.4 – SREF Decoding

This detection helps to synchronize the self-refresh base. The master’s bit SDSRF brings the whole system in self-refresh mode. The slave recognizes the SREF and set therefore its bit SDSRF. The current SDRDIV counters will be frozen until the self-refresh is exited.

11.5 – Bus Transition Cycle

The bus transition cycle is used to arbitrate between the six controllers. When one processor receives bus masterchip from the other, it executes a precharge all (PREA) before its first access to SDRAM only if the previous master had accessed SDRAM.

12 – Optimizing the Performance

12.1 – SDRAMs in Parallel

The market doesn't offer the combination of big page size (e.g. 1024 words) and a wide interface (32 bit), which would be desirable for DSP applications. Parallel connection does the trick: the advantage of good SDRAM performance for the price higher hardware requirements.

Note: The SHARC bank supports maximal 16M x 32bit.

The table below lists the possible configuration of different pages sizes creating a 32-bit I/O structure:

256 words

Number	Type	Banks	whole size	capacity load	
1	2M x 32	4	2M x 32	low	
2	4M x 16	4	4M x 32	low	
2	1M x 16	2	1M x 32	low	(ADSP-21065L EZ Kit Lite)

512 words

Number	Type	Banks	whole size	capacity load
4	2M x 8	2	2M x 32	medium
4	8M x 8	4	8M x 32	medium
2	8M x 16	4	8M x 32	medium

1024 words

Number	Type	Banks	whole size	capacity load
8	4M x 4	2	4M x 32	high
8	16M x 4	4	16M x 32	high
4	16M x 8	4	16M x 32	medium

Note: In cases of high capacity load, you should use the buffering feature.

Using a 2Mx32 SDRAM, the controller setting is: page=256, banks=4. Connecting two 1Mx16 parallel results also to 32bit but with 2 banks only (settings: page=256, banks=2).

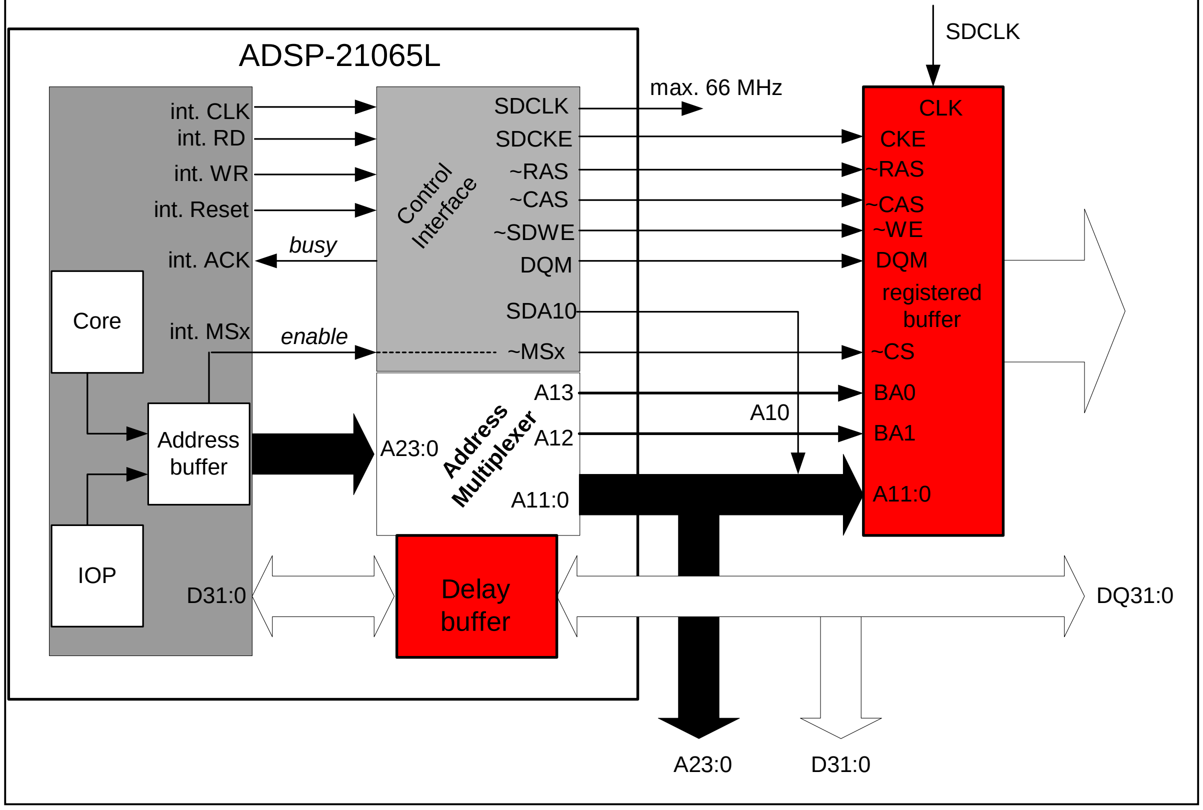
12.2 – External Buffering

In parallel connection one address and control bus feeds all devices. In order to meet the timing requirements, help against the capacitive load would be desirable. Therefore, the controller provides 2 features to cope with this situation:

- 2 x clock out lines (SDCLK0/SDCLK1) sharing the clock load
- Set SDBUF bit to 1(SDCTL) inserts a delay buffer to allow address- and command pipelining

Note: The external buffer reduces the power dissipation but increases the pipeline effects.

Figure 1: Signal chain: External buffering



12.3 – Using PC Modules

The maximum addressable size is 16Mx32bit for one SHARC bank only. The use of unbuffered PC DIMM modules (typical I/O sizes x32, x64 or x72 bits) is required but not so efficient, because the most vendors offer a size of x64 bits, which is commonly used for PCs. The SDRAM controller can handle a maximum of x32 bits. Moreover, depending on the size, the need of external address- and control buffers (pipelining) is required.
Note: The ADSP-21065L supports 16M x 32bit or one 64Mbyte module.

12.4 – Rules for Optimized Performance

Some rules to optimize the performance for external code execution:

- Use straight line code for sequential SDRAM addressing if possible, because the burst gives the best performance
- Use delayed branches instructions *jump (db), call (db)* in a page instead of non delayed branches
- Remember the controller's address mapping scheme, don't mix up the PMA (LDF) and SDRAM addresses by mistake
- Depending on the SDRAM's pages size and number of banks, place your code segments to minimize off page- and off bank accesses
- Use parallel connection for SDRAMs with big page size in order to obtain 32 bit (SDRAM: the bigger the page size, the smaller the I/O structure)
- Use DM data access loops for loops with less than 6 iterations

- Insert dummy PM data access to utilize the cache performance of DM data access loops with more than 6 iterations
- Use the SDRAM as code memory only, do not use it as data memory at the same time
- Use the optimized settings for the controller's state machine (*tRAS*, *tRP*, *CL*) depending on the speed grade and on the application's speed

13 – SDRAM and Booting

13.1 – Loader Kernel

When executing code or data storage from/to SDRAM, the code or data must be loaded in the device before runtime. This requires an initialized SDRAM before downloading the code during the boot scenario. The Tool's loader provides this capability.
2 standard loader executables are available to boot with EPROM or Host. Therefore, the kernels must be modified with the setup of SDRAM during the first 256 words of the loader kernel.

Note: Independent from boot mode, the SDRAM must be initialized before the user's application starts writing data or code to it. This is possible during the first 256-loader kernel words, otherwise the data will get corrupted.

Next table summarizes the steps:

- 1. ~RESET must be deasserted, the pins BSEL and ~BMS are sampled
- 2. Kernel (256 x 48 bit) is drawn down during hardwired DMA into the DSP (0x8000-0x80FF)
  (~BMS or ~HBG continuously asserted)
- 3. Interrupt generation starts kernel execution, <u>SDRAM and controller are initialized</u>
  (~MSx asserted for SDRAM setup)
- 4. User's code and data are loaded into the DSP (0x8110-) or SDRAM with the help of TAGs
  (~BMS, ~HBG and ~MSx are toggling requesting the bus)
- 5. Kernel is now overwritten with user's interrupt vector table
  (~BMS or ~HBG continuously asserted)
- 6. DSP starts code execution at address 0x8005 or (0x20000, SDRAM)

Note: Do not violate the Initialization time, VDD, VDDQ and CLKIN and clock must be stable for typical 200 µs before SDRAM power up mode.
Note: The Tools allows the simulation of the boot process.

13.2 – Booting Modes

The boot mode is selected by hardware after reset, depending on:

Pin	*BSEL*	*~BMS*	*hardwired DMA channel*
EPROM	1	output	8
Host	0	1	8

No boot	0	0	-

13.3 – In Circuit Emulation (ICE)

During hardware debug sessions, you can quickly start the SDRAM controller with the emulator's software.

- First, push the hardware reset button
- Second, load an *SDRAM_INIT.DXE* file to start the SDRAM controller.
- Third, load the *USERCODE.DXE* file to start the SDRAM specific application.

Note: Only a hardware reset of ADSP-21065L can reinitiate the power up sequence for new settings.
Note: The Read Latency is fully transparent during single step debug sessions.

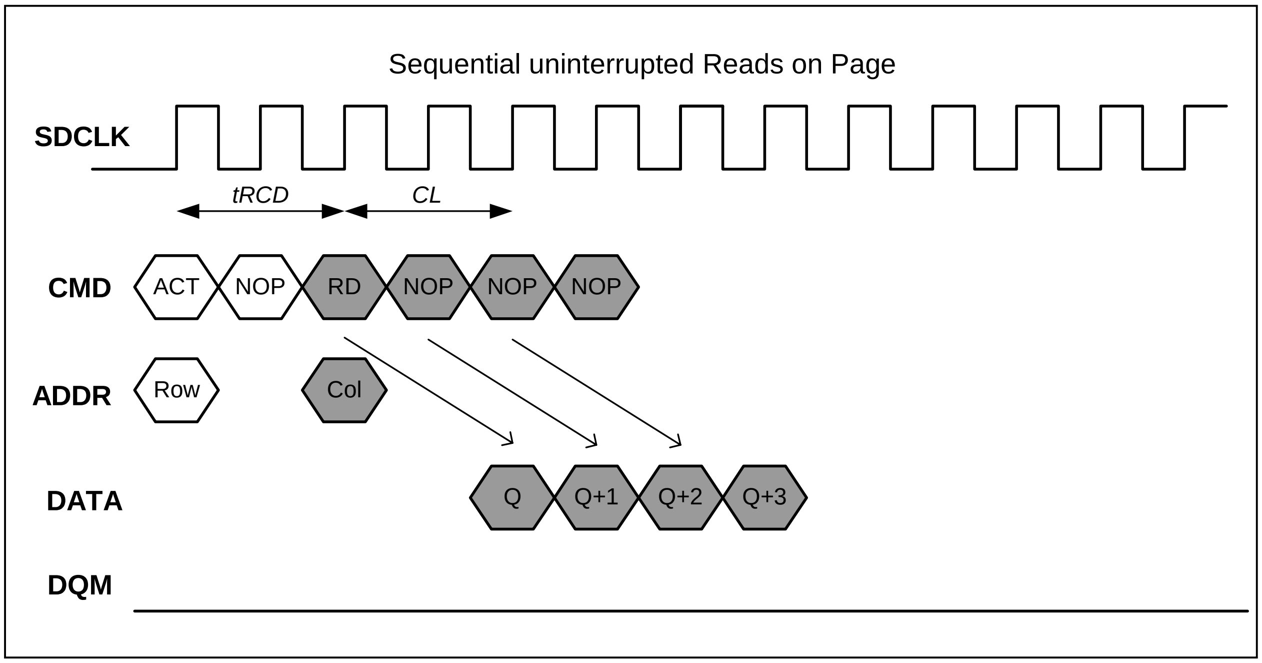
14 – Core and IOP Transfers to SDRAM

In this mode, the SDRAM is used as a source or destination for data transfers. To demonstrate the performance, following settings are used:

Silicon=ADSP-21065L Rev0.3
CLKIN=16,67 MHz
SDCLK=33,3 MHz
~MS0=zero wait states
Size=1Mx32bit
SDRDIV=513 cycles
tRAS=2 cycles
tRP=1 cycle
CL=2 cycles
tRCD=2 cycles

14.1 – Sequential Reads without Interruption

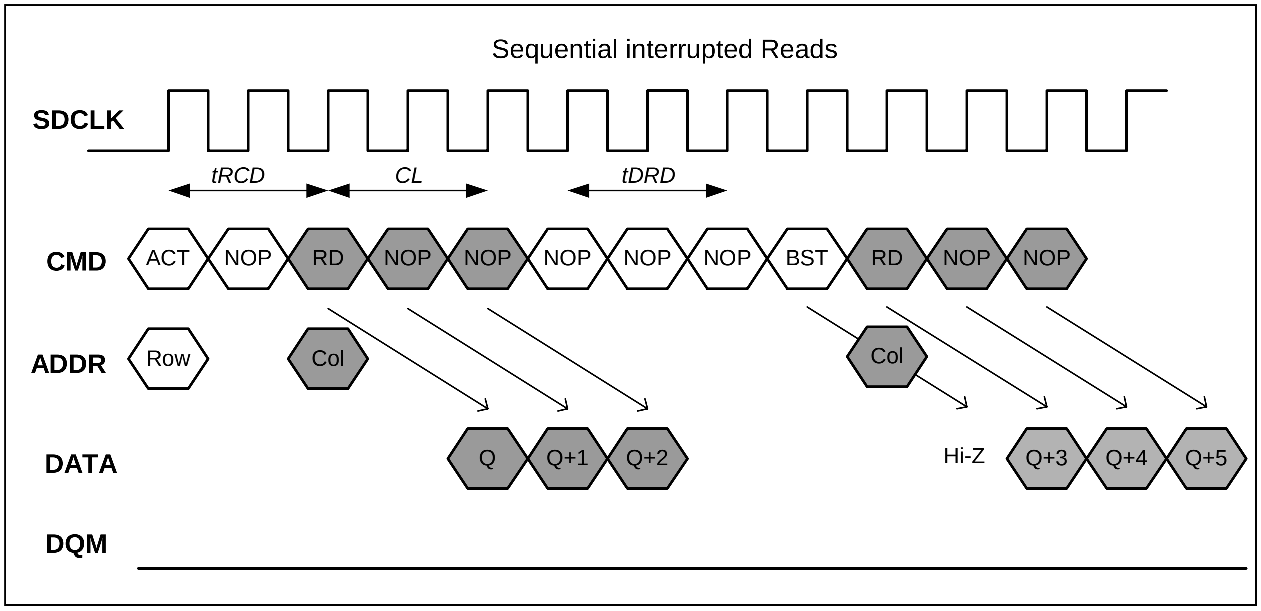
a) Core reads from SDRAM, no interruption caused.
b) IOP reads (EMx=1) from SDRAM, no interruption caused.



Nr. Cycles	*Core*	*Controller*	*Data*
1	**`r1=dm(dest_Q);`**	ACT	
2	int. ACK	NOP	
3	int. ACK	RD	
4	**`r2=dm(dest_Q+1);`**	NOP	
5	**`r3=dm(dest_Q+2);`**	NOP	Q
6	**`r4=dm(dest_Q+3);`**	NOP	Q+1
7		NOP	Q+2
8		NOP	Q+3

14.2 – Sequential Reads with minimum Interruption

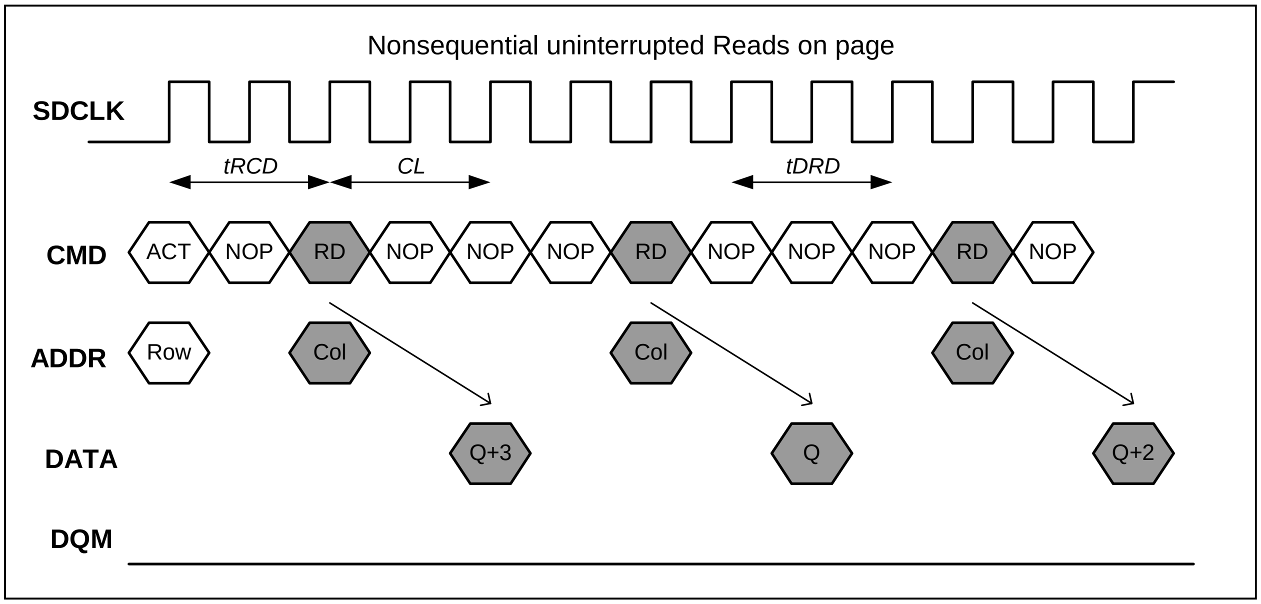
a) Core reads from SDRAM, interruption caused by a core access.
b) IOP reads (EMx=1) from SDRAM, interruption caused by a higher priority request to IOP.



Nr. Cycles	*Core*	*Controller*	*Data*
1	**r1=dm(dest_Q);**	ACT	
2	int. ACK	NOP	
3	int. ACK	RD	
4	**r2=dm(dest_Q+1);**	NOP	
5	**r3=dm(dest_Q+2);**	NOP	Q
6	int. ACK	NOP	Q+1
7	int. ACK	NOP	Q+2
8	int. ACK	NOP	
9	**r5=r6*r7;**	BST	
10	**r4=dm(dest_Q+3);**	RD	
11	**r5=dm(dest_Q+4);**	NOP	
12	**r6=dm(dest_Q+5);**	NOP	Q+3
13		NOP	Q+4
14		NOP	Q+5

14.3 – Non Sequential Reads without Interruption

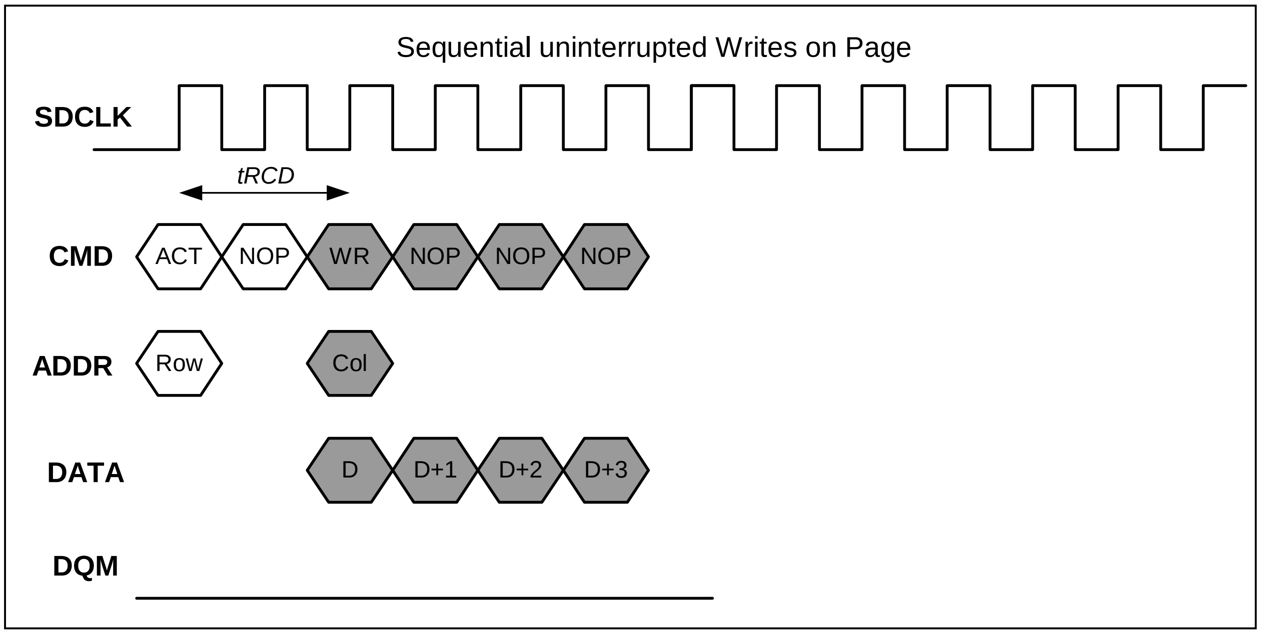
a) Core reads from SDRAM, no interruption caused.
b) IOP reads (EMx>1) from SDRAM, no interruption caused.



Nr. Cycles	*Core*	*Controller*	*Data*
1	**`r1=dm(dest_Q+3);`**	ACT	
2	`int. ACK`	NOP	
3	`int. ACK`	RD	
4	`int. ACK`	NOP	
5	`int. ACK`	NOP	Q+3
6	`int. ACK`	NOP	
7	**`r2=dm(dest_Q);`**	RD	
8	`int. ACK`	NOP	
9	`int. ACK`	NOP	Q
10	`int. ACK`	NOP	
11	**`r3=dm(dest_Q+2);`**	RD	
12	`int. ACK`	NOP	
13	`int. ACK`	NOP	Q+2
14	`int. ACK`	NOP	
15	`int. ACK`	BST	

14.4 – Sequential Writes without Interruption

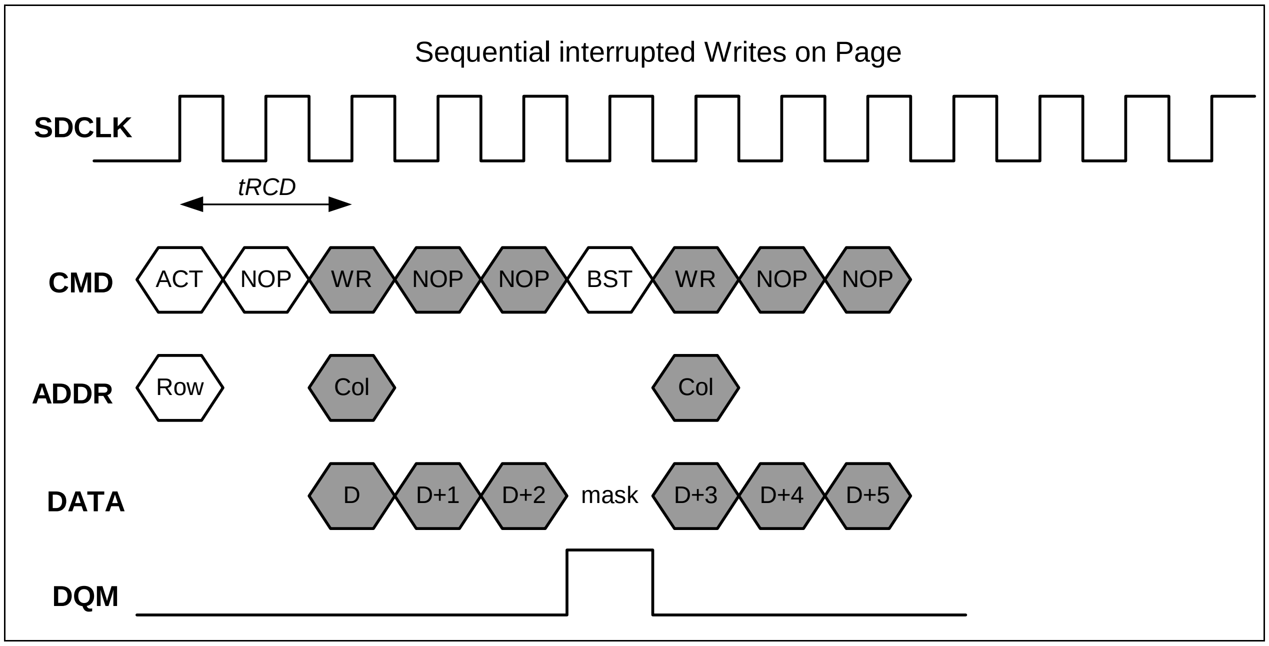
a) Core writes to SDRAM, no interruption caused.
b) IOP writes (EMx=1) to SDRAM, no interruption caused.



Nr. Cycles	*Core*	*Controller*	*Data*
1	**dm(dest_D)=r1;**	ACT	
2	int. ACK	NOP	
3	int. ACK	WR	D
4	**dm(dest_D+1)=r2;**	NOP	D+1
5	**dm(dest_D+2)=r3;**	NOP	D+2
6	**dm(dest_D+3)=r3;**	NOP	D+3

14.5 – Sequential Writes with minimum Interruption

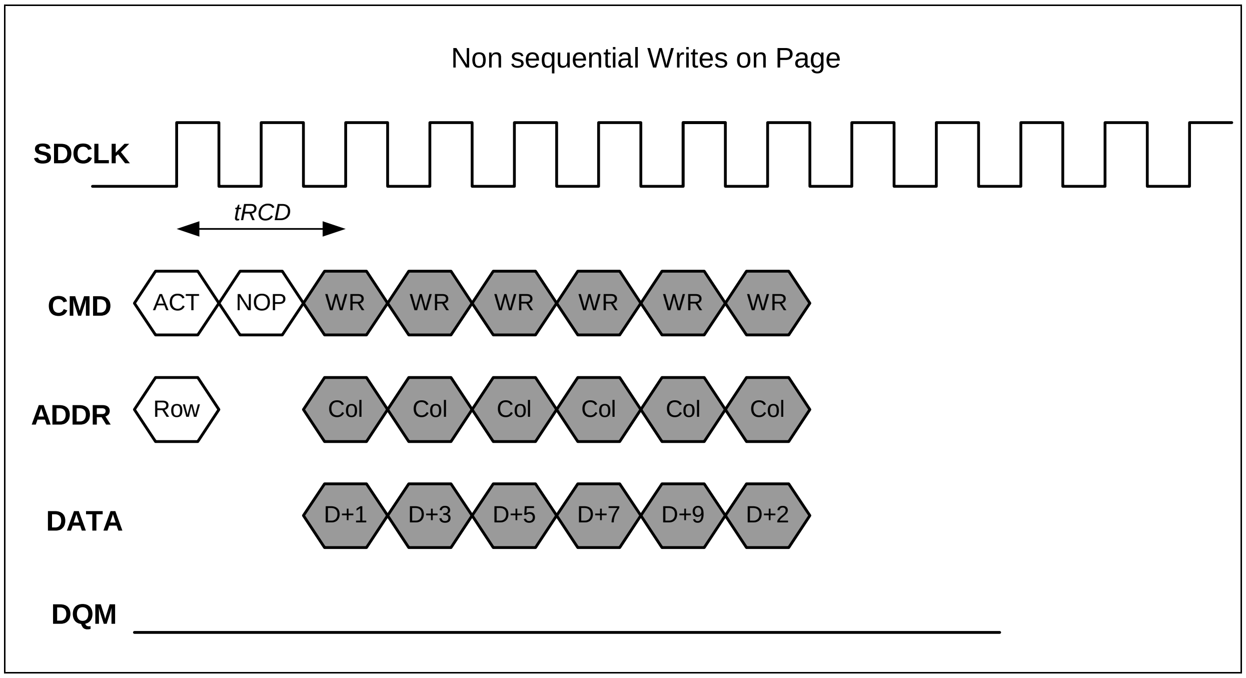
a) Core writes to SDRAM, interruption caused by core access.
b) IOP writes (EMx=1) to SDRAM, interruption caused by a higher priority request to IOP.



Nr. Cycles	*Core*	*Controller*	*Data*
1	**dm(dest_D)=r1;**	ACT	
2	int. ACK	NOP	
3	int. ACK	WR	D
4	**dm(dest_D+1)=r4;**	NOP	D+1
1	**dm(dest_D+2)=r5;**	NOP	D+2
2	**r0=r0+1;**	BST	
3	**dm(dest_D+3)=r6;**	NOP	D+3
6	**dm(dest_D+4)=r7;**	NOP	D+4
7	**dm(dest_D+5)=r8;**	NOP	D+5

14.6 – Non Sequential Writes without Interruption

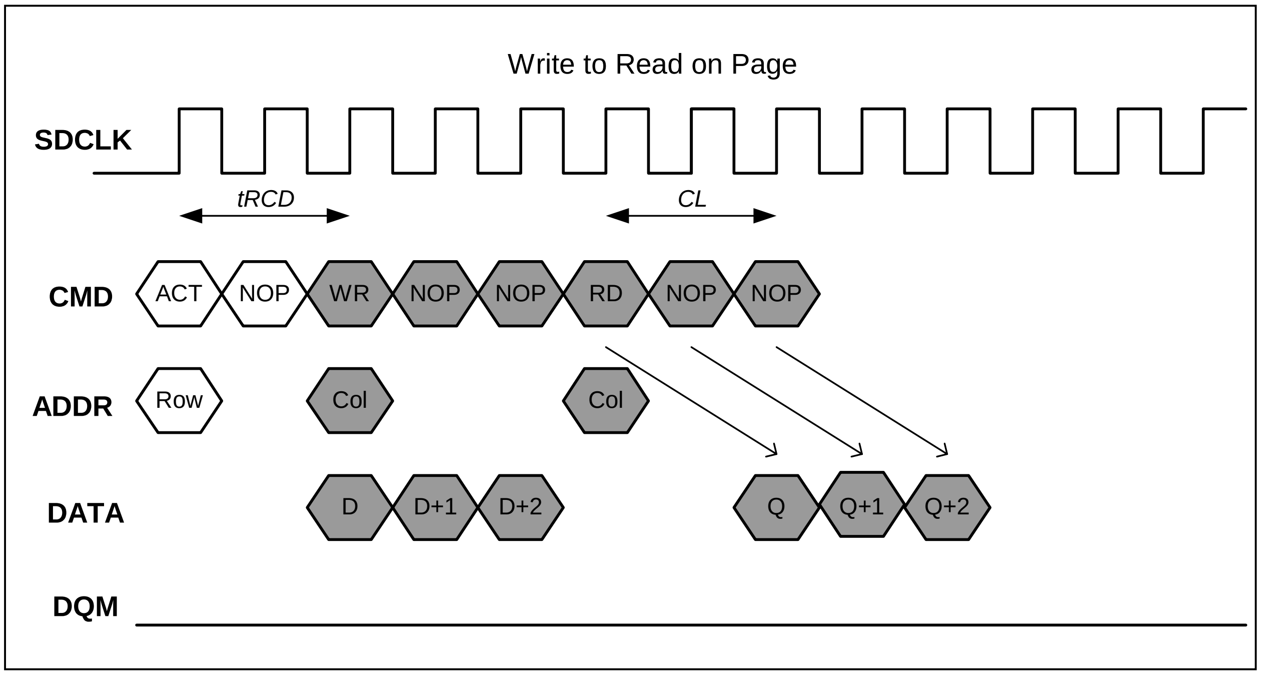
a) Core writes to SDRAM, no interruption caused.
b) IOP writes (EMx>1) to SDRAM, no interruption caused.



Nr. Cycles	*Core*	*Controller*	*Data*
1	**dm(dest_D+1)=r1;**	ACT	
2	int. ACK	NOP	
3	int. ACK	WR	D+1
4	**dm(dest_D+3)=r4;**	WR	D+3
5	**dm(dest_D+5)=r5;**	WR	D+5
6	**dm(dest_D+7)=r6;**	WR	D+7
7	**dm(dest_D+9)=r7;**	WR	D+9
8	**dm(dest_D+2)=r8;**	WR	D+2

14.7 – Minimum Write to Read Interval

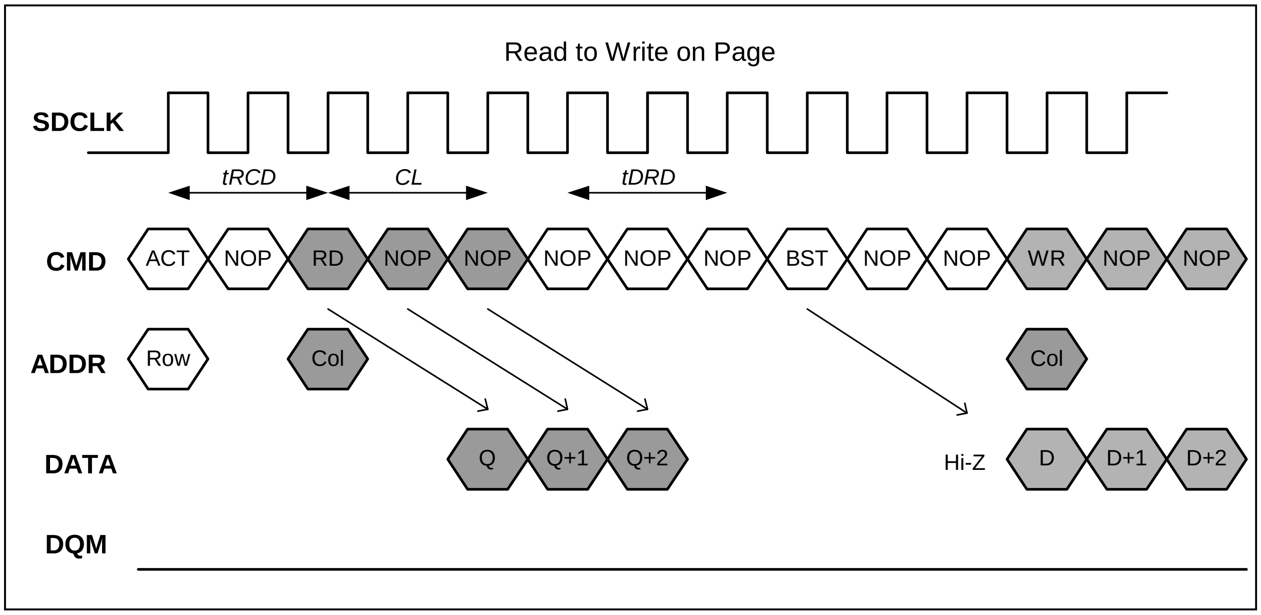
Core writes to and reads from SDRAM.



Nr. Cycles	*Core*	*Controller*	*Data*
1	**dm(dest_D)=r1;**	ACT	
2	int. ACK	NOP	
3	int. ACK	WR	D
4	**dm(dest_D+1)=r1;**	NOP	D+1
5	**dm(dest_D+2)=r1;**	NOP	D+2
6	**r6=dm(dest_Q);**	RD	
7	**r6=dm(dest_Q+1);**	NOP	
8	**r6=dm(dest_Q+2);**	NOP	Q
9		NOP	Q+1
10		NOP	Q+2

14.8 – Minimum Read to Write Interval

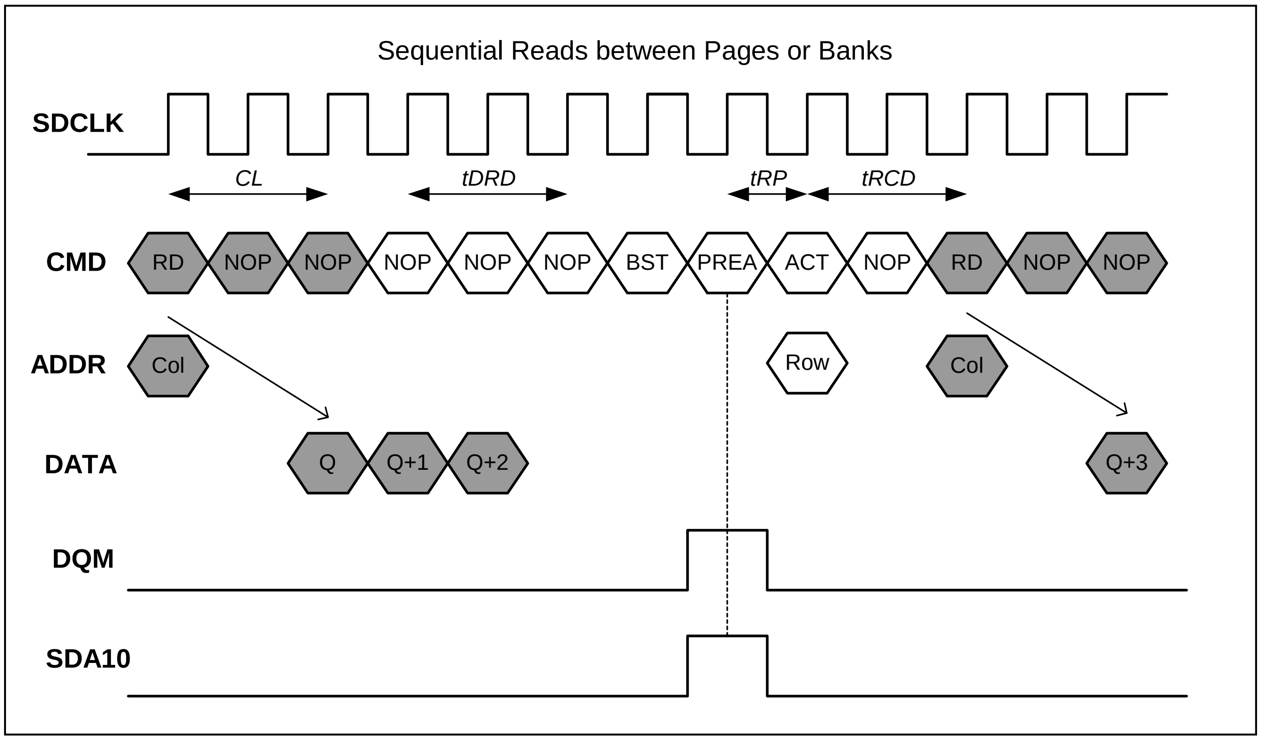
Core reads from and writes to SDRAM.



Nr. Cycles	*Core*	*Controller*	*Data*
1	**`r1=dm(dest_Q);`**	ACT	
2	`int. ACK`	NOP	
3	`int. ACK`	RD	
4	**`r1=dm(dest_Q+1);`**	NOP	
5	**`r1=dm(dest_Q+2);`**	NOP	Q
6	`int. ACK`	NOP	Q+1
7	`int. ACK`	NOP	Q+2
8	`int. ACK`	NOP	
9	`int. ACK`	BST	
10	`int. ACK`	NOP	
11	`int. ACK`	NOP	
12	**`dm(dest_D)=r6;`**	WR	D
13	**`dm(dest_D+1)=r6;`**	NOP	D+1
14	**`dm(dest_D+2)=r6;`**	NOP	D+2
15	**`r0=r0+1;`**	BST	

14.9 – Reads between Page/Bank

a) Core reads from SDRAM, interruption caused by another page or bank.
b) IOP reads (EMx=1) from SDRAM, interruption caused by another page or bank.

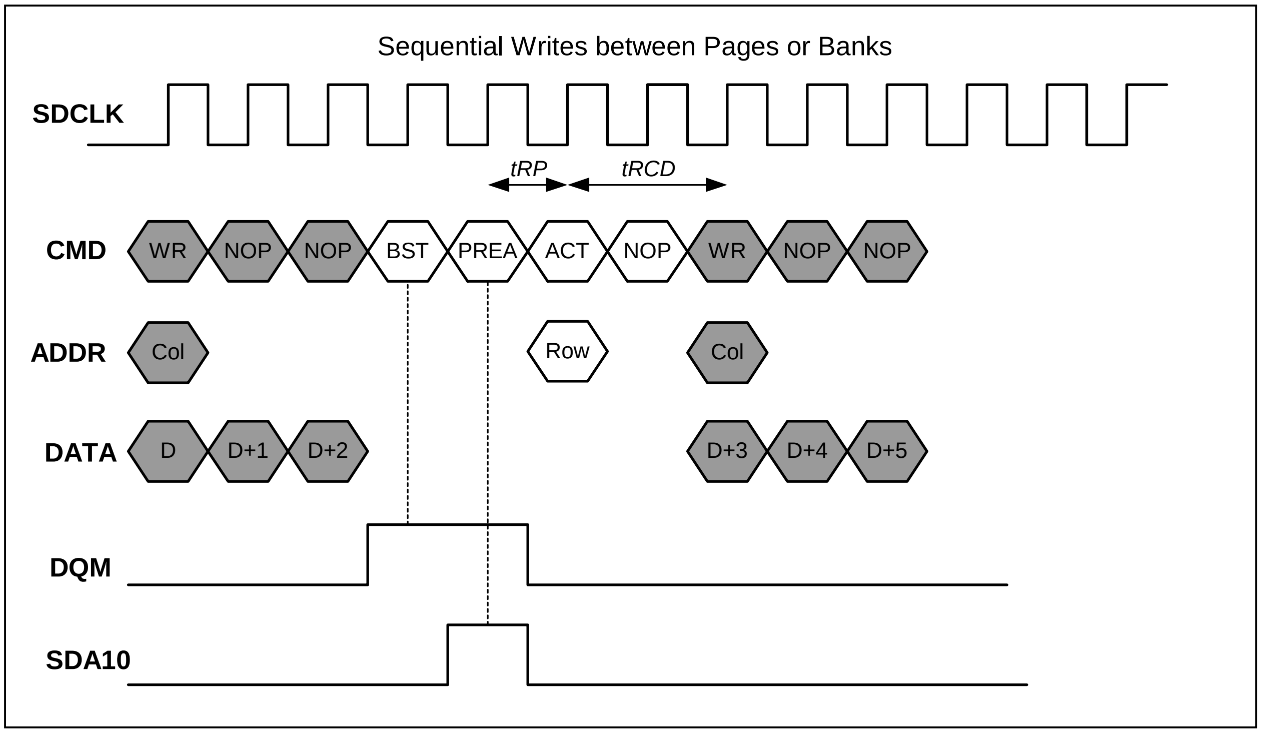


Nr. Cycles	*Core*	*Controller*	*Data*
1	**r1=dm(dest_Q);**	ACT	
2	int. ACK	NOP	
3	int. ACK	RD	
4	**r2=dm(dest_Q+1);**	NOP	
5	**r2=dm(dest_Q+2);**	NOP	Q
6	int. ACK	NOP	Q+1
7	int. ACK	NOP	Q+2
8	int. ACK	NOP	
9	int. ACK	BST	
10	int. ACK	PREA	
11	**r2=dm(dest_Q+3);**	ACT	
12	int. ACK	NOP	
13	int. ACK	RD	

14	**r2=dm(dest_Q+4);**	NOP	

14.10 – Writes between Page/Bank

a) Core writes to SDRAM, interruption caused by another page or bank.
b) IOP writes (EMx=1) to SDRAM, interruption caused by another page or bank.

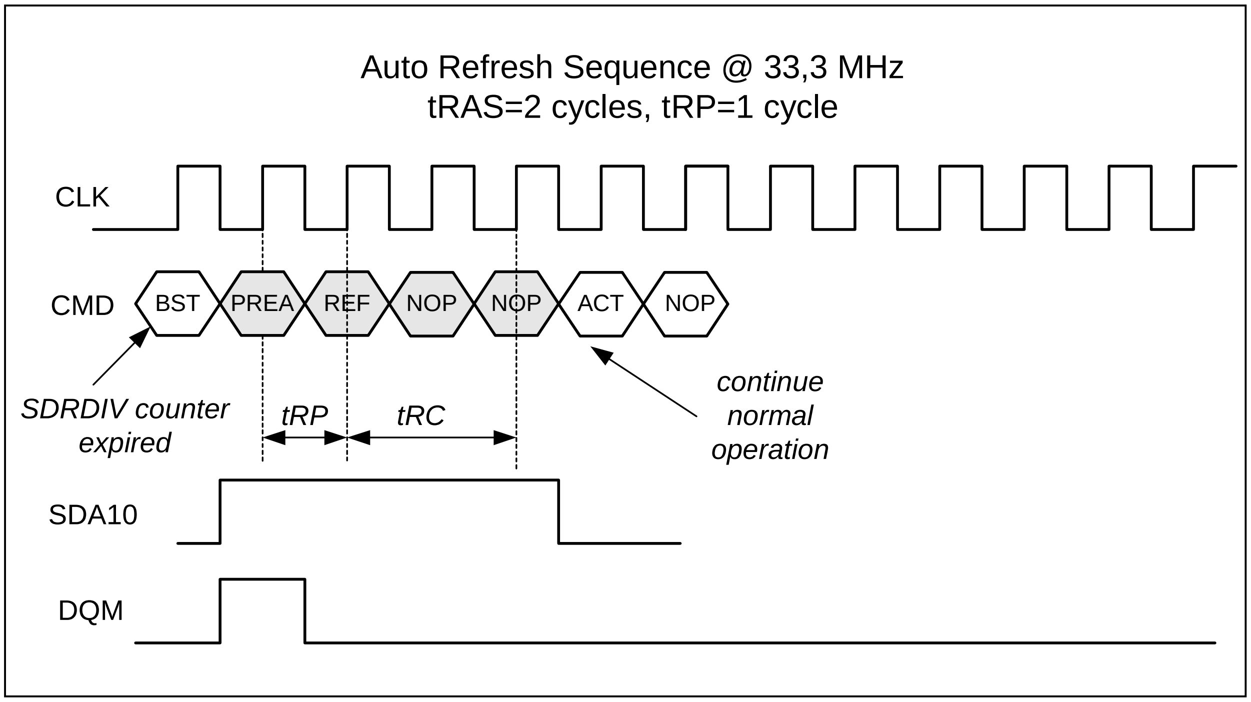


Nr. Cycles	*Core*	*Controller*	*Data*
1	**dm(dest_D)=r1;**	ACT	
2	int. ACK	NOP	
3	int. ACK	WR	D
4	**dm(dest_D+1)=r1;**	NOP	D+1
5	**dm(dest_D+2)=r1;**	NOP	D+2
6	int. ACK	BST	
7	int. ACK	PREA	
8	**dm(dest_D+3)=r1;**	ACT	
9	int. ACK;	NOP	
10	int. ACK;	WR	D+3
11	**dm(dest_D+4)=r1;**	NOP	D+4

12	**dm(dest_D+5)=r1;**	NOP	D+5

14.11 – Auto Refresh

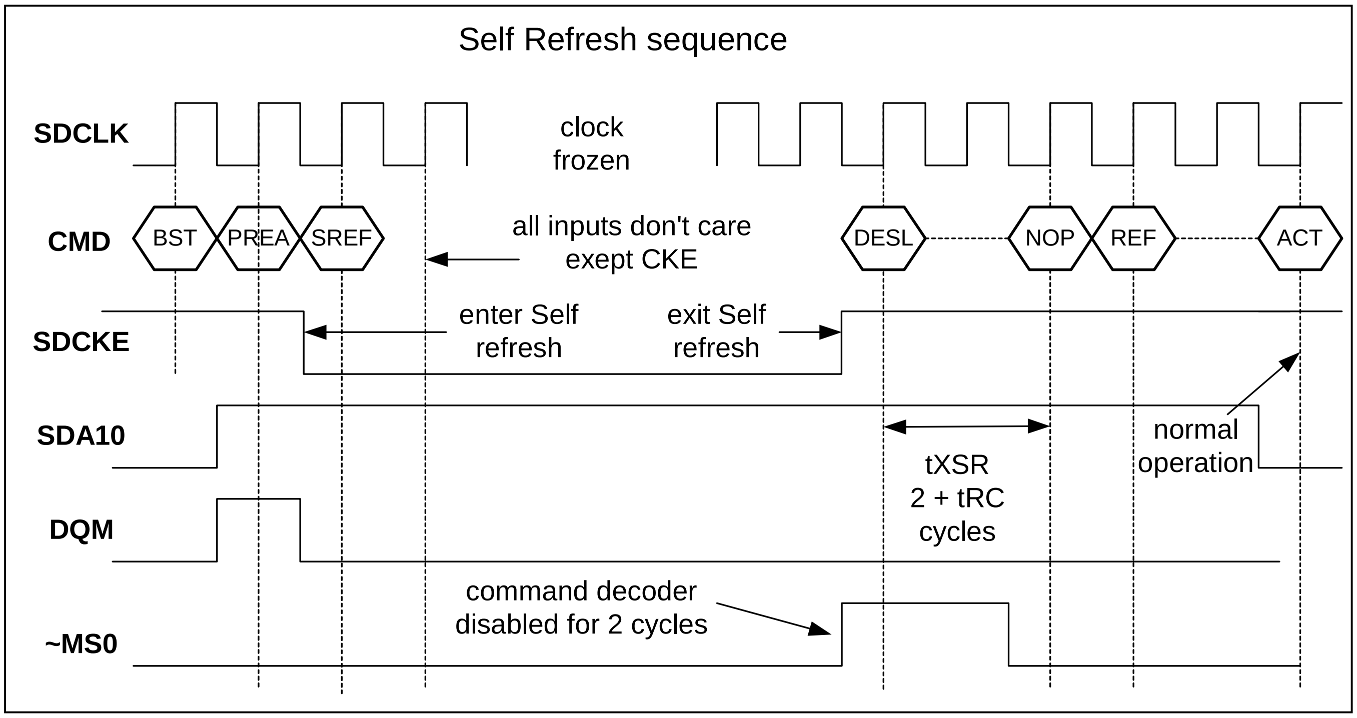
The refresh counter triggers refresh requests. The figure shows the refresh sequence: In the first place, all banks are precharged with SDA10 high and DQM high to avoid data conflict during precharge. The issued refresh follows right away. During refresh, other commands cannot be executed. The ratio between application time and refresh time is given by *tRC*=15,625µs/row:
e.g: 5 cycles/520 cycles x 100 = 0,96
This means, the refresh sequence requires 0,96 % of the whole performance at 33,3 MHz. By reducing the SDRDIV period, you can decrease the performance ratio to your needs.


Auto Refresh Sequence @ 33,3 MHz
tRAS=2 cycles, tRP=1 cycle
CLK
CMD
BST
PREA
REF
NOP
NOP
ACT
NOP
SDRDIV counter expired
tRP
tRC
continue normal operation
SDA10
DQM

Note: The auto refresh with 15,625 µs/row gives the best performance.
Note: An REF command starts an internal row refresh with CAS before RAS.

14.12 – Self Refresh

Next Figure explains this sequence:



Note: Only the CKE pin keeps control of the device in self refresh mode.

When the application sets the SDSRF bit in the IOCTL register, the controller will issue a BST, stopping the current action. In order to provide current information, all banks are precharged. During the PREA and the SREF command, the CKE transits from high to low thus enabling the self refresh mode. After entering, all the command inputs will get don't care status after one 1 cycle in order to reduce power consumption. Moreover the clock is frozen to reduce power consumption as well.
A SDRAM access from the SHARC starts the controller which ends the self refresh function. The CKE line transits high and the controller disables the command decoder for *tXSR* = 2+tRC cycles to restart the refresh time base. After *tRCmin*, the device executes an auto refresh and starts normal operation with the activate command after another *tRCmin*. The self refresh mode can be left after a certain period of time, which is calculated by the following equation:

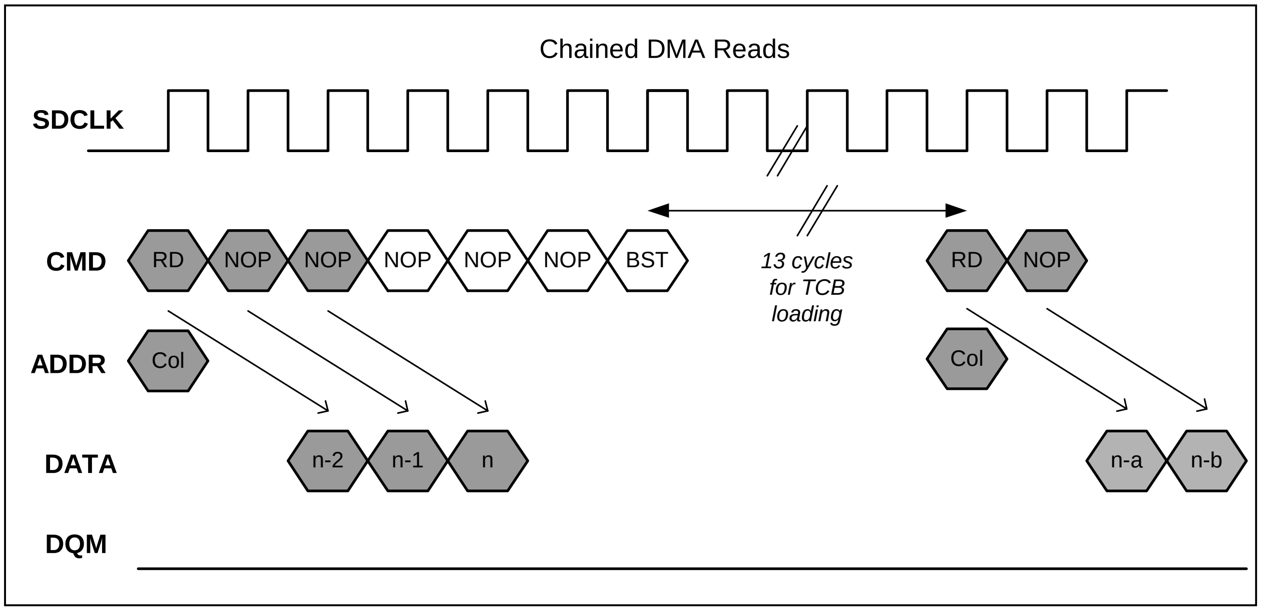
tExit = 2*tRC + 2 (cycles)

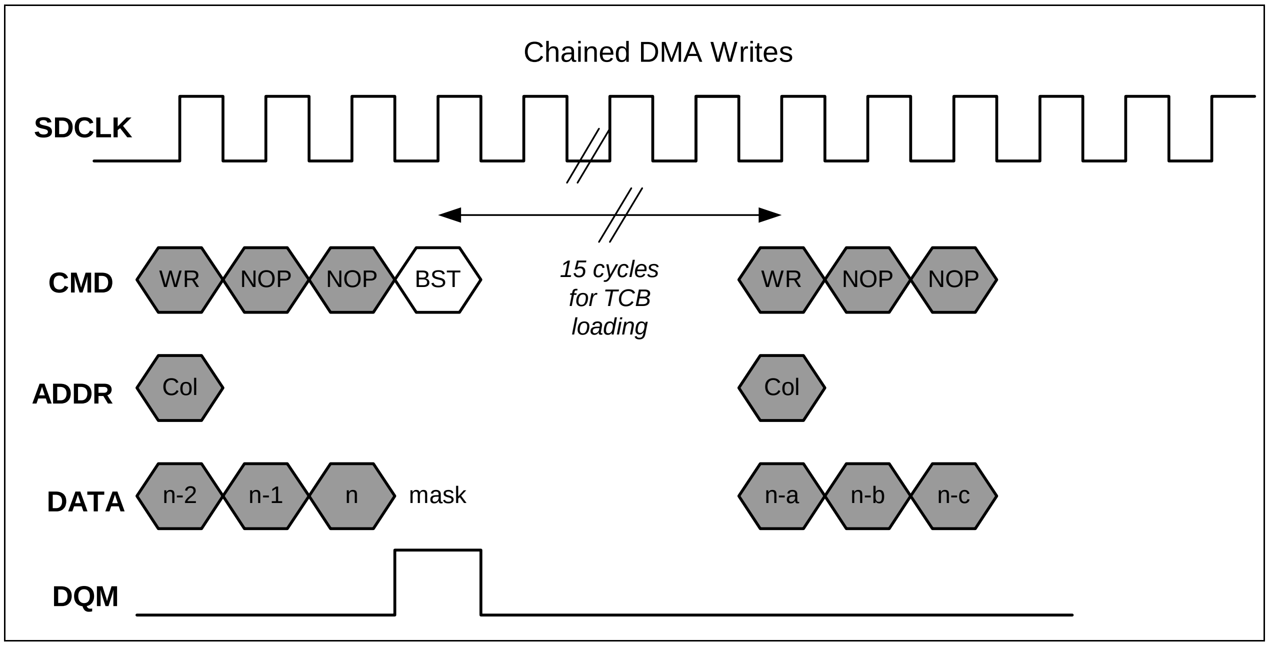
Note: Only the CKE pin keeps control of the device in self refresh mode.

14.13 – Chained DMA Transfers

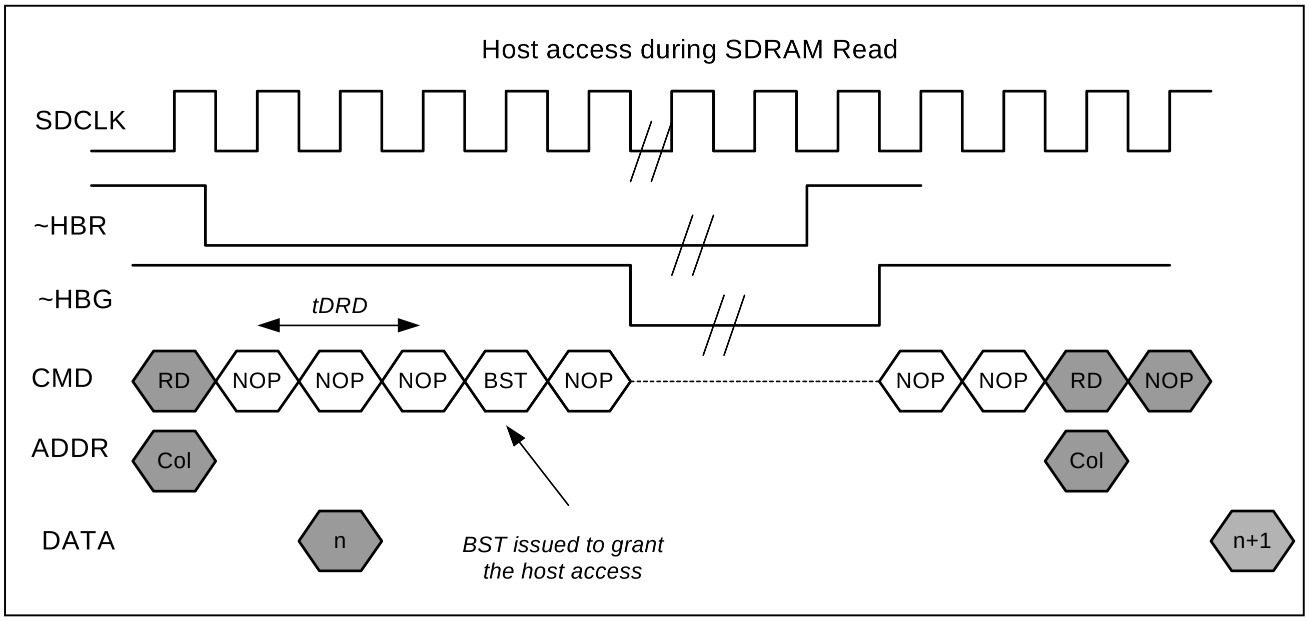
Loading the transfer control block (TCB) chain in between successive Master Mode DMA sequences. IOP reads/writes (EMx=1) from/to SDRAM, no interruption caused by a higher priority request to IOP.

Read from SDRAM:	13 SDCLK cycles
Write to SDRAM:	15 SDCLK cycles





14.14 – Host Access during Reads



A random host access during a read burst. After the detection of a request, the current burst operation of the SDRAM controller is interrupted and frozen. After deassertion of ~HBG, the controller continues bursting of the next data.
Note: The current bus master in a cluster will drive the SDRAM interface (refresh) during ~HBG low. The SDRAM bus master chip changes only with ~BRx in MMS.

References

- ADSP-21065L SHARC User's Manual, Analog Devices Inc.
- ADSP-21065L SHARC Technical Reference, Analog Devices Inc.
- ADSP-21065L Data Sheet, Analog Devices Inc.