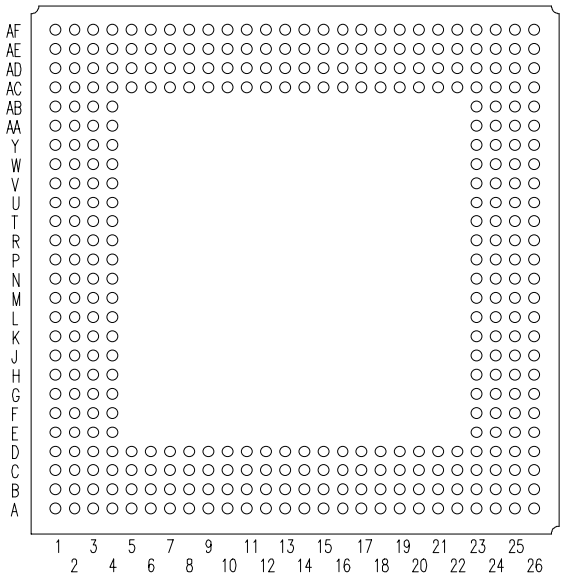
SPRS048 — APRIL 1998

- **Single Chip Parallel MIMD DSP**
- **Over 1.5 Billion RISC-like Operations per Second**
- **Master Processor (MP)**
  - **32-Bit RISC Processor**
  - **IEEE-754 Floating Point**
  - **4K-Byte Instruction Cache**
  - **4K-Byte Data Cache**
- **Two Parallel Processors (PPs)**
  - **32-Bit Advanced DSP Processors**
  - **64-Bit Opcode Provides Many Parallel Operations per Cycle**
  - **4K-Byte Instruction Cache, 4K-Byte Parameter RAM, and 8K Bytes of Data RAM per PP**
- **Transfer Controller (TC)**
  - **64-Bit Data Transfers**
  - **Up to 480M-Byte/s Transfer Rate**
  - **32-Bit Addressing**
  - **Direct EDO DRAM/VRAM Interface**
  - **Direct SDRAM Interface**
  - **Dynamic Bus Sizing**
  - **Intelligent Queuing and Cycle Prioritization**

GGP PACKAGE
(BOTTOM VIEW)



- **Big or Little Endian Operation**
- **44K Bytes of On-Chip RAM**
- **4G-Byte Address Space**
- **16.6 ns Cycle Time**
- **3.3-V Operation**
- **IEEE 1149.1 Test Port (JTAG)**

description

The TMS320C82 is a single chip, MIMD (multiple instruction/multiple data) parallel processor capable of performing over 1.5 billion operations per second. It consists of a 32-bit RISC Master Processor with a 120-MFlop IEEE Floating Point Unit, two 32-bit parallel-processing DSPs (PPs), and a Transfer Controller with up to 480 Mbyte/sec transfer rate. All the processors are tightly coupled via an on-chip crossbar which provides shared access to on-chip RAM. This performance and programmability make the ‘C82 ideally suited for video, imaging, and high-speed telecommunication applications.

architecture

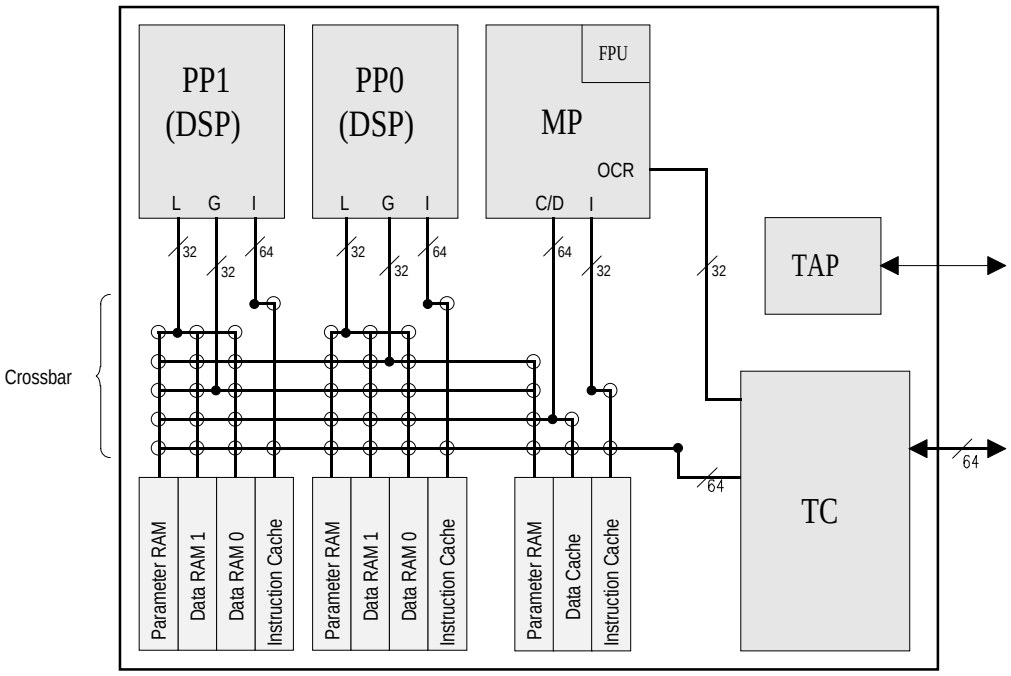


Figure 1. ‘C82 Block Diagram Showing Datapaths

The ‘C82 Block Diagram shows the major components of the ‘C82: the Master Processor (MP), Parallel Processors (PPs), Transfer Controller (TC), and JTAG Emulation Interface. Shared access to on-chip RAMs is achieved through the Crossbar. Crossbar connections are represented by ○. Each PP can perform three accesses per cycle through its Local, Global, and Instruction ports. The MP can access two RAMs per cycle through its Crossbar/Data and Instruction ports and the TC can access one RAM through its crossbar interface. Thus, up to nine simultaneous accesses are supported in each cycle. Addresses can be changed every cycle, allowing the crossbar matrix to be changed on a cycle-by-cycle basis. Contention between processors for the same RAM in the same cycle is resolved by a round-robin priority scheme. In addition to the Crossbar, a 32-bit data path exists between the MP and the TC. This allows the MP to access TC control registers which are memory-mapped into the MP's memory space.

pin assignments – numerical listing

PIN NO.	FUNCTION	PIN NO.	FUNCTION	PIN NO.	FUNCTION	PIN NO.	FUNCTION
A1	NC	B1	NC	C1	NC	D1	NC
A2	NC	B2	NC	C2	/DDIN	D2	/TRG/CAS
A3	NC	B3	/DBEN	C3	NC	D3	V_{SS}
A4	NC	B4	V_{DD}	C4	V_{DD}	D4	NC
A5	NC	B5	STATUS0	C5	V_{SS}	D5	STATUS1
A6	NC	B6	/CAS/DQM6	C6	V_{DD}	D6	/CAS/DQM7
A7	NC	B7	V_{SS}	C7	/CAS/DQM5	D7	V_{SS}
A8	NC	B8	V_{DD}	C8	/CAS/DQM3	D8	/CAS/DQM4
A9	NC	B9	/CAS/DQM1	C9	/CAS/DQM2	D9	V_{DD}
A10	NC	B10	V_{SS}	C10	/CAS/DQM0	D10	V_{SS}
A11	NC	B11	/HACK	C11	V_{DD}	D11	REQ
A12	NC	B12	CLKOUT	C12	V_{SS}	D12	V_{SS}
A13	NC	B13	LF	C13	V_{DD}	D13	V_{DDPLL}
A14	NC	B14	CLKIN	C14	V_{SSPLL}	D14	V_{SS}
A15	NC	B15	V_{DD}	C15	/HREQ	D15	/RESET
A16	NC	B16	/XPT3	C16	/XPT2	D16	/XPT1
A17	NC	B17	V_{SS}	C17	/XPT0	D17	/LINT4
A18	NC	B18	V_{DD}	C18	/EINT3	D18	/EINT2
A19	NC	B19	/EINT1	C19	TCK	D19	TMS
A20	NC	B20	/TRST	C20	TDI	D20	V_{SS}
A21	NC	B21	EMU1	C21	EMU0	D21	V_{DD}
A22	NC	B22	V_{DD}	C22	TDO	D22	V_{SS}
A23	NC	B23	AD31	C23	AD30	D23	NC
A24	NC	B24	AD29	C24	NC	D24	AD28
A25	NC	B25	NC	C25	V_{DD}	D25	V_{SS}
A26	NC	B26	NC	C26	NC	D26	NC

pin assignments – numerical listing (continued)

PIN NO.	FUNCTION	PIN NO.	FUNCTION	PIN NO.	FUNCTION	PIN NO.	FUNCTION
E1	NC	J23	V_{DD}	P1	NC	V23	V_{SS}
E2	V_{SS}	J24	AD20	P2	V_{DD}	V24	AD7
E3	/W	J25	AD19	P3	RCA5	V25	V_{SS}
E4	V_{SS}	J26	NC	P4	V_{DD}	V26	NC
E23	AD27	K1	NC	P23	AD13	W1	NC
E24	V_{SS}	K2	V_{DD}	P24	V_{DD}	W2	RCA11
E25	AD26	K3	RCA1	P25	AD14	W3	V_{SS}
E26	NC	K4	V_{SS}	P26	NC	W4	V_{SS}
F1	NC	K23	V_{SS}	R1	NC	W23	V_{DD}
F2	/RL	K24	V_{SS}	R2	RCA6	W24	AD6
F3	V_{DD}	K25	AD18	R3	V_{SS}	W25	V_{SS}
F4	DSF	K26	NC	R4	RCA7	W26	NC
F23	V_{DD}	L1	NC	R23	V_{SS}	Y1	NC
F24	AD25	L2	V_{DD}	R24	AD11	Y2	RCA12
F25	V_{DD}	L3	RCA2	R25	AD12	Y3	V_{DD}
F26	NC	L4	V_{DD}	R26	NC	Y4	RCA13
G1	NC	L23	V_{SS}	T1	NC	Y23	V_{DD}
G2	/EXCEPT0	L24	AD17	T2	V_{SS}	Y24	AD5
G3	/RAS	L25	V_{DD}	T3	V_{SS}	Y25	V_{DD}
G4	V_{DD}	L26	NC	T4	RCA8	Y26	NC
G23	AD24	M1	NC	T23	AD9	AA1	NC
G24	AD23	M2	V_{SS}	T24	AD10	AA2	V_{DD}
G25	V_{SS}	M3	RCA3	T25	V_{SS}	AA3	V_{DD}
G26	NC	M4	V_{DD}	T26	NC	AA4	V_{DD}
H1	NC	M23	V_{DD}	U1	NC	AA23	V_{SS}
H2	READY	M24	AD16	U2	V_{DD}	AA24	AD3
H3	V_{DD}	M25	V_{DD}	U3	RCA9	AA25	AD4
H4	/EXCEPT1	M26	NC	U4	V_{DD}	AA26	NC
H23	AD22	N1	NC	U23	AD8	AB1	NC
H24	AD21	N2	V_{SS}	U24	V_{DD}	AB2	RCA14
H25	V_{DD}	N3	RCA4	U25	V_{DD}	AB3	V_{SS}
H26	NC	N4	V_{SS}	U26	NC	AB4	RCA15
J1	NC	N23	V_{SS}	V1	NC	AB23	V_{DD}
J2	RCA0	N24	V_{SS}	V2	V_{DD}	AB24	AD2
J3	V_{SS}	N25	AD15	V3	RCA10	AB25	V_{SS}
J4	V_{SS}	N26	NC	V4	V_{SS}	AB26	NC

pin assignments – numerical listing (continued)

PIN NO.	FUNCTION	PIN NO.	FUNCTION	PIN NO.	FUNCTION	PIN NO.	FUNCTION
AC1	NC	AD1	NC	AE1	NC	AF1	NC
AC2	V_{SS}	AD2	RCA16	AE2	NC	AF2	NC
AC3	V_{SS}	AD3	NC	AE3	AD32	AF3	NC
AC4	NC	AD4	V_{DD}	AE4	AD33	AF4	NC
AC5	AD34	AD5	V_{SS}	AE5	AD35	AF5	NC
AC6	AD36	AD6	V_{DD}	AE6	AD37	AF6	NC
AC7	AD38	AD7	V_{SS}	AE7	AD39	AF7	NC
AC8	V_{DD}	AD8	AD40	AE8	V_{DD}	AF8	NC
AC9	V_{DD}	AD9	AD41	AE9	V_{SS}	AF9	NC
AC10	AD42	AD10	V_{SS}	AE10	V_{SS}	AF10	NC
AC11	AD43	AD11	AD44	AE11	V_{DD}	AF11	NC
AC12	AD45	AD12	AD46	AE12	V_{SS}	AF12	NC
AC13	AD47	AD13	AD48	AE13	V_{SS}	AF13	NC
AC14	AD49	AD14	V_{DD}	AE14	V_{DD}	AF14	NC
AC15	V_{SS}	AD15	AD50	AE15	V_{DD}	AF15	NC
AC16	AD52	AD16	V_{SS}	AE16	AD51	AF16	NC
AC17	AD54	AD17	AD53	AE17	V_{DD}	AF17	NC
AC18	AD56	AD18	AD55	AE18	V_{SS}	AF18	NC
AC19	AD57	AD19	V_{DD}	AE19	V_{DD}	AF19	NC
AC20	AD59	AD20	V_{SS}	AE20	AD58	AF20	NC
AC21	V_{DD}	AD21	AD60	AE21	V_{DD}	AF21	NC
AC22	V_{SS}	AD22	V_{SS}	AE22	AD61	AF22	NC
AC23	NC	AD23	V_{SS}	AE23	AD62	AF23	NC
AC24	V_{DD}	AD24	NC	AE24	AD63	AF24	NC
AC25	AD1	AD25	AD0	AE25	NC	AF25	NC
AC26	NC	AD26	NC	AE26	NC	AF26	NC

pin assignments – alphabetical listing

FUNCTION	PIN NO.	FUNCTION	PIN NO.	FUNCTION	PIN NO.
AD0	AD25	AD40	AD8	EMU0	C21
AD1	AC25	AD41	AD9	EMU1	B21
AD2	AB24	AD42	AC10	/EXCEPT0	G2
AD3	AA24	AD43	AC11	/EXCEPT1	H4
AD4	AA25	AD44	AD11	/HACK	B11
AD5	Y24	AD45	AC12	/HREQ	C15
AD6	W24	AD46	AD12	/LINT4	D17
AD7	V24	AD47	AC13	LF	B13
AD8	U23	AD48	AD13	/RAS	G3
AD9	T23	AD49	AC14	RCA0	J2
AD10	T24	AD50	AD15	RCA1	K3
AD11	R24	AD51	AE16	RCA2	L3
AD12	R25	AD52	AC16	RCA3	M3
AD13	P23	AD53	AD17	RCA4	N3
AD14	P25	AD54	AC17	RCA5	P3
AD15	N25	AD55	AD18	RCA6	R2
AD16	M24	AD56	AC18	RCA7	R4
AD17	L24	AD57	AC19	RCA8	T4
AD18	K25	AD58	AE20	RCA9	U3
AD19	J25	AD59	AC20	RCA10	V3
AD20	J24	AD60	AD21	RCA11	W2
AD21	H24	AD61	AE22	RCA12	Y2
AD22	H23	AD62	AE23	RCA13	Y4
AD23	G24	AD63	AE24	RCA14	AB2
AD24	G23	/CAS/DQM0	C10	RCA15	AB4
AD25	F24	/CAS/DQM1	B9	RCA16	AD2
AD26	E25	/CAS/DQM2	C9	READY	H2
AD27	E23	/CAS/DQM3	C8	REQ	D11
AD28	D24	/CAS/DQM4	D8	/RESET	D15
AD29	B24	/CAS/DQM5	C7	/RL	F2
AD30	C23	/CAS/DQM6	B6	STATUS0	B5
AD31	B23	/CAS/DQM7	D6	STATUS1	D5
AD32	AE3	CLKIN	B14	TCK	C19
AD33	AE4	CLKOUT	B12	TDI	C20
AD34	AC5	/DBEN	B3	TDO	C22
AD35	AE5	/DDIN	C2	TMS	D19
AD36	AC6	DSF	F4	/TRG/CAS	D2
AD37	AE6	/EINT1	B19	/TRST	B20
AD38	AC7	/EINT2	D18		
AD39	AE7	/EINT3	C18		

pin assignments – alphabetical listing (continued)

FUNCTION	PIN NO.	FUNCTION	PIN NO.	FUNCTION	PIN NO.	FUNCTION	PIN NO.
V_{DD}	B4	V_{DD}	U24	V_{SS}	C12	V_{SS}	V25
V_{DD}	B8	V_{DD}	U25	V_{SS}	D3	V_{SS}	W3
V_{DD}	B15	V_{DD}	V2	V_{SS}	D7	V_{SS}	W4
V_{DD}	B18	V_{DD}	W23	V_{SS}	D10	V_{SS}	W25
V_{DD}	B22	V_{DD}	Y3	V_{SS}	D12	V_{SS}	AA23
V_{DD}	C4	V_{DD}	Y23	V_{SS}	D14	V_{SS}	AB3
V_{DD}	C6	V_{DD}	Y25	V_{SS}	D20	V_{SS}	AB25
V_{DD}	C11	V_{DD}	AA2	V_{SS}	D22	V_{SS}	AC2
V_{DD}	C13	V_{DD}	AA3	V_{SS}	D25	V_{SS}	AC3
V_{DD}	C25	V_{DD}	AA4	V_{SS}	E2	V_{SS}	AC15
V_{DD}	D9	V_{DD}	AB23	V_{SS}	E4	V_{SS}	AC22
V_{DD}	D21	V_{DD}	AC8	V_{SS}	E24	V_{SS}	AD5
V_{DD}	F3	V_{DD}	AC9	V_{SS}	G25	V_{SS}	AD7
V_{DD}	F23	V_{DD}	AC21	V_{SS}	J3	V_{SS}	AD10
V_{DD}	F25	V_{DD}	AC24	V_{SS}	J4	V_{SS}	AD16
V_{DD}	G4	V_{DD}	AD4	V_{SS}	K4	V_{SS}	AD20
V_{DD}	H3	V_{DD}	AD6	V_{SS}	K23	V_{SS}	AD22
V_{DD}	H25	V_{DD}	AD14	V_{SS}	K24	V_{SS}	AD23
V_{DD}	J23	V_{DD}	AD19	V_{SS}	L23	V_{SS}	AE9
V_{DD}	K2	V_{DD}	AE8	V_{SS}	M2	V_{SS}	AE10
V_{DD}	L2	V_{DD}	AE11	V_{SS}	N2	V_{SS}	AE12
V_{DD}	L4	V_{DD}	AE14	V_{SS}	N4	V_{SS}	AE13
V_{DD}	L25	V_{DD}	AE15	V_{SS}	N23	V_{SS}	AE18
V_{DD}	M4	V_{DD}	AE17	V_{SS}	N24	V_{SSPLL}	C14
V_{DD}	M23	V_{DD}	AE19	V_{SS}	R3	/W	E3
V_{DD}	M25	V_{DD}	AE21	V_{SS}	R23	/XPT0	C17
V_{DD}	P2	V_{DDPLL}	D13	V_{SS}	T2	/XPT1	D16
V_{DD}	P4	V_{SS}	B7	V_{SS}	T3	/XPT2	C16
V_{DD}	P24	V_{SS}	B10	V_{SS}	T25	/XPT3	B16
V_{DD}	U2	V_{SS}	B17	V_{SS}	V4		
V_{DD}	U4	V_{SS}	C5	V_{SS}	V23		

signal descriptions

NAME	I/O	DESCRIPTION
LOCAL MEMORY INTERFACE		
AD63-AD40	I/O	**Data.** The upper 24 bits of data are read in/driven out over this bus.
AD39-AD32	I/O	**Data/Status.** This bus drives out the access code at row time. During column time, bits 39-32 of data are read in/driven out over this bus.
AD31-AD0	I/O	**Address/Data.** Outputs the 32-bit address at row time. During column time, the lower 32 bits of data are read in/driven out on this bus.
RCA16-RCA0	O	**Address.** Outputs the multiplexed row/column addresses.
/DBEN	O	**Data buffer enable.** This signal drives the active-low output enables on bidirectional transceivers which may be used to buffer input and output data on AD63-AD0.
/DDIN	O	**Data direction indicator.** Indicates the direction that data needs to pass through the transceivers. A low on this signal indicates a transfer from external memory into the ‘C82.
/EXCEPT1-/EXCEPT0	I	**Memory exception.** Two-bit encodings on these inputs request retries, faults, or configuration cache flushes. Additionally, these inputs are used to indicate the cycle type for refreshes.
READY	I	**Ready.** Indicates that the external device is ready to complete the memory cycle. This signal is driven inactive low by external circuitry to insert wait states into a memory cycle. READY is also used at reset to determine the endianness of the ‘C82. If READY is low at the rising edge of /RESET, the ‘C82 will operate in big-endian mode. If READY is high, the ‘C82 will operate in little-endian mode.
/RL	O	**Row latch.** The high-to-low transition of /RL can be used to latch the source information, status code, and 32-bit byte address present on AD39-AD0 at row time.
STATUS1-STATUS0	O	**Status.** Two-bit encoded outputs which indicate row, column, XPT end and idle conditions on the bus.
DRAM, VRAM, AND SDRAM CONTROL		
/CAS/DQM7-/CAS/DQM0	O	**Column address strobes.** These outputs drive the /CAS inputs of DRAMs and VRAMs or the DQM inputs of SDRAMs. The eight strobes provide byte write access to memory.
DSF	O	**Special function.** This signal is used to select special VRAM functions such as block write, load color register, and split register transfers, and SGRAM block writes.
/RAS	O	**Row address strobe.** The /RAS output drives the /RAS inputs of DRAMs, VRAMs, and SDRAMs.
/TRG/CAS	O	**Transfer/output enable or column address strobe.** /TRG/CAS is used as an output enable for DRAMs and VRAMs and as a transfer enable for VRAMs. /TRG/CAS also drives the /CAS inputs of SDRAMs.
/W	O	**Write enable.** /W is driven active-low prior to /CAS during DRAM/VRAM write cycles. During VRAM transfer cycles, /W is used to control the direction of the transfer. For SDRAM writes, /W is driven low concurrent with the DQM signals, and is also low during DCAB cycles.

signal descriptions (continued)

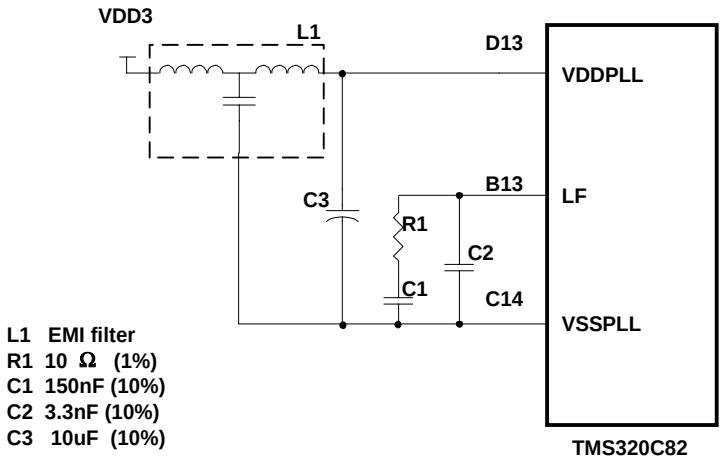
NAME	I/O	DESCRIPTION
		HOST INTERFACE
/HACK	O	**Host acknowledge.** The ‘C82 will drive this output low following an active /HREQ, to indicate that it has driven the local memory bus signals to high impedance and is relinquishing the bus. /HACK is driven high asynchronously following /HREQ being detected inactive, and the ‘C82 will resume driving the bus.
/HREQ	I	**Host request.** An external device drives this input active-low to request ownership of the local memory bus. When /HREQ is inactive high, the ‘C82 will own and drive the bus. /HREQ is internally synchronized to the ‘C82's internal clock. /HREQ is used at reset to determine the power-up state of the MP. If /HREQ is low at the rising edge of /RESET, the MP will come up running. If /HREQ is high, the MP will remain halted until the first interrupt occurrence on /EINT3.
REQ	O	**Internal cycle request.** This signal provides an indication that the ‘C82 is receiving a high-priority request (urgent refresh or XPT request). External logic can monitor this signal to determine if it is necessary to relinquish the local memory bus to the ‘C82.
		SYSTEM CONTROL
CLKIN	I	**Input clock.** This clock is used to generate the internal ‘C82 clocks to which all processor functions are synchronous.
LF	I	**Loop filter.** An external filter is connected to this pin to provide filtering for the C82’s on-chip PLL circuitry.
CLKOUT	O	**Local output clock.** This clock provides a way to synchronize external circuitry to internal timings. This clock is internally phase-locked to CLKIN.
/EINT1, /EINT2, /EINT3	I	**Edge-triggered interrupts.** These signals allow external devices to interrupt the MP on one of three interrupt levels (/EINT1 being the highest priority). The interrupts are rising-edge triggered. /EINT3 also serves as an unhalt signal. If the MP is powered-up halted, the first rising edge on /EINT3 will cause the MP to unhalt and fetch its reset vector. (The /EINT3 interrupt pending bit will not be set in this case.)
/LINT4	I	**Level-triggered interrupt.** This input provides an active-low level-triggered interrupt to the MP. Its priority falls below that of the edge-triggered interrupts. Any interrupt request should remain active-low until it is recognized by the ‘C82. The /LINT4 interrupt service routine is expected to clear the interrupt condition.
/RESET	I	**Reset.** The /RESET input is driven low to reset the ‘C82 (all processors). During reset, all internal registers are set to their initial state and all output pins are driven to high-impedance levels with the exception of CLKOUT, /HACK, and REQ, which continue to be driven. During the rising edge of /RESET, the MP reset mode and the ‘C82's operating endian are determined by the levels of /HREQ and READY pins, respectively.
/XPT3-/XPT0	I	**External Packet Transfer.** These encoded inputs are used by external devices to request a high-priority external packet transfer (XPT) by the TC. Fifteen XPT codes are supported. Code 1111 indicates that no request is being submitted. The XPT inputs should remain valid until the TC begins servicing the request.
		EMULATION CONTROL
EMU0, EMU1 †	I/O	**Emulation pins.** These two pins are used to support emulation host interrupts, special functions targeted at a single processor, and multiprocessor halt event communications.
TCK †	I	**Test clock.** This input provides the clock for the ‘C82's JTAG logic allowing it to be compatible with other JTAG devices, controllers, and test equipment designed for different clock rates.
TDI †	I	**Test data input.** This pin provides input data for all JTAG instructions and data scans of the ‘C82.
TDO	O	**Test data output.** This pin provides output data for all JTAG instructions and data scans of the ‘C82.
TMS †	I	**Test mode select.** This signal controls the JTAG state machine.
/TRST ‡	I	**Test reset.** This input resets the ‘C82's JTAG module. When active-low, all boundary scan logic is disabled, allowing normal ‘C82 operation.

† This pin has an internal pullup and may be left unconnected during normal operation.
‡ This pin has an internal pulldown and may be left unconnected during normal operation.

signal descriptions (continued)

NAME	I/O	DESCRIPTION
		POWER
V_{DD} §		**Power.** Nominal 3.3-volt power supply inputs.
V_{DDPLL}		**Power.** Nominal 3.3-volt power supply input for the on-chip PLL.
V_{SS} §		**Ground.** Electrical ground inputs.
V_{SSPLL}		**Ground.** Electrical ground input for the on-chip PLL (tied to V_{SS} internally).

§ For proper operation, all V_{DD} and V_{SS} pins must be connected externally.



NOTE: To ensure proper operation, the on-chip PLL should be powered with a stable supply. To minimize noise injection into the PLL, it is suggested that an external EMI filter be applied as shown. The RC filter network on the LF pin provides an external filter for the PLL.

Figure 2. PLL Support Circuitry

memory map

The ‘C82 has a 4G-byte address space. The lower 32M bytes are used to address internal RAM and memory-mapped registers.

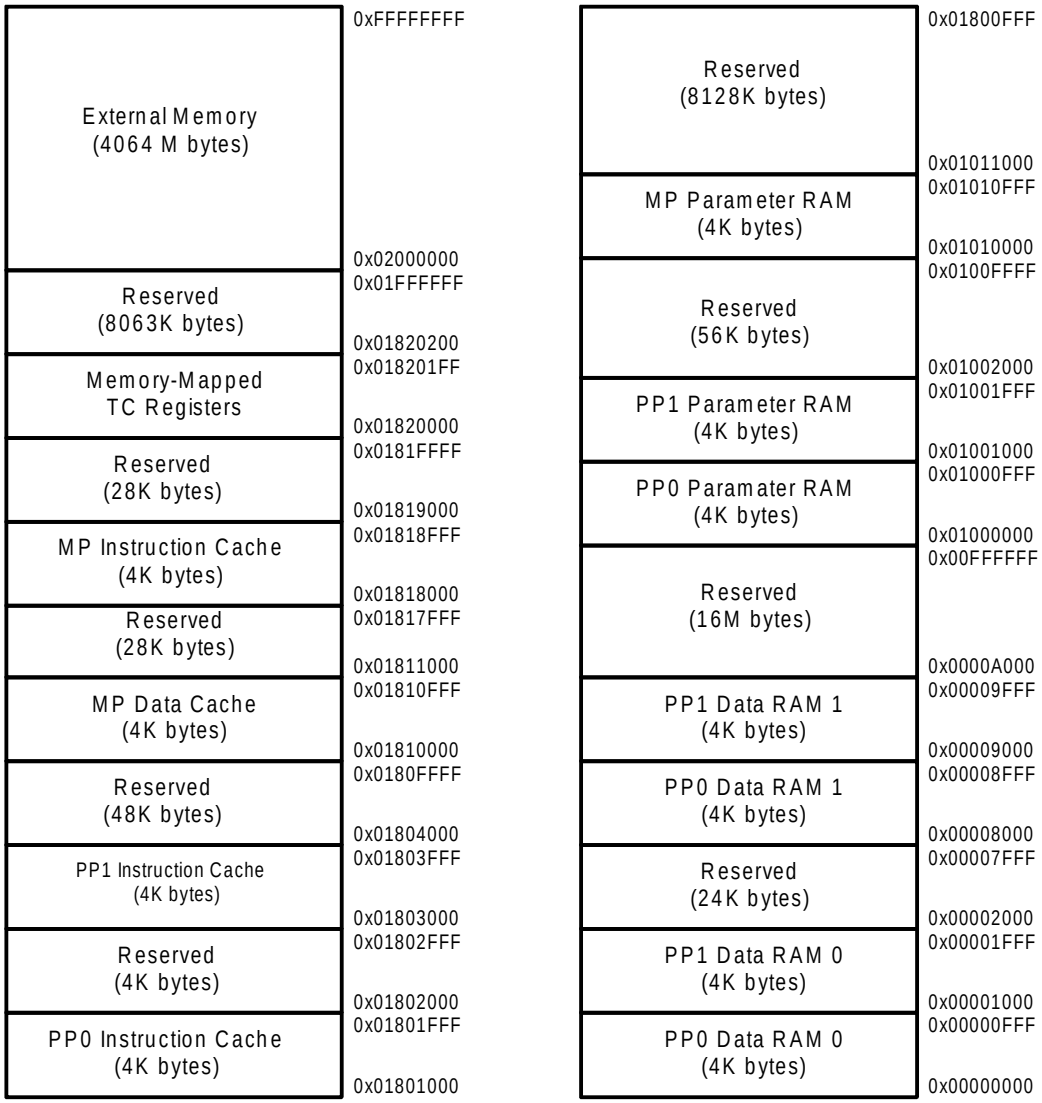


Figure 3. TMS320C82 Memory Map

master processor architecture

The Master Processor (MP) is a 32-bit RISC processor with an integral IEEE-754 floating-point unit. The MP has been designed for executing C code and is capable of performing at over 130k dhrystones. Major tasks which the MP will typically perform are:

- Task control and user interface
- Information processing and analysis
- IEEE-754 floating point (including graphics transforms)

functional block diagram

Figure 4 shows a block diagram of the master processor. Key features of the MP include:

- 32-bit RISC processor
  - Load/store architecture
  - 3 operand arithmetic and logical instructions
- 4K-byte instruction cache and 4K-byte data cache
  - 4-way set associative
  - LRU replacement
  - Data writeback
- 4K-byte non-cached parameter RAM
- Thirty-one 32-bit general-purpose registers
- Register and accumulator scoreboard
- 15-bit or 32-bit immediate constants
- 32-bit byte-addressing
- Scalable timer
- Leftmost-one and rightmost-one logic
- IEEE-754 floating-point hardware
  - Four double-precision floating-point vector accumulators
  - Vector floating-point instructions
    - Floating-point operation and parallel load or store
    - Multiply and accumulate
- High performance
  - 60 MIPS
  - 120 MFLOPS
  - Over 130,000 Dhrystones

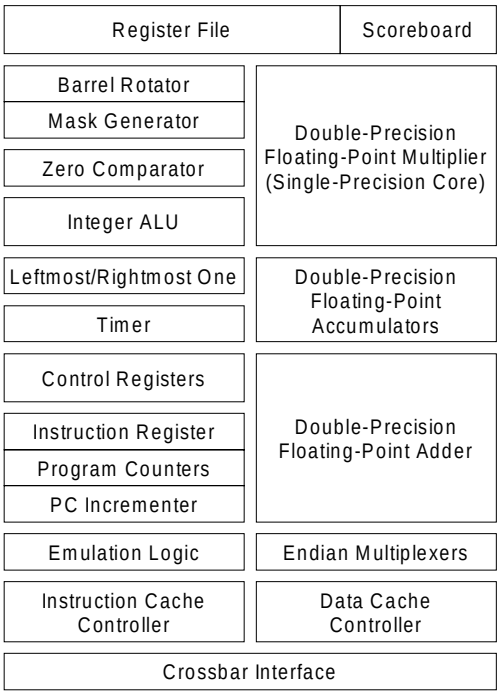


Figure 4. MP Block Diagram

general-purpose registers

The MP contains 31 32-bit general-purpose registers, R1 - R31. Register R0 always reads as zero and writes to it are discarded. Double-precision values are always stored in an even-odd register pair with the higher numbered register always holding the sign bit, and exponent. The R0/R1 pair is not available for this use. A scoreboard keeps track of which registers are awaiting loads or the result of a previous instruction and stalls the pipeline until the register contains valid data. As a recommended software convention, R1 is typically used as a stack pointer and R31 as a return address link register.

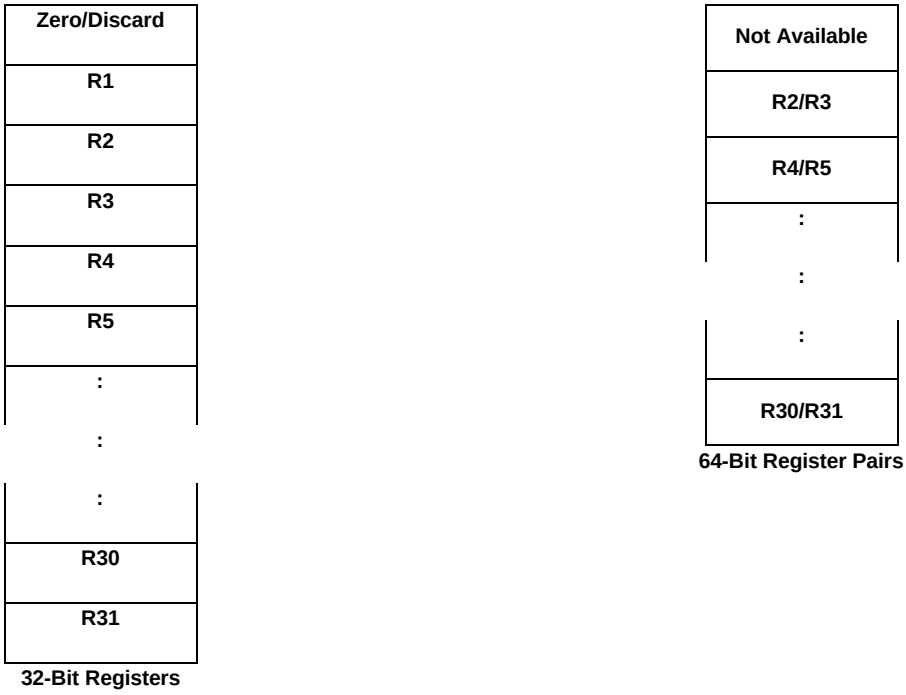


Figure 5. MP General-Purpose Registers

The 32-bit registers may contain signed-integer, unsigned-integer, or single-precision floating-point values. Signed and unsigned bytes and halfwords are sign-extended or zero-filled. Doublewords may be stored in a 64-bit even/odd register pair. Double-precision floating-point values are referenced using the even register number or the register pair. Figure 6 through Figure 8 show the register data formats.

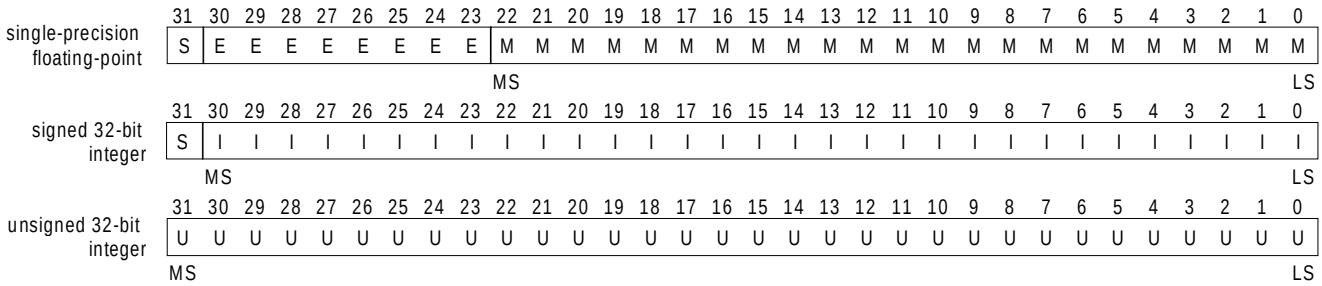


Figure 6. MP Register 32-Bit Data Formats

general-purpose registers (continued)

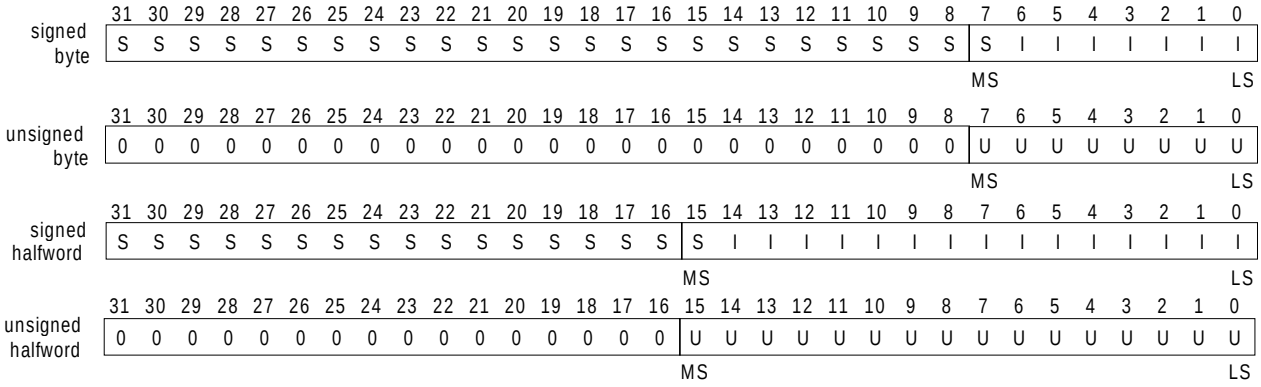


Figure 7. MP Register 8-Bit and 16-Bit Data

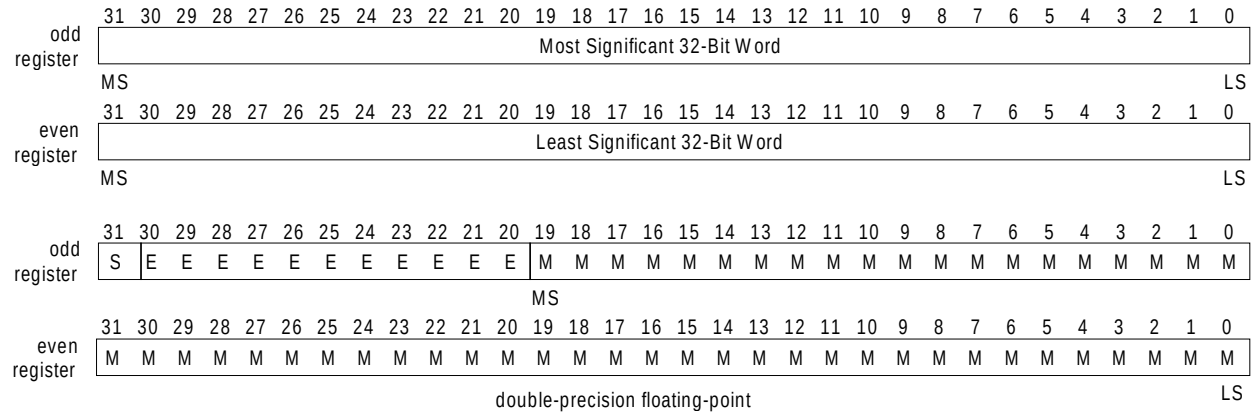


Figure 8. MP Register 64-Bit Data

double-precision floating-point accumulators

There are four double-precision floating-point registers to accumulate intermediate floating-point results.

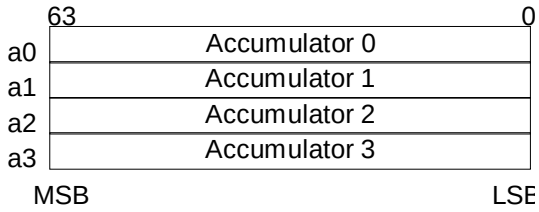


Figure 9. Floating-Point Accumulators

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77251-1443

control registers

In addition to the general-purpose registers, there are a number of control registers that are used to represent the state of the processor. The control register numbers of the accessible registers are shown in Table 1.

Table 1. MP Control Registers

NUMBER	NAME	DESCRIPTION
0x0000	EPC	Exception Program Counter
0x0001	EIP	Exception Instruction Pointer
0x0002	CONFIG	Configuration
0x0003		Reserved
0x0004	INTPEN	Interrupt Pending
0x0005		Reserved
0x0006	IE	Interrupt Enable
0x0007		Reserved
0x0008	FPST	Floating-Point Status
0x0009		Reserved
0x000A	PPERROR	PP Error Indicators
0x000B		Reserved
0x000C		Reserved
0x000D	PKTREQ	Packet Request Register
0x000E	TCOUNT	Current Counter Value
0x000F	TSCALE	Counter Reload Value
0x0010	FLTOP	Faulting Operation
0x0011	FLTADR	Faulting Address
0x0012	FLTTAG	Faulting Tag
0x0013	FLTDTL	Faulting Data (low)
0x0014	FLTDTH	Faulting Data (high)

NUMBER	NAME	DESCRIPTION
0x0015-0x001F		Reserved
0x0020	SYSSTK	System Stack Pointer
0x0021	SYSTMP	System Temporary Register
0x0022-0x002F		Reserved
0x0030	MPC	Emulator Exception Program Cntr
0x0031	MIP	Emulator Exception Instruction Ptr
0x0032		Reserved
0x0033	ECOMCNTL	Emulator Communication Control
0x0034	ANASTAT	Emulation Analysis Status Reg
0x0035-0x0038		Reserved
0x0039	BRK1	Emulation Breakpoint 1 Reg.
0x003A	BRK2	Emulation Breakpoint 2 Reg.
0x003B-0x01FF		Reserved
0x0200 - 0x020F	ITAG0-15	Instruction Cache Tags 0 to 15
0x0300	ILRU	Instruction Cache LRU Register
0x0400-0x040F	DTAG0-15	Data Cache Tags 0 to 15
0x0500	DLRU	Data Cache LRU Register
0x4000	IN0P	Vector Load Pointer 0
0x4001	IN1P	Vector Load Pointer 1
0x4002	OUTP	Vector Store Pointer

pipeline registers

The MP uses a three-stage Fetch, Execute, Access pipeline. The primary pipeline registers are manipulated implicitly by branch and trap instructions and are not accessible by the user. The exception and emulation pipeline registers are user-accessible as control registers. All pipeline registers are 32 bits.

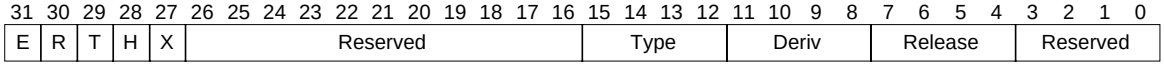
Table 2. MP FEA Pipeline Registers

	Program Execution Mode		
	Normal	Exception	Emulation
Program Counter	PC	EPC	MPC
Instruction Pointer	IP	EIP	MIP
Instruction Register	IR		

- Instruction Register (IR): contains the instruction being executed
- Instruction Pointer (IP): points to the instruction being executed
- Program Counter (PC): points to the instruction being fetched
- Exception/Emulator Instruction Pointer (EIP/MIP): points to the instruction that would have been executed had the exception / emulation trap not occurred.
- Exception/Emulator Program Counter (EPC/MPC): points to the instruction to be fetched on returning from the exception / emulation trap.

config register (0x0002)

The CONFIG register controls or reflects the state of certain options.



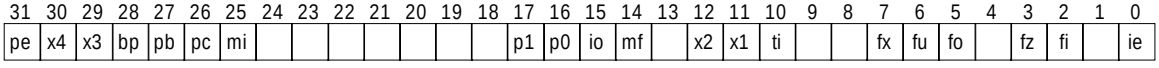
E - Endian Mode; 0 = big endian, 1 = little endian, Read only
R - PP Data RAM Round Robin; 0 = variable, 1 = fixed, Read/Write
T - TC PT Round Robin; 0 = variable, 1 = fixed, Read/Write
H - High-Priority MP Events; 0 = disabled, 1 = enabled , Read/Write
X - Externally Initiated Packet Transfers; 0 = disabled, 1 = enabled, Read/Write

Type - Number of PPs in device, Read only
Deriv - C8x family derivative, Read only (0x2)
Release - TMS320C82 version number, Read only

Figure 10. CONFIG Register

interrupt enable register (0x0006)

The IE register contains enable bits for each of the interrupts/traps. The global interrupt enable bit (ie) and the individual interrupt enable must be set in order for an interrupt to occur.



pe - PP error
x4 - external interrupt 4 ($\overline{\text{LINT4}}$)
x3 - external interrupt 3 ($\overline{\text{EINT3}}$)
bp - bad packet transfer
pb - packet transfer busy
pc - packet transfer complete
mi - message (MP self) interrupt

p1 - PP1 message interrupt
p0 - PP0 message interrupt
io - integer overflow
mf - memory fault
x2 - external interrupt 2 ($\overline{\text{EINT2}}$)
x1 - external interrupt 1 ($\overline{\text{EINT1}}$)
ti - MP timer interrupt

fx - floating point inexact
fu - floating point underflow
fo - floating point overflow
fz - floating point divide by zero
fi - floating point invalid
ie - global interrupt enable

Figure 11. IE Register

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77251-1443

interrupt pending register (0x0004)

The bits in INTPEN show the current state of each interrupt/trap. Pending interrupts will not occur unless the ie bit and corresponding interrupt enable bit are set (note: some memory faults are nonmaskable). Software must write a "1" to the appropriate INTPEN bit to clear an interrupt.

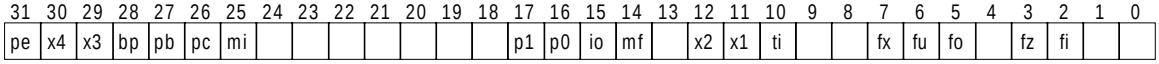
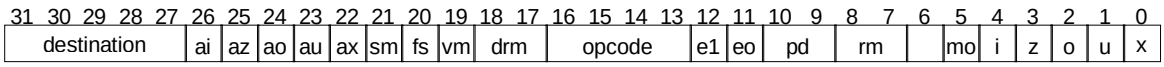


Figure 12. INTPEN Register

floating-point status register (0x0008)

FPST contains status and control information for the FPU. Bits 17-21 are read/write FPU control bits. Bits 22-26 are read/write accumulated status bits. All other bits show the status of the last FPU instruction to complete and are read-only bits.



dest - destination register
ai - accumulated value invalid
az - accumulated divide by zero
ao - accumulated overflow
au - accumulated underflow
ax - accumulated inexact
sm - sequential mode select
fs - floating point stall
vm - vector fast mode

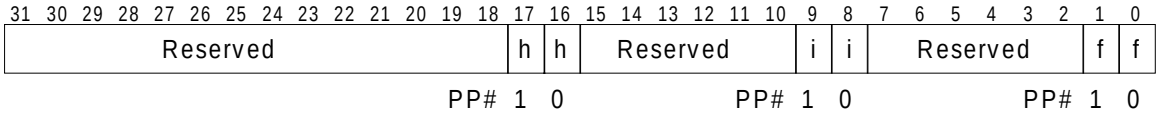
drm - rounding
00 - nearest 10 - positive ∞
01 - zero 11 -negative ∞
opcode - last opcode
e1 - 10th MSB of exponent
e0 - 9th MSB of exponent
pd - destination precision
00 - single float 10 - signed int
01 - double float 11 - unsigned int

rm - rounding
00 - nearest 10 positive ∞
01 - zero 11 -negative ∞
mo - int multiply overflow
i - invalid
z - divide by zero
o - overflow
u - underflow
x - inexact

Figure 13. FPST Register

PP error register (0x000A)

The bits in the PPERROR register reflect Parallel Processor errors. The MP may use these when a PP Error occurs to determine the cause of the error.

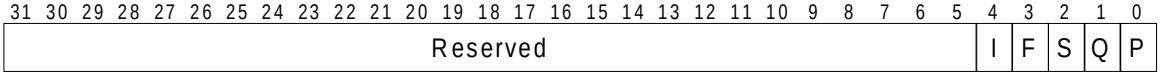


h - PP halted
i - PP illegal instruction
f - PP fault type; 0 = icache, 1 = DEA

Figure 14. PPERROR Register

packet transfer request register (0x000D)

PKTREQ controls the submission and priority of packet-transfer requests. It also indicates if a PT is currently active.

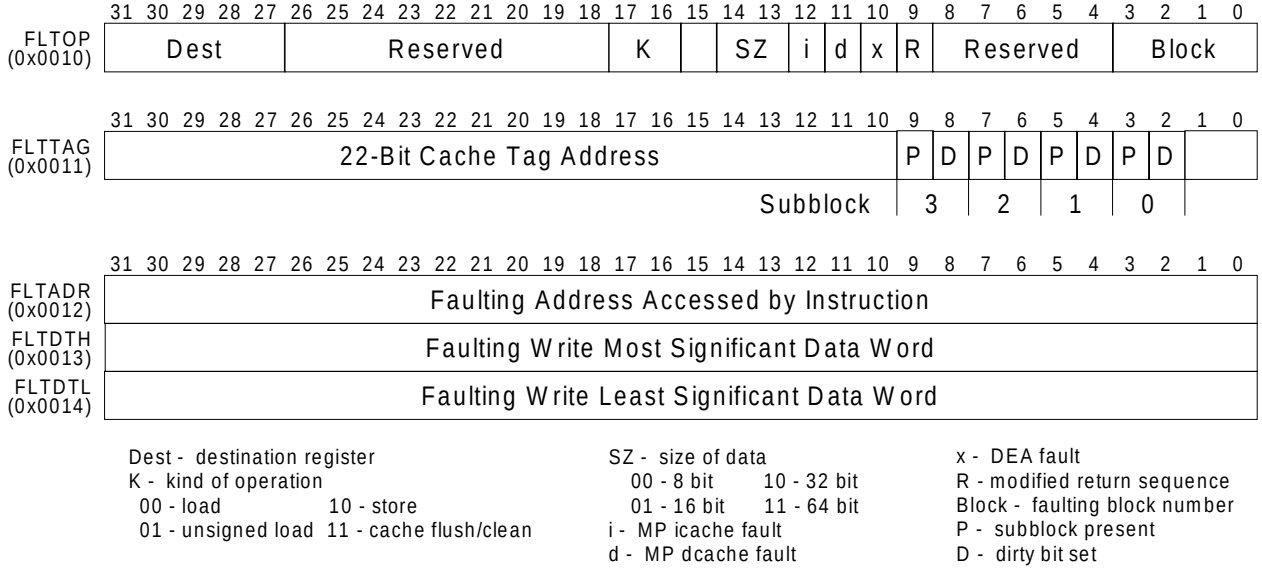


I - immediate (urgent) priority selected
F - high (foreground) priority selected
S - suspend packet transfer
Q - packet transfer queued; Read only
P - submit packet transfer request

Figure 15. PKTREQ Register

memory fault registers

The five read-only memory fault registers contain information about memory address exceptions.



Dest - destination register
K - kind of operation
00 - load
10 - store
01 - unsigned load
11 - cache flush/clean

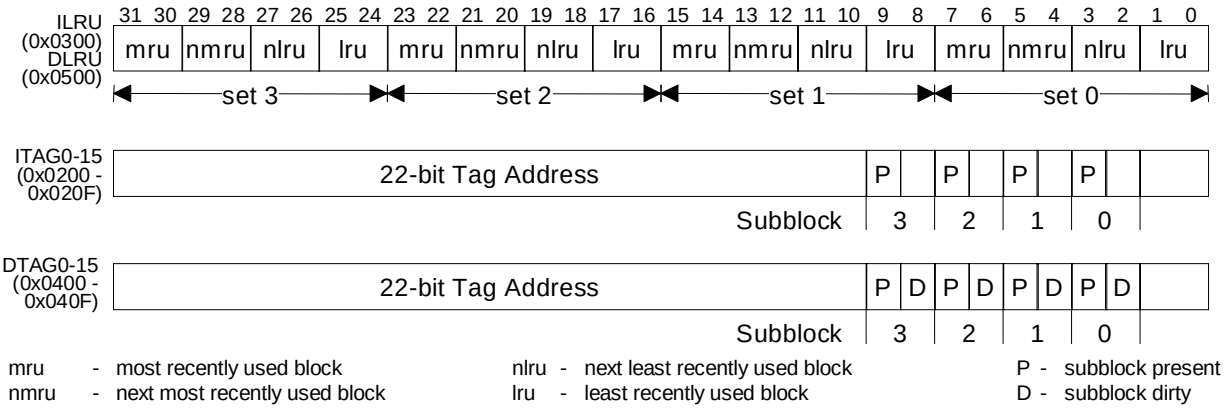
SZ - size of data
00 - 8 bit
10 - 32 bit
01 - 16 bit
11 - 64 bit
i - MP icache fault
d - MP dcache fault

x - DEA fault
R - modified return sequence
Block - faulting block number
P - subblock present
D - dirty bit set

Figure 16. Memory Fault Registers

cache registers

The ILRU and DLRU registers track least recently used information for the sixteen instruction cache and sixteen data cache blocks. The ITAGxx registers contain block addresses and the present flags for each subblock. DTAGxx registers are identical to ITAGxx registers but include dirty bits for each subblock.



mru - most recently used block
nmru - next most recently used block
nlru - next least recently used block
lru - least recently used block
P - subblock present
D - subblock dirty

mru, nmru, nlru, and lru have the value 0, 1, 2, or 3 representing the block number and are mutually exclusive for each set.

Figure 17. Cache Registers

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77251-1443

cache architecture

The MP contains two four-way set associative 4K caches—one for instructions and one for data. Each cache is divided into four sets with four blocks in each set. Each block represents 256 bytes of contiguous instructions or data and is aligned to a 256-byte address boundary. Each block is partitioned into eight subblocks that each contain sixteen 32-bit words and are aligned to 64-byte boundaries within the block. Cache misses cause one subblock to be loaded into cache. Figure 18 shows the cache architecture for one of the four sets in each cache. Figure 19 shows how addresses map into the cache using the cache tags and address bits.

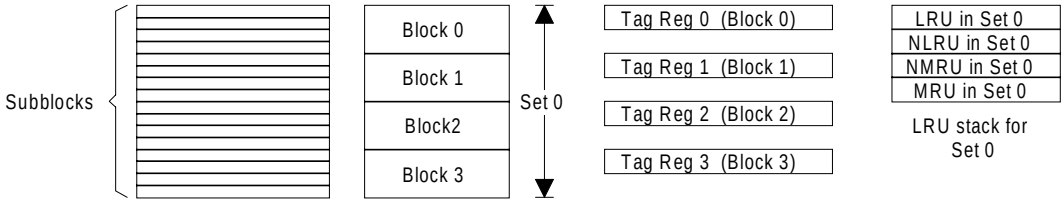


Figure 18. MP Cache Architecture (x4 Sets)

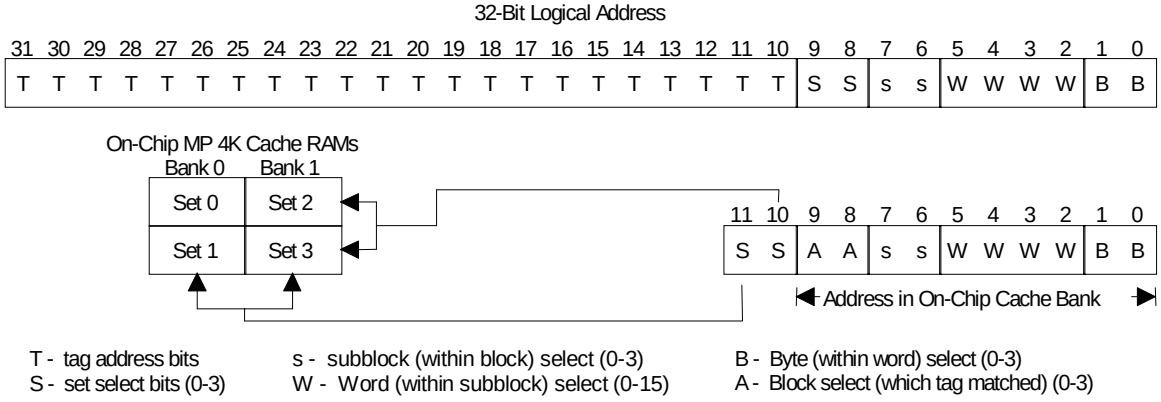


Figure 19. Cache Addressing

MP parameter RAM

The parameter RAM is a non-cacheable, 4K-byte, on-chip RAM which contains MP interrupt vectors, MP requested TC task buffers, and a general-purpose area. Figure 20 shows the parameter RAM address map.

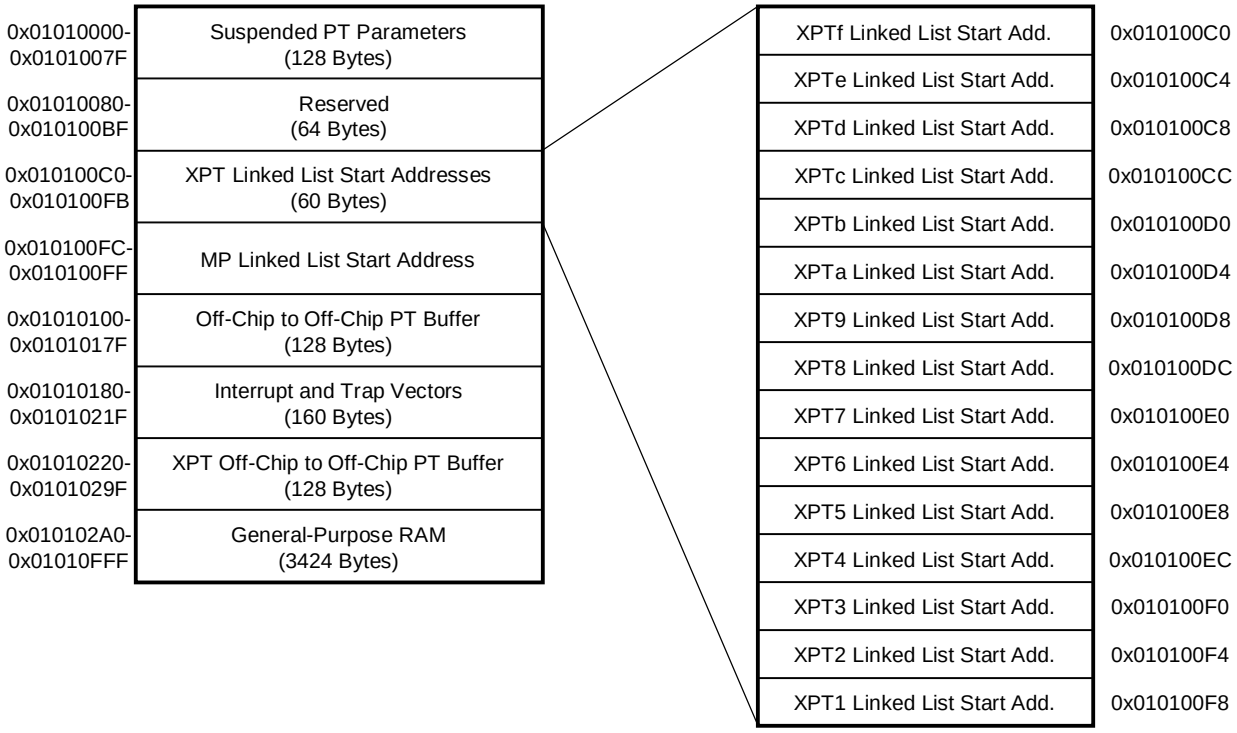


Figure 20. MP Parameter RAM

MP interrupt vectors

The MP interrupts and traps and their vector addresses are shown in Table 3 and Table 4.

Table 3. Maskable Interrupts

IE BIT (TRAP #)	NAME	VECTOR ADDRESS	MASKABLE INTERRUPT
0	ie	0x01010180	
2	fi	0x01010188	Floating-point invalid
3	fz	0x0101018C	Floating-point divide by zero
5	fo	0x01010194	Floating-point overflow
6	fu	0x01010198	Floating-point underflow
7	fx	0x0101019C	Floating-point inexact
8	f0	0x010101A0	Reserved
9	f1	0x010101A4	Reserved
10	ti	0x010101A8	MP timer
11	x1	0x010101AC	External interrupt 1 (/EINT1)
12	x2	0x010101B0	External interrupt 2 (/EINT2)
14	mf	0x010101B8	Memory fault
15	io	0x010101BC	Integer overflow
16	p0	0x010101C0	PP0 message
17	p1	0x010101C4	PP1 message
18	p2	0x010101C8	Reserved
19	p3	0x010101CC	Reserved
25	mi	0x010101E4	MP message
26	pc	0x010101E8	Packet transfer complete
27	pb	0x010101EC	Packet transfer busy
28	bp	0x010101F0	Bad packet transfer
29	x3	0x010101F4	External interrupt 3 (/EINT3)
30	x4	0x010101F8	External interrupt 4 (/LINT4)
31	pe	0x010101FC	PP error

Table 4. Nonmaskable Traps

TRAP NUMBER	NAME	VECTOR ADDRESS	NONMASKABLE TRAP
32	e1	0x01010200	Emulator trap1 (reserved)
33	e2	0x01010204	Emulator trap2 (reserved)
34	e3	0x01010208	Emulator trap3 (reserved)
35	e4	0x0101020C	Emulator trap4 (reserved)
36	fe	0x01010210	Floating-point error
37		0x01010214	Reserved
38	er	0x01010218	Illegal MP instruction
39		0x0101021C	Reserved
72 to 415		0x010102A0 to 0x010107FC	System- or user-defined

MP opcode formats

The three basic classes of MP instruction opcodes are short immediate, three register, and long immediate. The opcode structure for each class of instruction is shown in Figure 21.

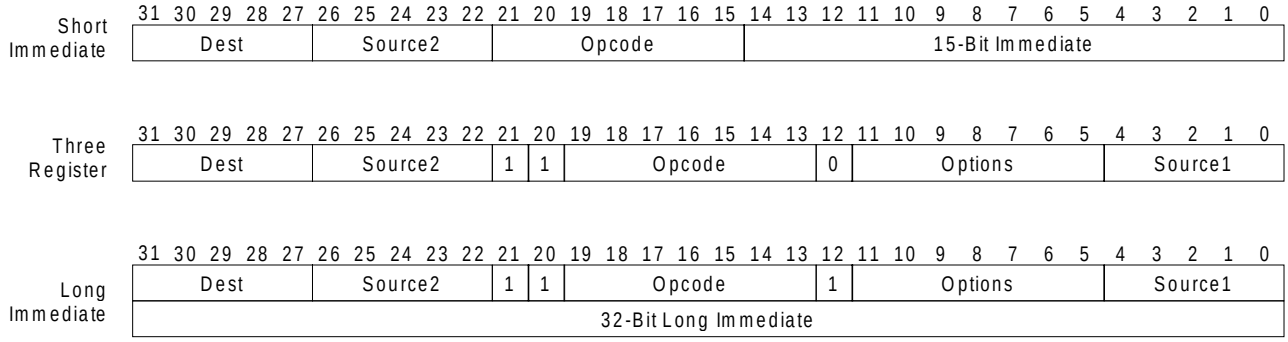


Figure 21. MP Opcode Formats

MP opcode summary

The opcode formats for the MP are shown in Table 5 through Table 7. Table 8 summarizes the master processor instruction set.

MP opcode summary (continued)

Table 5. Short Immediate Opcodes

	31 30 29 28 27	26 25 24 23 22	21 20 19 18 17 16 15	14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
illop0	Dest	Source	0 0 0 0 0 0 0	Unsigned Immediate
trap	- - - - E	- - - - -	0 0 0 0 0 0 1	Unsigned Trap Number
cmnd	- - - - -	- - - - -	0 0 0 0 0 1 0	Unsigned Immediate
rdcr	Dest	- - - - -	0 0 0 0 1 0 0	Unsigned Control Register Number
swcr	Dest	Source	0 0 0 0 1 0 1	Unsigned Control Register Number
brcr	- - - - -	- - - - -	0 0 0 0 1 1 0	Unsigned Control Register Number
shift.dz	Dest	Source	0 0 0 1 0 0 0	- - - i n Endmask Rotate
shift.dm	Dest	Source	0 0 0 1 0 0 1	- - - i n Endmask Rotate
shift.ds	Dest	Source	0 0 0 1 0 1 0	- - - i n Endmask Rotate
shift.ez	Dest	Source	0 0 0 1 0 1 1	- - - i n Endmask Rotate
shift.em	Dest	Source	0 0 0 1 1 0 0	- - - i n Endmask Rotate
shift.es	Dest	Source	0 0 0 1 1 0 1	- - - i n Endmask Rotate
shift.iz	Dest	Source	0 0 0 1 1 1 0	- - - i n Endmask Rotate
shift.im	Dest	Source	0 0 0 1 1 1 1	- - - i n Endmask Rotate
and.tt	Dest	Source2	0 0 1 0 0 0 1	Unsigned Immediate
and.tf	Dest	Source2	0 0 1 0 0 1 0	Unsigned Immediate
and.ft	Dest	Source2	0 0 1 0 1 0 0	Unsigned Immediate
xor	Dest	Source2	0 0 1 0 1 1 0	Unsigned Immediate
or.tt	Dest	Source2	0 0 1 0 1 1 1	Unsigned Immediate
and.ff	Dest	Source2	0 0 1 1 0 0 0	Unsigned Immediate
xnor	Dest	Source2	0 0 1 1 0 0 1	Unsigned Immediate
or.tf	Dest	Source2	0 0 1 1 0 1 1	Unsigned Immediate
or.ft	Dest	Source2	0 0 1 1 1 0 1	Unsigned Immediate
or.ff	Dest	Source2	0 0 1 1 1 1 0	Unsigned Immediate
ld	Dest	Base	0 1 0 0 M SZ	Signed Offset
ld.u	Dest	Base	0 1 0 1 M SZ	Signed Offset
st	Source	Base	0 1 1 0 M SZ	Signed Offset
dcache	- - - - F	Source2	0 1 1 1 M 0 0	Signed Offset
bsr	Link	- - - - -	1 0 0 0 0 0 A	Signed Offset
jsr	Link	Base	1 0 0 0 1 0 A	Signed Offset
bbz	BITNUM	Source	1 0 0 1 0 0 A	Signed Offset
bbo	BITNUM	Source	1 0 0 1 0 1 A	Signed Offset
bcnd	Cond	Source	1 0 0 1 1 0 A	Signed Offset
cmp	Dest	Source2	1 0 1 0 0 0 0	Signed Immediate
add	Dest	Source2	1 0 1 1 0 0 U	Signed Immediate
sub	Dest	Source2	1 0 1 1 0 1 U	Signed Immediate

- \- Reserved bit (code as 0)
- A Annul delay slot instruction if branch taken
- E Emulation trap bit
- F Clear present flags
- i Invert endmask
- M Modify, write modified address back to register
- n Rotate sense for shifting
- SZ Size (0=byte, 1=halfword, 2=word, 3=doubleword)
- U Unsigned form

MP opcode summary (continued)

Table 6. Long Immediate and Three Register Opcodes

	31 30 29 28	27	26 25 24 23 22	21 20 19 18 17 16 15 14 13	12	11 10 9 8 7 6 5	4 3 2 1 0
trap	- - - -	E	- - - - -	1 1 0 0 0 0 0 0 1	I	- - - - - - -	IND TR
cmnd	- - - - -		- - - - -	1 1 0 0 0 0 0 1 0	I	- - - - - - -	Source1
rdcr	Dest		- - - - -	1 1 0 0 0 0 1 0 0	I	- - - - - - -	IND CR
swcr	Dest		Source	1 1 0 0 0 0 1 0 1	I	- - - - - - -	IND CR
brcr	- - - - -		- - - - -	1 1 0 0 0 0 1 1 0	I	- - - - - - -	IND CR
shift.dz	Dest		Source	1 1 0 0 0 1 0 0 0	I	i n Endmask	Rotate
shift.dm	Dest		Source	1 1 0 0 0 1 0 0 1	I	i n Endmask	Rotate
shift.ds	Dest		Source	1 1 0 0 0 1 0 1 0	I	i n Endmask	Rotate
shift.ez	Dest		Source	1 1 0 0 0 1 0 1 1	I	i n Endmask	Rotate
shift.em	Dest		Source	1 1 0 0 0 1 1 0 0	I	i n Endmask	Rotate
shift.es	Dest		Source	1 1 0 0 0 1 1 0 1	I	i n Endmask	Rotate
shift.iz	Dest		Source	1 1 0 0 0 1 1 1 0	I	i n Endmask	Rotate
shift.im	Dest		Source	1 1 0 0 0 1 1 1 1	I	i n Endmask	Rotate
and.tt	Dest		Source2	1 1 0 0 1 0 0 0 1	I	- - - - - - -	Source1
and.tf	Dest		Source2	1 1 0 0 1 0 0 1 0	I	- - - - - - -	Source1
and.ft	Dest		Source2	1 1 0 0 1 0 1 0 0	I	- - - - - - -	Source1
xor	Dest		Source2	1 1 0 0 1 0 1 1 0	I	- - - - - - -	Source1
or.tt	Dest		Source2	1 1 0 0 1 0 1 1 1	I	- - - - - - -	Source1
and.ff	Dest		Source2	1 1 0 0 1 1 0 0 0	I	- - - - - - -	Source1
xnor	Dest		Source2	1 1 0 0 1 1 0 0 1	I	- - - - - - -	Source1
or.tf	Dest		Source2	1 1 0 0 1 1 0 1 1	I	- - - - - - -	Source1
or.ft	Dest		Source2	1 1 0 0 1 1 1 0 1	I	- - - - - - -	Source1
or.ff	Dest		Source2	1 1 0 0 1 1 1 1 0	I	- - - - - - -	Source1
ld	Dest		Base	1 1 0 1 0 0 M SZ	I	S D - - - - -	Offset
ld.u	Dest		Base	1 1 0 1 0 1 M SZ	I	S D - - - - -	Offset
st	Source		Base	1 1 0 1 1 0 M SZ	I	S D - - - - -	Offset
dcache	- - - -	F	Source2	1 1 0 1 1 1 M 0 0	I	0 0 - - - - -	Source1
bsr	Link		- - - - -	1 1 1 0 0 0 0 0 A	I	- - - - - - -	Offset
jsr	Link		Base	1 1 1 0 0 0 1 0 A	I	- - - - - - -	Offset
bbz	BITNUM		Source	1 1 1 0 0 1 0 0 A	I	- - - - - - -	Target
bbo	BITNUM		Source	1 1 1 0 0 1 0 1 A	I	- - - - - - -	Target
bcnd	Cond		Source	1 1 1 0 0 1 1 0 A	I	- - - - - - -	Target
cmp	Dest		Source2	1 1 1 0 1 0 0 0 0	I	- - - - - - -	Source1
add	Dest		Source2	1 1 1 0 1 1 0 0 U	I	- - - - - - -	Source1
sub	Dest		Source2	1 1 1 0 1 1 0 1 U	I	- - - - - - -	Source1

- \- Reserved bit (code as 0)
- D Direct external access bit
- E Emulation trap bit
- F Clear present flags
- i Invert endmask
- I Long immediate
- M Modify, write modified address back to register
- n Rotate sense for shifting
- S Scale offset by data size
- SZ Size (0=byte, 1=halfword, 2=word, 3=doubleword)

MP opcode summary (continued)

Table 7. Miscellaneous Instruction Opcode

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
vadd	Mem Dst					Source2/Dest					1	1	1	1	0	-	0	0	0	I	-	m	P	-	d	m	s	Source1				
vsub	Mem Dst					Source2/Dest					1	1	1	1	0	-	0	0	1	I	-	m	P	-	d	m	s	Source1				
vmpy	Mem Dst					Source2/Dest					1	1	1	1	0	-	0	1	0	I	-	m	P	-	d	m	s	Source1				
vmsub	Mem Dst					Dest					1	1	1	1	0	a	0	1	1	I	a	m	P	Z	-	m	-	Source1				
vrnd(FP)	Mem Dst					Dest					1	1	1	1	0	a	1	0	0	I	a	m	P	PD		m	s	Source1				
vrnd (Int)	Mem Dst					Dest					1	1	1	1	0	-	1	0	1	I	-	m	P	-	d	m	s	Source1				
vmac	Mem Dst					Source2					1	1	1	1	0	a	1	1	0	I	a	m	P	Z	-	m	-	Source1				
vmsc	Mem Dst					Source2					1	1	1	1	0	a	1	1	1	I	a	m	P	Z	-	m	-	Source1				
fadd	Dest					Source2					1	1	1	1	1	0	0	0	0	I	-	PD		P2		P1		Source1				
fsub	Dest					Source2					1	1	1	1	1	0	0	0	1	I	-	PD		P2		P1		Source1				
fmpy	Dest					Source2					1	1	1	1	1	0	0	1	0	I	-	PD		P2		P1		Source1				
fdiv	Dest					Source2					1	1	1	1	1	0	0	1	1	I	-	PD		P2		P1		Source1				
frndx	Dest					-	-	-	-	-	1	1	1	1	1	0	1	0	0	I	-	PD		RM		P1		Source1				
fcmp	Dest					Source2					1	1	1	1	1	0	1	0	1	I	-	-	-	P2		P1		Source1				
fsqrt	Dest					-	-	-	-	-	1	1	1	1	1	0	1	1	1	I	-	PD		-	-	P1		Source1				
lmo	Dest					Source					1	1	1	1	1	1	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-
rmo	Dest					Source					1	1	1	1	1	1	0	0	1	-	-	-	-	-	-	-	-	-	-	-	-	-
estop	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1	0	-	-	-	-	-	-	-	-	-	-	-	-	-
illopF	-	-	-	-	-	-	-	-	-	-	1	1	1	1	1	1	1	1	1	C	-	-	-	-	-	-	-	-	-	-	-	-

- \- Reserved (code as 0)
- a Floating-point accumulator select
- C Constant operands rather than register
- d Destination precision for vector (0=sp, 1=dp)
- I Long-immediate 32-bit data
- m Parallel memory operation specifier
- P Dest precision for parallel load/store (0=single, 1=double)
- P1 Precision of source1 operand
- P2 Precision of source2 operand
- PD Precision of destination result
- RM Rounding Mode (0=N, 1=Z, 2=P, 3=M)
- s Scale offset by data size
- Z Use 0 rather than accumulator

MP opcode summary (continued)

Table 8. Summary of MP Opcodes

INSTRUCTION	DESCRIPTION	INSTRUCTION	DESCRIPTION
add	Signed integer add	or.ff	Bitwise OR with 1's complement
and.tt	Bitwise AND	or.ft	Bitwise OR with 1's complement
and.ff	Bitwise AND with 1's complement	or.tf	Bitwise OR with 1's complement
and.ft	Bitwise AND with 1's complement	rdcr	Read control register
and.tf	Bitwise AND with 1's complement	rmo	Rightmost one
bbo	Branch bit one	shift.dz	Shift, disable mask, zero extend
bbz	Branch bit zero	shift.dm	Shift, disable mask, merge
bcnd	Branch conditional	shift.ds	Shift, disable mask, sign extend
br	Branch always	shift.ez	Shift, enable mask, zero extend
brcr	Branch control register	shift.em	Shift, enable mask, merge
bsr	Branch and save return	shift.es	Shift, enable mask, sign extend
cmnd	Send command	shift.iz	Shift, invert mask, zero extend
cmp	Integer compare	shift.im	Shift, invert mask, merge
dcache	Flush data cache subblock	st	Store register into memory
estop	Emulation stop	sub	Signed integer subtract
fadd	Floating-point add	swcr	Swap control register
fcmp	Floating-point compare	trap	Trap
fdiv	Floating-point divide	vadd	Vector floating-point add
fmpy	Floating-point multiply	vmac	Vector floating-point multiply and add to accumulator
frndx	Floating-point convert/round	vmpy	Vector floating-point multiply
fsqrt	Floating-point square root	vmsc	Vector floating-point multiply and subtract from accumulator
fsub	Floating-point subtract	vmsub	Vector floating-point subtract accumulator from source
illop	Illegal operation	vrnd(FP)	Vector round with floating-point input
jsr	Jump and save return	vrnd(Int)	Vector round with integer input
ld	Load signed into register	vsub	Vector floating-point subtract
ld.u	Load unsigned into register	xnor	Bitwise exclusive NOR
lmo	Leftmost one	xor	Bitwise exclusive OR
or.tt	Bitwise OR		

parallel processor architecture

The Parallel Processor (PP) is a 32-bit integer digital signal processor (DSP) optimized for imaging and graphics applications. The PP can execute in parallel a multiply, ALU operation, and two memory accesses within a single instruction. This internal parallelism allows a single PP to achieve over 500 million operations per second for certain algorithms. The PP has a three-input ALU that supports all 256 three input Boolean combinations and many combinations of arithmetic and Boolean functions. Data merging and bit-to-byte, bit-to-word, and bit-to-halfword translations are supported by hardware in the input data path to the ALU. Typical tasks performed by a PP include:

- Pixel-intensive processing
  - Motion estimation
  - Convolution
  - PixBLTs
  - Warp
  - Histogram
  - Mean square error
- Domain transforms
  - DCT
  - FFT
  - Hough
- Core graphics functions
  - Line
  - Circle
  - Shaded fills
  - Fonts
- Image Analysis
  - Segmentation
  - Feature extraction
- Bit-stream encoding/decoding
  - Data merging
  - Table look-ups

functional block diagram

Figure 22 shows a block diagram of a parallel processor. Key features of the PP include:

- 64-bit instruction word (supports multiple parallel operations)
- 3-stage pipeline for fast instruction cycle
- Numerous registers
  - 8 data, 10 address, 6 index registers
  - 20 other user-visible registers
- Data Unit
  - 16x16 integer multiplies (optional 8x8)
  - Splittable 3-input ALU
  - 32-bit barrel rotator
  - Mask generator
  - Multiple-status flag expander for translations to/from 1 bit-per-pixel space.
  - Conditional assignment of data unit results
  - Conditional source selection
  - Special processing hardware

    leftmost one / rightmost one

    leftmost bit change / rightmost bit change
- Memory addressing
  - 2 address units (global and local) provide up to two 32-bit accesses in parallel with data unit operation.
  - 12 addressing modes (immediate and indexed)
  - Byte, halfword, and word addressability
  - Scaled indexed addressing
  - Conditional assignment for loads
  - Conditional source selection for stores
- Program flow
  - Three hardware loop controllers

    zero overhead looping / branching

    nested loops

    multiple loop endpoints
  - Instruction cache management
  - PC mapped to register file
  - Interrupts for messages and context switching
- Algebraic assembly language

functional block diagram (continued)

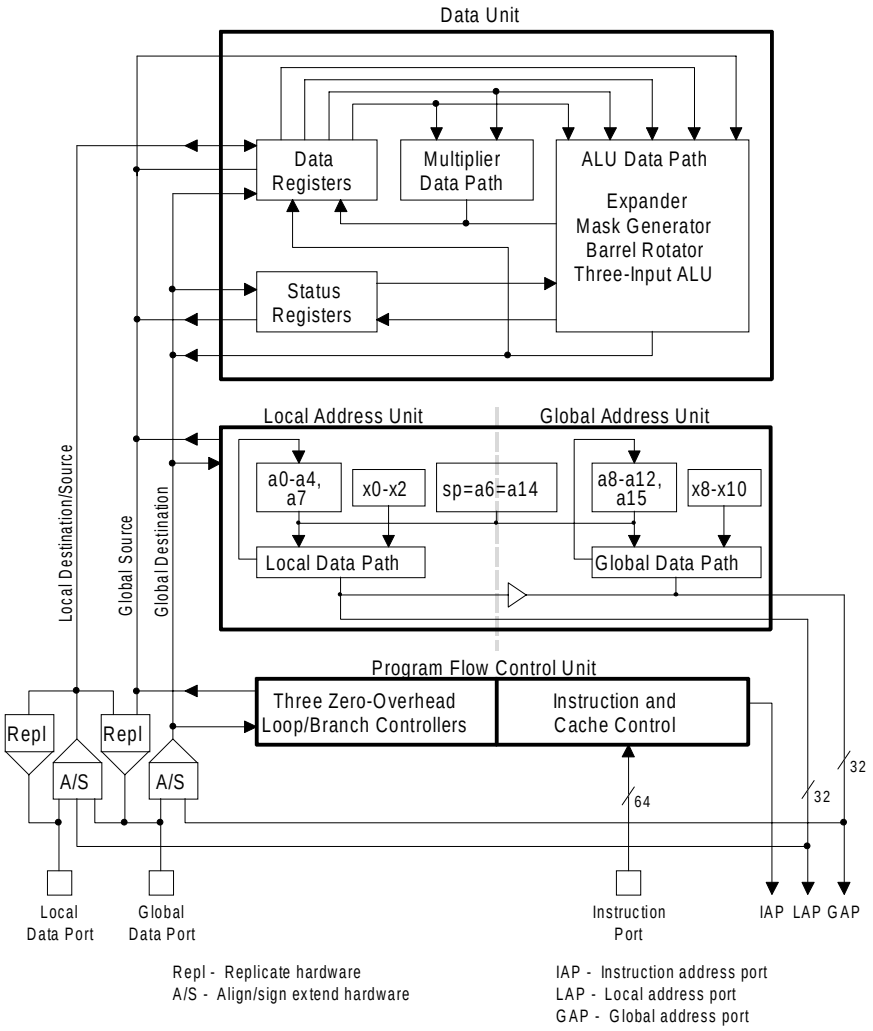


Figure 22. PP Block Diagram

PP registers

The PP contains many general-purpose registers. It also has a number of status registers and configuration registers. All PP registers are 32-bit registers. Below are the accessible registers of the various PP blocks.

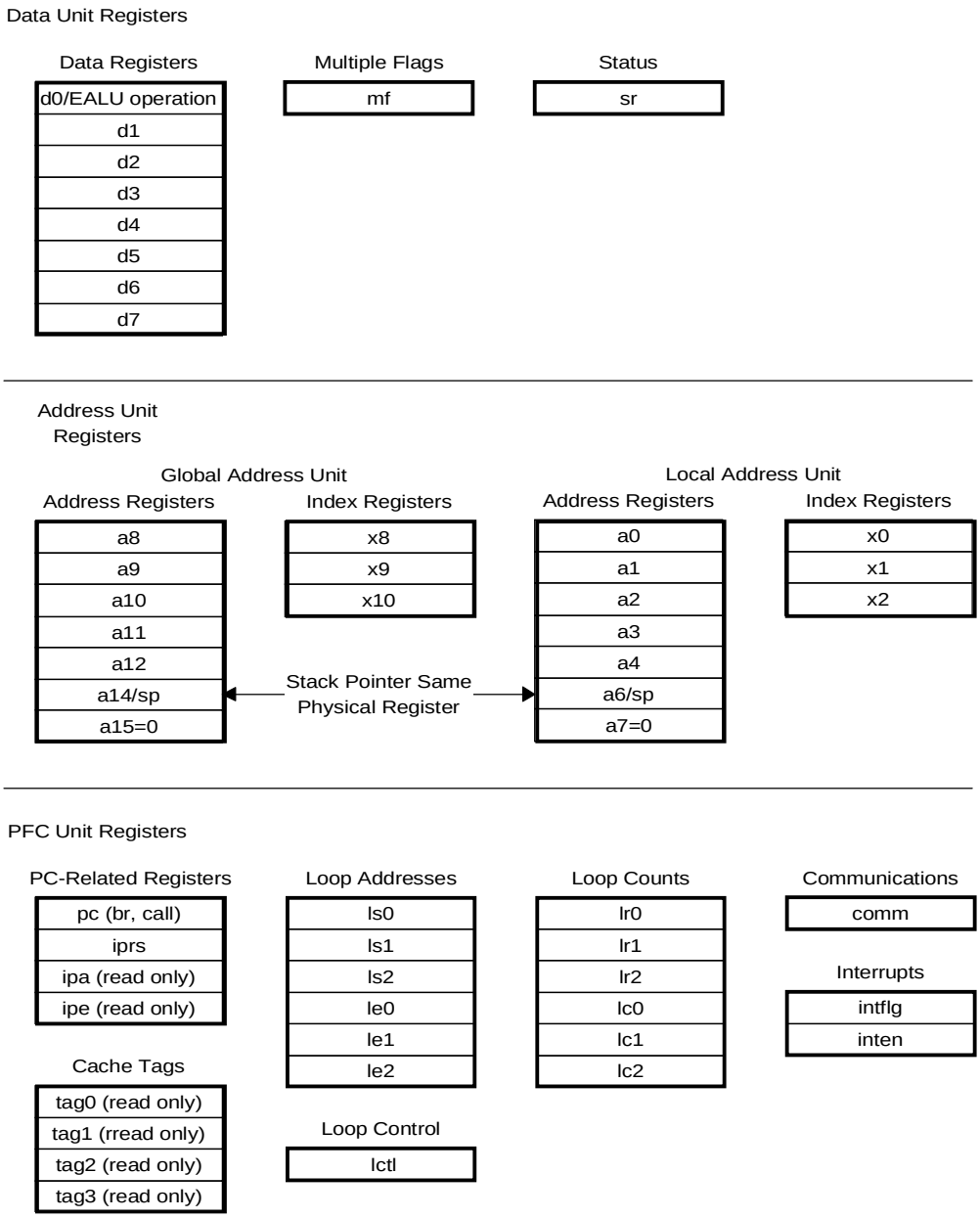


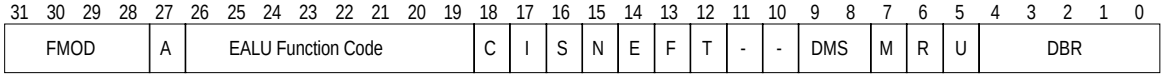
Figure 23. PP Registers

data unit registers

The data unit contains eight 32-bit general-purpose data registers (d0-d7) referred to as the D registers. The d0 register also acts as the control register for EALU operations.

d0 register

When used as the EALU control register, d0 has the format shown in Figure 24.



FMOD - function modifiers
A - arithemetic enable
C - EALU carry-in
I - invert-carry-in

S - sign extend
N - nonmultiple mask
E - explicit multiple carry-in
F - expanded mf
T - ALU saturate

DMS - default multiply shift amount
M - split multiply
R - rounded multiply
U - saturate multiplier output
DBR - default barrel rotate amount

Figure 24. d0 Format for EALU Operations

mf register

The multiple flags (mf) register records status information from each split ALU segment for multiple arithmetic operations. The mf register may be expanded to generate a mask for the ALU.

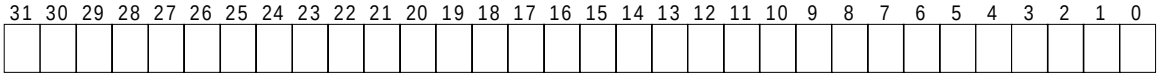
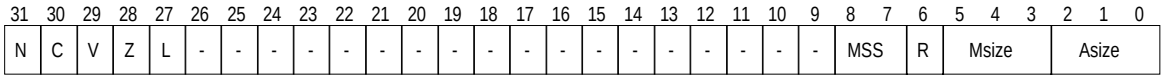


Figure 25. mf Register Format

sr register

The status register (sr) contains status and control bits for the PP ALU.



N - negative status bit
C - carry status bit
V - overflow status bit
Z - zero status bit
L - latched overflow (sticky)

MSS - mf status selection
00 - set by zero
01 - set by sign
10 - set by extended result
11 - reserved
R - rotation bit

Msize - expander data size
Asize - split ALU data size

Figure 26. sr Register Format

address unit registers

address registers

The address unit contains ten 32-bit address registers which contain the base address for address computations or may be used for general-purpose data. The registers a0 - a4 are used for local address computations and registers a8-a12 are used for global address computations.

index registers

The six 32-bit index registers contain index values for use with the address registers in address computations or may be used for general-purpose data. Registers x0-x2 are used by the local address unit and registers x8-x10 are used by the global address unit.

stack pointer

The stack pointer contains the address of the top of the PP's system stack. The stack pointer is addressed as a6 by the local address unit and as a14 by the global address unit.

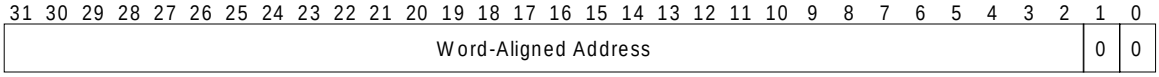


Figure 27. sp Register Format

zero register

The zero registers are read-as-zero address registers for the local address unit (a7) and global address unit (a15). Writes to the registers are ignored and may be specified when operational results are to be discarded.

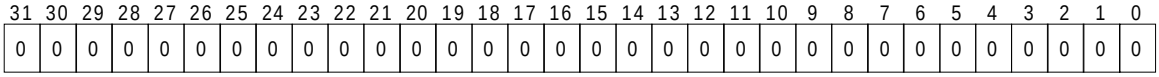


Figure 28. Zero Registers

PFC Unit Registers

loop registers

The loop registers control three levels of zero-overhead loops. The 32-bit loop start registers (ls0 - ls2) and loop end registers (le0 - le2) contain the starting and ending addresses for the loops. The loop counter registers (lc0 - lc2) contain the number of repetitions remaining in their associated loops. The lr0 - lr2 registers are loop reload registers used to support nested loops. The format for the loop control register (lctl) is shown in Figure 29. There are also six special write-only mappings of the loop reload registers. The lrs0 - lrs2 codes are used for fast initialization of ls*n*, lr*n*, and lc*n* registers for multi-instruction loops while the lrse0 - lrse2 codes are used for single instruction-loop fast initialization.

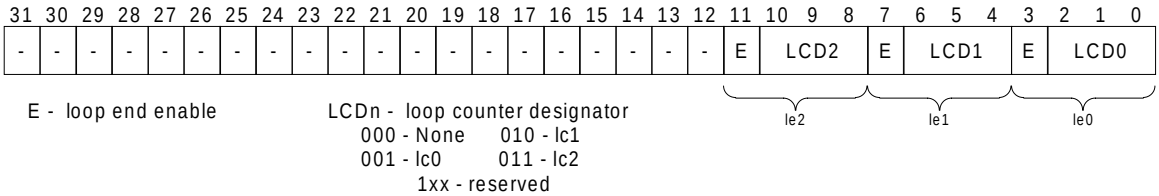


Figure 29. lctl Register

pipeline registers

The pfc unit contains a pointer to each stage of the PP pipeline. The pc contains the program counter which points to the instruction being fetched. The ipa points to the instruction in the address stage of the pipeline and the ipe points to the instruction in the execute stage of the pipeline. The instruction pointer return-from-subroutine (iprs) register contains the return address for a subroutine call.

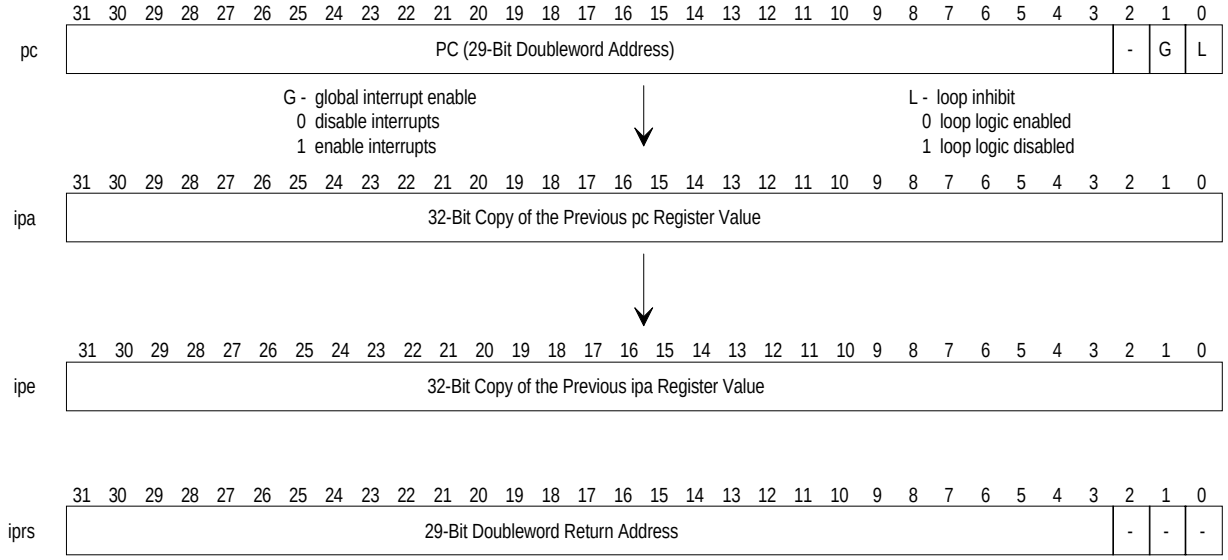


Figure 30. Pipeline Registers

interrupt registers

The interrupt enable register (inten) allows individual interrupts to be enabled and configures intflg operation. The interrupt flag register (intflg) contains interrupt flag bits.

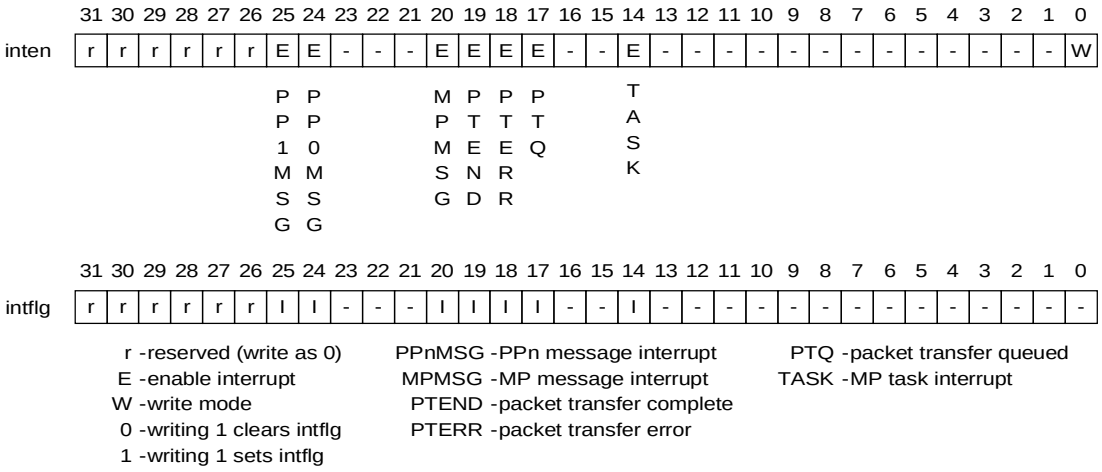


Figure 31. PP Interrupt Registers

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77251-1443

communication register

The comm register contains the packet transfer handshake bits and PP indicator bits.

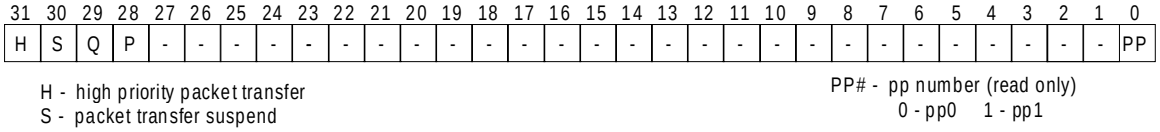


Figure 32. comm Register

cache tag registers

The tag0 - tag3 registers contain the tag address and subblock present bits for each cache block.

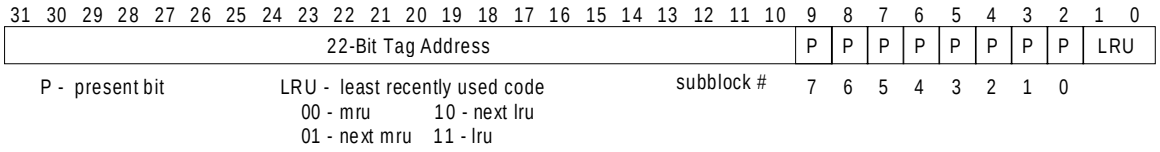


Figure 33. Cache Tag Registers

PP cache architecture

Each of the two PPs has its own 4K-byte instruction cache. Each cache is divided into four blocks and each block is divided into eight subblocks containing 16 64-bit instructions each. Cache misses cause one subblock to be loaded into cache. Figure 34 shows the cache architecture for one of the four sets in each cache. Figure 35 shows how addresses map into the cache using the cache tags and address bits.

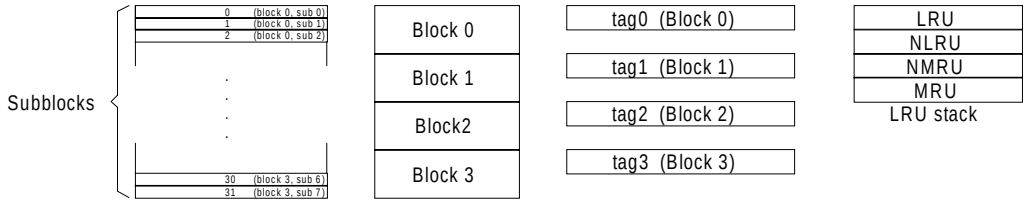


Figure 34. PP Cache Architecture

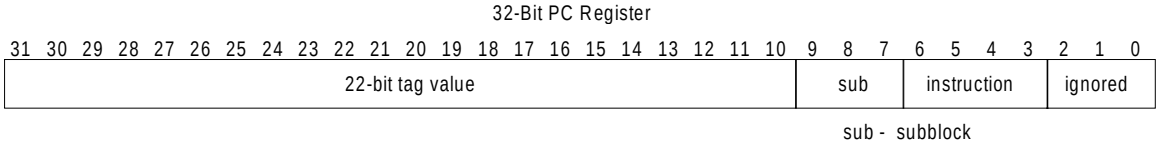


Figure 35. pc Register Cache Address Mapping

PP parameter RAM

The parameter RAM is a noncacheable, 4K-byte, on-chip RAM which contains PP interrupt vectors, PP requested TC task buffers, and a general-purpose area. Figure 36 shows the parameter RAM address map.

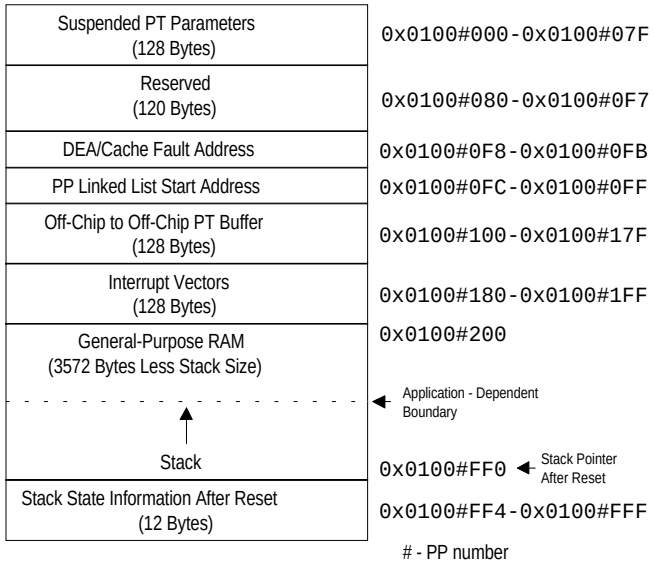


Figure 36. PP Parameter RAM

PP interrupt vectors

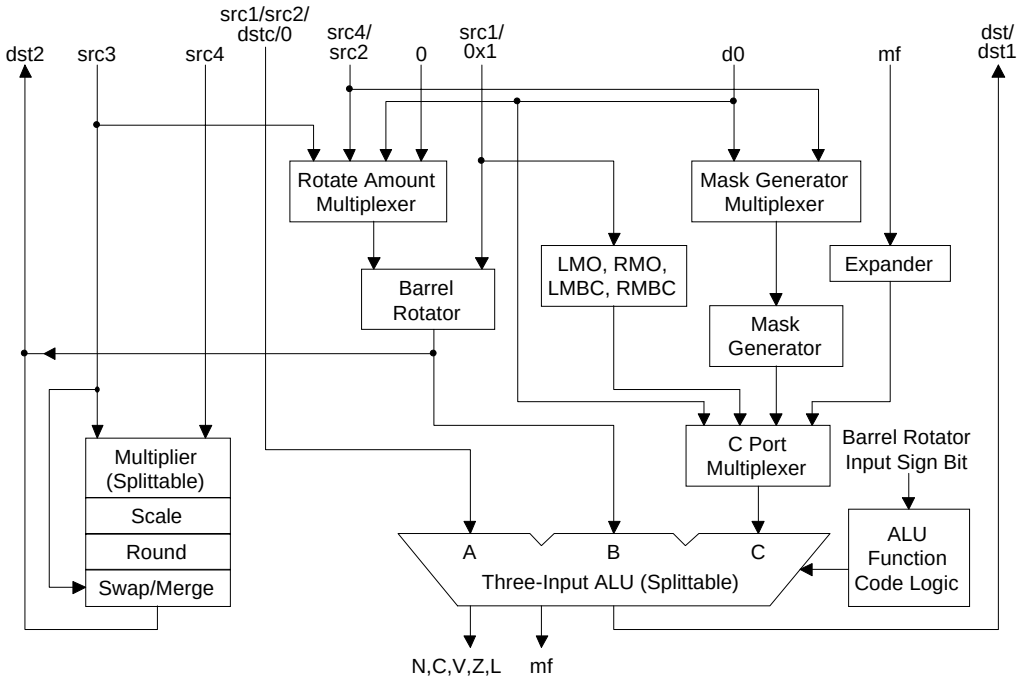
The PP interrupts and their vector addresses are shown in Table 9.

Table 9. PP Interrupt Vectors

NAME	VECTOR ADDRESS	INTERRUPT
TASK	0x0100#1B8	Task Interrupt
PTQ	0x0100#1C4	Packet Transfer Queued
PTERR	0x0100#1C8	Packet Transfer Error
PTEND	0x0100#1CC	Packet Transfer End
MPMSG	0x0100#1D0	MP Message
PP0MSG	0x0100#1E0	PP0 Message
PP1MSG	0x010101E4	PP1 Message

PP data unit architecture

The data unit has independent data paths for the ALU and the multiplier, each with its own set of hardware functions. The multiplier data path includes a 16x16 multiplier, a halfword swapper and rounding hardware. The ALU data path includes a 32-bit three-input ALU, a barrel rotator, mask generator, mf expander, left/right most one and left/right most bit change logic, and several multiplexers.



src1 - any register, D reg. only
for l/rmo, l/rmbc hardware
src2 - D reg. or sometimes 5/32-bit immediate
src3 - D reg. only
src4 - D reg. only
dst/dst1 - any register

dst2 - D reg. only
dstc - D reg. only (dest companion reg source)
0x1 - Constant
0 - Constant
d0 - 5 LSBs of d0

Figure 37. Data Unit Block Diagram

PP data unit architecture (continued)

The PP's ALU can be split into one 32-bit ALU, two 16-bit ALUs or four 8-bit ALUs. Figure 38 shows the multiple arithmetic data flow for the case of a four 8-bit split of the ALU (called multiple-byte arithmetic). The ALU operates as independent parallel ALUs where each ALU receives the same function code.

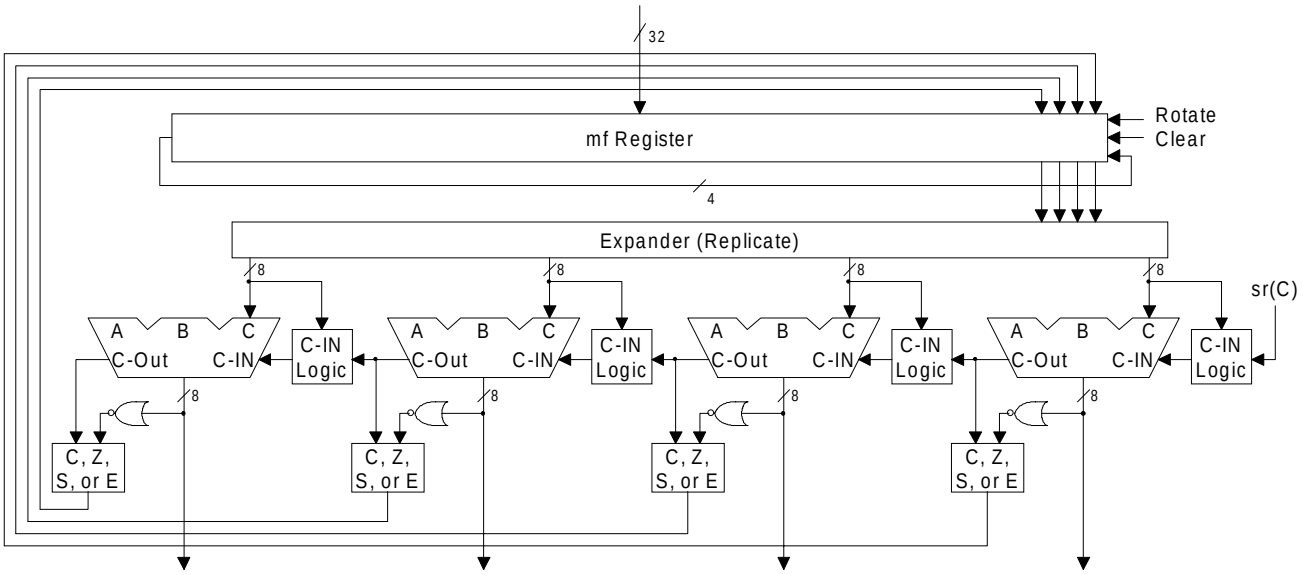


Figure 38. Multiple-Byte Arithmetic Data Flow

During EALU operations, the split ALU outputs may be saturated/clamped at maximum or minimum values. The ALU saturate feature is controlled by the T bit and the N bit in d0, as shown in Table 10. Saturation may only be specified for 32-bit signed arithmetic.

Table 10. ALU Saturate/Clamp Option

DO (EALU)		NON-MULTIPLE MASK/SATURATE-CLAMP OPTION
N	T	
0	0	Normal operation
0	1	Reserved
1	0	Non-multiple mask
1	1	Saturate-clamp-signed option

32-BIT SIGNED ALU:

Result = 0x7FFFFFFF	if	~Cout[31] & Cin[31]	Saturate at max positive value
Result = 0x80000000	if	Cout[31] & ~Cin[31]	Clamp at most negative value

PP multiplier

The PP's hardware multiplier can perform one 16x16 multiply with a 32-bit result or two 8x8 multiplies with two 16-bit results in a single cycle. A 16x16 multiply may use signed or unsigned operands as shown in Figure 39.

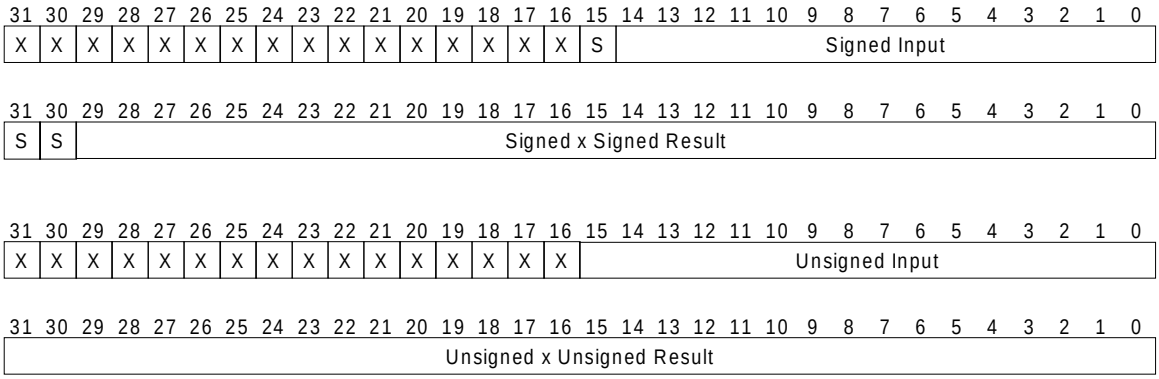


Figure 39. 16 x 16 Multiplier Data Formats

When performing two simultaneous 8x8 split multiplies. The first input word contains unsigned byte operands and the second input word may contain signed or unsigned byte operands. These formats are shown in Figure 40 and Figure 41.

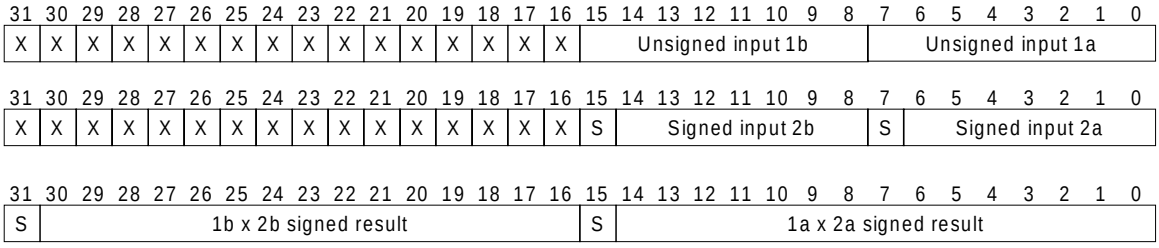


Figure 40. Signed Split-Multiply Data Formats

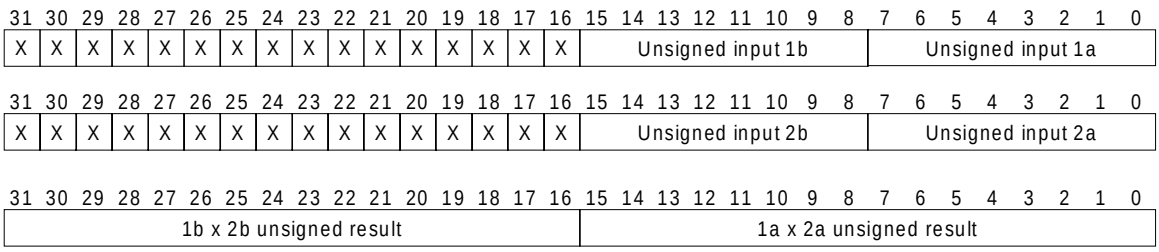


Figure 41. Unsigned Split-Multiply Data Formats

PP multiplier (continued)

Additionally, 16 x 16 multiplies may take on another form wherein the multiplier output is rounded by adding bit 15 to bit 16 of the result. The upper 16 bits of the result are written to the upper 16 bits of the destination register, while the lower 16 bits are filled with bits 31-16 of the first multiply source operand. This allows back-to-back multiplies to produce two rounded results, as shown below.

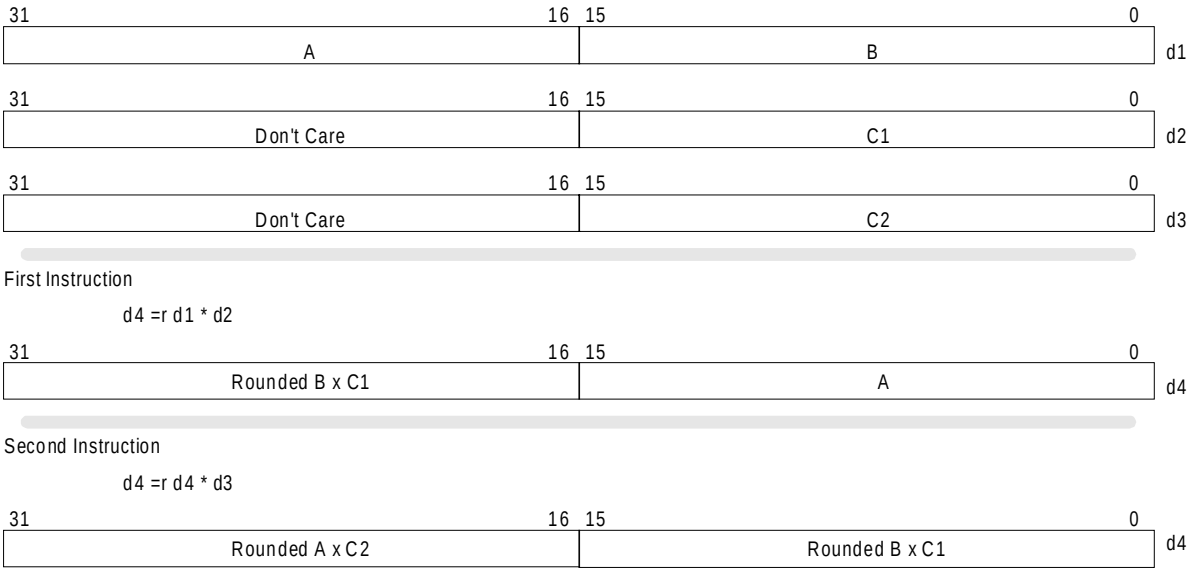


Figure 42. 16 x 16 Rounded Multiply

During MPY||EALU† operations, the multiplier output may be saturated as specified by the U bit in d0. Saturation is valid only for left shift of 0 and 1, as shown in Table 11. Like the ALU saturation option, multiplier saturation may only be specified during EALUs, and is valid only for signed multiplies. Multiplier and ALU saturation may be independently specified in a given EALU. Saturation is specified using the t function modifier.

Table 11. Multiplier Saturation

DMS		U	MULTIPLIER SATURATE OPTION
X	X	0	No saturation
0	0	1	set result to 0x3FFFFFFF (instead of 0x40000000)
0	1	1	set result to 0x7FFFFFFF (instead of 0x80000000)
1	X	1	No saturation

When rounding is enabled, the 16 LSBs of the result are protected (i.e., unaffected by the saturation option). In this case a pre-saturated result of 0x4000XXXX (DMS = 00) will be saturated to 0x3FFFXXXX, and a pre-saturated result of 0x7FFFXXXX (DMS = 01) is saturated to 0x8000XXXX. Saturation should not be specified with split multiplies.

†The || symbol indicates operations are to be performed in parallel.

PP program flow control unit architecture

The program flow control unit performs instruction fetching and decoding, loop control, and handshaking with the transfer controller. The pfc unit architecture is shown in Figure 43.

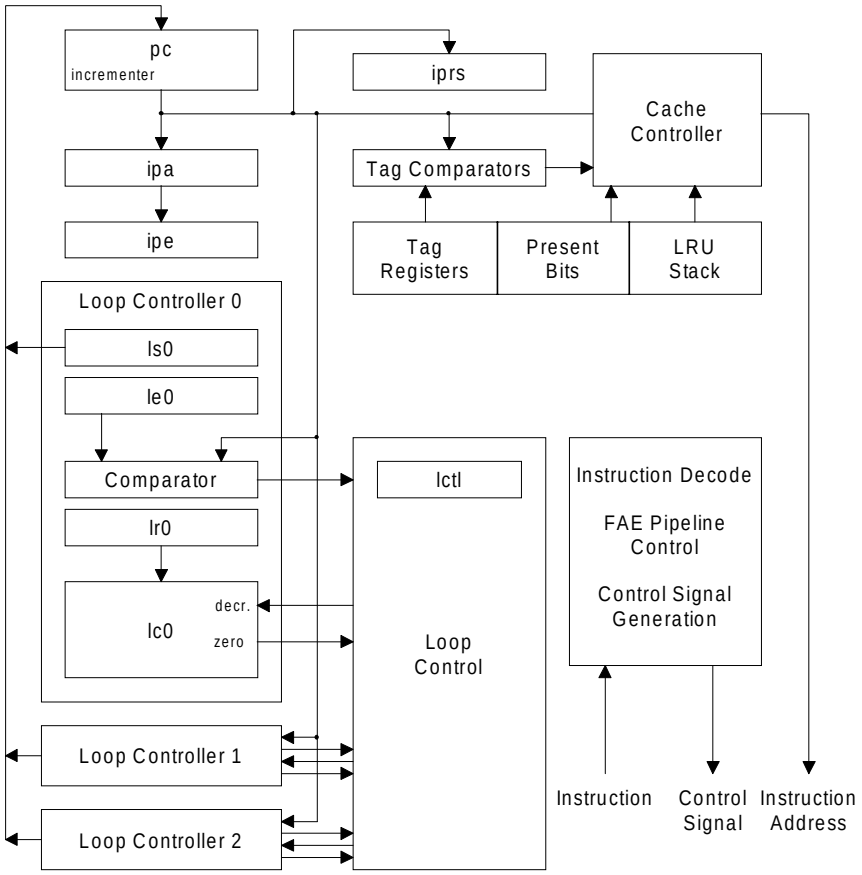


Figure 43. Program Flow Control Unit Block Diagram

The PP has a three-stage fetch, address, execute pipeline as shown in Figure 44. The pc, ipa, and ipe registers point to the address of the instruction in each stage of the pipeline. On each cycle in which the pipeline advances, ipa is copied into ipe, pc is copied into ipa, and the pc is incremented by one instruction (8 bytes).

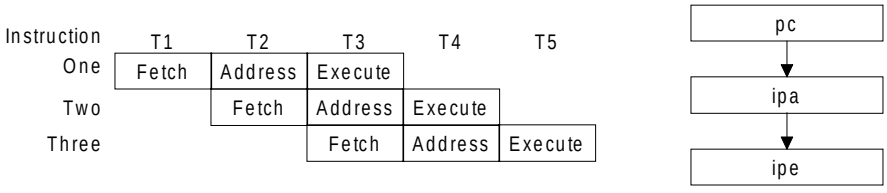


Figure 44. FAE Instruction Pipeline

PP address unit architecture

The PP has both a local and global address unit which operate independently of each other. The address units support twelve different addressing modes. In place of performing a memory access, either or both of the address units can perform an address computation that is written directly to a PP register instead of being used for a memory access. This address unit arithmetic provides additional arithmetic operation to supplement the data unit during compute-intensive algorithms.

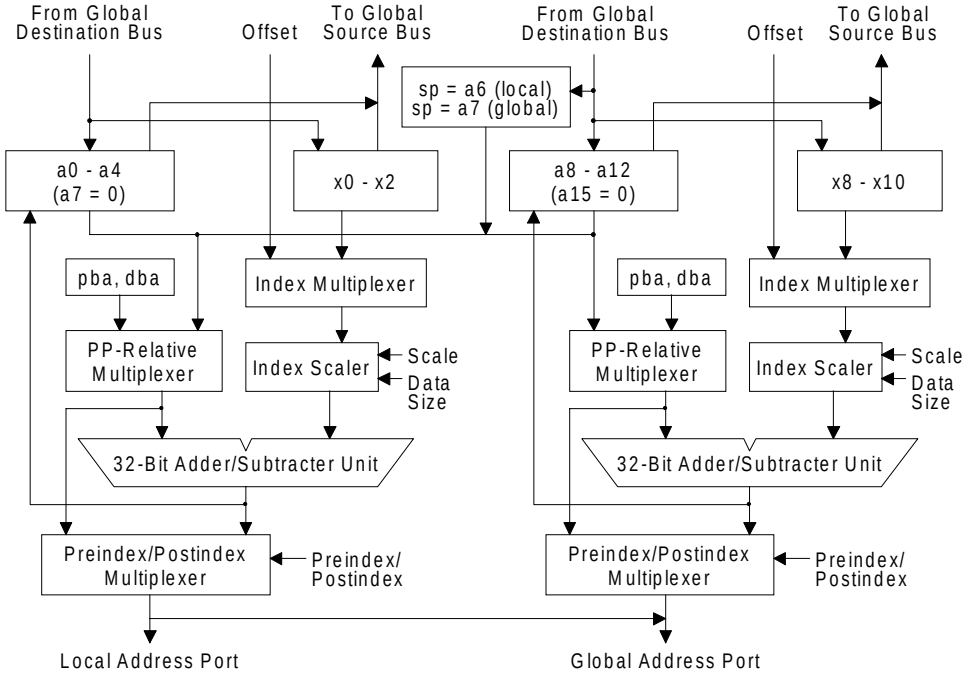


Figure 45. Address Unit Architecture

PP instruction set

PP instructions are represented by algebraic expressions for the operations performed in parallel by the multiplier, ALU, global address unit, and local address unit. The expressions use the || symbol to indicate operations that are to be performed in parallel. The PP ALU operator syntax is shown in Table 12. The data unit operations (multiplier and ALU) are summarized in Table 13 and the parallel transfers (global and local) are summarized in Table 14.

Table 12. PP Operators by Precedence

OPERATOR	FUNCTION
src1 [n] src1-1	Select odd (n=true) or even (n=false) register of D register pair based on negative condition code
( )	Subexpression delimiters
@mf	Expander operator
%	Mask generator
%%	Nonmultiple mask generator (EALU only)
%!	Modified mask generator (0xFFFFFFFF output for 0 input)
%%!	Nonmultiple shift right mask generator (EALU only)
\\	Rotate left
<<	Shift left (pseudo-op for rotate and mask)
>>u	Unsigned shift right
>> or >>s	Signed shift right
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
+	Addition
-	Subtraction
=[cond]	Conditional assignment
=[cond.pro]	Conditional assignment with status protection
=	Equate

PP instruction set (continued)

Table 13. Summary of Data Unit Operations

Operation	Base set ALUs
Description	Perform an ALU operation specifying ALU function, 2 src and 1 dest operand, and operand routing. ALU function is one of 256 three-input Boolean operations or one of 16 arithmetic operations combined with one of 16 function modifiers
Syntax	dst = [fmod] [{cond [.pro]}] ALU_EXPRESSION
Examples	d6 = (d6 ^ d4) & d2 d3 = [nn.nv] d1 -1
Operation	EALU || ROTATE
Description	Perform an extended ALU (EALU) operation (specified in d0) with one of two data routings to the ALU and optionally write the barrel rotator output to a second dest register. ALU operation is one of 256 Boolean or 256 arithmetic.
Syntax	dst1 = [{cond [.pro]}] ealu (src2, [dst2 =] [{cond}] src1 {n} src1-1 \\ src3, [%] src4) dst1 = [fmod] [{cond [.pro]}] ealu (label:EALU_EXPRESSION [|| dst2 = {cond} src1 [{n} src1-1 \\ src3])
Examples	d7 = [nn] ealu(d2, d6 = [nn] d3\\d1, %d4) d3 = mzc ealu(mylabel: d4 + (d5\\d6 & %d7) || d1 = d5\\d6)
Operation	MPY || ADD
Description	Perform a 16x16 multiply with optional parallel add or subtract. Condition code applies to both multiply and add.
Syntax	dst2 = [sign] [{cond}] src3 * src4 [|| dst = [{cond [.pro]}] src2 + src1 [{n} src1 -1]] dst2 = [sign] [{cond}] src3 * src4 [|| dst = [{cond [.pro]}] src2 - src1 [{n} src1 -1]]
Example	d7 = u d6 * d5 || d5 = d4 - d1
Operation	MPY || SADD
Description	Perform a 16x16 multiply with a parallel right-shift and add or subtract. Condition code applies to both multiply and shift and add.
Syntax	dst2 = [sign] [{cond}] src3 * src4 || dst = [{cond [.pro]}] src2 + src1 [{n} src1 -1] >> -d0 dst2 = [sign] [{cond}] src3 * src4 || dst = [{cond [.pro]}] src2 - src1 [{n} src1 -1] >> -d0
Examples	d7 = u d6 * d5 || d5 = d4 - d1 >> -d0
Operation	MPY || EALU
Description	Perform a multiply and an optional parallel EALU. Multiply can use rounding, scaling, or splitting features.
Syntax	Generic Form: dst2 = [sign] [{cond}] src3 * src4 || dst = [{cond [.pro]}] ealu[f] (src2, src1 [{n} src1 -1] \\ d0, %d0) dst2 = [sign] [{cond}] src3 * src4 || ealu() Explicit Form: dst2 = [sign] [opt] [{cond}] src3 * src4 [<<dms] || dst1 = [fmod] [{cond [.pro]}] ealu (label: EALU_EXPRESSION) dst2 = [sign] [opt] [{cond}] src3 * src4 [<<dms] || ealu (label)
Examples	d7 = [p] d5 * d3 || d2 = [p] ealu(d1, d6\\d0, %d0) ; generic form d2 = m d4 * d7 || d3 = ealu (mylabel: d3 + d2 >> 9) ; explicit form
Operation	divi
Description	Perform one iteration of unsigned divide algorithm. Generates one quotient bit per execution using iterative subtraction.
Syntax	dst1 = [{cond [.pro]}] divi (src2, dst2 = {cond} src1 [{n} src1-1])
Examples	d3 = divi (d1, d2 = d2) d3 = divi (d1, d2 = d3[n]d2)
Misc. Operations	dint; eint; dloop, eloop, qwait, nop
Description	Globally disable interrupts; globally enable interrupts; globally disable looping, globally enable looping; spin until comm Q bit is zero, do nothing in the data unit
Syntax	dint eint dloop eloop qwait nop

[ ] - optional parameter extension
{ } - square brackets ([ ]) must be used
sign - u=unsigned, s=signed
cond - condition code
f - use 1's complement of d0
fmod - function modifier
pro - protect status bits
dms - default multiply shift amount

PP instruction set (continued)

Table 14. Summary of Parallel Transfers

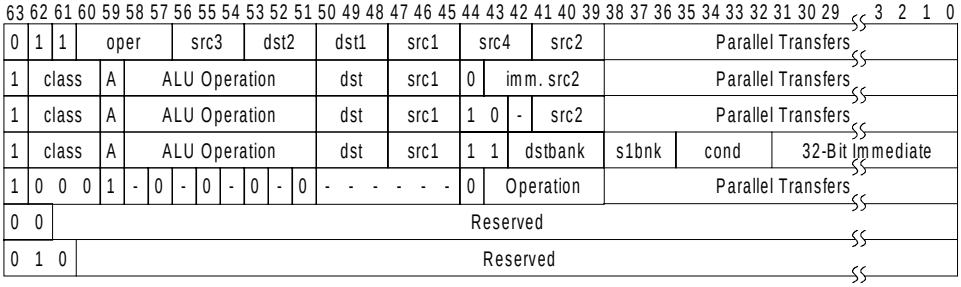
Operation	Load
Description	Transfer from memory into PP register
Syntax	dst = [sign] [size] [{cond}] * addrexp dst = [sign] [size] [{cond}] * an.element
Examples	d3 = u h [n] * (a9++=[2]) d1 = * a2.sMY_ELEMENT
Operation	Store
Description	Transfer from PP register into memory
Syntax	* addrexp = [size] src [{n} src-1] * an.element = [size] src [{n} src-1]
Examples	* --a2 = d3 *a9.sMY_ELEMENT = a3
Operation	Address Unit Arithmetic
Description	Compute address and store in PP register.
Syntax	dst = [size] [{cond}] & * addrexp dst = [size] [{cond}] & * an.element
Examples	d2 = &*(a3 + x0) a1 = &*a9.sMY_ELEMENT
Operation	Move
Description	Transfer from PP register to PP register
Syntax	dst = [g] [{cond}] src
Examples	x2 = mf d1 = g d3
Operation	Field Extract Move
Description	Transfer from PP register to PP register extracting and right-aligning one byte or halfword
Syntax	dst = [sign] [size item]
Example	d3 = u b2 d1
Operation	Field Replicate Move
Description	Transfer from PP register to PP register replicating the LSbyte or LShalfword to 32 bits.
Syntax	dst = r [size] {cond} src
Example	d7 = rh d3

[ ] - optional parameter extension
{ } - square brackets ([ ]) must be used
sign - u=unsigned, s=signed
cond - condition code
size - b=byte, h=halfword, w=word (default)
item - 0=byte0/halfword0, 1=byte1/halfword1, 2=byte2, 3=byte3
g - use global unit

PP opcode formats

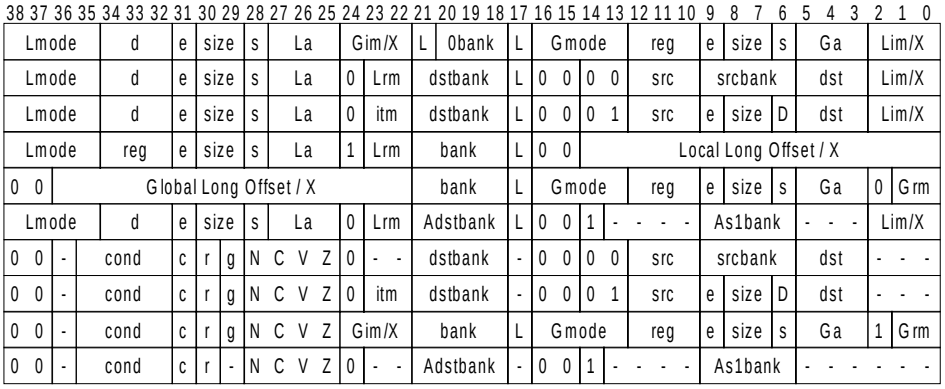
A PP instruction uses a 64-bit opcode. The opcode is essentially divided into a data unit portion and a parallel transfer portion. There are five data unit opcode formats comprising bits 39-63 of the opcode. Bits 0-38 of the opcode specify one of ten parallel transfer formats. An alphabetical list of the mnemonics used in Figure 46 for the data unit and parallel transfer portions of the opcode are shown in Table 15 and Table 16, respectively.

Data Unit Formats



A. Six-Operand (MPY||ADD, etc.)
B. Base Set ALU (5-Bit Immediate)
C. Base Set ALU (Register src2)
D. Base Set ALU (32-Bit Immediate)
E. Miscellaneous

Transfer Formats



1. Double Parallel
2. Move || Local
3. Field Move || Local
4. Local (Long Offset)
5. Global (Long Offset)
6. Non-D DU || Local
7. Conditional DU || Conditional Mode
8. Conditional DU || Conditional Field Move
9. Conditional DU || Conditional Global
10. Conditional Non-D DU

Figure 46. PP Opcode Formats

PP opcode formats (continued)

Table 15. Data Unit Mnemonics

MNEMONIC	FUNCTION
A	A = 1 selects arithmetic operations, A = 0 selects boolean operations
ALU Operation	For Boolean operation (A=0), select the eight ALU function signals. For Arithmetic operation (A=1), odd bits specify the ALU function and even bits define the ALU function modifiers.
class	Operation class, determines routing of ALU operands.
cond	condition code
dst	D register destination or lower three bits of non-D register code.
dst1	ALU dest. for MPY||ADD, MPY||EALU, or EALU||ROTATE operation. D register or lower three bits of non-D register code
dst2	Multiply dest. for MPY||ADD or MPY||EALU operation or rotate dest. for EALU||ROTATE operation. D register.
dstbank	ALU register bank.
imm.src2	5-bit immediate for src2 of ALU operation.
32-Bit Immediate	32-bit immediate for src2 of ALU operation.
oper	Six-operand data unit operation (MPY||ADD, MPY||SADD, MPY||EALU, EALU||ROTATE, divi)
Operation	Miscellaneous operation
src1	ALU source 1 register code (D register unless srcbank or s1bank is used)
src2	D register used as ALU source 2
src3	D register for multiplier source (MPY||ADD or MPY||EALU) or rotate amount (EALU||ROTATE)
src4	D register for ALU C port operand or EALU||ROTATE mask generator input or multiplier source 2 for MPY||ADD, MPY||EALU
s1bank	Bits 5-3 of src1 register code (bit 6 assumed to be 0)

PP opcode formats (continued)

Table 16. Parallel Transfer Mnemonics

MNEMONIC	FUNCTION
0bank	Bits 5-3 of global transfer source/destination register code (bit 6 assumed to be 0)
Adstbnk	Bits 6-3 of ALU destination register code
As1bank	Bits 6-3 of ALU source 1 register code
bank	Bits 6-3 of global (or local) store source or load destination
c	Conditional choice of D register for src1 operand of the ALU
C	Protect status register's carry bit
cond	Condition code
d	D register or lower three bits of register code for local transfer source/destination
D	Duplicate least significant data during moves
dst	The three lower bits of the register code for move or field move destination
dstbank	Bits 6-3 of move destination register code
e	Sign extend local (bit 31), sign extend global (bit 9)
g	Conditional global transfer
Ga	Global address register for load, store, or address unit arithmetic
Gim / X	Global address unit immediate offset or index register
Gmode	Global unit addressing mode
Grm	Global PP-relative addressing mode
itm	Number of item selected for field extract move
L	L = 1 selects load operation, L = 0 selects store / address unit arithmetic operation
La	Local address register for load, store or address unit arithmetic
Lim / X	Local address unit immediate offset or index register
Lmode	Local unit addressing mode
Lrm	Local PP-relative addressing mode
N	Protect status register's negative bit
r	Conditional write of ALU result
reg	Register number used with bank or 0bank for global load, store or address unit arithmetic
s	Enable index scaling. Additional index bit for byte accesses or arithmetic operations (bit 28, local; bit 6, global)
size	Size of data transfer (bits 30-29, local; bits 8-7, global)
src	Three lower bits of register code for register-register move source or non-field moves. D register source for field move
srcbank	Bits 6-3 of register code for register-register move source
V	Protect status register's overflow bit [protects L (latched overflow) also]
Z	Protect status register's zero bit
-	Unused bit (fill with 0)

PP opcode formats (continued)

Table 17 summarizes the supported parallel transfer formats, their formats, and whether the transfers are local or global. It also indicates the allowed ALU operations and whether conditions and status protection are supported.

Table 17. Parallel Transfer Format Summary

	ALU Operands				Global Transfer				Local Transfer			
					Move	Load/Store/AUA			Load/Store/AUA			
Format	dst1	src1	Cond	Status Protection	src → dst	s/d	Index	Rel	s/d	Index	Rel	Port
Double parallel	D	D	No	No	-	Lower	X/short	No	D	X/short	No	Local
Move | Local	D	D	No	No	Any → Any	-	-	-	D	X/short	Yes	Local
Field move | Local	D	D	No	No	D → Any	-	-	-	D	X/short	No	Local
Global (long offset)	D	D	No	No	-	Any	X/long	Yes		-	-	-
Local (long offset)	D	D	No	No	-	-	-	-	Any	X/long	Yes	Global
Non-D DU | Local	Any	Any	No	No	-	-	-	-	D	X/short	Yes	Global
Conditional move	D	D	Yes	Yes	Any → Any	-	-	-		-	-	-
Cond. field move	D	D	Yes	Yes	D → Any	-	-	-	-	-	-	-
Cond. global	D	D	Yes	Yes	-	Any	X/short	Yes	-	-	-	-
Cond. non-D DU	Any	Any	Yes	Yes	-	-	-	-	-	-	-	-
32-bit imm. base ALU	Any	Lower	Yes	No	-							

DU - data unit AUA - address unit arithmetic s/d - source/destination register Rel - relative addressing support

Table 18 shows the encoding used in the opcodes to specify particular PP registers. A 3-bit register field contains the three LSBs. The register codes are used for the src, src1, src2, src3, src4, dst, dst1, dst2, d, reg, Ga, La, Gim/X, and Lim/X opcode fields. The four MSBs specify the register bank which is concatenated to the register field for the full 7-bit code. The register bank codes are used for the dstbank, s1bnk, srcbank, 0bank, bank, Adstbnk, and As1bank opcode fields. When no associated bank is specified for a register field in the opcode, the D register bank is assumed. When the MSB of the bank code is not specified in the opcode (as in 0bank and s1bank) it is assumed to be 0, indicating a lower register.

PP opcode formats (continued)

Table 18. PP Register Codes

Lower Registers (MSB of Bank = 0)					
Coding			Coding		
Bank	Reg	Register	Bank	Reg	Register
0000	000	a0	0100	000	d0
0000	001	a1	0100	001	d1
0000	010	a2	0100	010	d2
0000	011	a3	0100	011	d3
0000	100	a4	0100	100	d4
0000	101	reserved	0100	101	d5
0000	110	a6 (sp)	0100	110	d6
0000	111	a7 (zero)	0100	111	d7
0001	000	a8	0101	000	reserved
0001	001	a9	0101	001	sr
0001	010	a10	0101	010	mf
0001	011	a11	0101	011	reserved
0001	100	a12	0101	100	reserved
0001	101	reserved	0101	101	reserved
0001	110	a14 (sp)	0101	110	reserved
0001	111	a15 (zero)	0101	111	reserved
0010	000	x0	0110	000	reserved
0010	001	x1	0110	001	reserved
0010	010	x2	0110	010	reserved
0010	011	reserved	0110	011	reserved
0010	100	reserved	0110	100	reserved
0010	101	reserved	0110	101	reserved
0010	110	reserved	0110	110	reserved
0010	111	reserved	0110	111	reserved
0011	000	x8	0111	000	pc/call
0011	001	x9	0111	001	ipa/br
0011	010	x10	0111	010	ipe #
0011	011	reserved	0111	011	iprs
0011	100	reserved	0111	100	inten
0011	101	reserved	0111	101	intflg
0011	110	reserved	0111	110	comm
0011	111	reserved	0111	111	lctl

Upper Registers (MSB of Bank = 1)					
Coding			Coding		
Bank	Reg	Register	Bank	Reg	Register
1000	000	reserved	1100	000	lc0
1000	001	reserved	1100	001	lc1
1000	010	reserved	1100	010	lc2
1000	011	reserved	1100	011	reserved
1000	100	reserved	1100	100	lr0
1000	101	reserved	1100	101	lr1
1000	110	reserved	1100	110	lr2
1000	111	reserved	1100	111	reserved
1001	000	reserved	1101	000	lrse0
1001	001	reserved	1101	001	lrse1
1001	010	reserved	1101	010	lrse2
1001	011	reserved	1101	011	reserved
1001	100	reserved	1101	100	lrs0
1001	101	reserved	1101	101	lrs1
1001	110	reserved	1101	110	lrs2
1001	111	reserved	1101	111	reserved
1010	000	reserved	1110	000	ls0
1010	001	reserved	1110	001	ls1
1010	010	reserved	1110	010	ls2
1010	011	reserved	1110	011	reserved
1010	100	reserved	1110	100	le0
1010	101	reserved	1110	101	le1
1010	110	reserved	1110	110	le2
1010	111	reserved	1110	111	reserved
1011	000	reserved	1111	000	reserved
1011	001	reserved	1111	001	reserved
1011	010	reserved	1111	010	reserved
1011	011	reserved	1111	011	reserved
1011	100	reserved	1111	100	tag0 #
1011	101	reserved	1111	101	tag1 #
1011	110	reserved	1111	110	tag2 #
1011	111	reserved	1111	111	tag3 #

read only

data unit operation code

For data unit opcode Format A, a 4-bit operation code specifies one of sixteen six-operand operations and an associated data path. See Table 19 for six-operand operation codes.

Table 19. Six-Operand Format Operation Codes

oper Field Bit				
60	59	58	57	Operation Type
0	u	0	s	MPY || ADD
0	u	1	f	MPY || EALU
1	0	f	k	EALU || ROTATE
1	0	1	0	divi
1	1	u	s	MPY || SADD

u - unsigned
s - subtract
f - 1's complement function code
k - use mask or mf expander

operation class code

The base set ALU opcodes (Formats B, C, D) use an operation class code to specify one of eight different routings to the A, B, and C ports of the ALU. See Table 20.

Table 20. Base Set ALU Class Summary

Class	Destination	A Port	B Port	C Port
0 0 0	dst	src2	src1	@mf
0 0 1	dst	dstc	src1 \\ d0	src2
0 1 0	dst	dstc	src1	%src2
0 1 1	dst	dstc	src1 \\ src2	%src2
1 0 0	dst	src2	src1 \\ d0	%d0
1 0 1	dst	src2	src1 \\ d0	@mf
1 1 0	dst	dstc	src1	src2
1 1 1	dst	src1	1 \\ src2	src2

\\ - rotate left
% - Mask generation
@mf - expand function
dstc - companion D reg.

ALU operation code

For base set ALU Boolean opcodes (A=0), the ALU function is formed by a sum of Boolean products selected by the ALU Operation opcode bits as shown in Table 21. For base set arithmetic opcodes (A=1), the four odd ALU Operation bits specify an arithmetic operation as described in Table 22, while the four even bits specify one of the ALU function modifiers as shown in Table 23.

ALU operation code (continued)

Table 21. Base Set ALU Boolean Function Codes

OPCODE BIT	PRODUCT TERM
58	A & B & C
57	~A & B & C
56	A & ~B & C
55	~A & ~B & C
54	A & B & ~C
53	~A & B & ~C
52	A & ~B & ~C
51	~A & ~B & ~C

Table 22. Base Set Arithmetics

OPCODE BITS				CARRY IN	ALGEBRAIC DESCRIPTION	NATURAL FUNCTION	MODIFIED FUNCTION (IF DIFFERENT FROM NATURAL FUNCTION)
57	55	53	51				
0	0	0	0	x			
0	0	0	1	1	A - (B | C)	A - B <1<	
0	0	1	0	0	A + (B & ~C)	A + B <0<	
0	0	1	1	1	A - C	A - C	
0	1	0	0	1	A - (B | ~C)	A - B >1>	(A - (B & C)) if sign=0
0	1	0	1	1	A - B	A - B	
0	1	1	0	C(n)	A - (B & @mf | -B & ~@mf)	A + B / A - B	if class 0 or 5
				1/0	A + |B|	A + B / A - B	if class 1-4 or 6-7, A-B if sign=1
0	1	1	1	1	A - (B & C)	A - B>0>	
1	0	0	0	0	A + (B & C)	A + B>0>	
1	0	0	1	~C(n)	A + (B & @mf | -B & ~@mf)	A - B / A + B	if class 0 or 5
				0/1	A - |B|	A - B / A + B	if class 1-4 or 6-7, A+B if sign=1
1	0	1	0	0	A + B	A + B	
1	0	1	1	0	A + (B | ~C)	A + B >1>	(A + (B & C)) if sign=0
1	1	0	0	0	A + C	A + C	
1	1	0	1	1	A - (B & ~C)	A - B <0<	
1	1	1	0	0	A + (B | C)	A + B <1<	
1	1	1	1	0	(A & C) + (B & C)	field A + B	

C(n) - LSB of each part of C port register
<0< - zero-extend shift left
<1< - one-extend shift left
>0> - zero-extend shift right
>1> - one-extend shift right

ALU operation code (continued)

Table 23. Function Modifier Codes

FUNCTION MODIFIER BITS				MODIFICATION PERFORMED
58	56	54	52	
0	0	0	0	Normal operation
0	0	0	1	cin
0	0	1	0	%! if maskgen instruction, lmo if not maskgen
0	0	1	1	%! and cin if maskgen instruction, rmo if not maskgen
0	1	0	0	A port = 0
0	1	0	1	A port = 0 and cin
0	1	1	0	A port = 0 and %! if maskgen, lmbc if not maskgen
0	1	1	1	A port = 0, %! and cin if maskgen, rmbc if not maskgen
1	0	0	0	mf bit(s) set by carry out(s). (mc)
1	0	0	1	mf bit(s) set based on status register MSS field. (me)
1	0	1	0	Rotate mf by Asize, mf bit(s) set by carry out(s). (mrc)
1	0	1	1	Rotate mf by Asize, mf bit(s) set based on status register MSS field. (mre)
1	1	0	0	Clear mf, mf bit(s) set by carry out(s). (mzc)
1	1	0	1	Clear mf, mf bit(s) set based on status register MSS field. (mze)
1	1	1	0	No setting of bits in mf register. (mx)
1	1	1	1	Reserved

cin - carry in
%! - modified mask generator
lmo - leftmost one
rmo - rightmost one
lmbc - leftmost bit change
rmbc - rightmost bit change

miscellaneous operation code

For data unit opcode Format E, the Operation field selects one of the miscellaneous operations.

Table 24. Miscellaneous Operation Codes

OPCODE BITS					MNEMONIC	OPERATION
43	42	41	40	39		
0	0	0	0	0	nop	No data unit operation. Status not modified.
0	0	0	0	1	qwait	Wait until comm Q bit is clear
0	0	0	1	0	eint	Global interrupt enable
0	0	0	1	1	dint	Global interrupt disable
0	0	1	0	0	eloop	Global loop enable
0	0	1	0	1	dloop	Global loop disable
0	0	1	1	x	reserved	
0	1	x	x	x	reserved	
1	x	x	x	x	reserved	

addressing mode codes

The Lmode (bits 35-38) and Gmode (bits 13-16) of the opcode specify the local and global transfer for various parallel transfer opcode formats (Lmode in formats 1,2,3,4, and 6 and Gmode in formats 1,5, and 9). The coding for the addressing mode fields is shown in Table 25.

Table 25. Addressing Mode Codes

CODING	EXPRESSION	DESCRIPTION
00xx		Nop (nonaddressing mode operation)
0100	*(an ++= xm)	Postaddition of index register, with modify
0101	*(an --= xm)	Postsubtraction of index register, with modify
0110	*(an ++= imm)	Postaddition of immediate, with modify
0111	*(an --= imm)	Postsubtraction of immediate, with modify
1000	*(an + xm)	Preaddition of index register
1001	*(an - xm)	Presubtraction of index register
1010	*(an + imm)	Preaddition of immediate
1011	*(an - imm)	Presubtraction of immediate
1100	*(an += xm)	Preaddition of index register, with modify
1101	*(an -= xm)	Presubtraction of index register, with modify
1110	*(an += imm)	Preaddition of immediate, with modify
1111	*(an -= imm)	Presubtraction of immediate, with modify

an - address register in l/g address unit imm - immediate offset
xm - index register in same unit as an register

L, e codes

The L and e bits combine to specify the type of parallel transfer performed. For the local transfer, L and e are bits 21 and 31, respectively. For the global transfer, L and e are bits 17 and 9, respectively.

Table 26. Parallel Transfer Type

L	e	PARALLEL TRANSFER
1	0	Zero-extend load
1	1	Sign-extend load
0	0	Store
0	1	Address unit arithmetic

size codes

The size code specifies the data transfer size. For field moves (parallel transfer Format 3), only byte and halfword data sizes are valid. See Table 27.

Table 27. Transfer Data Size

CODING	DATA SIZE
00	Byte (8 bits)
01	Halfword (16 bits)
10	Word (32 bits)
11	Reserved

relative addressing mode codes

The Lrm and Grm opcode fields allow the local address or global address units, respectively to select PP-relative addressing as shown in Table 28.

Table 28. Relative Addressing Mode Codes

CODING	RELATIVE ADDRESSING MODE
00	Normal (absolute addressing)
01	Reserved
10	PP-relative dba
11	PP-relative pba

dba - Data RAM 0 base is base address
pba - Parameter RAM base is base address

condition codes

In the four conditional parallel transfer opcodes (Formats 7-10), this field specifies one of sixteen condition codes to be applied to the data unit operation source, data unit result, or global transfer based on the setting of the c, r, and g bits, respectively. The condition codes are shown in Table 29. For the 32-bit immediate data unit opcode (Format D), the condition applies to the data unit result only.

Table 29. Condition Codes

CONDITION BITS				MNEMONIC	DESCRIPTION	STATUS BIT COMBINATION
35	34	33	32			
0	0	0	0	u	Unconditional (default)	None
0	0	0	1	p	Positive	~N & ~Z
0	0	1	0	ls	Lower than or same	~C | Z
0	0	1	1	hi	Higher than	C & ~Z
0	1	0	0	lt	Less than	(N & ~V) | (~N & V)
0	1	0	1	le	Less than or equal	(N & ~V) | (~N & V) | Z
0	1	1	0	ge	Greater than or equal	(N & V) | (~N & ~V)
0	1	1	1	gt	Greater than	(N & V & ~Z) | (~N & ~V & ~Z)
1	0	0	0	hs, c	Higher than or same, carry	C
1	0	0	1	lo, nc	Lower than, no carry	~C
1	0	1	0	eq, z	Equal, zero	Z
1	0	1	1	ne, nz	Not equal, not zero	~Z
1	1	0	0	v	Overflow	V
1	1	0	1	nv	No overflow	~V
1	1	1	0	n	Negative	N
1	1	1	1	nn	Nonnegative	~N

EALU operations

Extended ALU (EALU) operations allow the execution of more advanced ALU functions than those specified in the base set ALU opcodes. The opcode for EALU instructions contains the operands for the operation while the d0 register extends the opcode by specifying the EALU operation to be performed. The format of d0 for EALU operations is shown in Figure 24.

EALU Boolean functions

EALU operations support all 256 Boolean ALU functions plus the flexibility to add 1 or a carry-in to Boolean sum. The Boolean function performed by the ALU is:

(F0 & (~A & ~B & ~C)) | F1 & (A & ~B & ~C) | F2 & (~A & B & ~C) | F3 & (A & B & ~C) | F4 & (~A & ~B & C) | F5 & (A & ~B & C) | F6 & (~A & B & C) | F7 & (A & B & C)) [+1 | +cin]

Table 30. EALU Boolean Function Codes

d0 BIT	ALU FUNCTION SIGNAL	PRODUCT TERM
26	F7	A & B & C
25	F6	~A & B & C
24	F5	A & ~B & C
23	F4	~A & ~B & C
22	F3	A & B & ~C
21	F2	~A & B & ~C
20	F1	A & ~B & ~C
19	F0	~A & ~B & ~C

EALU arithmetic functions

EALU operations support all 256 arithmetic functions provided by the three-input ALU plus the flexibility to add 1 or a carry-in to the result. The arithmetic function performed by the ALU is:

$$f(A,B,C) = A \,\&\, f1(B,C) + f2(B,C) \; [+1 \mid cin]$$

f1(B,C) and f2(B,C) are independent Boolean combinations of the B and C ALU inputs. The ALU function is specified by selecting the desired f1 and f2 subfunction and then XORing the f1 and f2 code from Table 31 to create the ALU function code for bits 19-26 of d0. Additional operations such as absolute values and signed shifts can be performed using d0 bits which control the ALU function based on the sign of one of the inputs.

EALU arithmetic functions (continued)

Table 31. ALU f1(B,C) and f2(B,C) Subfunctions

f1 CODE	f2 CODE	SUBFUNCTION	COMMON USAGE
00	00	0	Zero the term
AA	FF	-1	-1 (All 1s)
88	CC	B	B
22	33	-B -1	Negate B
A0	F0	C	C
0A	0F	-C -1	Negate C
80	C0	B & C	Force bits in B to 0 where bits in C are 0
2A	3F	-(B & C) - 1	Force bits in B to 0 where bits in C are 0 and negate
A8	FC	B | C	Force bits in B to 1 where bits in C are 1
02	03	-(B | C) - 1	Force bits in B to 1 where bits in C are 1 and negate
08	0C	B & ~C	Force bits in B to 0 where bits in C are 1
A2	F3	-(B & ~C) -1	Force bits in B to 0 where bits in C are 1 and negate
8A	CF	B | ~C	Force bits in B to 1 where bits in C are 0
20	30	-(B | ~C) -1	Force bits in B to 1 where bits in C are 0 and negate
28	3C	(B & ~C) | ((-B - 1) & C)	Choose B if C = all 0s and -B if C = all 1s
82	C3	(B & C) | ((-B - 1) & ~C)	Choose B if C = all 1s and -B if C = all 0s

transfer controller architecture

The transfer controller (TC) is a combined memory controller and DMA (direct memory access) machine. It handles the movement of data within the ‘C82 system as requested by the master processor, parallel processors, and external devices. The transfer controller performs the following data movement and memory control functions:

- MP and PP instruction cache fills
- MP data cache fills and dirty block write-back
- MP and PP direct external accesses (DEAs)
- MP and PP packet transfers
- Externally initiated packet transfers (XPTs)
- Shift register transfer (SRT) packet transfers for updating VRAM-based frame buffers
- DRAM/SDRAM refresh
- Host bus request

TC functional block diagram

A functional block diagram of the transfer controller is shown in Figure 47. Key features of the TC include:

- Crossbar Interface
  - 64-bit data path
  - Single-cycle access
- External Memory Interface
  - 4 GByte address range

TC functional block diagram (continued)

- Internal memory configuration-cache stores up to six sets of bank information. Features programmable:
  - bus size : 8, 16, 32, or 64 bits
  - page size
  - bank size
  - address multiplexing
  - cycle timing
  - block-write mode
  - bank priority
- Big or little endian operation

- Cache, VRAM, refresh controller
  - Programmable refresh rate
  - VRAM block write support
- Independent Src and Dst addressing
  - Autonomous addressing based on packet transfer parameters
  - Data read and write at different rates
  - Numerous data merging and alignment functions performed during transfer
- Intelligent request prioritization

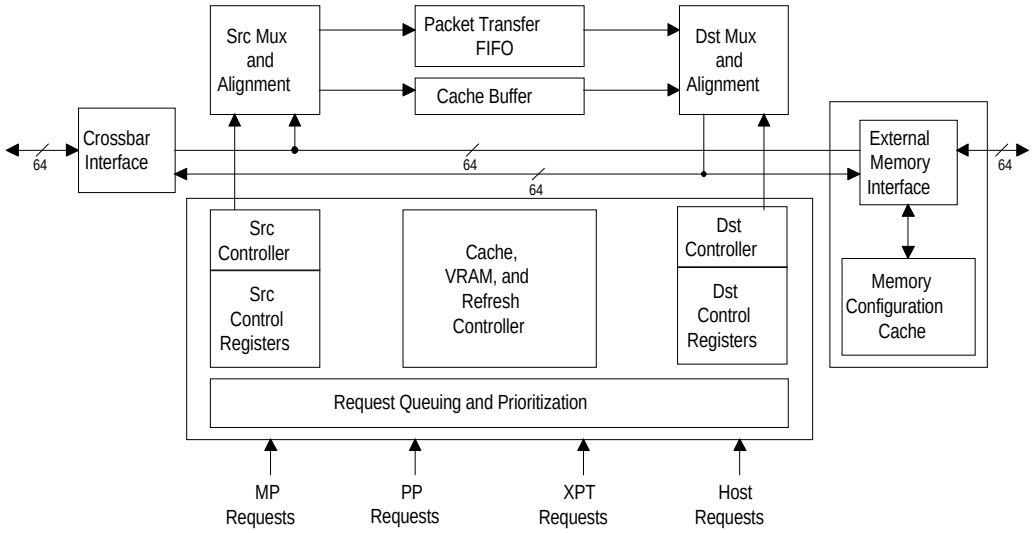


Figure 47. TC Block Diagram

transfer controller registers

The TC contains four on-chip memory-mapped registers accessible by the MP.

REFCNTL register (0x01820000)

The REFCNTL register controls refresh cycles.

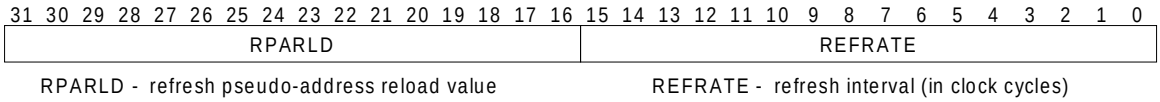


Figure 48. REFCNTL Register

PTMIN register (0x01820004)

The PTMIN register determines the minimum number of cycles that a packet transfer will execute before being suspended by a higher-priority packet transfer. Short-form XPTs may interrupt long-form PTs without suspending them.

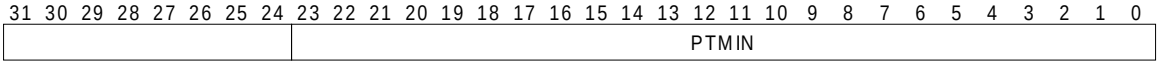


Figure 49. PTMIN Register

PTMAX register (0x01820008)

The PTMAX register determines the maximum number of cycles after PTMIN has elapsed that a packet transfer will execute before timing out.

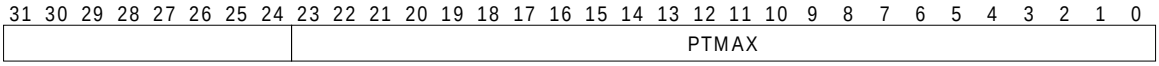


Figure 50. PTMAX Register

FLTSTS register (0x0182000C)

The FLTSTS register indicates the cause of a memory-access fault. Fault status bits are cleared by writing a 1 to the appropriate bit.

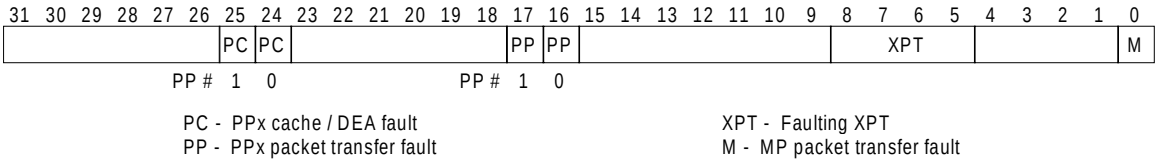


Figure 51. FLTSTS Register

packet transfer parameters

The most efficient method for data movement in a TMS320C82 system is through the use of packet transfers (PTs). Packet transfers allow the TC to autonomously move blocks of data between a specified src and dst memory region. Requests for the TC to execute a packet transfer may be made by the MP, PPs, or external devices. A packet transfer parameter table describing the data packet and how it is to be transferred must be programmed in on-chip memory before the transfer is requested. The TC on the TMS320C82 supports short- and long-form packet transfers. The PT parameter tables for both formats are shown in Figure 52 and Figure 53.

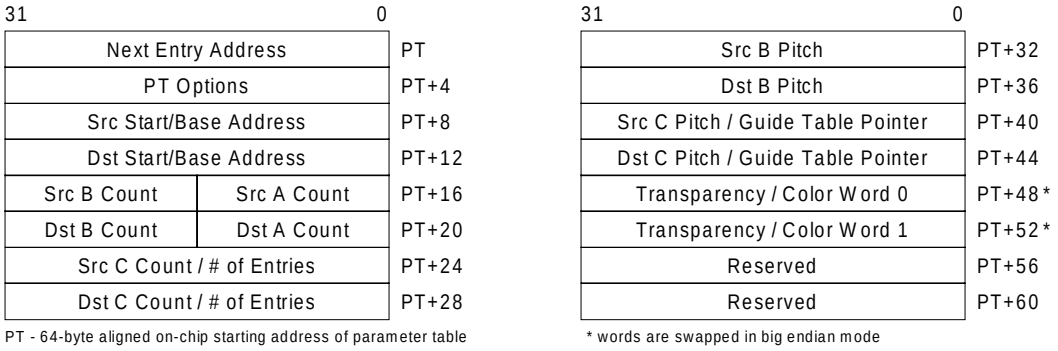


Figure 52. Packet Transfer Parameter Table - Long Form

packet transfer parameters (continued)

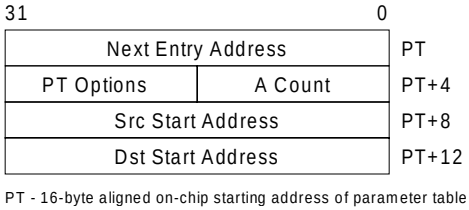


Figure 53. Packet Transfer Parameter Table - Short Form

PT options field

The PT Options field of the parameter table controls the type of Src and Dst transfer that the TC performs. The format of the options field is shown in Figure 54.

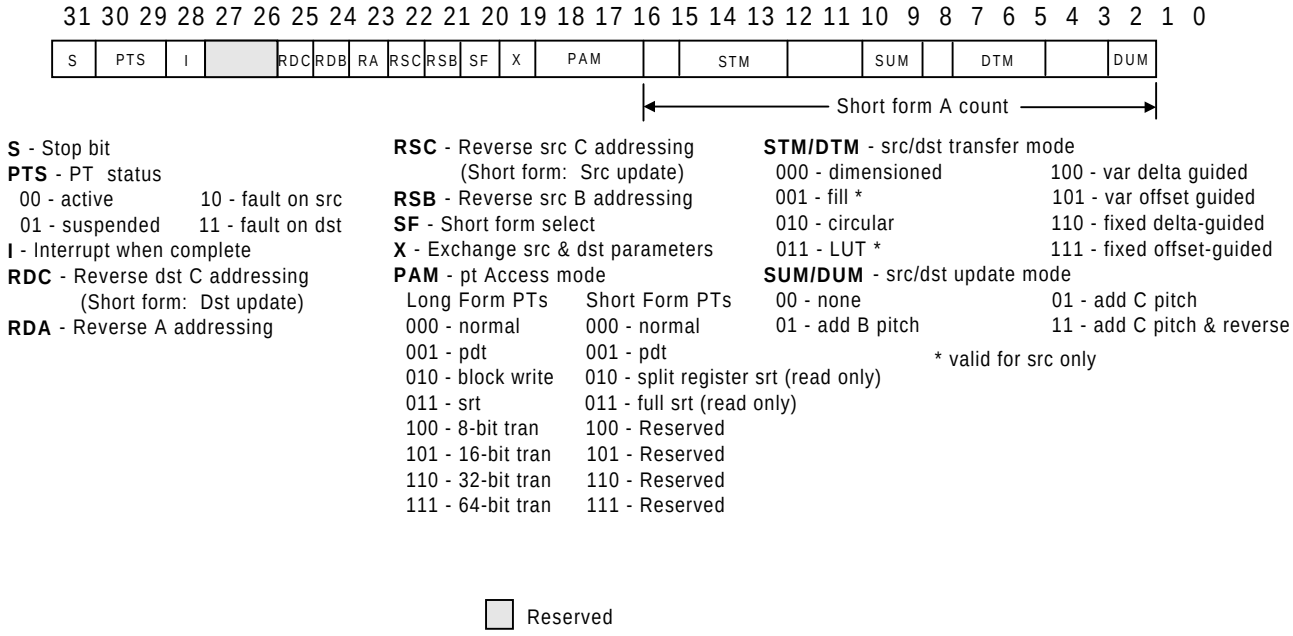


Figure 54. PT Options Field

status codes

The TMS320C82 outputs status information on two busses which describe the type of cycle being performed. During row time, status codes are output on AD[39:32] and STATUS[1:0]. The cycle type may be latched using /RL and used by external logic to perform memory bank decoding or enable special hardware features. The STATUS[1:0] pins indicate idle cycles and ending XPT accesses.

Table 32. Row Time Status Codes (AD[39:32])

AD[39:36]	SOURCE	AD[35:32]	ACTIVITY
0 0 0 0	'C82	0 0 0 0	Read
0 0 0 1	XPT1	0 0 0 1	Write
0 0 1 0	XPT2	0 0 1 0	PDT Read
0 0 1 1	XPT3	0 0 1 1	PDT Write
0 1 0 0	XPT4	0 1 0 0	PT Full SRT Read
0 1 0 1	XPT5	0 1 0 1	Reserved
0 1 1 0	XPT6	0 1 1 0	PT Split SRT Read
0 1 1 1	XPT7	0 1 1 1	Reserved
1 0 0 0	XPT8	1 0 0 0	SDRAM MRS
1 0 0 1	XPT9	1 0 0 1	Block Write
1 0 1 0	XPTa	1 0 1 0	Reserved
1 0 1 1	XPTb	1 0 1 1	Load Color Register
1 1 0 0	XPTc	1 1 0 0	Refresh
1 1 0 1	XPTd	1 1 0 1	SDRAM DCAB
1 1 1 0	XPTe	1 1 1 0	Bank Configuration
1 1 1 1	XPTf	1 1 1 1	Idle

Table 33. Memory Cycle Status

STATUS[1:0]	ACTIVITY
0 0	Idle/DCAB/Drain
0 1	Row access
1 0	XPT end
1 1	Column access

memory bank configuration

Before an access can begin, the ‘C82 must know certain information about the memory bank it is addressing. Information about the bus size, address shifting, memory speed, and bank size is read in during a special configuration cycle and stored in an on-chip memory configuration cache. The ‘C82 memory configuration cache holds up to six entries, which are maintained with a multi-priority level least recently used algorithm. The memory configuration cache is only accessible by the TC; software may not modify the contents of the cache. The bank configuration fields are read in over the AD bus in four consecutive byte-wide reads, and have the format shown in Figure 55.

RCA[1:0]	AD[7:0] or AD[63:56] 7 6 5 4 3 2 1 0	FUNCTION GROUP
0 0	E | TO | - | CT	cycle timing
0 1	PS | B | AS	address control
1 0	R | - | BW | WS | BS	bus size
1 1	- | PL | MS	bank properties

Figure 55. Bank Configuration Cycle Fields

memory exceptions

Retry and fault conditions are encoded on the /EXCEPT[1:0] pins on the ‘C82. Additionally, a configuration cache flush may be requested over these pins. The /EXCEPT[1:0] codes are shown in Table 34.

Table 34. Memory Exception Codes

/EXCEPT[1:0]		MEMORY EXCEPTION
0	0	Configuration cache flush
0	1	Fault
1	0	Retry or page request
1	1	None

Support for exceptions is not mandatory. Exception support may be enabled or disabled on a bank-by-bank basis. During the bank configuration cycle, if the E bit is set to one, exception support will be enabled for that particular bank. When E is set to 0, except codes of 00 and 01 are ignored during accesses to that bank. Page requests are not sensitive to the E bit.

read turn around

Data is driven by, and read into, the ‘C82 over the AD[63:0] bus. Additionally, at row time address and status information is output over this bus. Because of this, the potential exists for a drive conflict, particularly when reading from slow devices such as EPROMs. To compensate for this, extra cycles may be added to the end of read bursts to allow memories and drivers to turn off. Similarly, PDT write cycles contain the minimum number of turnoff cycles as their PDT read cycle counterparts. The number of turnoff cycles is controlled by the TO field in the bank configuration cache entry. The TO encodings are shown in Table 35.

Table 35. TO (Turnoff) Cycle Encoding

TO(1:0)		EXTRA TURNOFF CYCLES
0	0	None
0	1	1
1	0	2
1	1	3

cycle time selection

The ‘C82 supports fourteen sets of memory timings to interface with various memory types. The cycle timing is selected by the value input in the CT(3:0) field during a bank configuration cycle. The selected timing remains in effect for all accesses made to that bank while its configuration is in cache.

Table 36. Cycle Timing Selection

CT(3:0)				CYCLE TIMING
0	0	0	0	DRAM: pipelined 1 cycle/column EDO
0	0	0	1	DRAM: unpipelined 1 cycle/column EDO
0	0	1	0	DRAM: unpipelined 2 cycle/column EDO
0	0	1	1	DRAM: unpipelined 3 cycle/column EDO
0	1	0	0	SRAM: synchronous 1 cycle/column
0	1	0	1	SRAM: asynchronous 1 cycle/column
0	1	1	0	SRAM: asynchronous 2 cycle/column
0	1	1	1	SRAM: asynchronous 3 cycle/column
1	0	0	0	SDRAM: burst length 1; CAS latency 2
1	0	0	1	SDRAM: burst length 1; CAS latency 3
1	0	1	0	SDRAM: burst length 1; CAS latency 4
1	0	1	1	Reserved
1	1	0	0	SDRAM: burst length 2; CAS latency 2
1	1	0	1	SDRAM: burst length 2; CAS latency 3
1	1	1	0	SDRAM: burst length 2; CAS latency 4
1	1	1	1	Reserved

page sizing

Whenever the ‘C82 performs an external access, it must track the current page boundary of the addressed bank. When a page boundary is crossed, a new row access must be performed. This is accomplished by comparing certain bits of the logical address bus. Because the location of the logical address on the RCA[16:0] bus is dependent on bus size (controlled by BS(1:0)), page size is also affected by the BS(1:0) inputs. Table 37 outlines the effective page sizes as a function of PS(3:0) and BS(1:0).

Table 37. Page Size

PS(3:0)	BS(1:0)	LOGICAL ADDRESS BITS COMPARED	PAGE SIZE
0 0 0 0	0 0	None	1 byte
	0 1	None	2 bytes
	1 0	None	4 bytes
	1 1	None	8 bytes
0 0 0 1	0 0	31:3	8 bytes
	0 1	31:4	16 bytes
	1 0	31:5	32 bytes
	1 1	31:6	64 bytes
0 0 1 0	0 0	31:4	16 bytes
	0 1	31:5	32 bytes
	1 0	31:6	64 bytes
	1 1	31:7	128 bytes
0 0 1 1	0 0	31:5	32 bytes
	0 1	31:6	64 bytes
	1 0	31:7	128 bytes
	1 1	31:8	256 bytes
0 1 0 0	0 0	31:6	64 bytes
	0 1	31:7	128 bytes
	1 0	31:8	256 bytes
	1 1	31:9	512 bytes
0 1 0 1	0 0	31:7	128 bytes
	0 1	31:8	256 bytes
	1 0	31:9	512 bytes
	1 1	31:10	1K byte
0 1 1 0	0 0	31:8	256 bytes
	0 1	31:9	512 bytes
	1 0	31:10	1K byte
	1 1	31:11	2K bytes
0 1 1 1	0 0	31:9	512 bytes
	0 1	31:10	1K byte
	1 0	31:11	2K bytes
	1 1	31:12	4K bytes
1 0 0 0	0 0	31:10	1K byte
	0 1	31:11	2K bytes
	1 0	31:12	4K bytes
	1 1	31:13	8K bytes
1 0 0 1	0 0	31:11	2K bytes
	0 1	31:12	4K bytes
	1 0	31:13	8K bytes
	1 1	31:14	16K bytes
1 0 1 0	0 0	31:12	4K bytes
	0 1	31:13	8K bytes
	1 0	31:14	16K bytes
	1 1	31:15	32K bytes
1 0 1 1	0 0	31:13	8K bytes
	0 1	31:14	16K bytes
	1 0	31:15	32K bytes
	1 1	31:16	64K bytes
1 1 0 0	0 0	31:14	16K bytes
	0 1	31:15	32K bytes
	1 0	31:16	64K bytes
	1 1	31:17	128K bytes
1 1 0 1	0 0	31:15	32K bytes
	0 1	31:16	64K bytes
	1 0	31:17	128K bytes
	1 1	31:18	256K bytes
1 1 1 0	0 0	31:16	64K bytes
	0 1	31:17	128K bytes
	1 0	31:18	256K bytes
	1 1	31:19	512K bytes
1 1 1 1	0 0	31:17	128K bytes
	0 1	31:18	256K bytes
	1 0	31:19	512K bytes
	1 1	31:20	1M byte

address multiplexing

In order to support various RAM devices, the TMS320C82 provides multiplexed row and column addresses on the RCA bus. A full 32-bit address is always output on AD[31:0] at row time. This value can be latched with /RL to provide bank-decoding. The actual address lines to memory should be connected to the RCA bus.

In order to support a wide variety of memory types, the logical address output on RCA is a function of both bus size and address shift. The row-column address multiplexing on the RCA bus is shown in Table 39. Memories and peripherals should be physically connected to RCA[N:0], where N is the number of address pins on the device. As the logical address bits output on the RCA bus is a function of both address shift and bus size, this connection is valid regardless of the memory architecture.

When performing peripheral device packet transfers, the lower bits (logical address bits 0, 1, and 2) may be lost depending on the configuration of the memory bank. To compensate for this, the B bit may be set for memory banks which must support PDTs. When set, the TC will place the missing lower address bits on the upper bits of the RCA bus during PDT accesses. External logic may decode these bits to access the byte information. The bit replacement functionality is shown in Table 38. Note that the row address output on RCA[16:0] is unaffected.

Table 38. PDT Address Bit Replacement (B=1)

BUS	LOGICAL ADDRESS BITS OUTPUT ON RCA[16:0] DURING COLUMN TIME																
WIDTH	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1 byte	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2 bytes	0	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
4 bytes	1	0	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
8 bytes	2	1	0	16	15	14	13	12	11	10	9	8	7	6	5	4	3

address multiplexing (continued)

Table 39. RCA Address Multiplexing (B = 0)

Cycle	AS[2:0]	BS[1:0]	Logical address bits output on RCA (16:0)																	
			16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	1 1 1	1 1	x	x	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
Col			19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
Row	1 1 1	1 0	x	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Col			18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
Row	1 1 1	0 1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15
Col			17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Row	1 1 1	0 0	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14
Col			16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	1 1 0	1 1	x	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Col			19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
Row	1 1 0	1 0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15
Col			18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
Row	1 1 0	0 1	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14
Col			17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Row	1 1 0	0 0	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
Col			16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	1 0 1	1 1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15
Col			19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
Row	1 0 1	1 0	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14
Col			18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
Row	1 0 1	0 1	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
Col			17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Row	1 0 1	0 0	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
Col			16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	1 0 0	1 1	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14
Col			19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
Row	1 0 0	1 0	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
Col			18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
Row	1 0 0	0 1	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
Col			17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Row	1 0 0	0 0	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
Col			16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	0 1 1	1 1	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13
Col			19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
Row	0 1 1	1 0	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
Col			18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
Row	0 1 1	0 1	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
Col			17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Row	0 1 1	0 0	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
Col			16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	0 1 0	1 1	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
Col			19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
Row	0 1 0	1 0	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
Col			18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
Row	0 1 0	0 1	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
Col			17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Row	0 1 0	0 0	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
Col			16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	0 0 1	1 1	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
Col			19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
Row	0 0 1	1 0	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
Col			18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
Row	0 0 1	0 1	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9
Col			17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Row	0 0 1	0 0	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
Col			16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Row	0 0 0	1 1	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
Col			19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3
Row	0 0 0	1 0	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
Col			18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
Row	0 0 0	0 1	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Col			17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Row	0 0 0	0 0	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Col			16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

user-defined wait states

Some memory architectures (most notably DRAM and SDRAM) may require wait states to be added to the ‘C82’s external interface. For memories and peripherals requiring a larger row time for decoding purposes, the R bit may be set in the configuration cache entry for those memory banks which will cause the ‘C82 to automatically insert a single additional state into row time of the external memory cycle. For DRAM and SRAM accesses, the additional state is inserted between ad2 and rl1 (with /RAS high). For SDRAM accesses, the state is inserted between ac2 and the column pipeline (to increase the time between ACTV and a READ/WRT command).

Table 40. Row (R) Time Wait State Selection

MEMORY TYPE	R	ADDITIONAL CYCLE INSERTED
CT=00xx (EDO DRAM)	0	None
	1	1 between ad2 and rl1 (/RAS high)
CT=01xx (SRAM)	0	None
	1	1 between ad2 and rl1 (/RAS high)
CT=1xxx (SDRAM)	0	None
	1	1 between ac2 and first column access

Additionally, the WS field in the configuration cache entry for each memory bank allows the user to automatically insert a predefined number of wait states into the column time pipeline without using the READY input. This significantly enhances the ‘C82s ability to interface with slower peripherals at higher clock rates. When WS(1:0) is set to a nonzero value, wait states will be inserted into the column time pipeline. If the READY signal is asserted, then additional wait states will be inserted as per the normal sampling mechanism of the READY input (during the final cycle of each column access).

Similar to the READY input, the WS(1:0) field should only be set to a nonzero value for 2 and 3 cyc/col accesses. Wait states inserted due to this field will output a status code of 11 (column time) on STATUS[1:0] so that the system may differentiate between the default wait states and pipeline bubbles.

Table 41. WS (Wait State) Encoding

MEMORY TYPE	WS[1:0]	ADDITIONAL CYCLES INSERTED (REFERENCES /CAS/DQM)
CT=001x (EDO DRAM)	00	None
	01	1 low
	10	1 low, 1 high
	11	2 low, 1 high
CT=011x (SRAM)	00	None
	01	1 low
	10	2 low
	11	3 low

Note: The WS(1:0) field should be set to 00 for single-cycle memory banks.

block write support

The TMS320C82 supports three modes of VRAM block write. The block-write mode is selectable so that software may specify block writes without knowing what type of block-write the addressed memory supports. Block writes are only supported for 64-bit busses. During block-write and load-color-register cycles, the block-write mode specified by the BW(1:0) bits of the bank configuration block will be used. Table 42 lists the block-write modes associated with the BW(1:0) bits.

Table 42. Block Write Selection

BW(1:0)	BLOCK-WRITE MODE
00	Simulated
01	Reserved
10	4x
11	8x

bus sizing

The ‘C82 supports data bus sizes of 8, 16, 32, or 64 bits. The value input in the BS(1:0) field of the bank-configuration cycle indicates the bus size of the addressed memory. This determines the maximum number of bytes that the ‘C82 can transfer during each column access. If the number of bytes to be transferred exceeds the bus size, multiple accesses will automatically be performed to complete the transfer.

Table 43. Bus Size Selection

BS(1:0)	BUS SIZE
00	8 bits
01	16 bits
10	32 bits
11	64 bits

The selected bus size also determines which portion of the data bus will be used for the transfer. For 64-bit memory, the entire data bus is used. For 32-bit memory, AD[31:0] are used in little-endian mode and AD[63:32] are used in big-endian mode. 16-bit busses use AD[15:0] and AD[63:48]; and 8-bit busses use AD[7:0] and AD[63:56] for little- and big-endian, respectively. The ‘C82 always aligns data to the proper portion of the bus and activates the appropriate /CAS/DQM strobes. During read cycles, all /CAS/DQM strobes will be active (low). During write cycles, only those /CAS/DQM strobes corresponding to bytes actually being written will be activated.

cache priority level

The ‘C82 memory configuration cache can contain up to six entries. Once full, memory accesses to unconfigured banks require that one of the entries in cache be flushed and a configuration cycle for the desired bank be performed. The cache uses a multilevel, least-recently-used algorithm to determine which entry to flush. Because certain entries (i.e., code space, data space) may pertain to more time-critical functions than others, each of the six entries in the cache can be assigned a priority of high, medium, or low; indicated by the PL(1:0) bits of the configuration cycle fields for that bank. The LRU algorithm is implemented for each priority level separately; that is, when it becomes necessary to discard an entry, the least recently used entry of lowest priority present is discarded. The priority level encoding is shown below.

Table 44. Cache Priority Levels

PL(1:0)		PRIORITY LEVEL
0	0	Low
0	1	Medium
1	0	Reserved
1	1	High

bank size

Since many memory types may require the same configuration, the bank configuration contains a memory-bank-size field called MS(4:0). This field specifies the size of the memory bank to which the cache entry pertains, and thus which address bits should be compared to determine if a cache miss has occurred. The MS(4:0) codings are shown in Table 45.

Table 45. Memory Bank Size

MS(4:0)	ADDRESS BITS COMPARED	BANK SIZE
0 0 0 0 0	None	4G bytes
0 0 0 0 1	31	2G bytes
0 0 0 1 0	31:30	1G byte
0 0 0 1 1	31:29	512M bytes
0 0 1 0 0	31:28	256M bytes
0 0 1 0 1	31:27	128M bytes
0 0 1 1 0	31:26	64M bytes
0 0 1 1 1	31:25	32M bytes
0 1 0 0 0	31:24	16M bytes
0 1 0 0 1	31:23	8M bytes
0 1 0 1 0	31:22	4M bytes
0 1 0 1 1	31:21	2M bytes
0 1 1 0 0	31:20	1M byte
0 1 1 0 1	31:19	512K bytes
0 1 1 1 0	31:18	256K bytes
0 1 1 1 1	31:17	128K bytes
1 0 0 0 0	31:16	64K bytes
1 0 0 0 1	31:15	32K bytes
1 0 0 1 0	31:14	16K bytes
1 0 0 1 1	31:13	8K bytes
1 0 1 0 0	31:12	4K bytes
1 0 1 0 1	31:11	2K bytes
1 0 1 1 0	31:10	1K bytes
1 0 1 1 1	31:9	512 bytes
1 1 0 0 0	31:8	256 bytes
1 1 0 0 1	31:7	128 bytes
1 1 0 1 0	31:6	64 bytes
1 1 0 1 1	31:5	32 bytes
1 1 1 0 0	31:4	16 bytes
1 1 1 0 1	31:3	8 bytes
1 1 1 1 0	31:2	4 bytes
1 1 1 1 1	31:1	2 bytes

refresh controller

The ‘C82 has an on-chip refresh controller that schedules refresh cycles to be performed by the TC. Refresh rate is programmable via the TC’s REFCNTL register. A refresh pseudo-address is output on AD[16:1] during refreshes, which may be used for bank-decoding. The refresh pseudo-address is decremented once for each refresh cycle that is performed. When it decrements to 0, it is reloaded with the value in the upper 16 bits of the REFCNTL register.

refresh controller (continued)

A refresh cycle is indicated by the status code 0x00001100 on AD[39:32] at row time. During a refresh cycle, information input on /EXCEPT[1:0] tells the TC what type of refresh cycle to perform. The /EXCEPT[1:0] encoding for refresh cycles is shown in Table 46. A retried refresh cycle is immediately terminated, and the refresh pseudo-address is not decremented.

Table 46. Refresh Cycles

/EXCEPT[1:0]		REFRESH MODE
0	0	DRAM (3 cycles /RAS high)
0	1	DRAM (4 cycles /RAS high)
1	0	Retry
1	1	SDRAM

SDRAM support

The TMS320C82 provides direct support for synchronous DRAM (SDRAM) and graphics RAM (SGRAM). During ‘C82 power-up refresh cycles, the external system must signal the presence of these memories by inputting an /EXCEPT[1:0] code of 11. This causes the ‘C82 to perform an SDRAM deactivate (DCAB) command.

Additionally, the ‘C82 will perform an SDRAM mode register set (MRS) cycle following a memory bank configuration cycle if that cycle specifies an SDRAM cycle timing code. No further MRS cycles will be performed for that bank as long as it remains in cache.

The MRS cycle is required to initialize the SDRAM for operation. Information about the burst length and read latency (CAS latency) is input to the SDRAM via its address inputs. The MRS value generated by the ‘C82 is shown in Figure 56. It should be noted that CAS latency four reads are intended for use with CAS latency three SDRAMs, and thus the MRS cycle is performed as such.

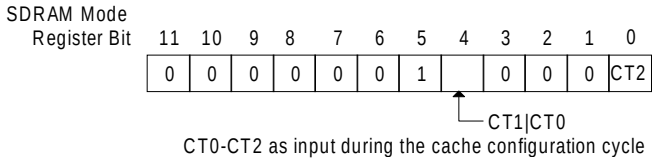


Figure 56. MRS Value

Because the MRS register is programmed through the SDRAM address inputs, the alignment of the MRS data to the ‘C82 logical address bits is adjusted for the bus size as shown in Figure 57. The appearance of the MRS bits on the ‘C82 physical address bus (RCA[16:0]) is dependent on the address multiplexing as selected by the AS(2:0) field.

	C82 Logical Address Bits															
BS(1:0)	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0 0	X	X	X	X	11	10	9	8	7	6	5	4	3	2	1	0
0 1	X	X	X	11	10	9	8	7	6	5	4	3	2	1	0	X
1 0	X	X	11	10	9	8	7	6	5	4	3	2	1	0	X	X
1 1	X	11	10	9	8	7	6	5	4	3	2	1	0	X	X	X

Figure 57. MRS Value Alignment

memory cycles

TMS320C82 external memory cycles are generated by the TC's external memory controller. The controller's state machine generates a sequence of states which define the transition of the memory interface signals. The state sequence is dependent on the cycle timing selected for the bank being accessed. Memory cycles consist of row states and the column pipeline.

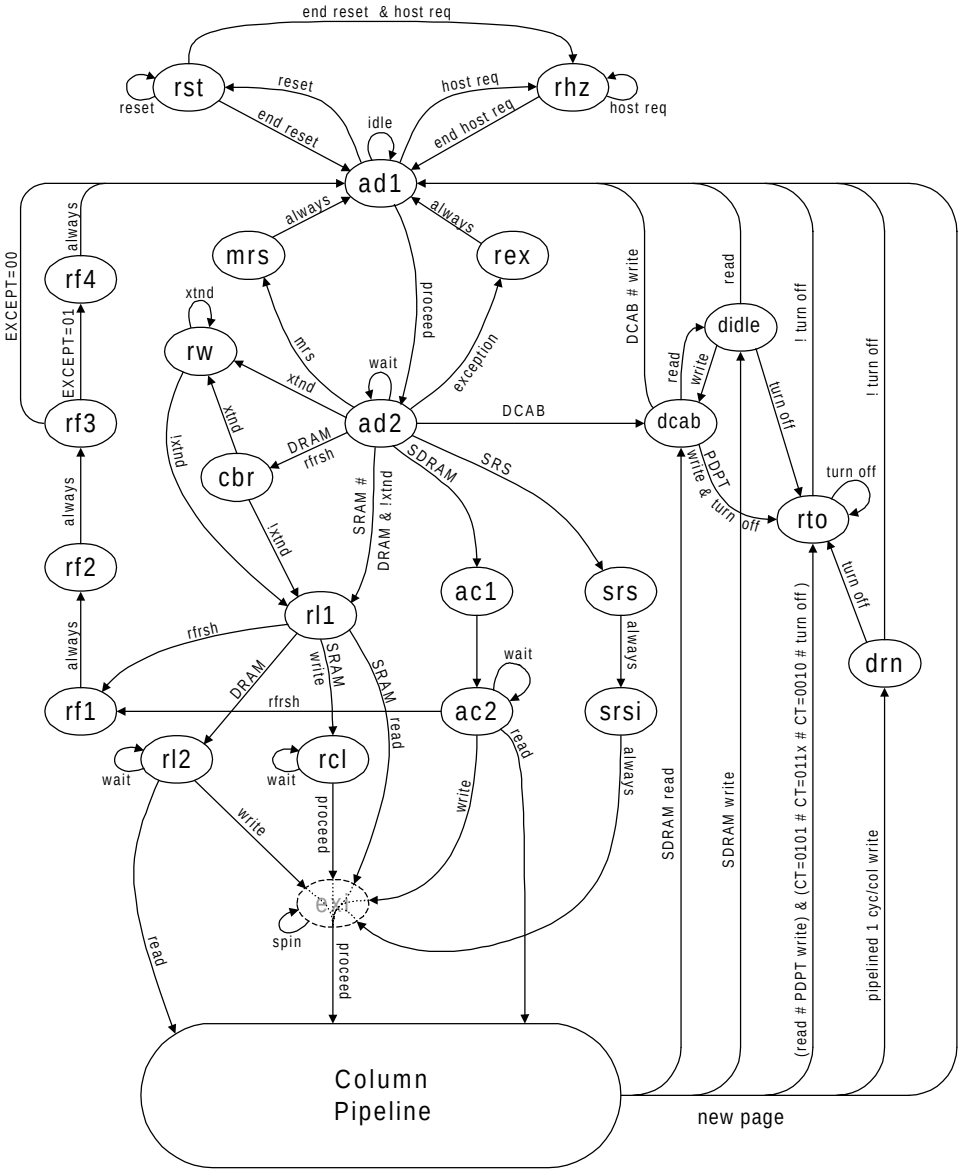


Figure 58. Memory Cycle State Diagram

row states

The row states make up the row time of each memory access. They occur when each new page access begins. The transition indicators determine the conditions that cause transitions to another state.

row states (continued)

Table 47. Row Time Memory States

STATE	DESCRIPTION
ad1	The first address state, common to all memory accesses. All signals are driven inactive and outputs are address and status. State is repeated if idle.
ad2	The second address state, common to all memory accesses. /RL is asserted and /DDIN is driven according to the data transfer direction. READY is sampled.
rl1	For DRAM and SRAM cycles. /RAS is asserted, AD[63:0] is placed in high impedance, and /DBEN is asserted if required. READY is sampled during cbr refresh cycles.
rl2	For DRAM cycles. Transitions of /TRG/CAS, /W, and /DSF may occur. State ensures sufficient /RAS low time before /CAS/DQM[7:0] are activated. READY is sampled.
ac1	For SDRAM cycles. ACTV command is generated on /RAS, /TRG/CAS, and /W. /DBEN is asserted if required, and AD[63:0] is placed in high impedance.
ac2	For SDRAM cycles. No transitions occur, but state ensures sufficient time between ACTV command and subsequent READ or WRT commands.
mrs	MRS command cycle.
cbr	Occurs before rl1 for DRAM refreshes to ensure /CAS/DQM is asserted a sufficient amount of time prior to assertion of /RAS.
rw	For all DRAM cycles, one or more of these states may be inserted if the previous page access was to the same bank. No transitions occur. State ensures sufficient /RAS high time.
rcl	Column pipeline load; common to all SRAM write cycles. One or more of these states may be inserted to ensure that the pipeline is properly loaded. READY is sampled.
srs	For SDRAM SRS cycles. /DBEN is asserted.
srsi	For SDRAM SRS cycles. Idle cycle required to allow the pipeline to load.
rex	Row time exception state. Ensures sufficient /RAS low time before returning to state ad1. State may be repeated as required.
rf1	First refresh cycle. Common to all refreshes. No signal transitions occur.
rf2	Second refresh cycle. Common to all refreshes. No signal transitions occur.
rf3	Third refresh cycle. Common to all refreshes. No signal transitions occur.
rf4	For slow (/EXCEPT[1:0] = 01) DRAM refresh cycles. Fourth refresh cycle which allows an additional cycle of /RAS low time.
exi	One or more of these states may be inserted when exceptions are enabled for the addressed bank of memory. State may be repeated as required. State is not present if exceptions are not supported.
didle	Idle cycle for SDRAM write cycles ensures enough time between the WRT command and bank deactivation (DCAB).
dcab	Deactivation cycle for SDRAM cycles.
drn	For 1 cycle/column pipelined DRAM writes, all /CAS/DQM[7:0] are activated to drain the DRAM pipeline.
rto	Turnoff cycle for all DRAM and SRAM reads; and DRAM and nonsynchronous SRAM PDT writes. All output signals except for /RL are driven inactive. Additional rto cycles will be performed according to the value of the TO[1:0] bank configuration field.
rst	Reset state. During reset, all ‘C82 signals with the exceptions of /HACK, REQ, and CLKOUT are placed in the high-impedance state.
rhz	High-impedance state. Occurs during host requests and repeats until bus is released by the host. All ‘C82 signals with the exceptions of /HACK, REQ, and CLKOUT are placed in the high-impedance state.

row states (continued)

Table 48. State Transition Indicators

INDICATOR	DESCRIPTION
any cycle	Continuation of current cycle.
/EXCEPT=xx	State change occurs for indicated /EXCEPT[1:0] value (as latched in ad2).
exception	Memory exception—retry, fault, or configuration cache flush request.
retry	/EXCEPT = 10.
wait	READY input sampled low in ad2, rl1 (CBR refresh only), rl2, rcl, and last column state; repeat current state.
spin	Internally generated wait state to allow TC pipeline to load/resolve contention.
new page	The next access requires a page change (new row access)
turn off	Turn off, specified by default or by TO[1:0] field in bank configuration.
xtnd	Internally generated transition to insert additional cycles in order to support exceptions.

external memory timing examples

The following sections contain descriptions of the various C82 memory cycles and illustrate the signal transitions for those cycles. Memory cycles may be separated into three basic categories: DRAM cycles for use with EDO DRAM and VRAM; SRAM cycles for use with SRAM and peripherals; and SDRAM cycles for use with SDRAM and SGRAM.

DRAM cycles

The DRAM cycles are page-mode accesses consisting of a row access followed by one or more column accesses. Column accesses may be one, two, or three clock cycles in length with 2 and 3 cycle accesses allowing the insertion of wait states to accommodate slow devices. Idle cycles can occur after necessary column accesses have completed or between column accesses due to “bubbles” in the TC data flow pipeline. The pipeline diagrams in Figure 59 show the pipeline stages for each access type and when the /CAS/DQM signal corresponding to the column access is activated.

DRAM cycles (continued)

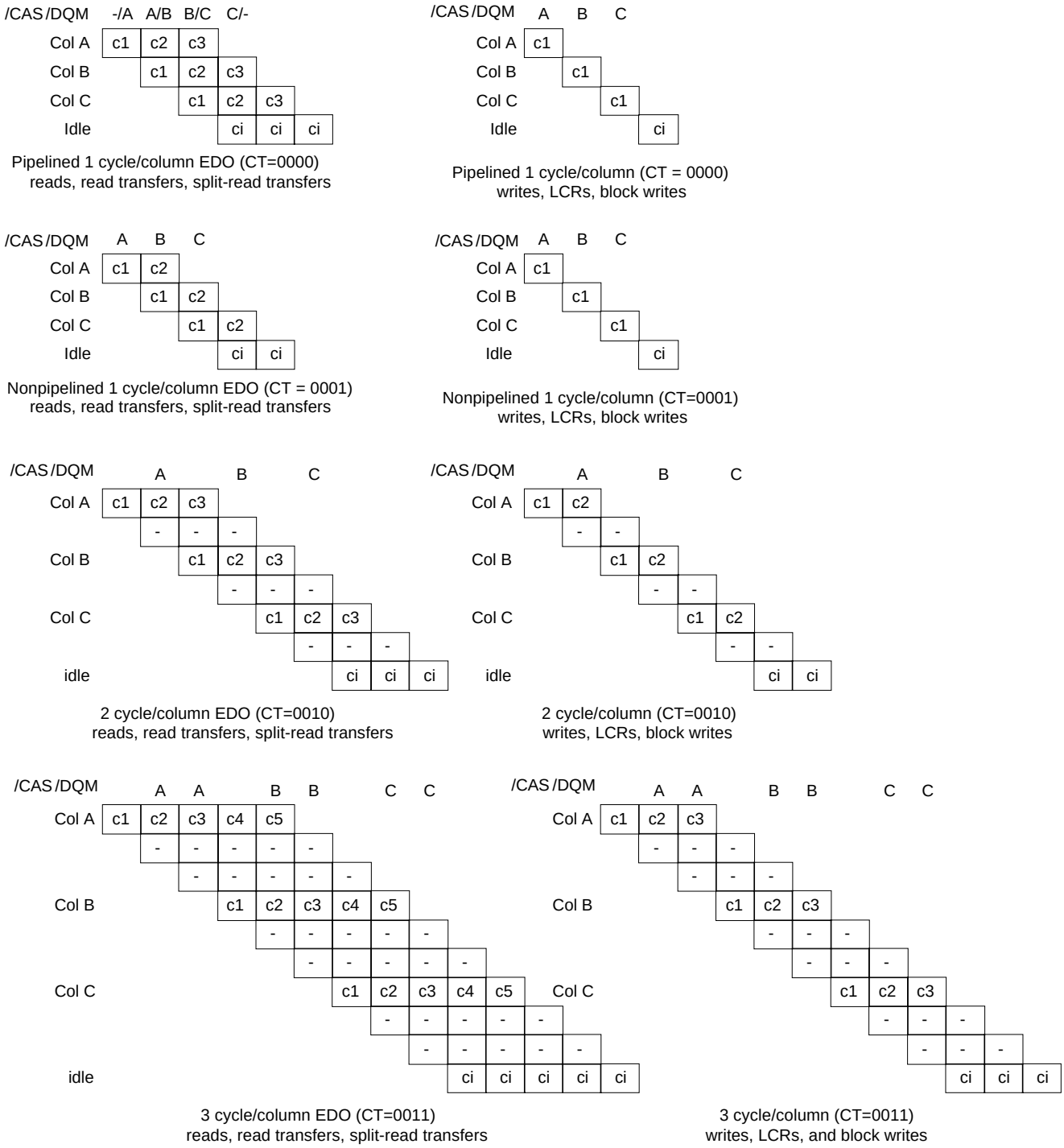


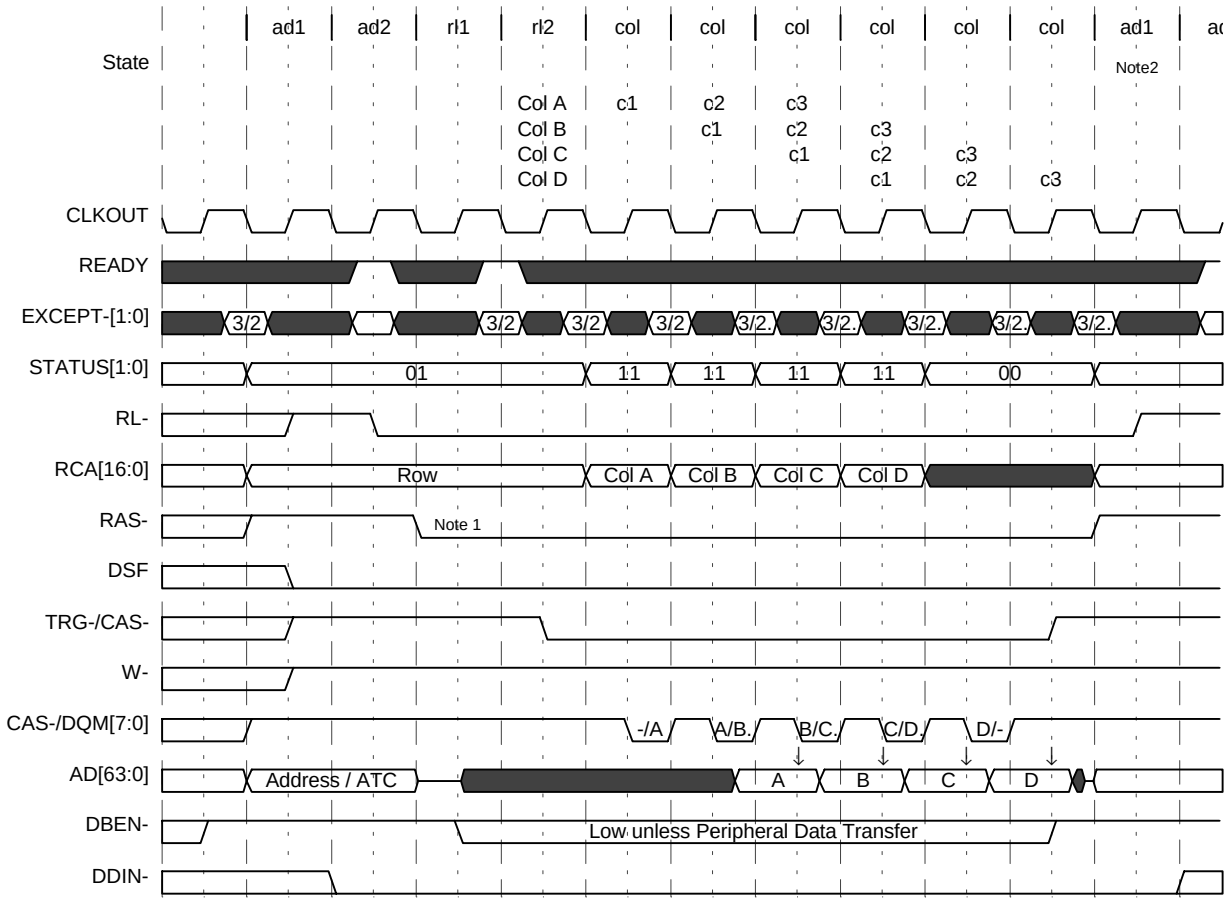
Figure 59. DRAM Cycle Column Pipelines

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77251-1443

read cycles

Read cycles transfer data or instructions from external memory to the ‘C82. The cycles can occur as a result of a packet transfer, cache request, or DEA request. During the cycle, /W is held high, /TRG/CAS is driven low after /RAS to enable memory output drivers, and /DBEN and /DDIN are low so that data transceivers may drive into the ‘C82. During column time, the TC places AD[63:0] into high impedance, allowing it to be driven by the memory, and latches input data during the appropriate column state. The TC always reads 64 bits and extracts and aligns the appropriate bytes. Invalid bytes for bus sizes of less than 64 bits are discarded. During peripheral device packet transfers, /DBEN and /DDIN remain high.

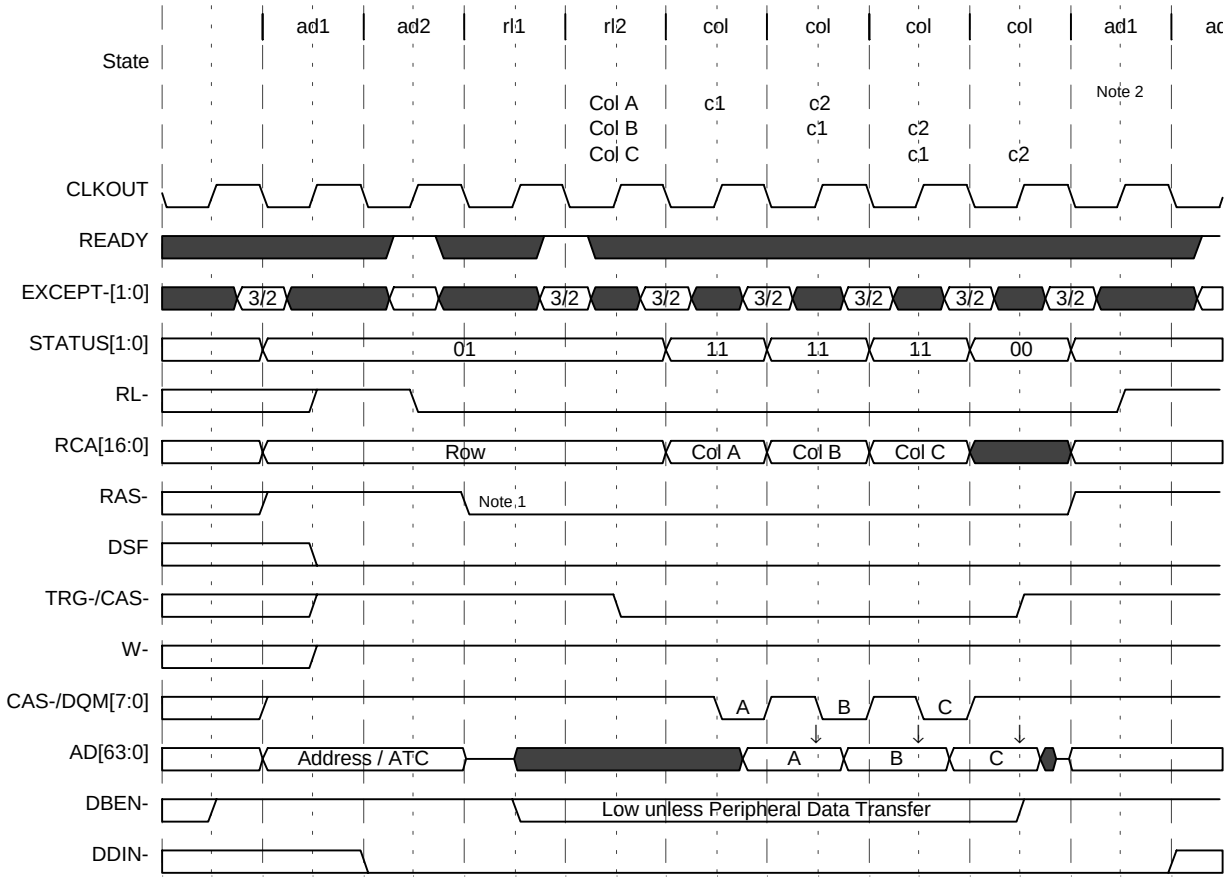
For DRAM reads, the minimum number of cycles between ad1 and the first column access is four whether exceptions are supported or not.



Notes: 1. Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.
2. Turnoff cycles will be inserted between rto and ad1 as specified by the bank configuration.

Figure 60. 1 cycle/column Pipelined EDO DRAM Read Cycle

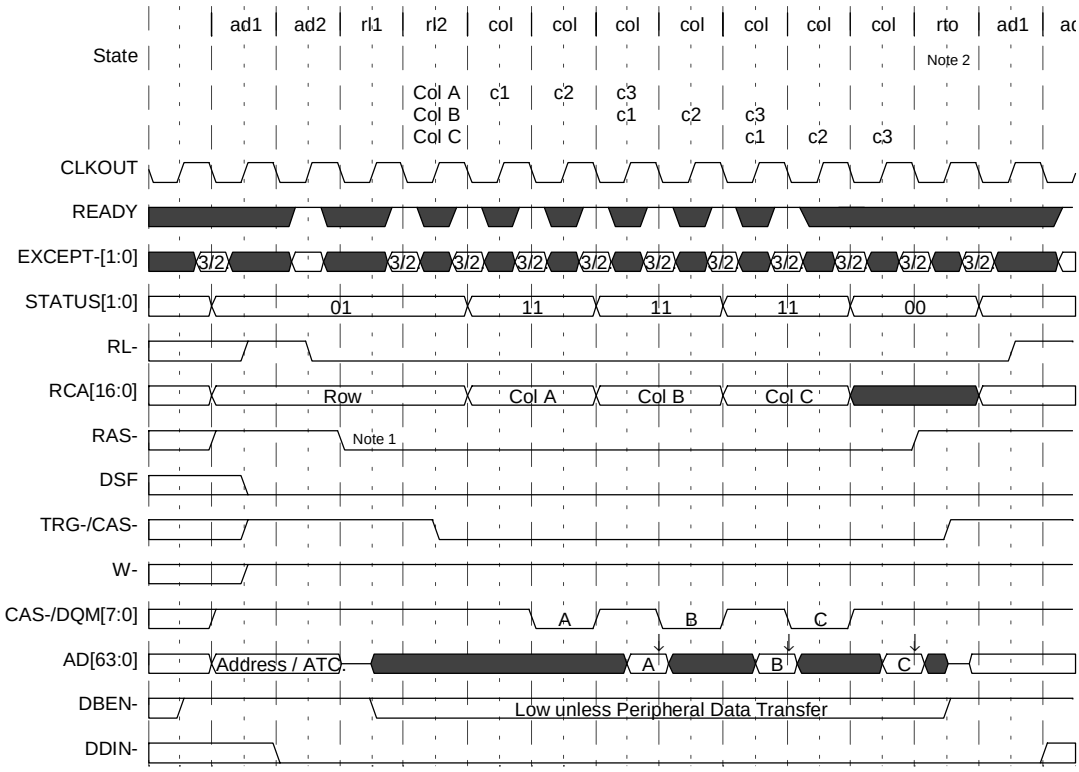
read cycles (continued)



Notes: 1. Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.
2. Turnoff cycles will be inserted between rto and ad1 as specified by the bank configuration.

Figure 61. 1 cycle/column EDO DRAM Read Cycle

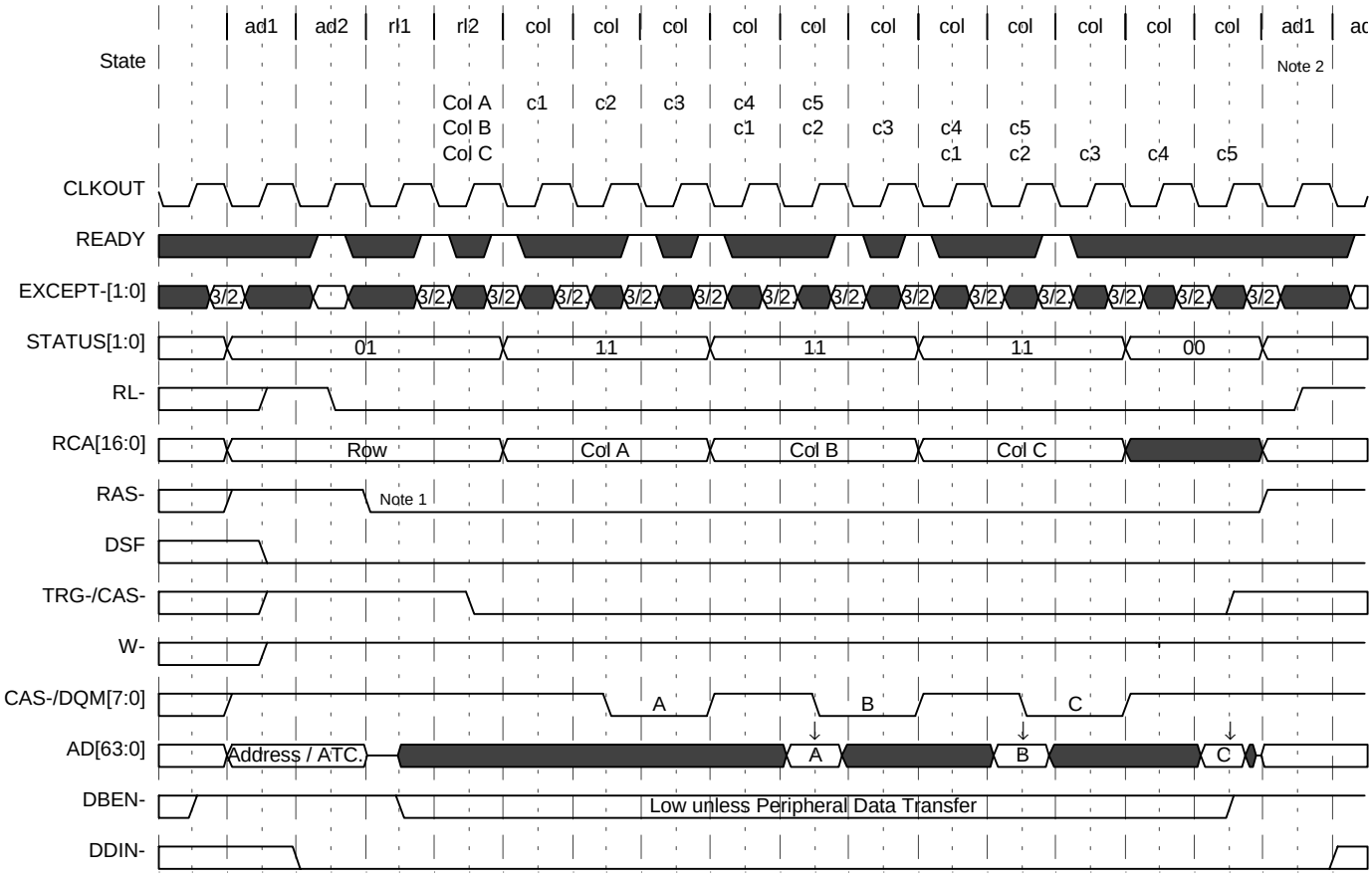
read cycles (continued)



Notes: 1. Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.
2. Turnoff cycles will be inserted between rto and ad1 as specified by the bank configuration.

Figure 62. 2 cycles/column EDO DRAM Read Cycle

read cycles (continued)



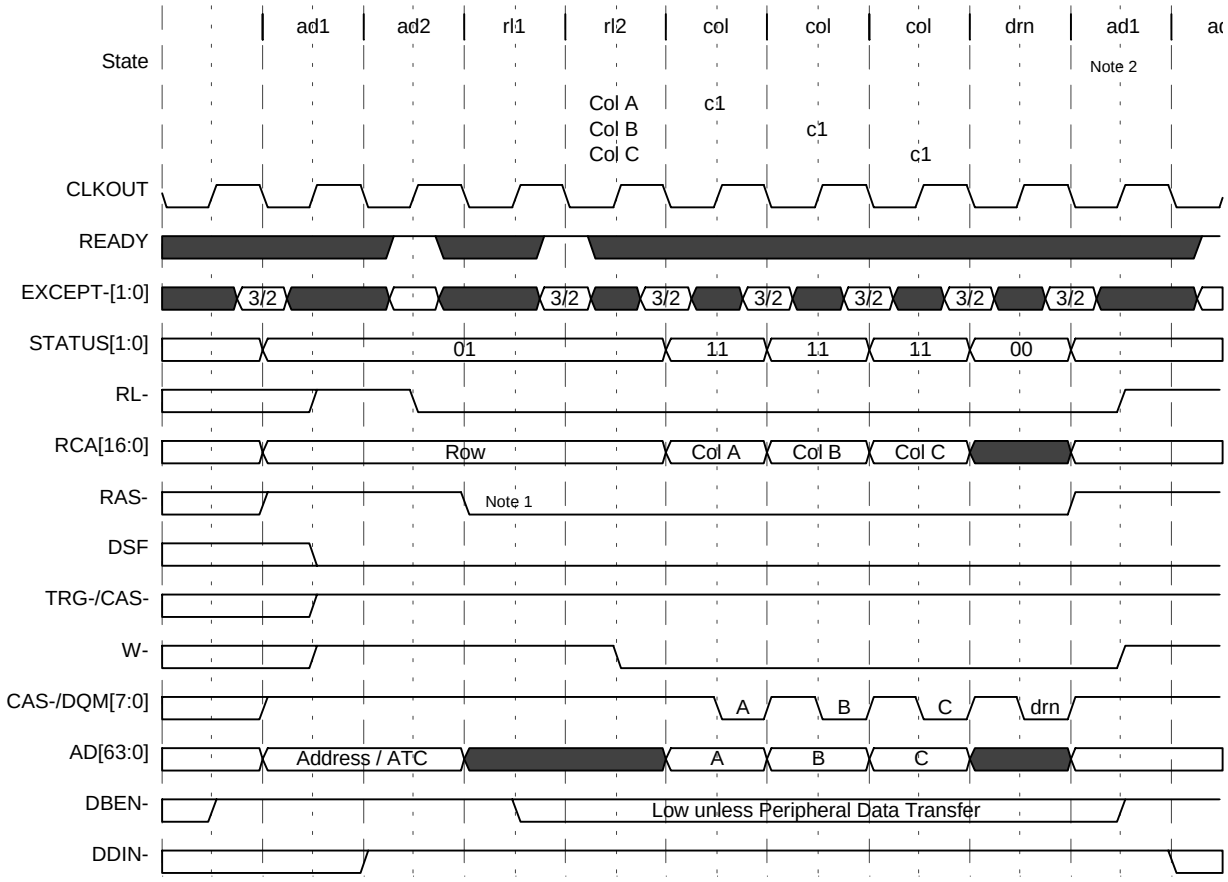
Notes: 1. Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 4 cycles.
2. Turnoff cycles will be inserted between rto and ad1 as specified by the bank configuration.

Figure 63. 3 cycle/column EDO DRAM Read Cycle

write cycles

Write cycles transfer data from the ‘C82 to external memory. These cycles can occur as a result of a packet transfer, a DEA request, or an MP data cache write-back. During the cycle /TRG/CAS is held high, /W is driven low after the fall of /RAS to enable early write cycles, and /DDIN is high so that data transceivers drive toward memory. The TC drives data out on AD[63:0] and indicates valid bytes by activating the appropriate /CAS/DQM strobes. During peripheral device packet transfers, /DBEN remains high and AD[63:0] is placed in high impedance so that the peripheral device may drive data into the memory. Additionally, the number of turnoff cycles identified by the TO(1:0) field for the addressed bank of memory will be applied during PDT write cycles.

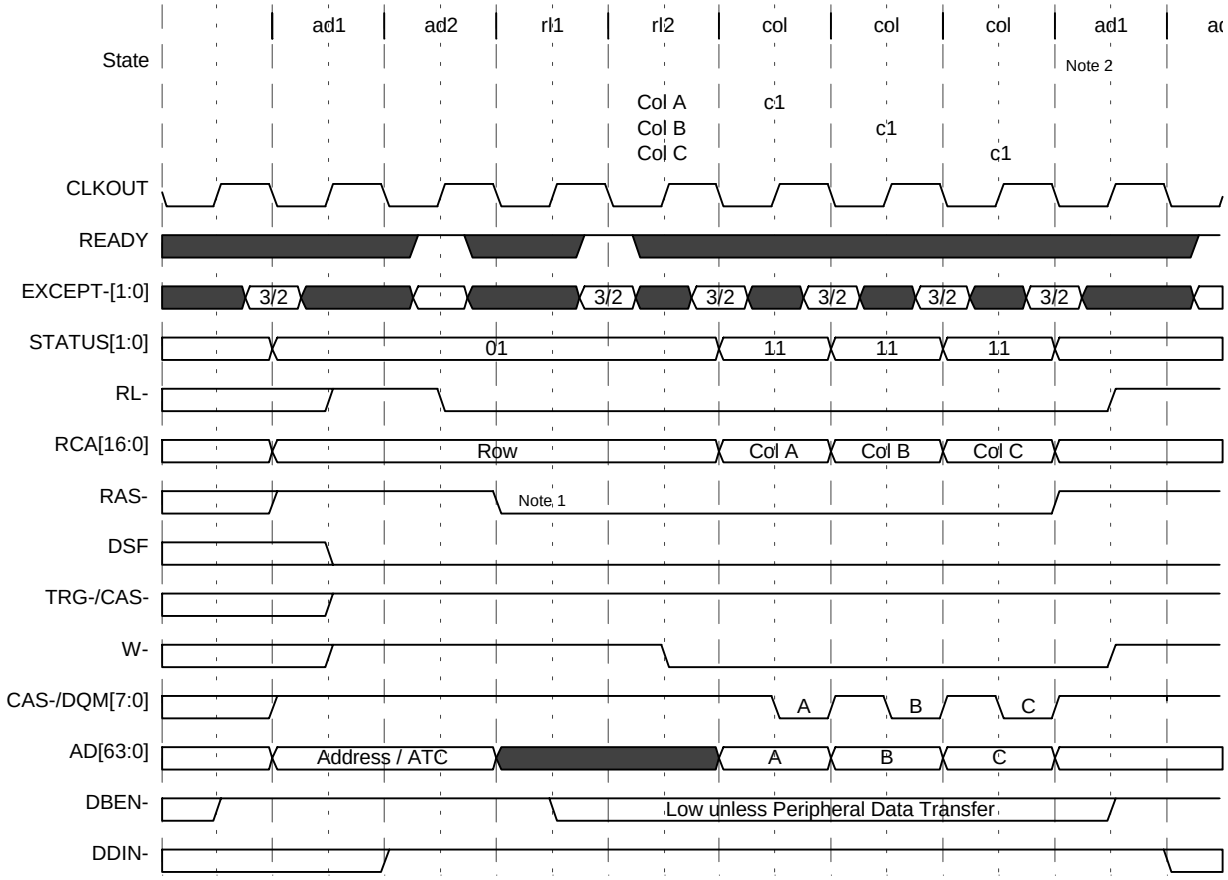
Exceptions are not supported in the following diagrams. Support for exceptions increases the minimum number of cycles between ad1 and the first column state from 4 to 6.



Notes: 1. Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.
2. During peripheral data transfer, turnoff cycles will be inserted prior to ad1 as specified by the bank configuration.

Figure 64. 1 cycle/column Pipelined DRAM Write Cycle

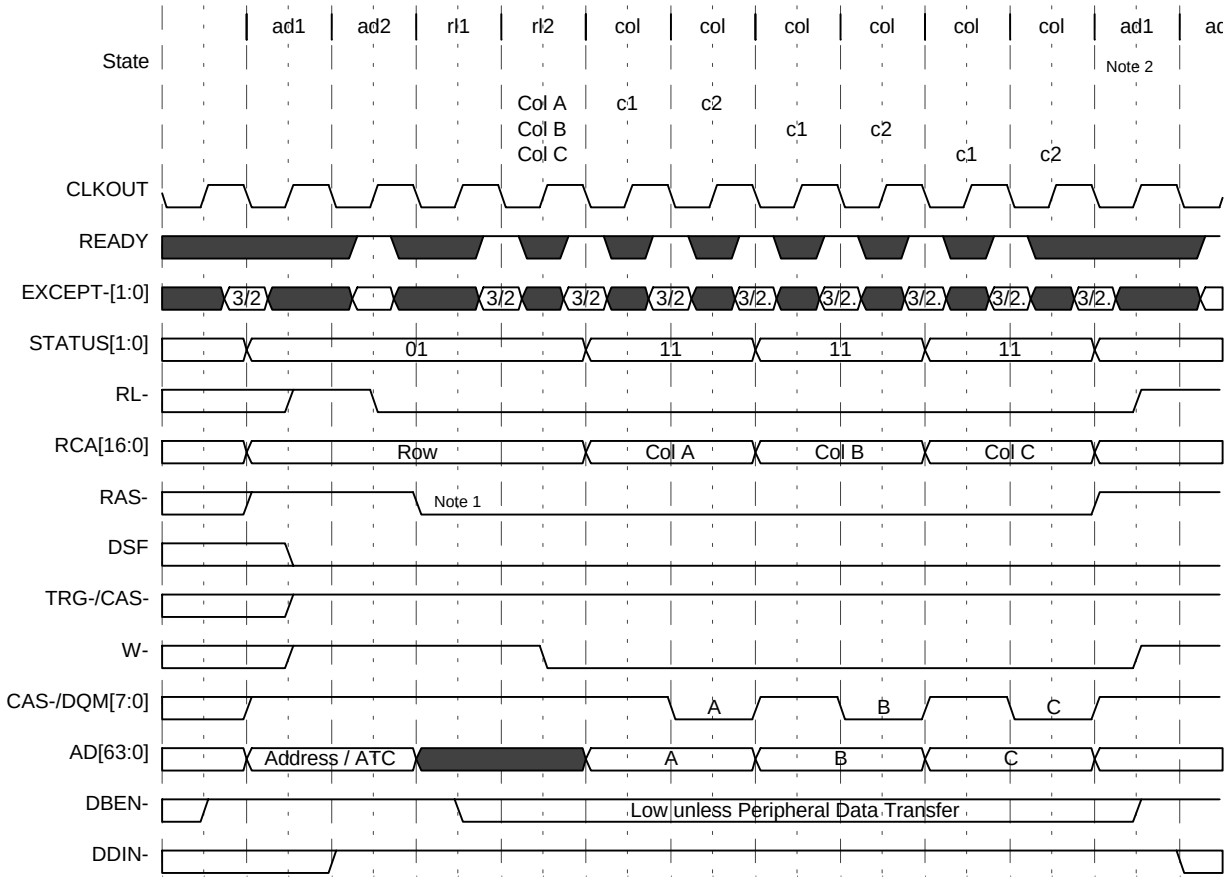
write cycles (continued)



Notes: 1. Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.
2. During peripheral data transfer, turnoff cycles will be inserted prior to ad1 as specified by the bank configuration.

Figure 65. 1 cycle/column DRAM Write Cycle

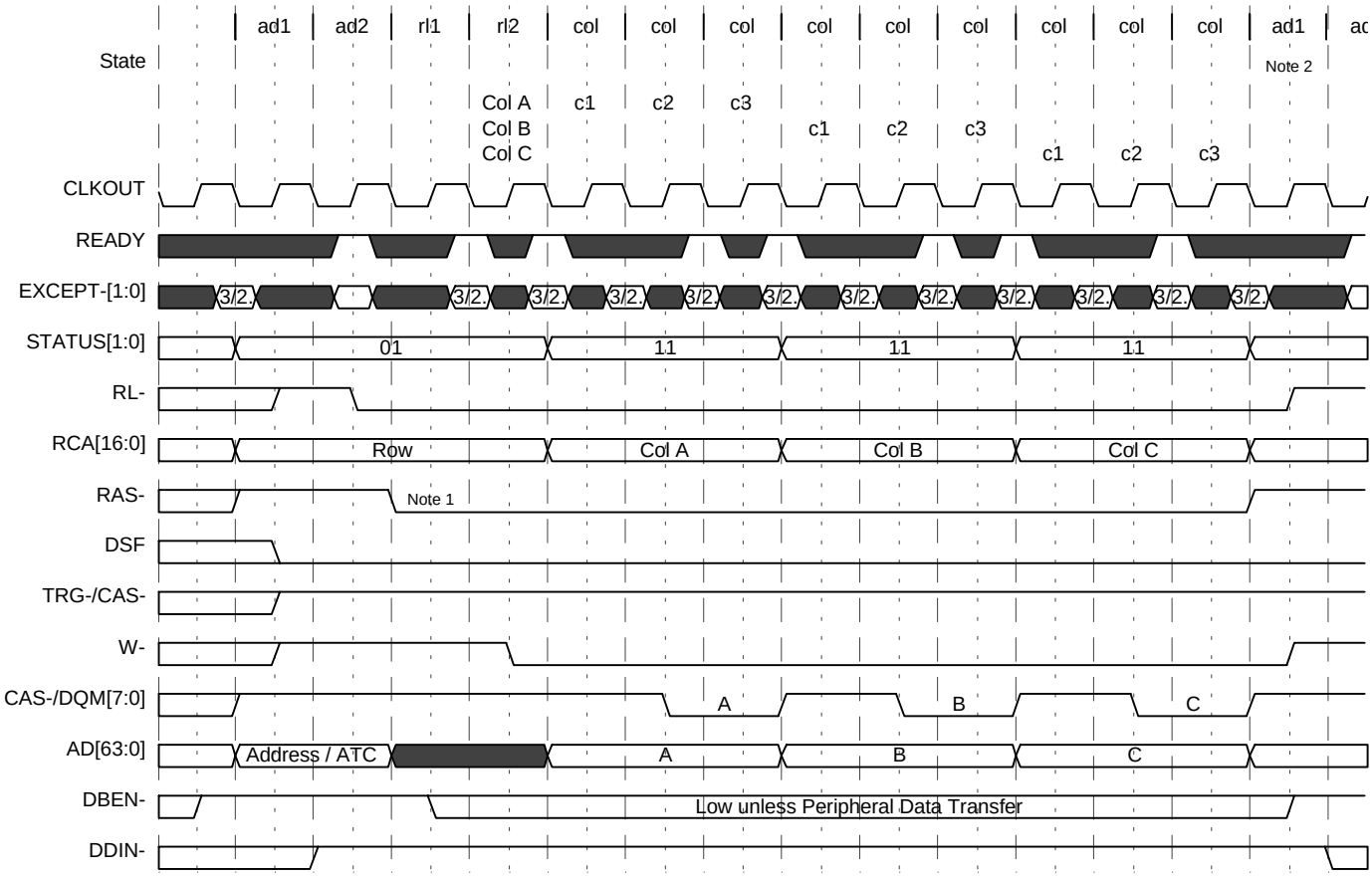
write cycles (continued)



Notes: 1. Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.
2. During peripheral data transfer, turnoff cycles will be inserted prior to ad1 as specified by the bank configuration.

Figure 66. 2 cycle/column DRAM Write Cycle

write cycles (continued)



Notes: 1. Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 4 cycles.
2. During peripheral data transfer, turnoff cycles will be inserted prior to ad1 as specified by the bank configuration.

Figure 67. 3 cycle/column DRAM Write Cycle

load color register cycles

Load color register (LCR) cycles are used to load a VRAM's color register prior to performing a block write. LCR cycles are supported only on 64-bit data busses. Because an LCR writes into a VRAM, it closely resembles a normal write cycle. The difference is that the DSF output is high at both the fall of /RAS and the fall of /CAS/DQM. Also, because the VRAM color register is a single location, only one column access occurs.

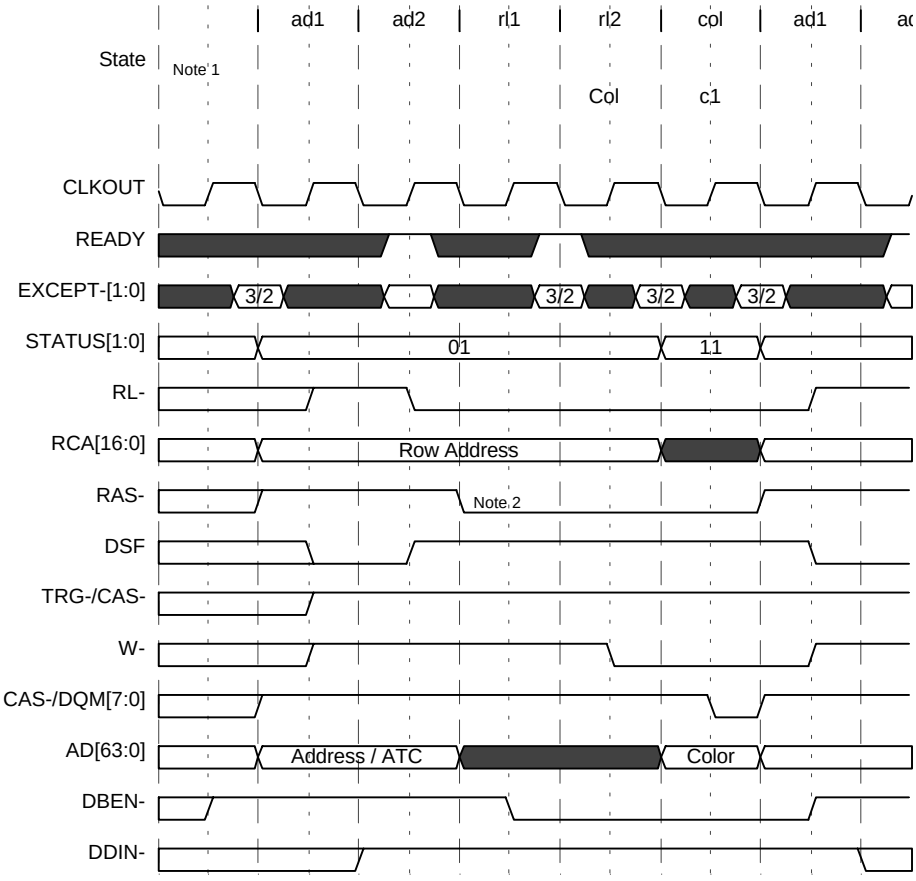
The row address output by the TC is used for bank decode only. Normally all VRAM banks should be selected during an LCR cycle because another LCR will not occur when a block write memory page change occurs. The column address output during an LCR is likewise irrelevant as the VRAM color register is the only location written. All /CAS/DQM strobes are active during an LCR cycle.

If exception support for a given bank is enabled, the /EXCEPT[1:0] inputs are sampled during LCR column states and must be at valid levels. A retry code (/EXCEPT[1:0] = 10) at column time has no effect, however, because only one column access is performed.

If the BW (block write) field of the configuration cache entry for the given bank indicate that the addressed memory supports only simulated block writes, the LCR cycle will be changed into a normal write cycle at the start of the simulated block write.

Exceptions are not supported in the following diagrams. Support for exceptions increases the minimum number of cycles between ad1 and the first column state from 4 to 6.

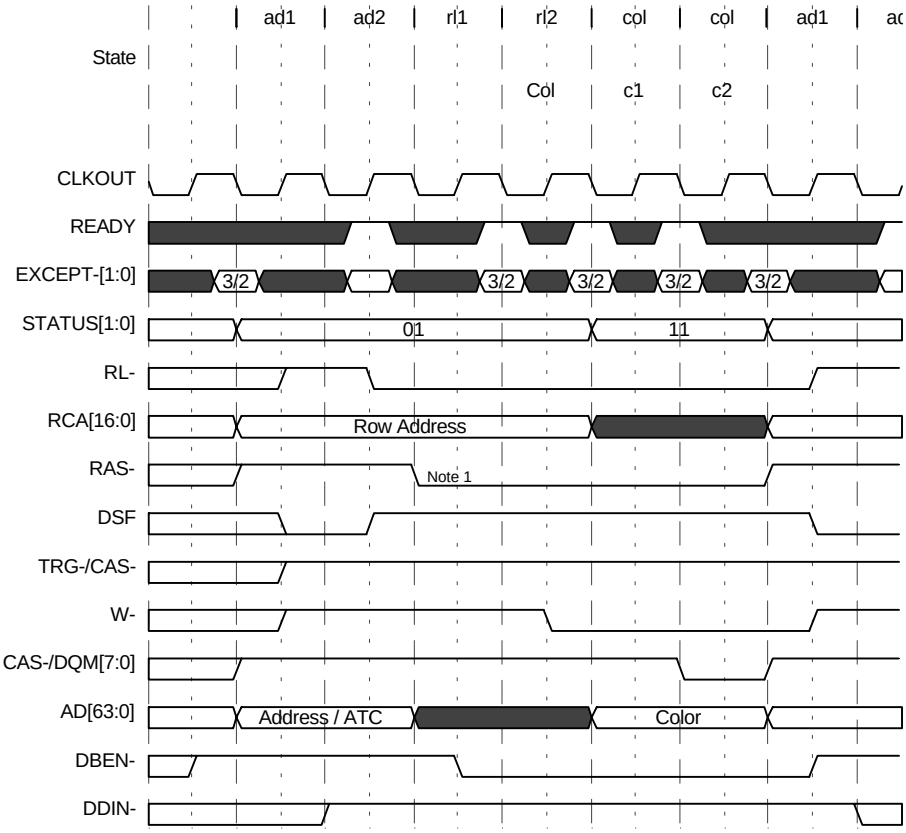
load color register cycles (continued)



Notes: 1. These timings apply to piplined and nonpiplined 1 cycle/column DRAM.
2. Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.

Figure 68. 1 cycle/column Load Color Register (LCR) Cycle

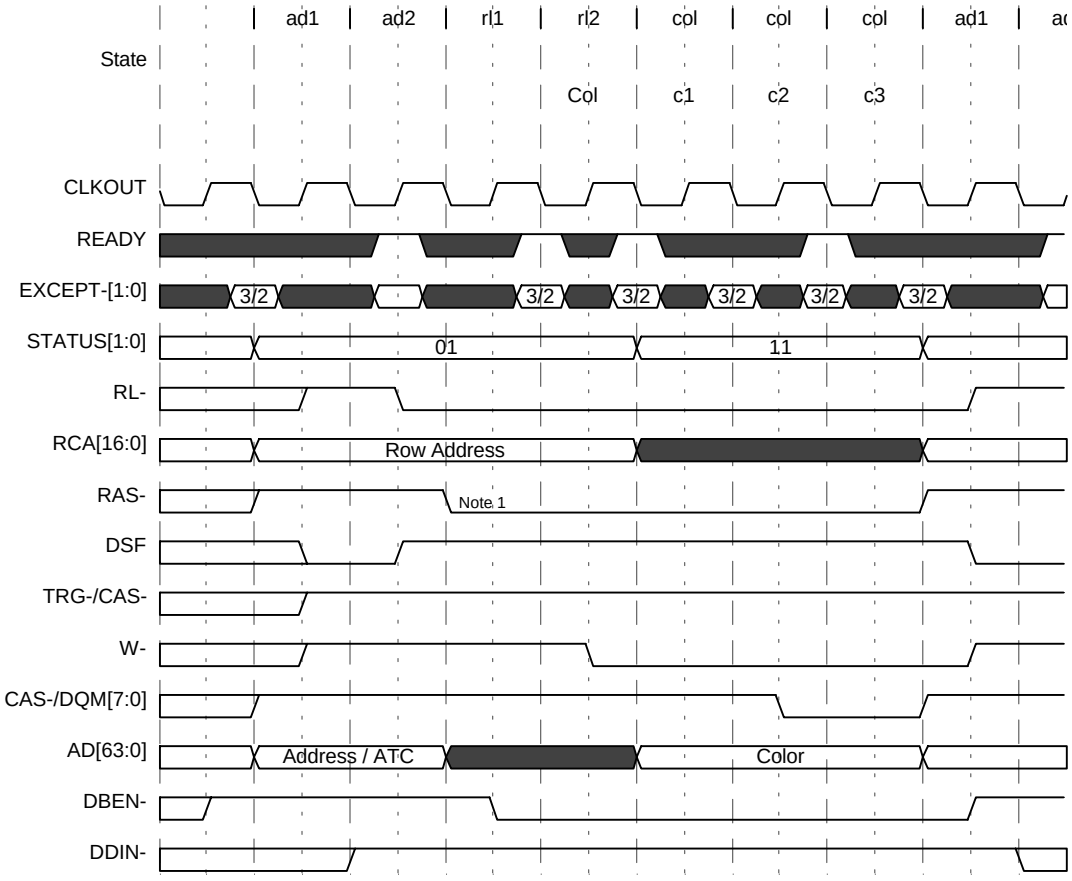
load color register cycles (continued)



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.

Figure 69. 2 cycle/column Load Color Register (LCR) Cycle

load color register cycles (continued)



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 4 cycles.

Figure 70. 3 cycle/column Load Color Register (LCR) Cycle

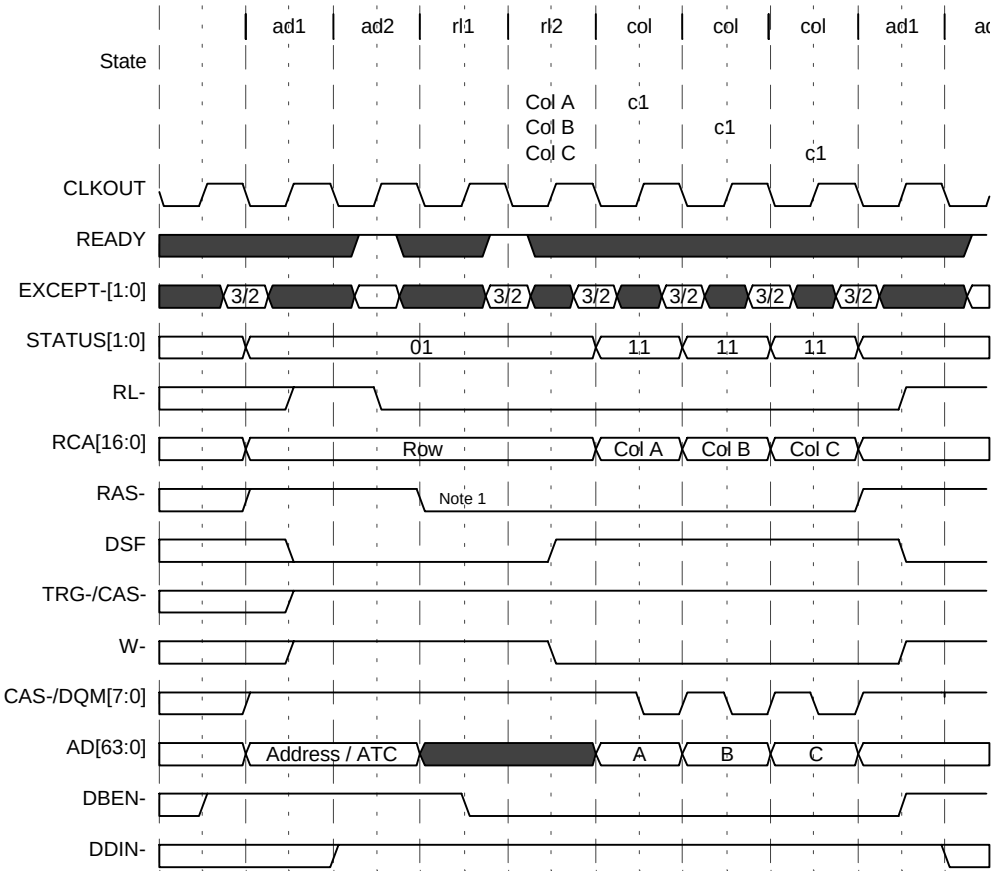
block-write cycles

Block-write cycles cause the data stored in the VRAM color registers to be written to the memory locations enabled by the appropriate data bits on the AD[63:0] bus. This allows up to a total of 64 bytes (depending on the block-write type) to be written in a single column access. The TMS320C82 supports 4x and 8x block-writes. Selection of block-write type is controlled by the BW field input during the appropriate bank configuration cycle. A block-write cycle is indicated by a row-time status code of 1001 output on AD[35:32]. The block-write cycle is identical to a standard write cycle with the following exceptions:

- DSF is high at the fall of /CAS/DQM, enabling the block-write function of the VRAMs.
- Only 64-bit data bus widths are supported; consequently, all /CAS/DQM signals will be active (low).
- Block writes always begin with a new row access. Upon completion of a block-write, the memory interface returns to the ad1 state to await the next access.
- The address output on AD[63:0] during column accesses represent the column locations to be written using the color register value. Depending on the type of block-write being performed, all of the data bits may not be used by the VRAM.
- The two or three LSBs of address output on RCA[16:0] are ignored by the VRAMs because the column locations are specified by the value on the data bus.

Exceptions are not supported in Figure 71, Figure 72, and Figure 73. Support for exceptions increases the minimum number of cycles between ad1 and the first column state from 4 to 6.

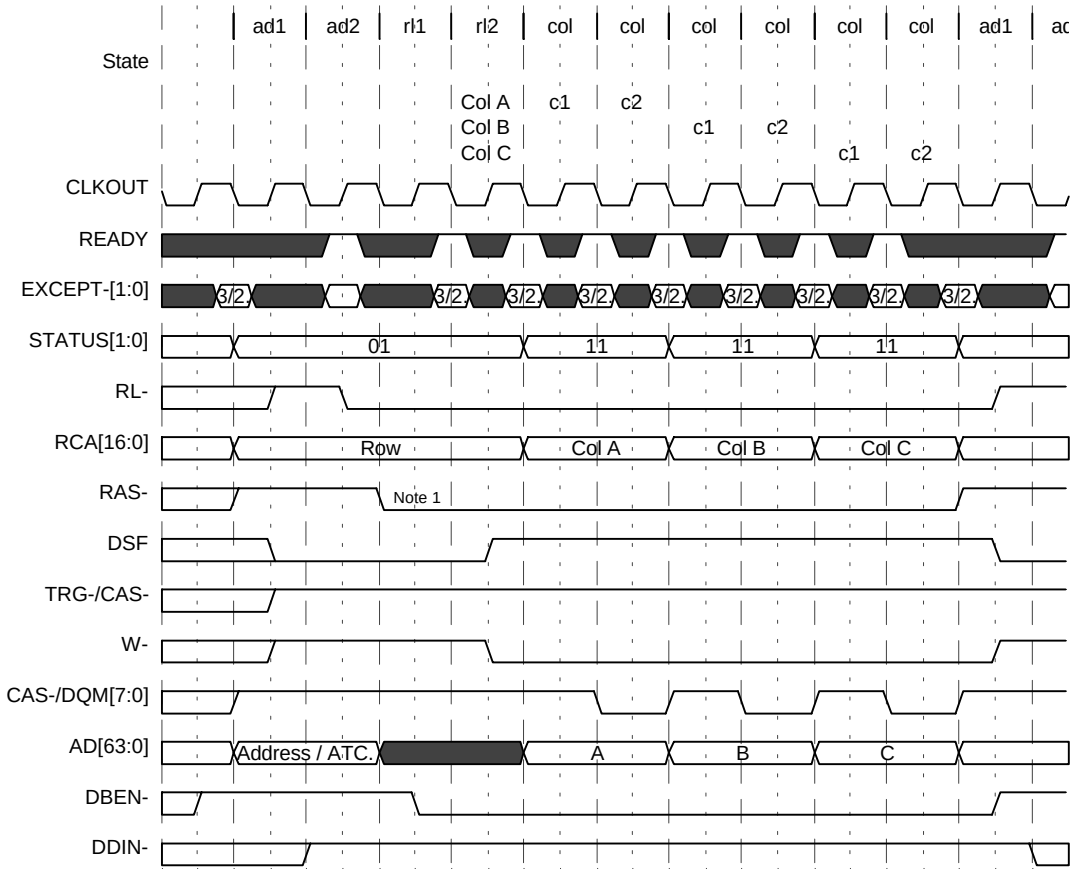
block-write cycles (continued)



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.

Figure 71. 1 cycle/column Block-Write Cycle

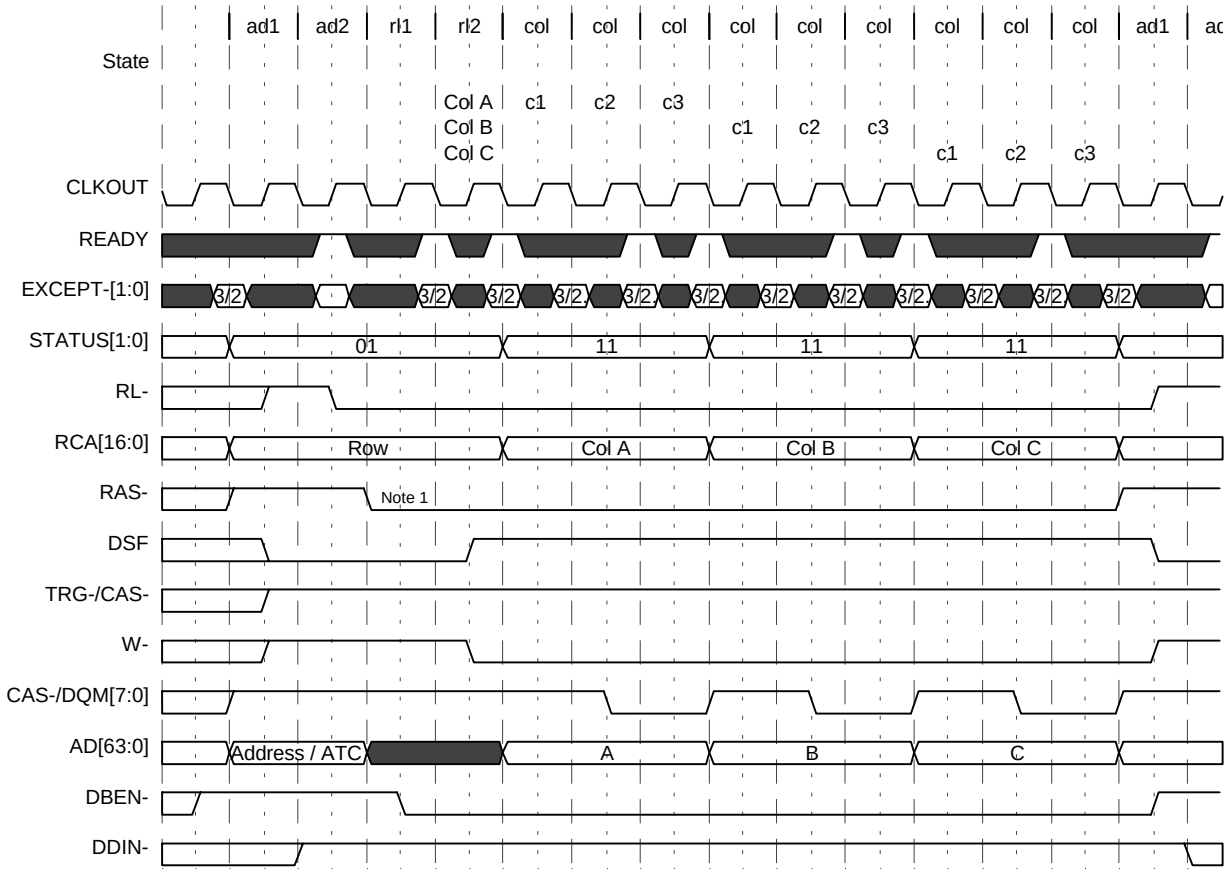
block-write cycles (continued)



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.

Figure 72. 2 cycle/column Block-Write Cycle

block-write cycles (continued)



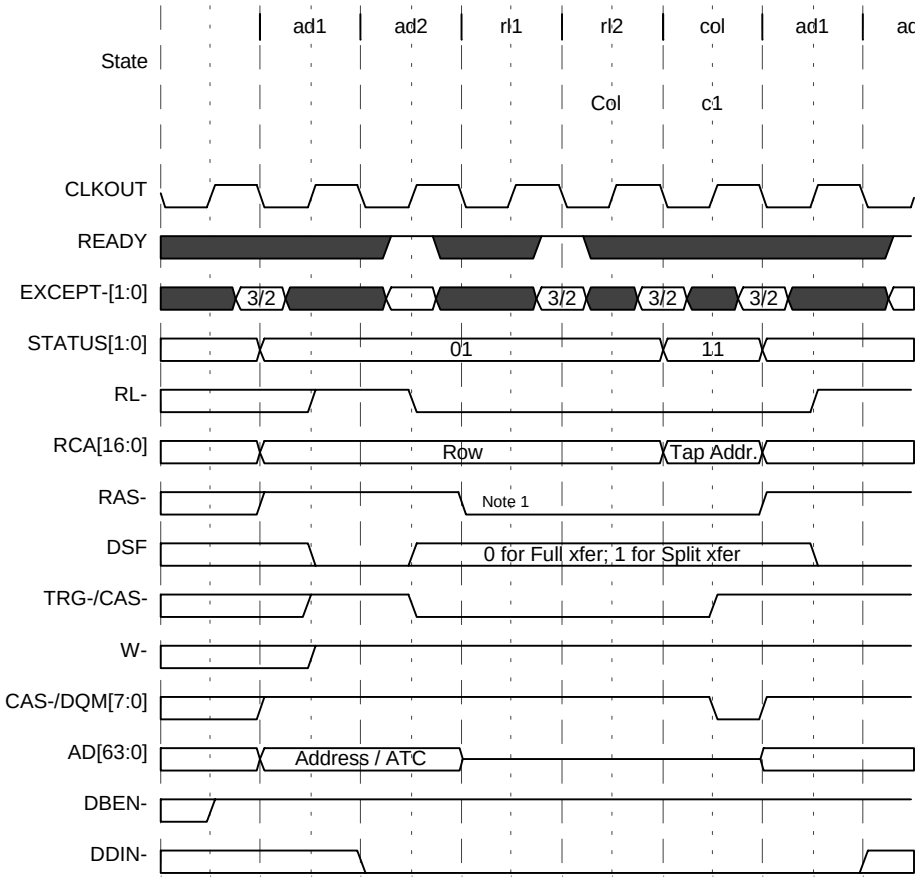
Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 4 cycles.

Figure 73. 3 cycle/column Block-Write Cycle

read and split-read transfers

Read and split-read transfers resemble a standard read cycle. These cycles are performed as the result of a packet transfer submission to the TC which specifies the SRT mode. The ‘C82 supports both read and split-read transfers. These cycles are designed to transfer a row of data from the VRAM memory array into the VRAM SAM register. Since no data is actually transferred over the system bus during an SRT cycle, the AD[63:0] bus is placed in high impedance. The /TRG/CAS output is driven low prior to the fall of /RAS to indicate a transfer cycle. Only a single column access is performed, therefore, while /EXCEPT[1:0] are required to be at valid levels, retries will have no effect if asserted at column time. The value output on RCA[16:0] at column time represents the SAM tap point.

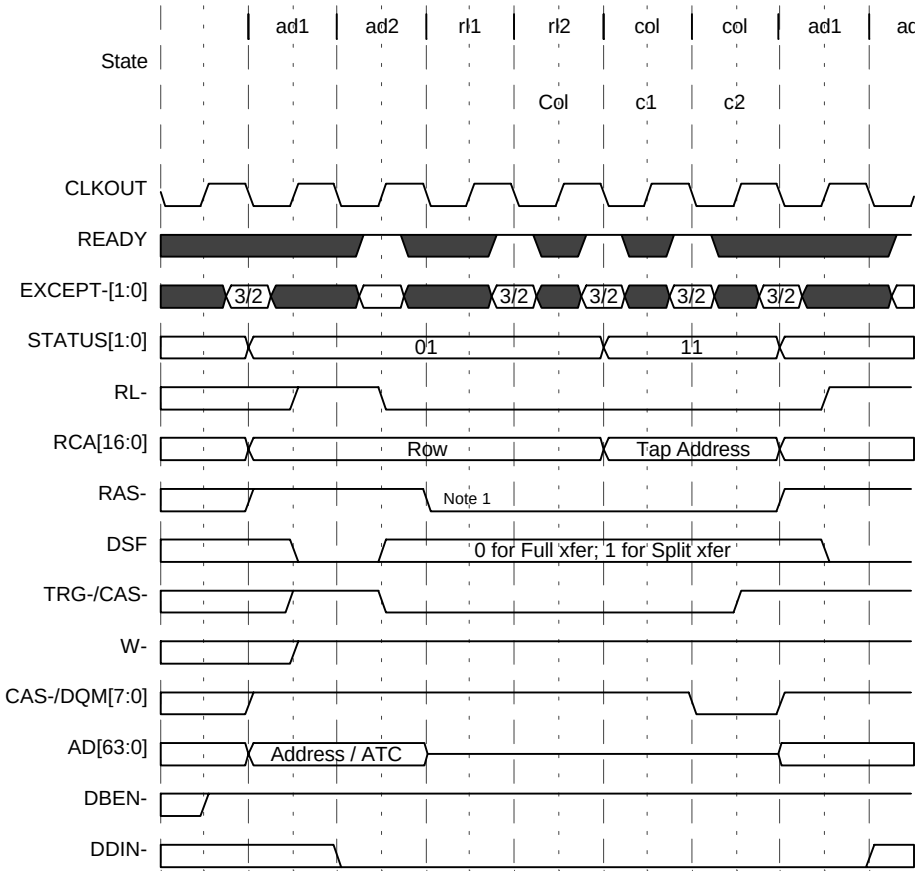
read and split-read transfers (continued)



Notes: 1. Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.
2. These timings apply to piplined and nonpiplined 1 cycle/column DRAM.

Figure 74. 1 cycle/column Memory-to-Register Transfer Cycle

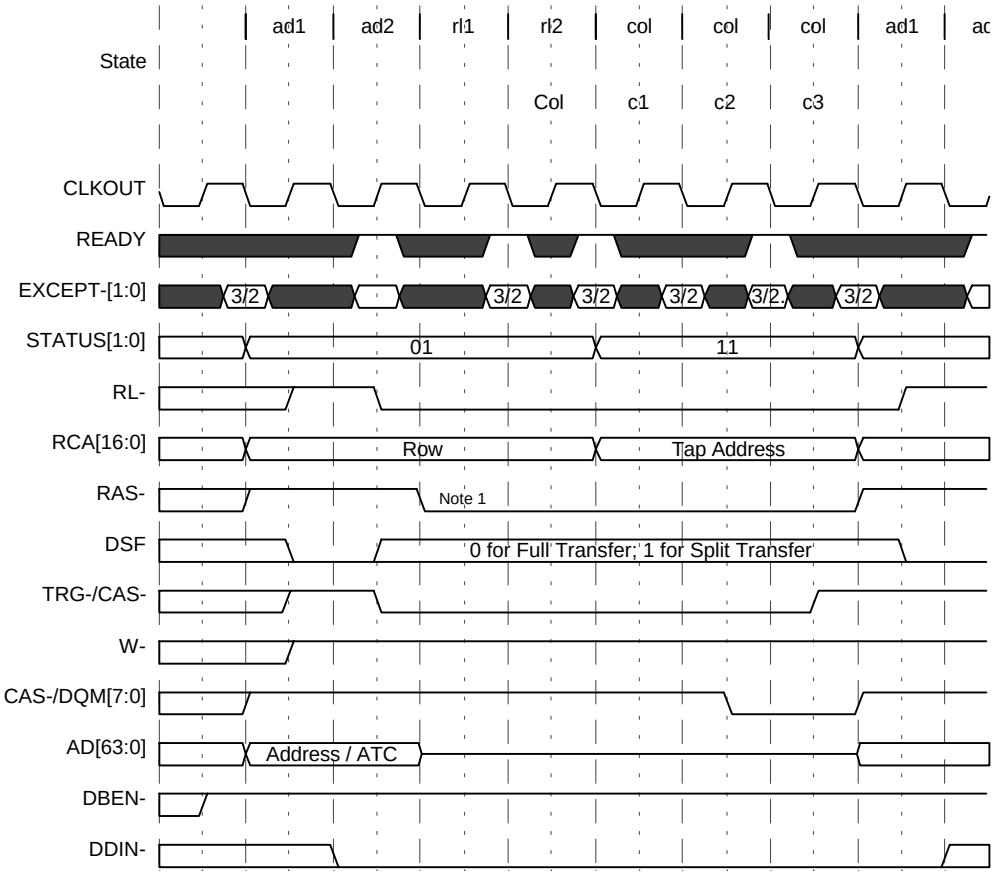
read and split-read transfers (continued)



Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 3 cycles.

Figure 75. 2 cycle/column Memory-to-Register Transfer Cycle

read and split-read transfers (continued)

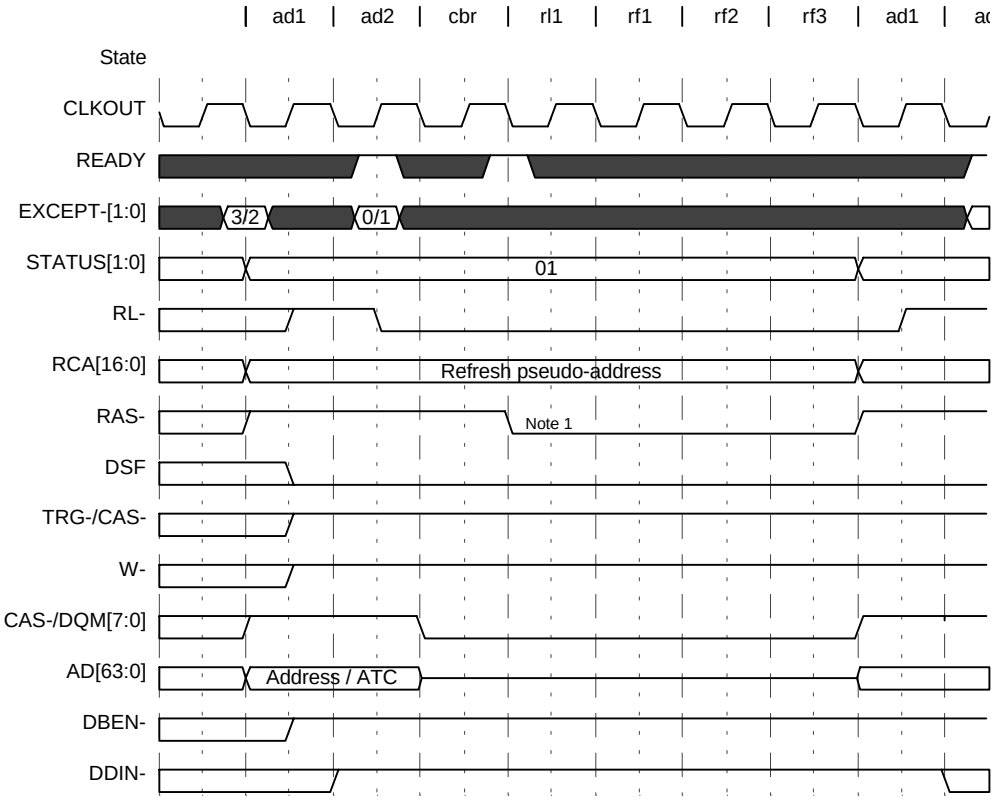


Note 1: Additional cycles will be inserted between ad2 and rl1 as required to ensure RAS- high of at least 4 cycles.

Figure 76. 3 cycle/column Memory-to-Register Transfer Cycle

DRAM refresh cycle

The DRAM refresh cycle is performed when the TC receives a DRAM cycle timing input (/EXCEPT[1:0] = 0X) at the start of a refresh cycle. The TC performs /CAS-before-/RAS (CBR) refresh cycles, wherein the refresh address is generated internal to the memory device. The ‘C82 outputs a 16-bit pseudo-address (used for refresh bank decode) on RCA[16:1]. The pseudo-address is decremented once for each refresh that is performed.



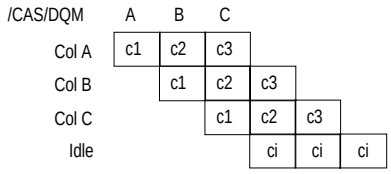
Note 1: RAS- cannot be high for less than the 3 cycles required when /EXCEPT[1:0] = 00.
An additional cycle will be inserted between cbr and rl1 when /EXCEPT[1:0] = 01.
An additional cycle will be inserted between rf3 and ad1 when /EXCEPT[1:0] = 01.

Figure 77. DRAM Refresh

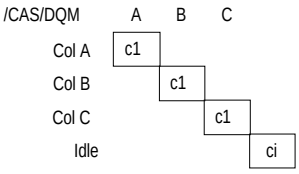
SRAM cycles

Similar to DRAM cycles, the SRAM cycles are page-mode accesses consisting of a row access followed by one or more column accesses. Column accesses may be one, two, or three clock cycles in length with 2 and 3 cycle accesses allowing the insertion of wait states to accommodate slow devices. Idle cycles can occur after necessary column accesses have completed or between column accesses due to “bubbles” in the TC data flow pipeline. The pipeline diagrams in Figure 78 show the pipeline stages for each access type and when the /CAS/DQM signal corresponding to the column access is activated.

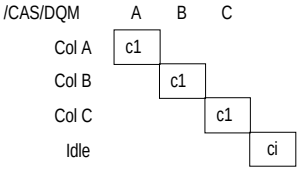
SRAM cycles (continued)



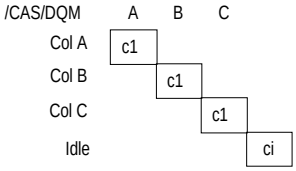
Synchronous SRAM (CT=0100) reads



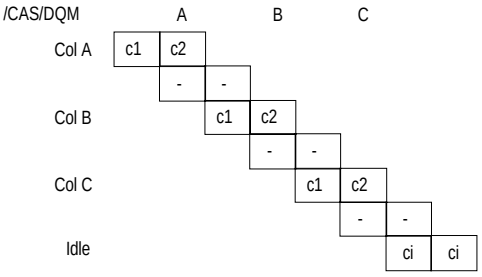
Synchronous SRAM (CT = 0100) writes



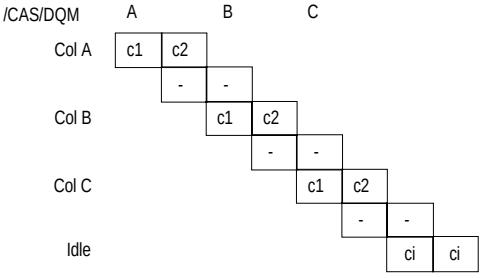
Asynchronous 1 cycle/column SRAM (CT = 0101) reads



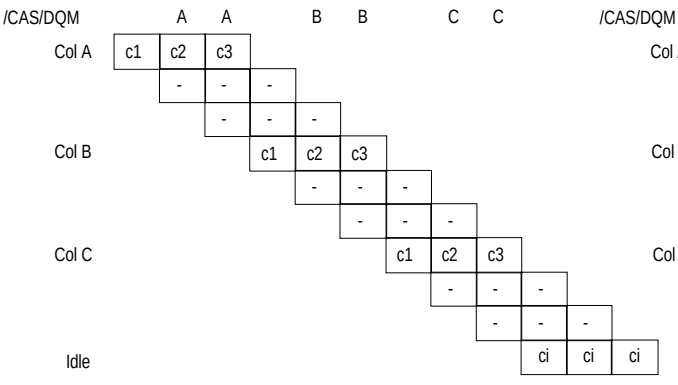
Asynchronous 1 cycle/column (CT=0101) writes



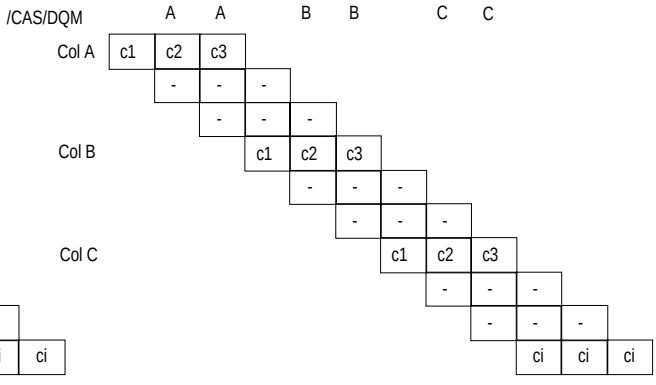
2 cycle/column SRAM (CT=0110) reads



2 cycle/column (CT=0110) writes



3 cycle/column SRAM (CT=0111) reads



3 cycle/column (CT=0111) writes

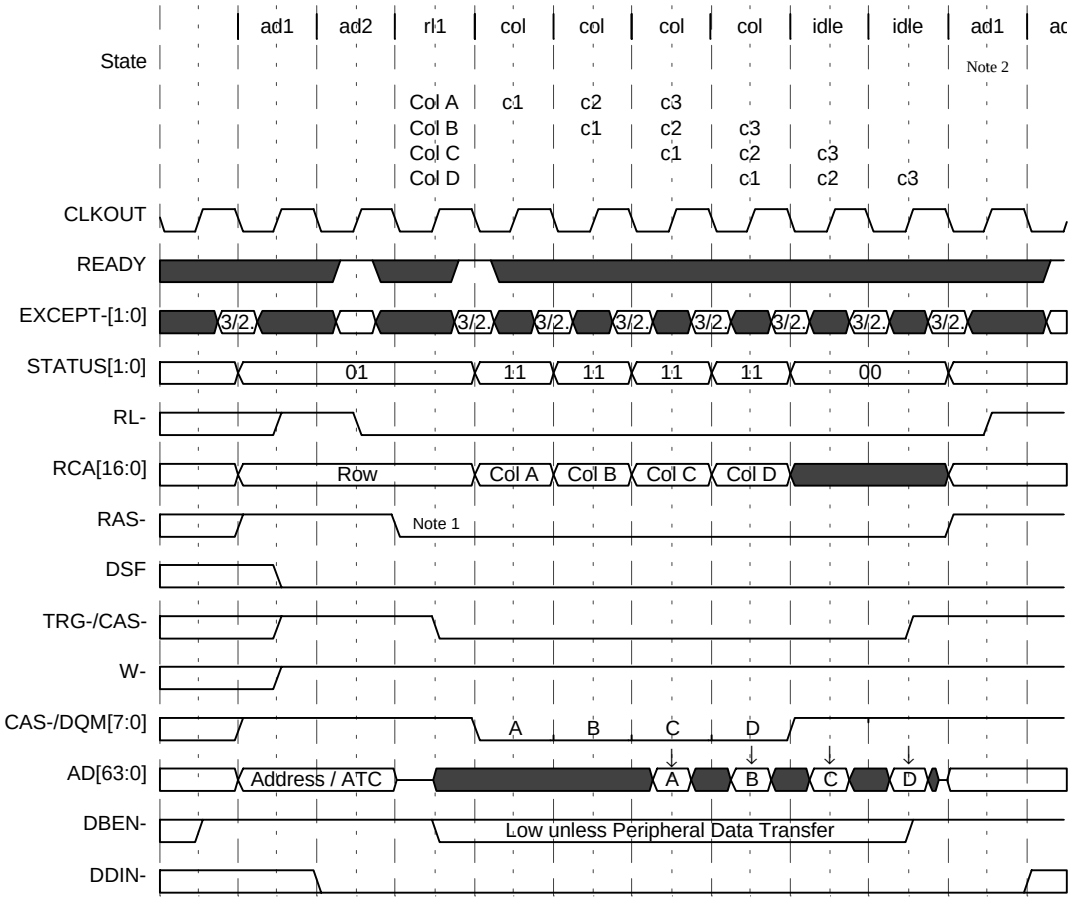
Figure 78. SRAM Cycle Column Pipelines

read cycles

SRAM read cycles transfer data or instructions from external memory to the ‘C82. SRAM read cycles are designed to interface with SRAM and other peripherals with SRAM-like I/O interfaces. During the cycle, /W is held high, /TRG/CAS is driven low after /RAS to enable memory output drivers, and /DBEN and /DDIN are low so that data transceivers may drive into the ‘C82. During column time, the TC places AD[63:0] into high impedance, allowing it to be driven by the memory, and latches input data during the appropriate column state. The TC always reads 64 bits and extracts and aligns the appropriate bytes. Invalid bytes for bus sizes of less than 64 bits are discarded. During peripheral device packet transfers, /DBEN and /DDIN remain high.

Exceptions are not supported in Figure 79, Figure 80, Figure 81, and Figure 82. Support for exceptions increases the minimum number of cycles between ad1 and the first column state from 3 to 4.

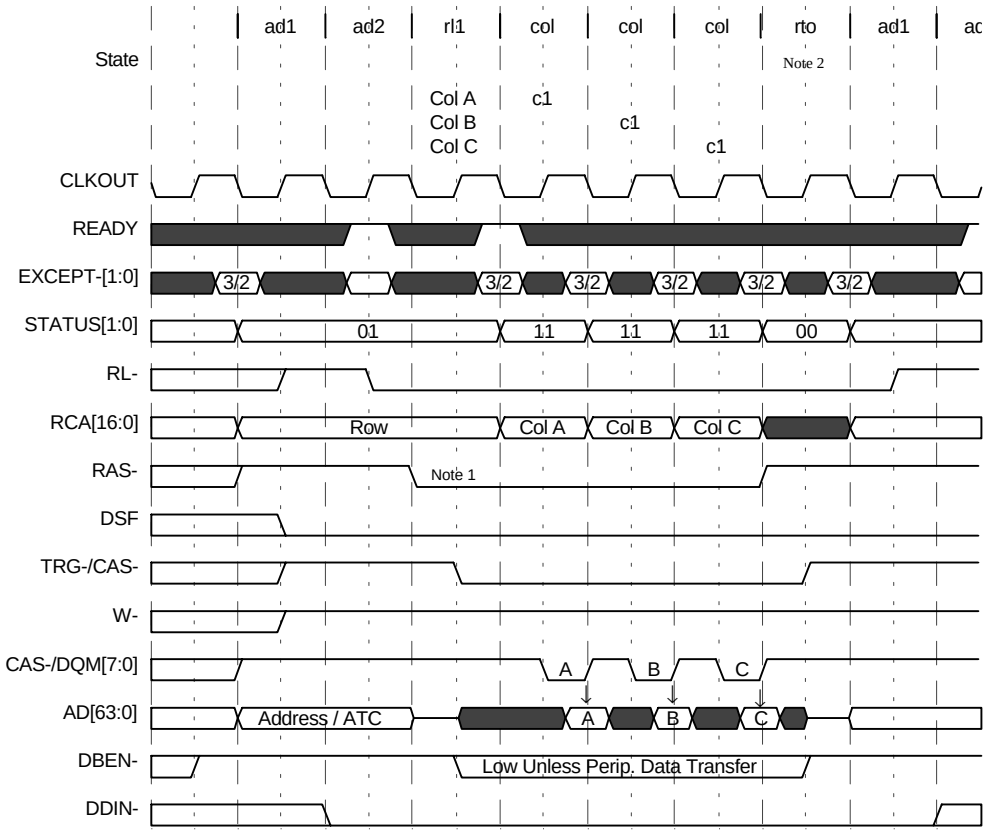
read cycles (continued)



Notes: 1. No RAS- high time requirements apply to these cycles.
2. Turnoff cycles will be inserted between rto and ad1 as specified by the bank configuration.

Figure 79. Synchronous SRAM Read Cycle

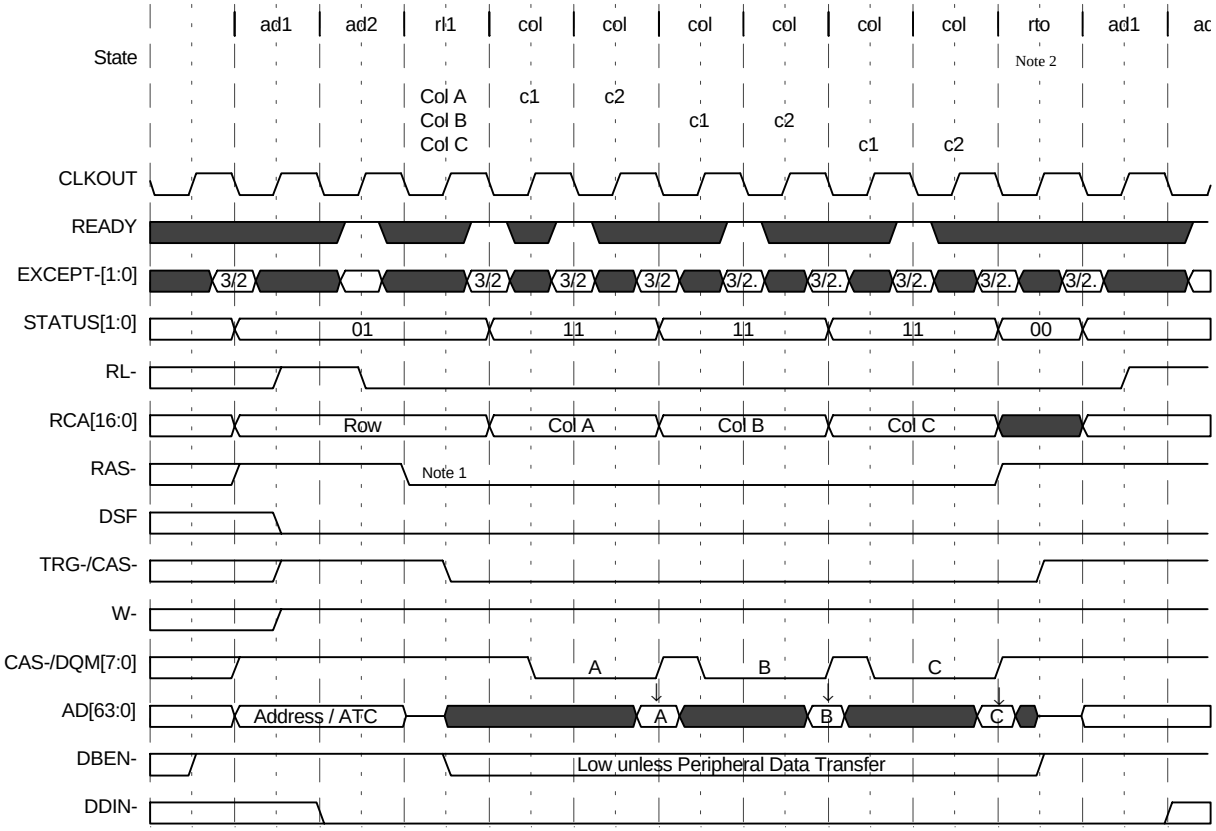
read cycles (continued)



Notes: 1. No RAS- high time requirements apply to these cycles.
2. Additional turnoff cycles will be inserted between rto and ad1 as specified by the bank configuration.

Figure 80. 1 cycle/column SRAM Read Cycle

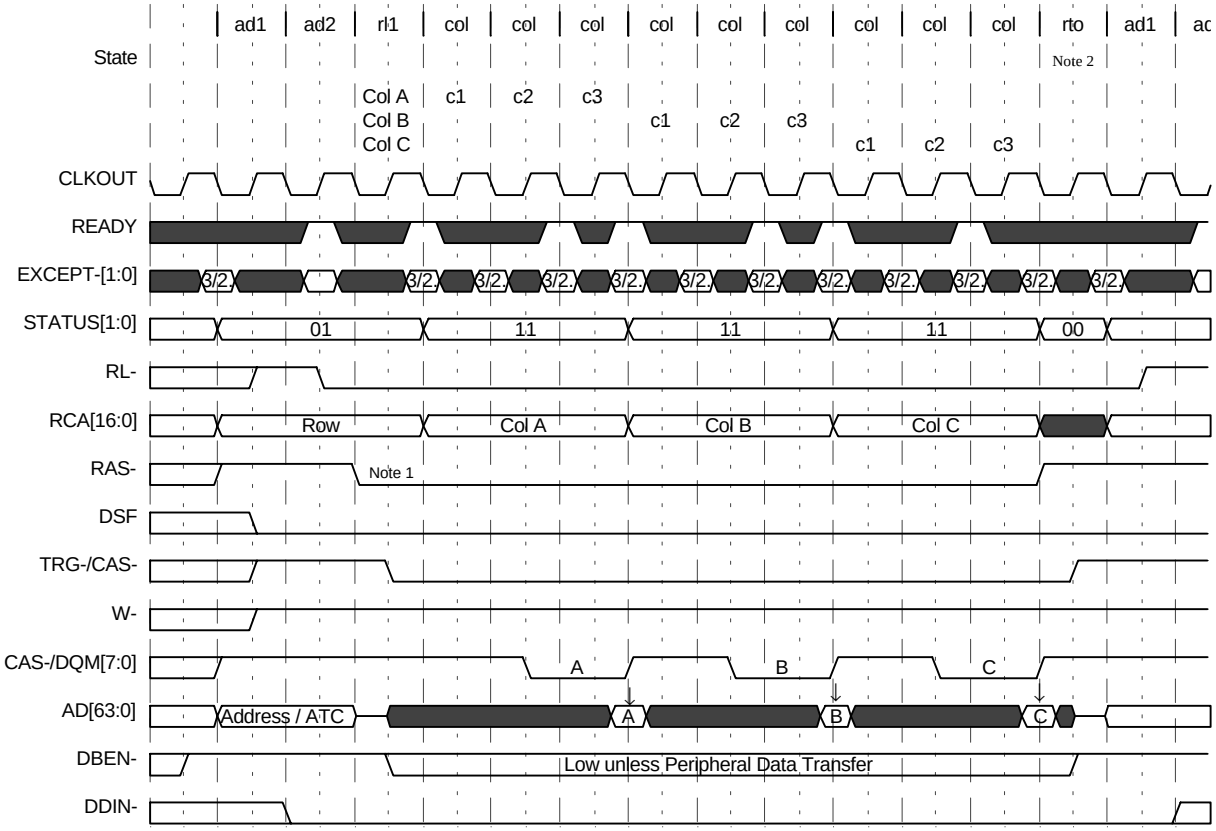
read cycles (continued)



Notes: 1. No RAS- high time requirements apply to these cycles.
2. Additional turnoff cycles will be inserted between rto and ad1 as specified by the bank configuration.

Figure 81. 2 cycle/column SRAM Read Cycle

read cycles (continued)



Notes: 1. No RAS- high time requirements apply to these cycles.
2. Additional turnoff cycles will be inserted between rto and ad1 as specified by the bank configuration.

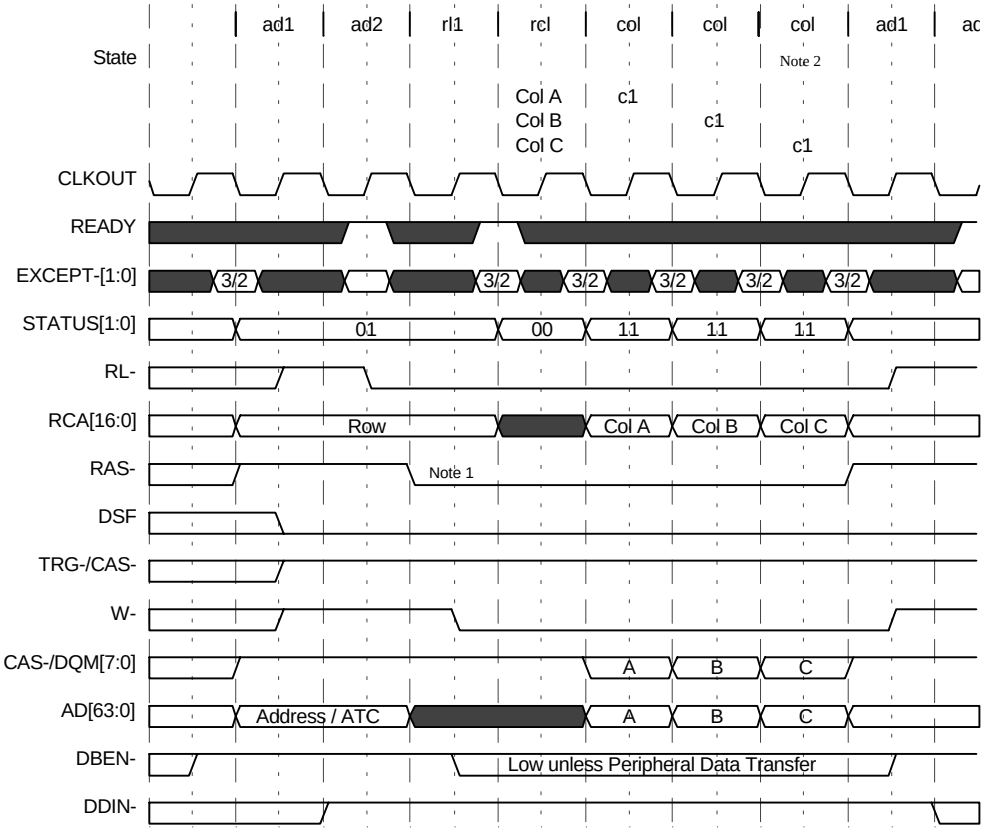
Figure 82. 3 cycle/column SRAM Read Cycle

write cycles

Write cycles transfer data from the ‘C82 to external memory. These cycles can occur as a result of a packet transfer, a DEA request, or an MP data cache write-back. During the cycle /TRG/CAS is held high, /W is driven low after the fall of /RAS to enable early write cycles, and /DDIN is high so that data transceivers drive toward memory. The TC drives data out on AD[63:0] and indicates valid bytes by activating the appropriate /CAS/DQM strobes. The /CAS/DQM signals may be used as chip enables (/CE) for SRAMs and peripherals (i.e., /CE-controlled writes). During peripheral device packet transfers, /DBEN remains high and AD[63:0] is placed in high impedance so that the peripheral device may drive data into the memory. Additionally, the number of turnoff cycles identified by the TO(1:0) field for the addressed bank of memory will be applied during PDT write cycles.

Exceptions are not supported in Figure 83, Figure 84, Figure 85, and Figure 86. Support for exceptions increases the minimum number of cycles between ad1 and the first column state from 4 to 6.

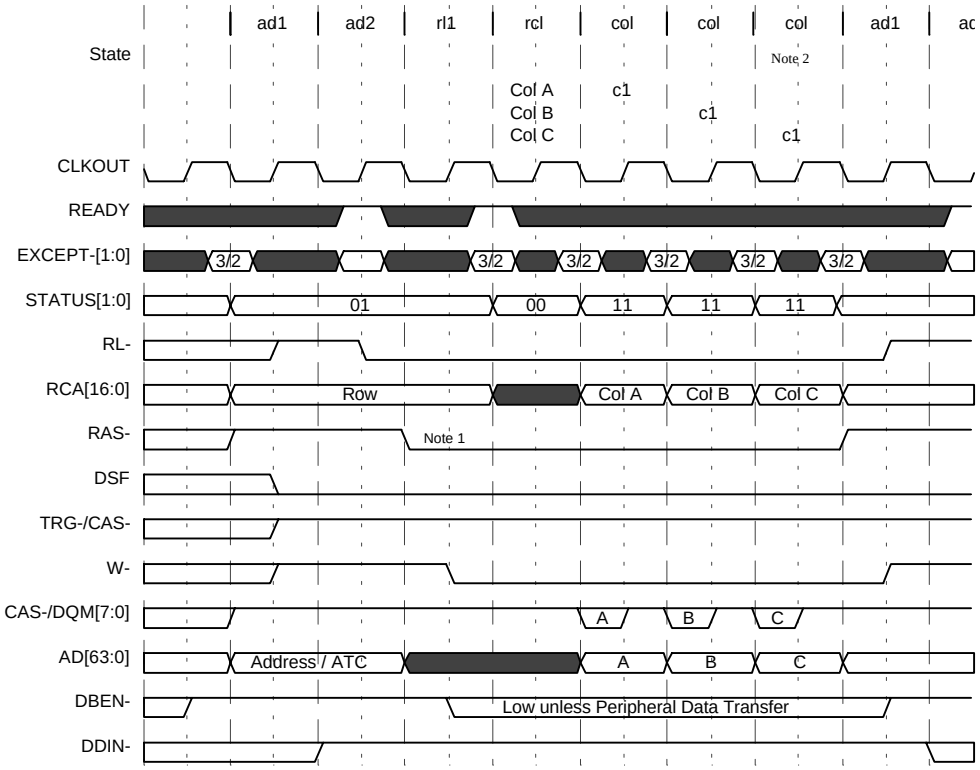
write cycles (continued)



Notes: 1. No RAS- high time requirements are applicable to these cycles.
2. During peripheral data transfer, turnoff cycles will be inserted prior to ad1 as specified by the bank configuration.

Figure 83. Synchronous SRAM Write Cycle

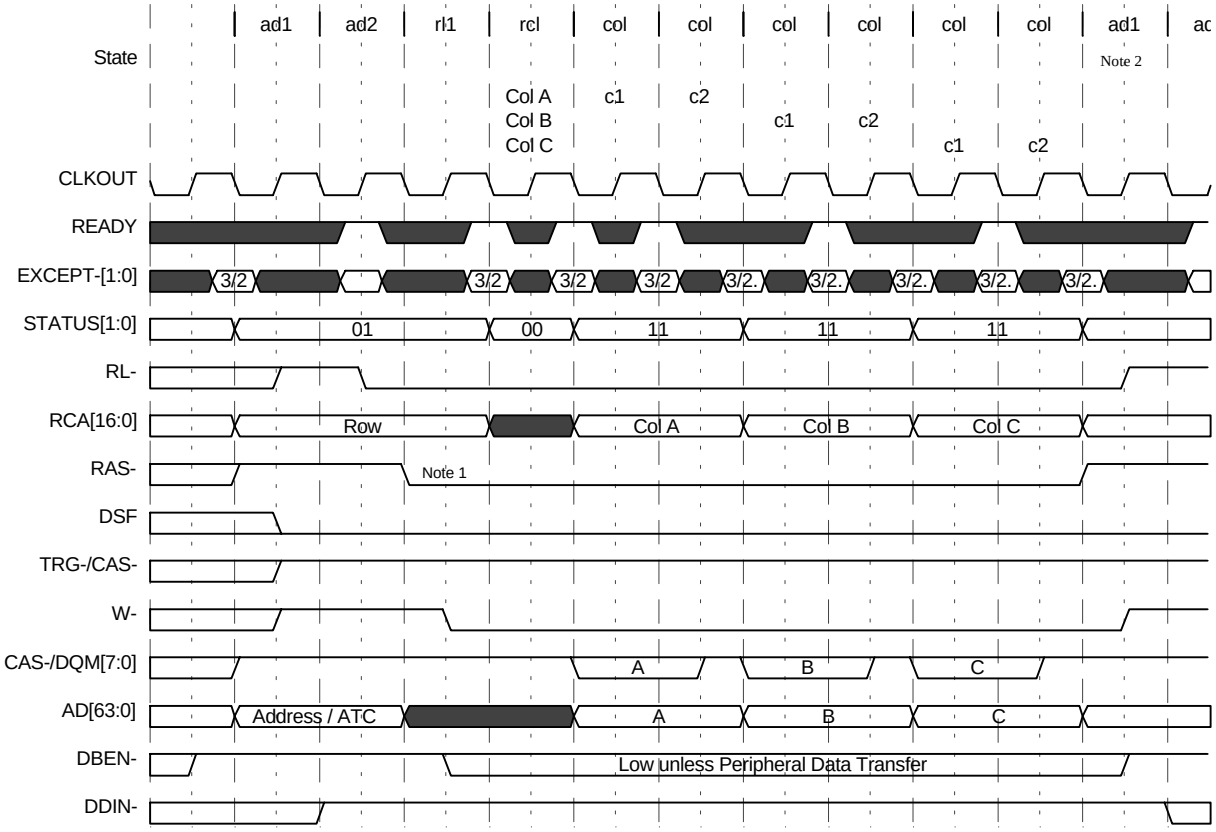
write cycles (continued)



Notes: 1. No RAS- high time requirements are applied to these cycles.
2. During peripheral data transfer, turnoff cycles will be inserted prior to ad1 as specified by the bank configuration.

Figure 84. 1 cycle/column SRAM Write Cycle

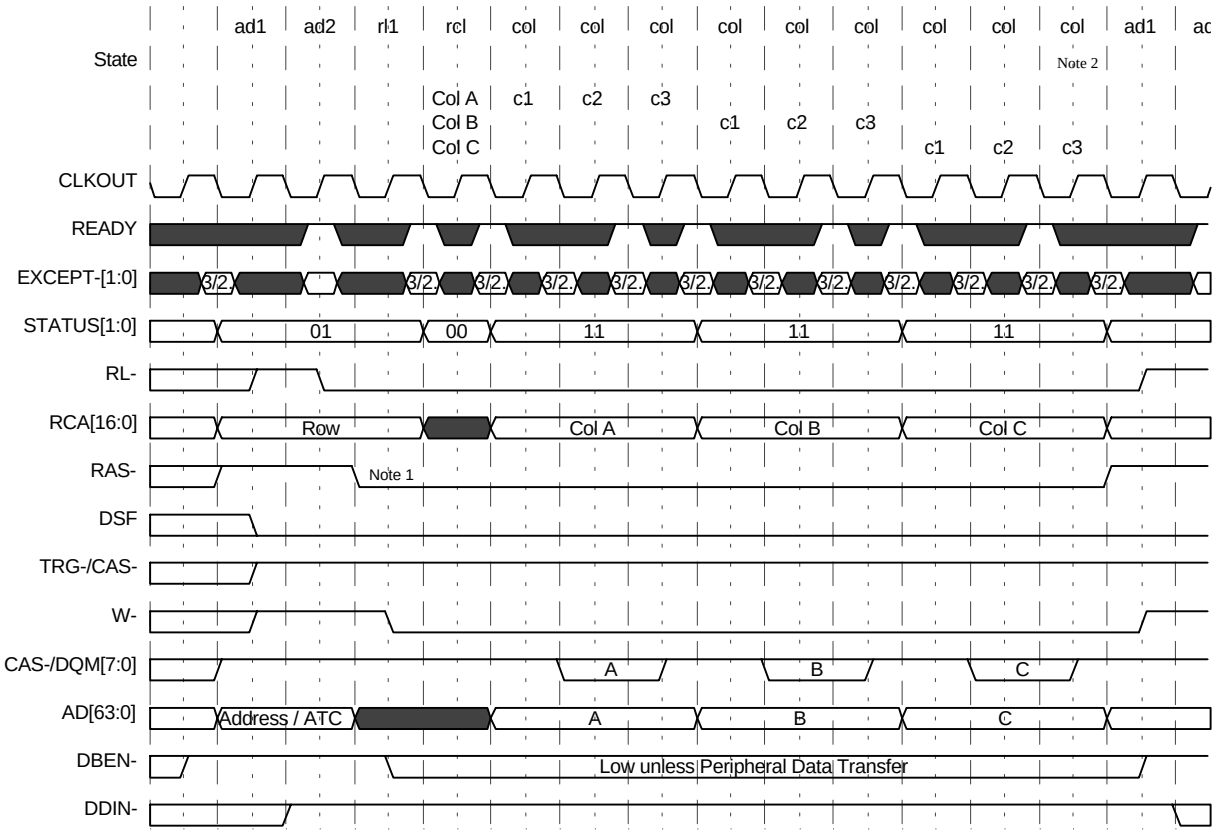
write cycles (continued)



Notes: 1. No RAS- high time requirements apply to these cycles.
2. During peripheral data transfer, turnoff cycles will be inserted prior to ad1 as specified by the bank configuration.

Figure 85. 2 cycle/column SRAM Write Cycle

write cycles (continued)



Notes: 1. No RAS- high time requirements are applied to these cycles.
2. During peripheral data transfer, turnoff cycles will be inserted prior to ad1 as specified by the bank configuration.

Figure 86. 3 cycle/column SRAM Write Cycle

SDRAM cycles

The SDRAM cycles support the use of SDRAM and SGRAM devices for single-cycle memory accesses. While SDRAM cycles use the same state sequences as DRAM cycles, the memory control signal transitions are modified to perform SDRAM command cycles. The supported SDRAM commands are:

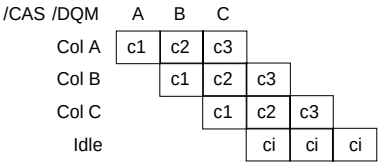
- DCAB Deactivate (precharge) all banks
- ACTV Activate the selected bank and select the row
- READ Input starting column address and start read operation
- WRT Input starting column address and start write operation
- MRS Set SDRAM mode register
- REFR Auto-refresh cycle with internal address
- SRS Set special register (color register)
- BLW Block write

SDRAM cycles begin with an activate (ACTV) command followed by the requested column accesses. When a memory page change occurs, the selected bank is deactivated with a DCAB command.

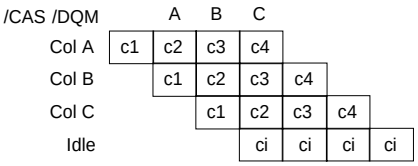
The TMS320C82 supports CAS latencies of 2, 3, or 4 cycles and burst lengths of 1 or 2. These are selected by the CT field read in during a bank-configuration cycle for the given bank. It should be noted that CAS latency 4 accesses are intended for use with CAS latency 3 SDRAM-like devices.

The column pipelines for SDRAM accesses are shown in Figure 87. Idle cycles can occur after necessary column accesses have completed or between column accesses due to “bubbles” in the TC data flow pipeline. The pipeline diagrams show the pipeline stages for each access type and when the /CAS/DQM signal corresponding to the column access is activated.

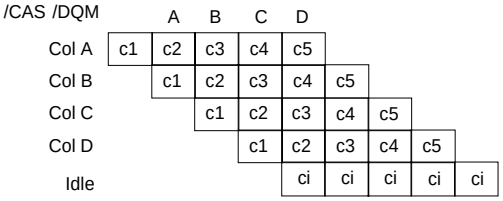
SDRAM cycles (continued)



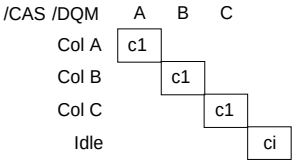
Burst length 1, 2 cycle latency (CT = 1000)
reads, read transfers, split-read transfers



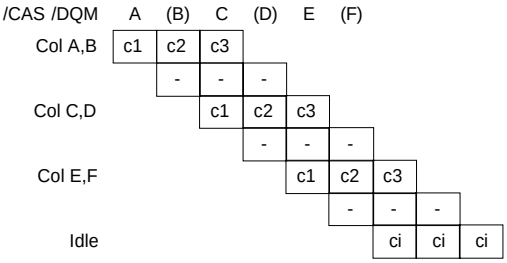
Burst length 1, 3 cycle latency (CT = 1001)
reads, read transfers, split-read transfers



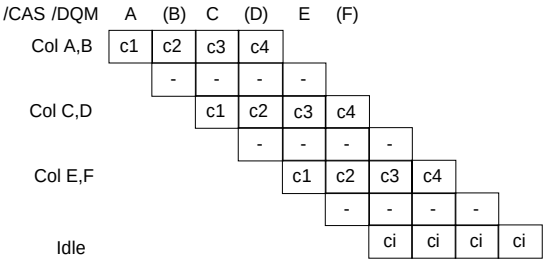
Burst length 1, 4 cycle latency (CT = 1010)
reads, read transfers, split-read transfers



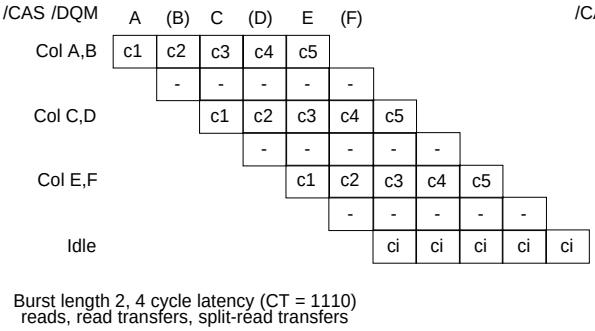
Burst length 1
writes, block writes, SRSs



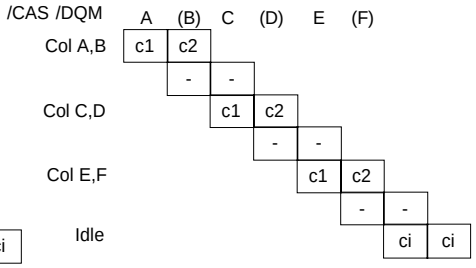
Burst length 2, 2 cycle latency (CT = 1100)
reads, read transfers, split-read transfers



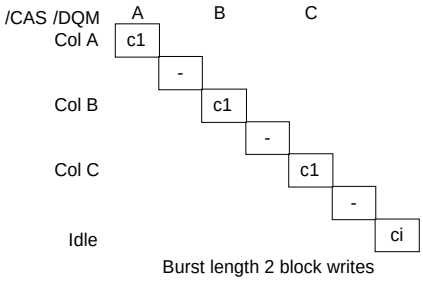
Burst length 2, 3 cycle latency (CT = 1101)
reads, read transfers, split-read transfers



Burst length 2, 4 cycle latency (CT = 1110)
reads, read transfers, split-read transfers



Burst length 2 writes

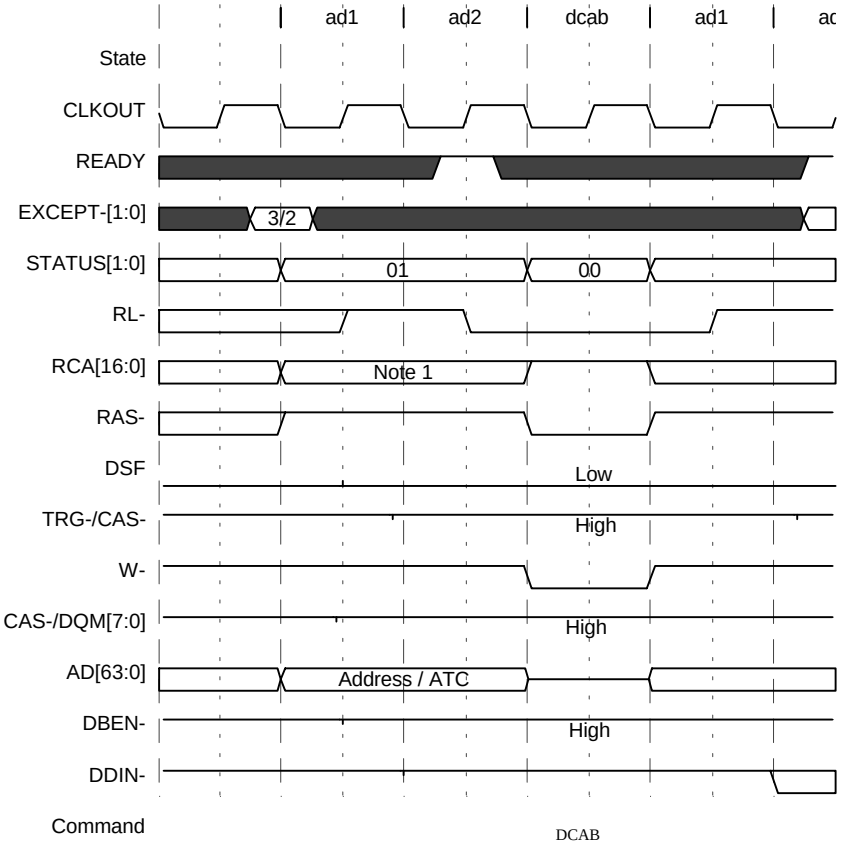


Burst length 2 block writes

Figure 87. SDRAM Column Pipelines

special SDRAM cycles

In order to properly initialize SDRAM, the TMS320C82 provides support for two special SDRAM cycles. During the power-up refresh sequence, the first refresh cycle that receives an SDRAM bank code (/EXCEPT[1:0] = 11) will be abandoned and a power-up deactivate (DCAB) cycle is performed. Only one DCAB operation is performed, so it is expected that all SDRAM banks will be decoded for this operation. The ‘C82 also performs an MRS cycle immediately following a bank-configuration cycle if that cycle returned a SDRAM CT code. No further MRS cycles for that bank are performed as long as the bank configuration remains cached.



Note 1: The row address is a don’t care.

Figure 88. SDRAM Power-up Deactivate

special SDRAM cycles (continued)

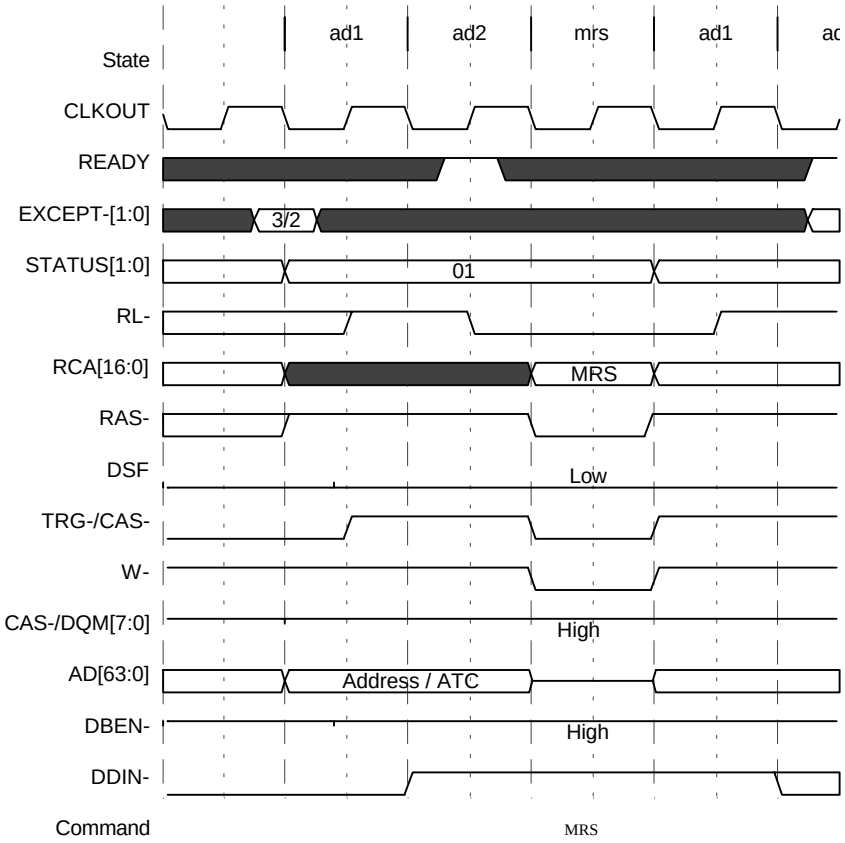


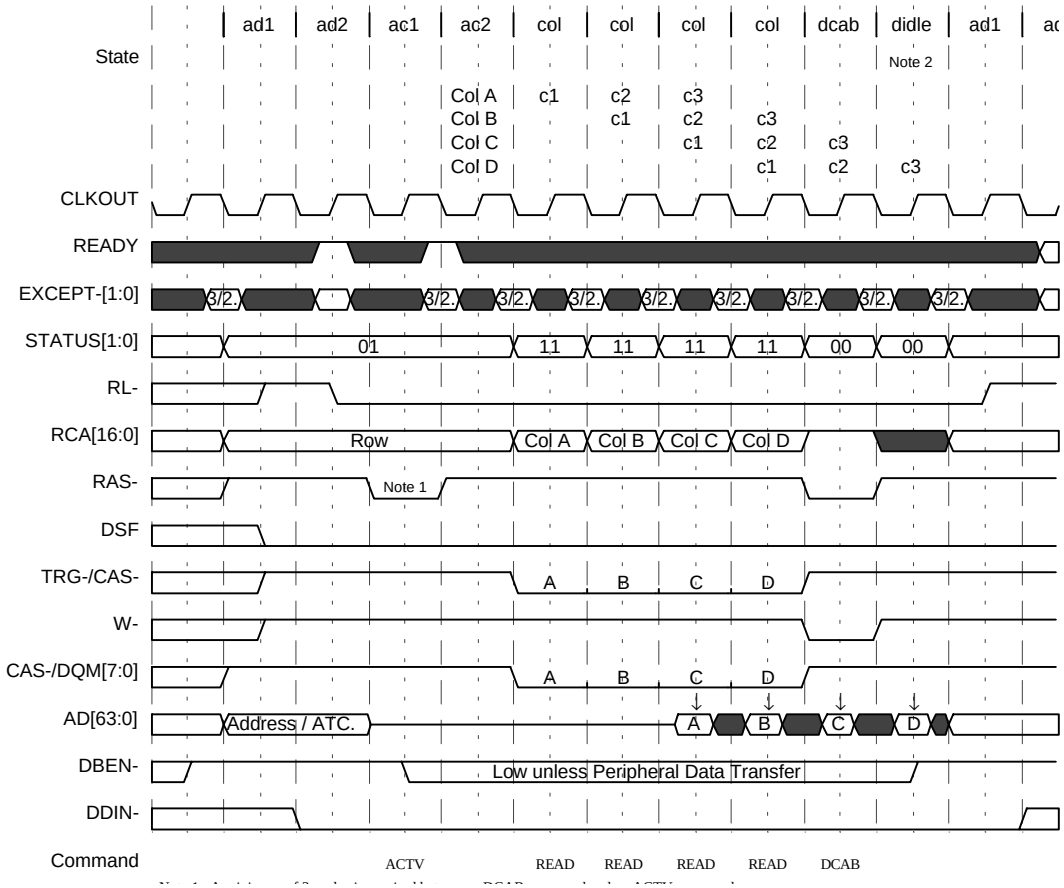
Figure 89. SDRAM Mode Register Set

SDRAM read cycles

Read cycles begin with an activate command to activate the bank and select the row. The TC outputs the column address and activates the /TRG/CAS strobe for each read command. For burst length 1 accesses, a read command can occur on each cycle. For burst length 2 accesses, a read command may occur every two cycles. During column time, the TC places AD[63:0] into high impedance, allowing it to be driven by the memory and latches input data during the appropriate column state. The TC always reads 64 bits and extracts and aligns the appropriate bytes. Invalid bytes for bus sizes of less than 64 bits are discarded. The /CAS/DQM strobes are activated two cycles before input data is latched (three cycles before in the case of CAS latency 4 accesses). If the second column in a burst is not required, then /CAS/DQM is not activated. During peripheral device packet transfers, /DBEN remains high.

For SDRAM reads, the minimum number of cycles between ad1 and the first column access is four whether exceptions are supported or not.

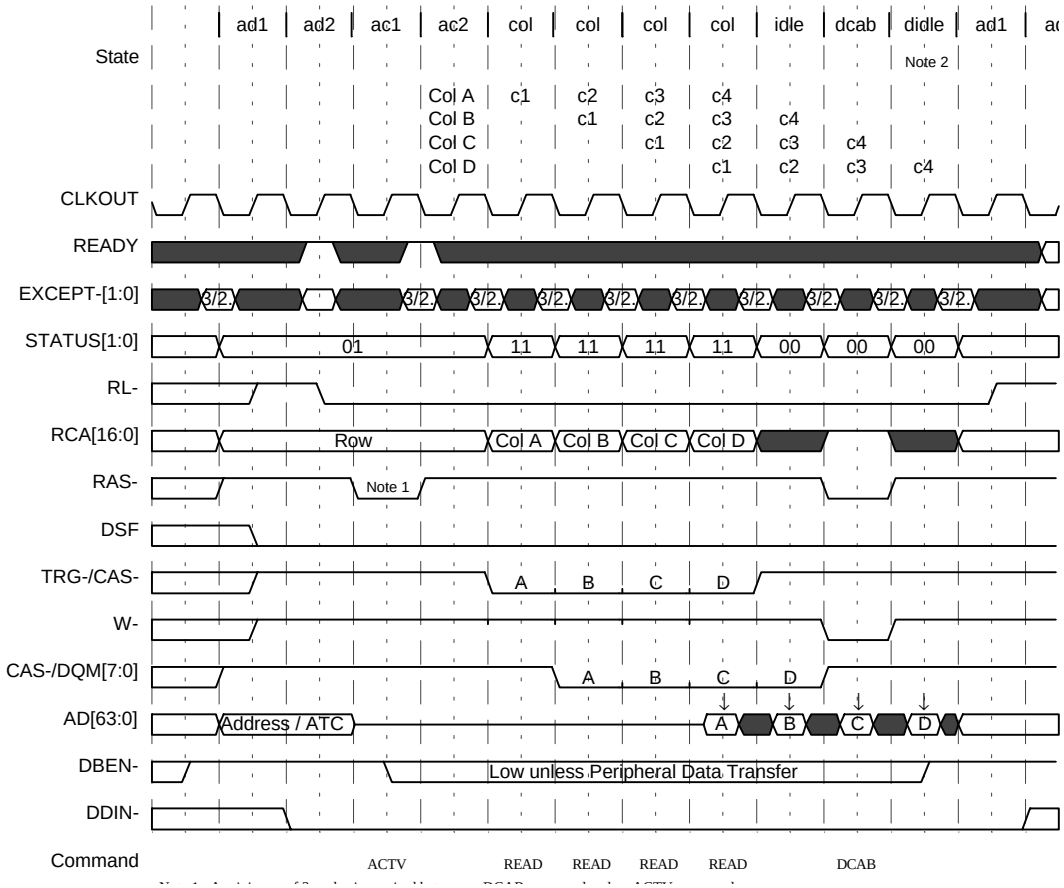
SDRAM read cycles (continued)



Notes: 1. A minimum of 3 cycles is required between a DCAB command and an ACTV command.
2. Turnoff cycles will be inserted between didle and ad1 as specified by the bank configuration.

Figure 90. SDRAM Burst Length 1, 2 Cycle Latency Read

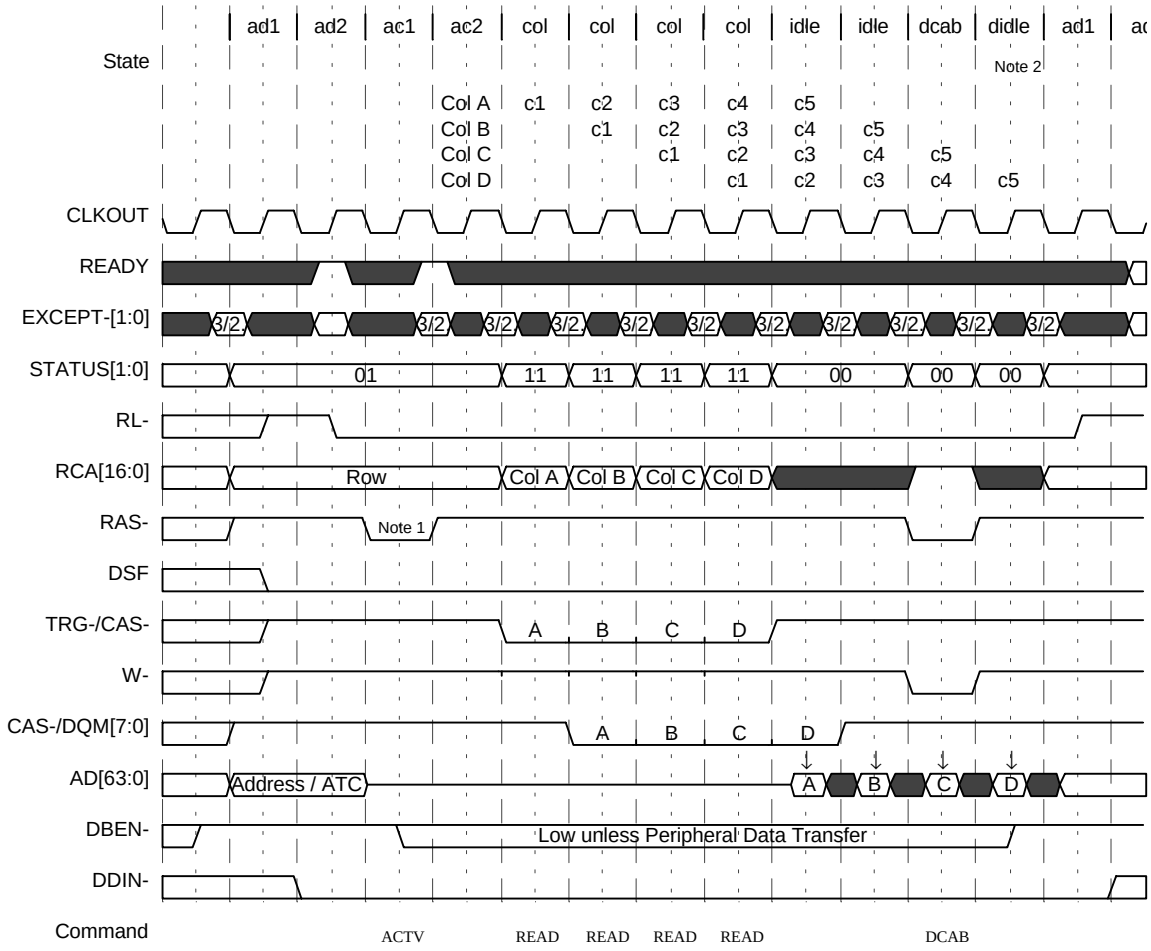
SDRAM read cycles (continued)



Notes: 1. A minimum of 3 cycles is required between a DCAB command and an ACTV command.
2. Turnoff cycles will be inserted between didle and ad1 as specified by the bank configuration.

Figure 91. SDRAM Burst Length 1, 3 Cycle Latency Read

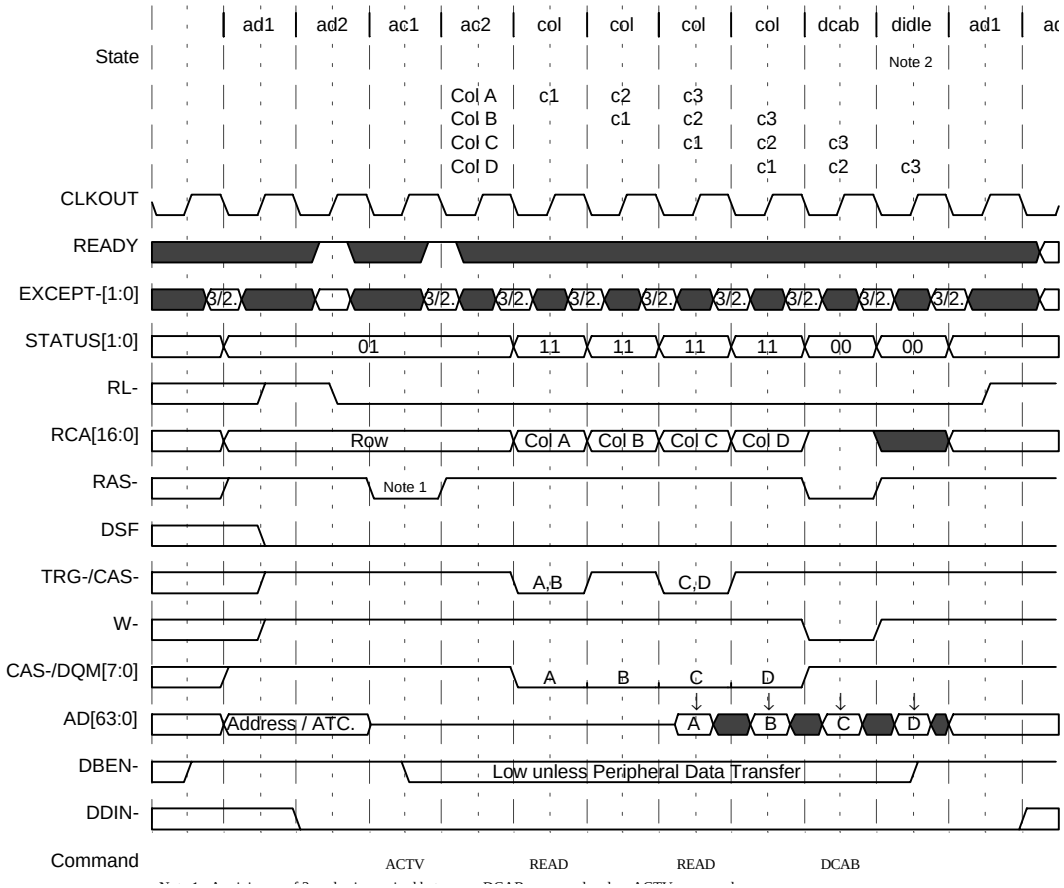
SDRAM read cycles (continued)



Notes: 1. A minimum of 3 cycles is required between a DCAB command and an ACTV command.
2. Turnoff cycles will be inserted between didle and ad1 as specified by the bank configuration.

Figure 92. SDRAM Burst Length 1, 4 Cycle Latency Read

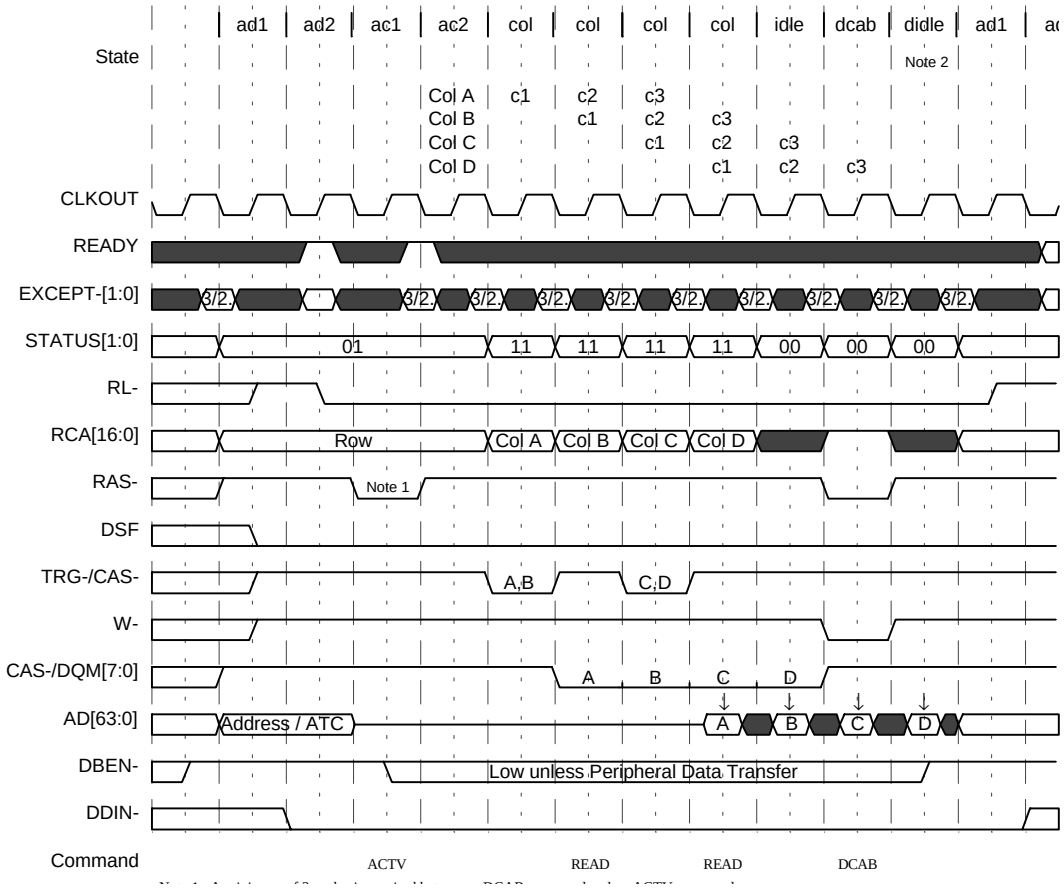
SDRAM read cycles (continued)



Notes: 1. A minimum of 3 cycles is required between a DCAB command and an ACTV command.
2. Turnoff cycles will be inserted between didle and ad1 as specified by the bank configuration.

Figure 93. SDRAM Burst Length 2, 2 Cycle Latency Read

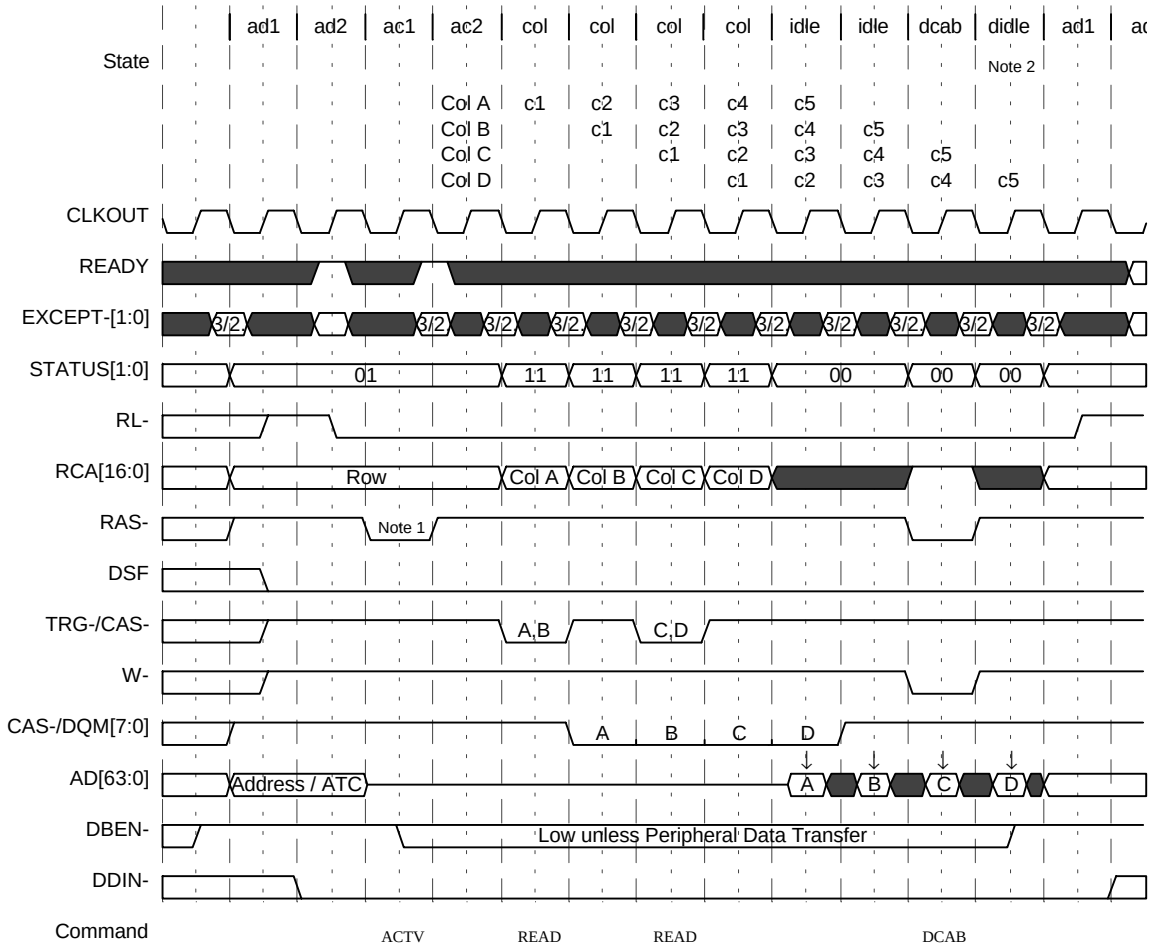
SDRAM read cycles (continued)



Notes: 1. A minimum of 3 cycles is required between a DCAB command and an ACTV command.
2. Turnoff cycles will be inserted between didle and ad1 as specified by the bank configuration.

Figure 94. SDRAM Burst Length 2, 3 Cycle Latency Read

SDRAM read cycles (continued)



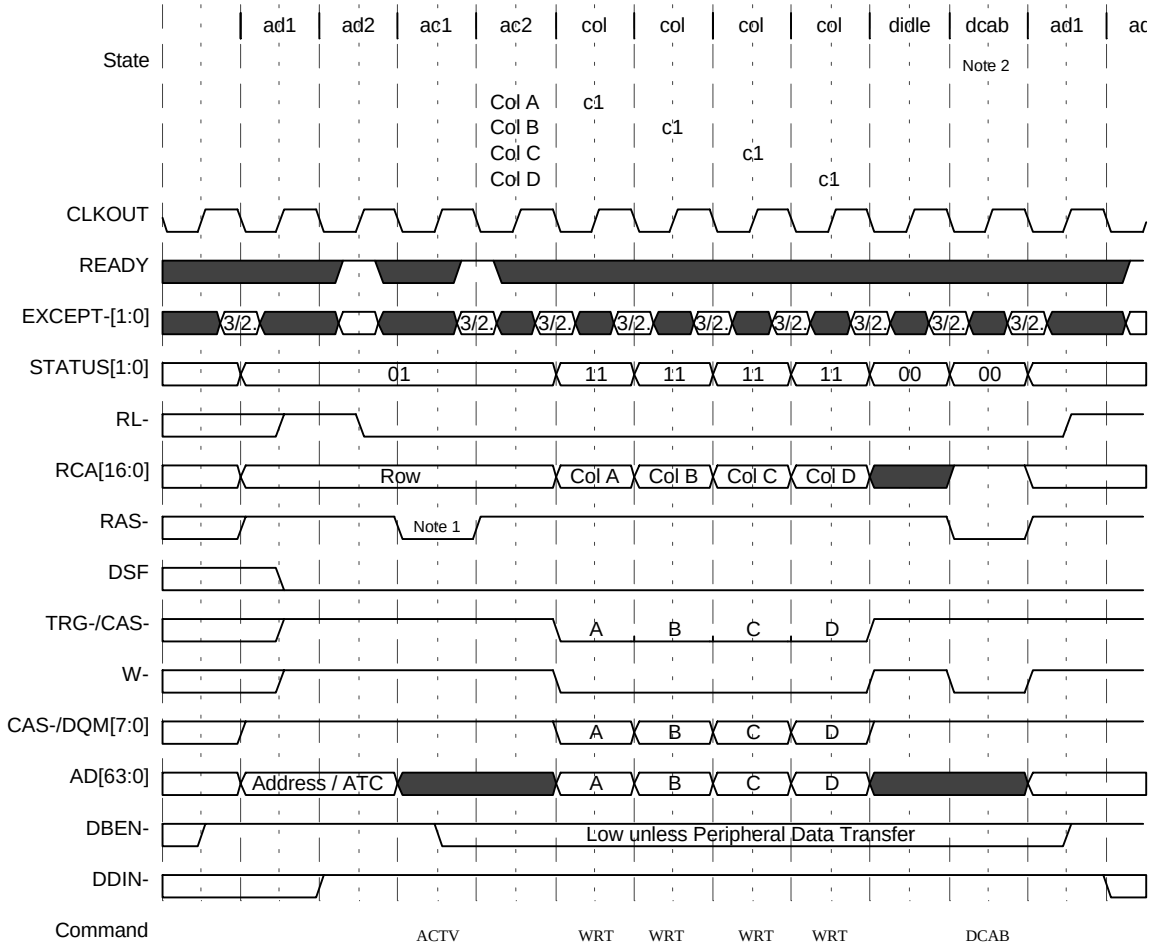
Notes: 1. A minimum of 3 cycles is required between a DCAB command and an ACTV command.
2. Turnoff cycles will be inserted between didle and ad1 as specified by the bank configuration.

Figure 95. SDRAM Burst Length 2, 4 Cycle Latency Read

SDRAM write cycles

Write cycles begin with an activate command to activate the bank and select the row. The TC outputs the column address and activates the /TRG/CAS and /W strobes for each write command. For burst length 1 accesses, a write command can occur on each cycle. For burst length 2 accesses, a write command may occur every two cycles. The TC drives data out on AD[63:0] during each cycle of an active write command and indicates valid bytes by driving the appropriate /CAS/DQM strobes low. During peripheral device packet transfers, /DBEN remains high and AD[63:0] is placed in high impedance so that the peripheral may drive data into the memories. Additionally, the number of turn off cycles identified by the TO(1:0) field for the addressed bank of memory will be applied during PDT write cycles.

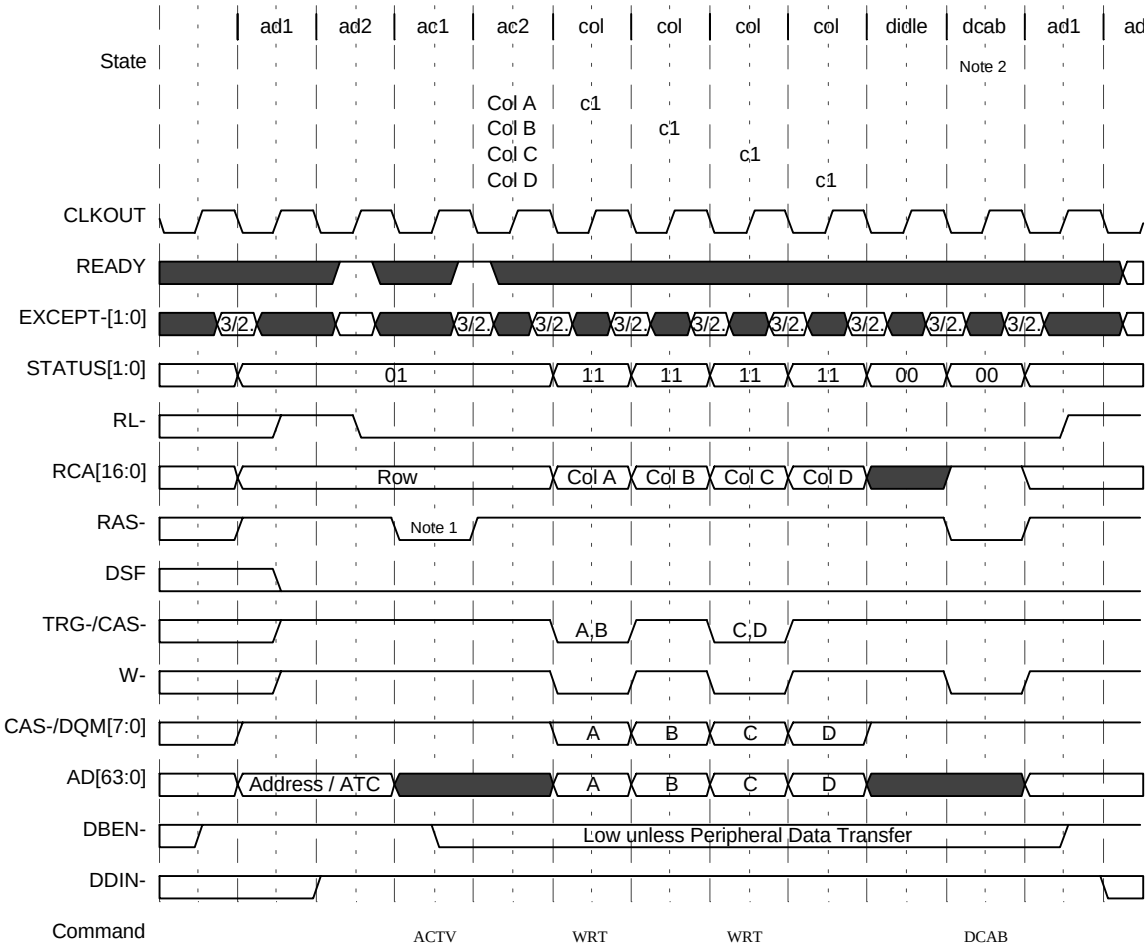
Exceptions are not supported in Figure 96 and Figure 97. Support for exceptions increases the minimum number of cycles between ad1 and the first column state from 4 to 6.



Notes: 1. A minimum of 3 cycles is required between a DCAB command and an ACTV command.
2. During peripheral data transfers, turnoff cycles will be inserted prior to ad1 as specified by the bank configuration.

Figure 96. SDRAM Burst Length 1 Write

SDRAM write cycles (continued)



Notes: 1. A minimum of 3 cycles is required between a DCAB command and an ACTV command.
2. During peripheral data transfers, turnoff cycles will be inserted prior to ad1 as specified by the bank configuration.

Figure 97. SDRAM Burst Length 2 Write

special register set cycle

Special register set (SRS) cycles are used to program control registers within an SGRAM. The ‘C82 only supports programming of the color register for use with block writes. The cycle is similar to a single-burst-length-1 write cycle but DSF is driven high. The values output on the ‘C82’s RCA[16:0] bus causes the color register to be selected as shown in Figure 98. The color register value is output on the AD[63:0] bus.

Exception are not supported in Figure 99; support for exceptions increases the number of cycles between ad1 and the column access from 4 to 6.

SDRAM Addr Pin	BS	A8	A7	A6	A5	A4	A3	A2	A1	A0
SDRAM Function	0	0	0	LC	LM	LS	Stop Reg			
'C82 Output Value	0	0	0	1	0	0	0	0	0	0

Figure 98. Special Register Set Value

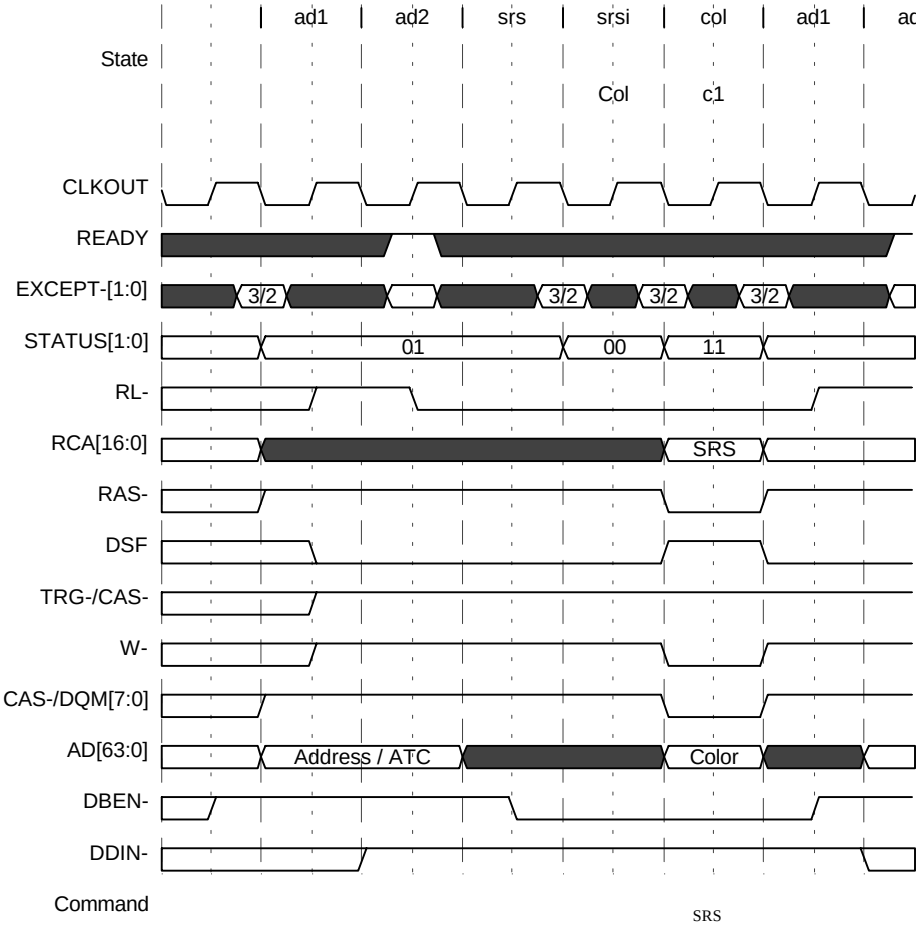
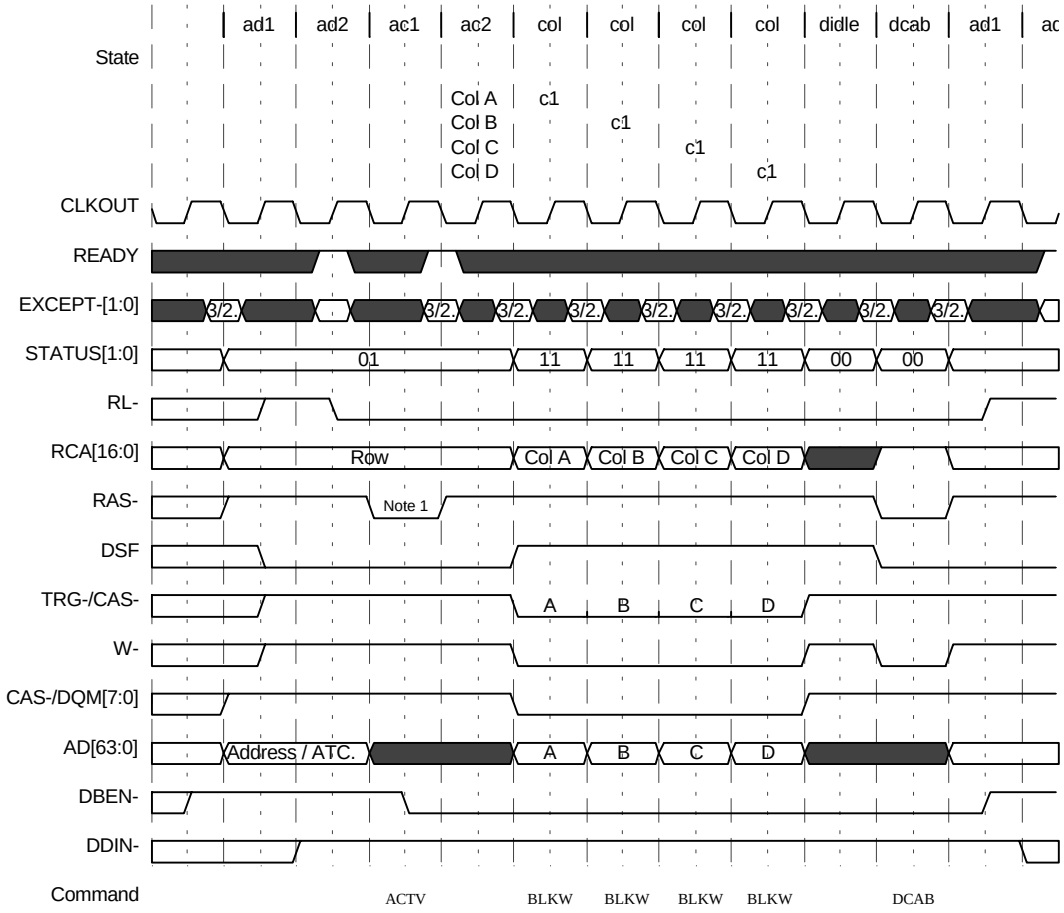


Figure 99. SDRAM SRS Cycle

SDRAM block-write cycles

Block-write cycles allow SGRAMs to write a stored color value to multiple column locations in a single access. Block-write cycles are similar to write cycles except that DSF is driven high to indicate a block-write command. Because burst is not supported for block write, burst length 2 accesses generate a single block write every other clock cycle.

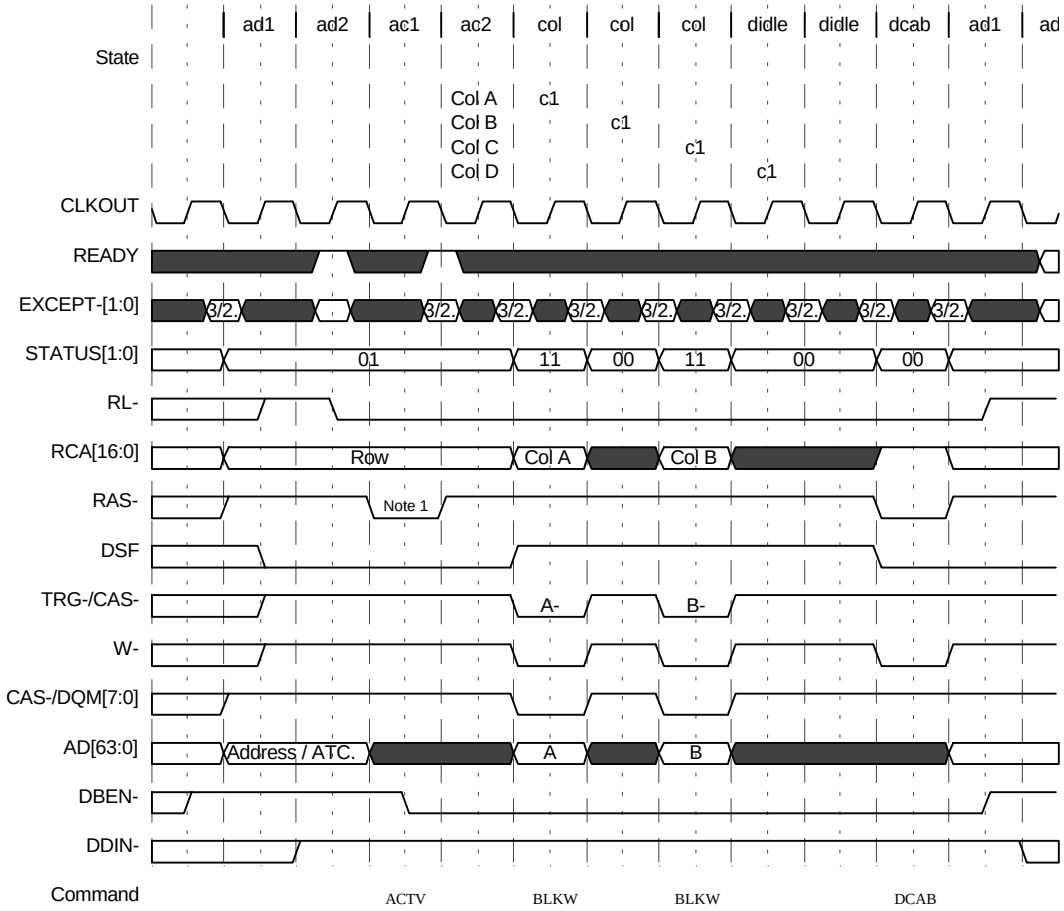
Exceptions are not supported in Figure 100 and Figure 101. Support for exceptions increases the minimum number of cycles between ad1 and the first column state from 4 to 6.



Note 1: A minimum of 3 cycles is required between a DCAB command and an ACTV command.

Figure 100. SDRAM Burst Length 1 Block Write

SDRAM block-write cycles (continued)

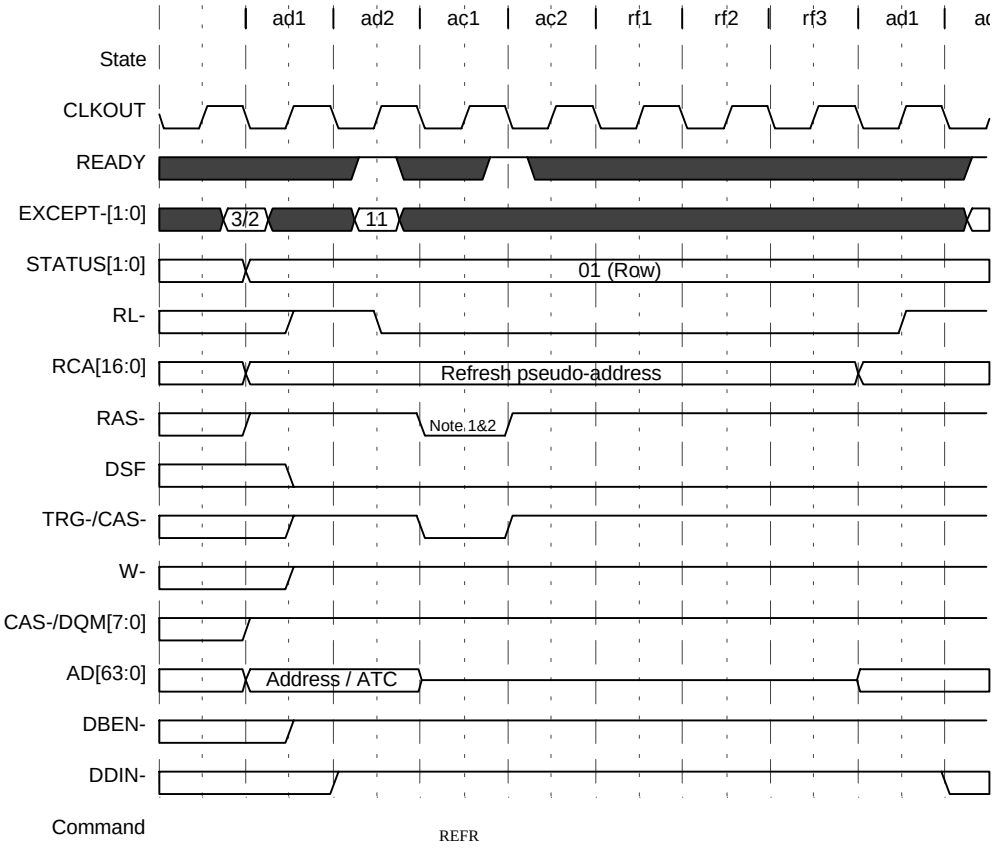


Note 1: A minimum of 3 cycles is required between a DCAB command and an ACTV command.

Figure 101. SDRAM Burst Length 2 Block Write

SDRAM refresh cycle

The SDRAM refresh cycle is performed when the TC receives an SDRAM cycle timing input (/EXCEPT[1:0] = 11) at the start of a refresh cycle. The /RAS and /TRG/CAS outputs are driven low for 1 cycle to strobe a refresh command (REFR) into the SDRAM. The refresh address is generated internal to the SDRAM. The ‘C82 outputs a 16-bit pseudo-address (used for refresh bank decode) on RCA[16:1]. The pseudo-address is decremented once for each refresh that is performed.



Notes: 1. A minimum of three cycles is required between the CLKOUT edges of a DCAB command and a subsequent ACTV command.
2. A minimum of seven cycles is required between the CLKOUT edges of a REFR command and a subsequent REFR or ACTV command.

Figure 102. SDRAM Refresh

bank configuration cycle

The ‘C82 bank configuration cycle is required each time a new bank of memory is to be accessed whose configuration is not in the memory-configuration cache. The memory cache contains six entries. Entry replacement is based on a multiple-level least-recently-used algorithm. The least-recently-used lowest-priority entry will be flushed from the cache if all six entries are used.

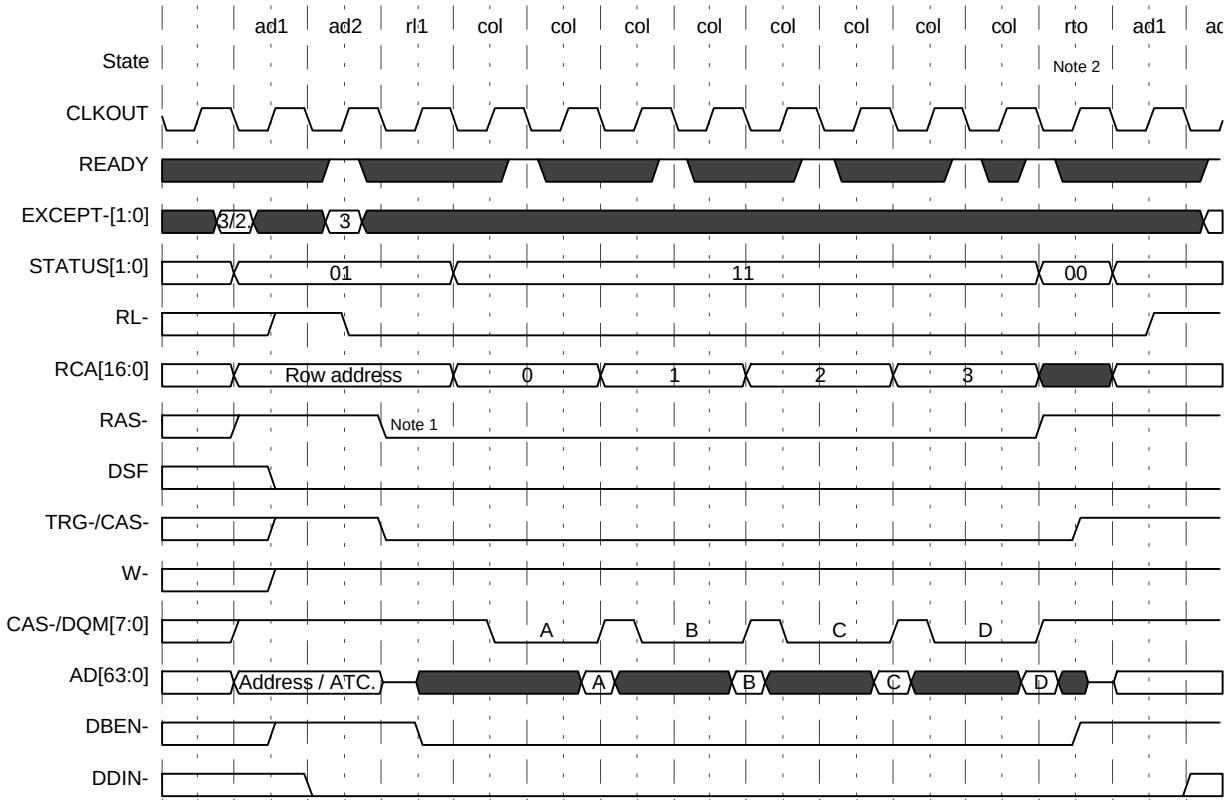
Bank configuration cycles may occur as the result of a cache miss, packet transfer, DEA, or may be requested by the system by inputting an exception code of 00 on the /EXCEPT[1:0] inputs during the ad2 state of a memory access. This exception code forces the bank configuration to be flushed and a new configuration for that bank to be read in. This is particularly useful for systems that use “shadow mapping” to decode two different memory types to the same address.

A configuration cache fill resembles a normal read cycle, with a few exceptions. Most notable are:

- Accesses are byte width only.
- Only four accesses are performed.

In four consecutive reads, the bank configuration fields described in Figure 55 are read in over either AD[63:56] (big-endian mode) or AD[7:0] (little-endian mode). This sequence is atomic, and no pipeline bubbles will occur. The RCA bus outputs the sequence 00, 01, 02, and 03 for the four accesses. The configuration cache cycle is indicated by a row time status code of 1110 output on AD[35:32]. It is anticipated that external logic will latch that status code (using /RL) and several upper address bits (AD[31:xx]) and decode RCA[1:0] to respond with the appropriate bank configuration information.

bank configuration cycle (continued)



Notes: 1. No RAS- high time requirements are applied to these cycles.
2. Additional turnoff cycles can be inserted by adding waitstates during the rto (turnoff) cycle. This capability is unique to this cycle.

Figure 103. Bank Configuration Cycle

host interface

The ‘C82 contains a simple three-pin mechanism by which a host or other device may gain control of the ‘C82 local memory bus. The /HREQ input may be driven low by the host to request the ‘C82’s bus. Once the TC has completed the current memory access, it will place the local bus (except CLKOUT) into a high-impedance state. It will then drive the /HACK output low to indicate that the host device owns and may drive the bus. The REQ output reflects the highest-priority cycle request being received internally by the TC. The host can monitor this output to determine if It needs to relinquish the local bus back to the ‘C82.

Table 49. REQ Output

REQ	ASSOCIATED INTERNAL TC REQUEST
1	Urgent refresh or XPT
0	All other activity

device reset

The TMS320C82 is reset when the /RESET input is driven low. The ‘C82 outputs will immediately go into a high-impedance state with the exception of CLKOUT, /HACK, and REQ. While /RESET is low, all internal register are set to their default values and internal logic is reset.

On the rising edge of /RESET, the state of READY is sampled to determine if big-endian (READY=0) or little-endian (READY=1) operation is selected. The state of /HREQ is also sampled to determine if the master processor will come up running (/HREQ=0) or halted (/HREQ=1). All other inputs and data lines are don’t cares during device reset.

Once /RESET is high, the ‘C82 will drive the high-impedance signals to their inactive values. The TC will then perform 32 refresh cycles to initialize system memory. If, during initialization refresh, the TC receives an SDRAM cycle timing code (/EXCEPT[1:0] = 11), it will perform an SDRAM DCAB cycle to initialize SDRAM and then continue with the refresh cycles.

After completing initialization refresh, if the MP is running, the TC will perform a bank configuration cycle for the bank at address 0xFFFFFFC0. This is the cache subblock which contains the starting MP instruction located at 0xFFFFFFF8. If the MP comes up halted, the configuration cycle and instruction cache fills will not take place until the first occurrence of an /EINT3 interrupt to unhalt the MP.

absolute maximum ratings†

Supply voltage range, V_{DD} (see Note 1) ..... -0.3 V to 4 V
Input voltage range, V_I ..... -0.3 V to 4 V
Output voltage range ..... -0.3 V to 4 V
Operating case temperature range, T_C ..... 0°C to 85°C
Storage temperature range ..... -55°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: All voltage levels are with respect to ground (V_{SS}).

recommended operating conditions

	PARAMETER	MIN	NOM	MAX	UNIT
V_{DD}	Supply voltage	3.165	3.3	3.465	V
V_{DDPLL}	Phase-locked loop supply voltage (see Note 2)	3.165	3.3	3.465	V
V_{SS}	Supply voltage (see Note 3)	0	0	0	V
V_{SSPLL}	Phase-locked loop supply voltage (see Note 2)	0	0	0	V
I_{OH}	High-level output current			-400	µA
I_{OL}	Low-level output current			2	mA
T_C	Operating case temperature	0		85	°C

NOTES: 2. The VDDPLL pin should be supplied through an EMI filter coupled to VSSPLL. Care should be taken to provide a minimum inductance path between VSSPLL and system ground.
3. In order to minimize noise on VSS, care should be taken to provide a minimum inductance path between the VSS pins and system ground.

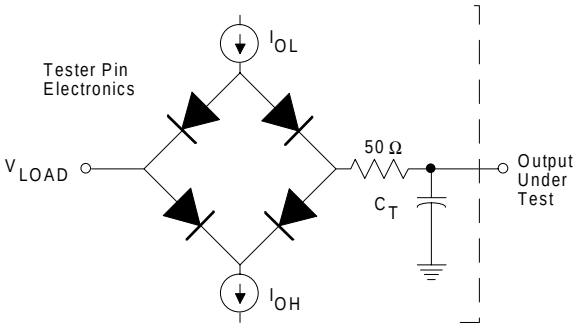
electrical characteristics over full ranges of supply voltage and operating case temperature (unless otherwise noted)

	PARAMETER	TEST CONDITIONS‡	MIN	TYP	MAX	UNIT
V_{IH}	High-level input voltage		2		V_{DD} + 0.3	V
V_{IL}	Low-level input voltage		-0.3		0.8	V
V_{OH}	High-level output voltage	V_{DD} = min, I_{OH} = max	2.4	§		V
V_{OL}	Low-level output voltage	V_{DD} = max, I_{OH} = min			0.6	V
I_O	Output current, leakage (high-impedance)	V_{DD} = max, V_O = 2.8V			20	µA
	(except EMU0, and EU1)	V_{DD} = max, V_O = 0.6V			-20	
I_I	Input current (except TCK, TDI, and TMS)	V_I = V_{SS} to V_{DD}			±20	µA
I_{DD}	Supply current (See Note 4)	V_{DD} = max, 60 MHz		1.2	2.2	A
		V_{DD} = max, 50 MHz		1.1	2.1	
I_{DDPLL}	PLL supply current				150	mA
C_I	Input capacitance			10		pF
C_O	Output capacitance			10		pF

NOTE 4: Maximum supply current is derived from a test case that generates the theoretical maximum data flow using a worst casecheckerboard data pattern on a sustained cycle-by-cycle basis. Typical supply current is derived from a test case which attempts to emulate typical use conditions of the on-chip processors with random data. Typical I_{DD} will vary from application to application based on data flow, transitions, and on-chip processor utilization.
For conditions shown as MIN/MAX, use the appropriate value specified under the recommended operating conditions.

‡ All typical values are at V_{DD} = 3.3 V, T_A = 25°C
§ Typical steady state V_{OH} will not exceed V_{DD}

PARAMETER MEASUREMENT INFORMATION



Where: I_{OL} = 2.0 mA (all outputs)
I_{OH} = 400 µA (all outputs)
V_{LOAD} = 1.5 V
C_T = 60 pF typical load circuit distributed capacitance

Figure 104. Test Load Circuit

signal transition levels

TTL-level outputs are driven to a minimum logic-high level of 2.4 V and to a maximum logic-low level of 0.6 V. Figure 105 shows the TTL-level outputs.

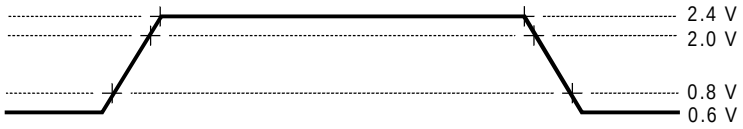


Figure 105. TTL Level Outputs

TTL-output transition times are specified as follows:

For a *high-to-low* transition on a TTL-compatible output signal, the level at which the output is said to be no longer high is 2 V, and the level at which the output is said to be low is 0.8 V.

For a *low-to high* transition, the level at which the output is said to be no longer low is 0.8 V, and the level at which the output is said to be high is 2 V.

Figure 106 shows the LVTTL-level inputs

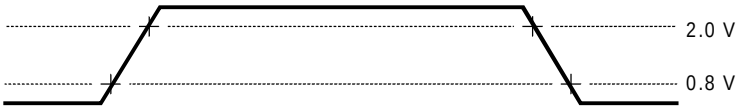


Figure 106. LVTTL Level Inputs

LVTTL-compatible input transition times are specified as follows:

For a *high-to-low transition* on a input signal, the level at which the input is said to be no longer high is 2 V, and the level at which the input is said to be low is 0.8 V.

For a *low-to-high transition* on an input signal, the level at which the input is said to be no longer low is 0.8 V, and the level at which the input is said to be high is 2 V.

timing parameter symbology

Timing parameter symbols used herein were created in accordance with JEDEC Standard 100-A. In order to shorten the symbols, some of the pin names and other related terminology have been abbreviated as follows:

A	AD[31:0]
AD	AD[63:0]
ADS	AD[39:32]
CAS	/CAS/DQM[7:0]
CKI	CLKIN
CKO	CLKOUT
CMP	/EXCEPT[1:0], READY
D	AD[63:0]
EIN	/EINT1, /EINT2, /EINT3
EMU	EMU0, EMU1
EXC	/EXCEPT[1:0]
HAK	/HACK
HRQ	/HREQ
LIN	/LINT4
MID	AD[63:0], RCA[16:0], STATUS[1:0]
OUT	AD[63:0], /CAS/DQM[7:0], /DBEN, /DDIN, DSF, /RAS, /RL, STATUS[1:0], /TRG/CAS, /W
STS	STATUS[1:0]
RAS	/RAS
RCA	RCA[16:0]
RDY	READY
RST	/RESET
REQ	REQ
RL	/RL
TCK	TCK
TDI	TDI
TDO	TDO
TMS	TMS
TRS	/TRST
XPT	/XPT[3:0]

LOWERCASE SUBSCRIPTS AND THEIR MEANINGS ARE:

a	access time
c	cycle time (period)
d	delay time
h	hold time
su	setup time
t	transition time
w	pulse duration (width)

THE FOLLOWING LETTERS AND SYMBOLS AND THEIR MEANINGS ARE:

D	Driven
H	High
L	Low
V	Valid
X	Unknown, changing, or don't-care level
Z	High impedance

general notes on timing parameters

The period of CLKOUT may be equal to, or is twice the period of CLKIN ($t_{c(CKI)}$) depending on whether or not PLL mode is selected. In PLL mode, CLKOUT is the same period as CLKIN. In non-PLL mode, the period of CLKOUT will be twice the period of CLKIN, or $2t_{c(CKI)}$. The half cycle time (t_H) that appears in the following tables is one-half the output clock period.

All output signals from the ‘C82 (including CLKOUT) are derived from an internal clock such that all output transitions for a given half-cycle occur with a minimum of skewing relative to each other.

The signal combinations shown in the following timing diagrams may not necessarily represent actual cycles. For actual cycle examples, refer to the appropriate cycle description section of this data sheet.

CLKIN timing requirements

NO.	PARAMETER		C82-50		C82-60		UNIT
			MIN	MAX	MIN	MAX	
1	$t_{c(CKI)}$	Period of CLKIN	10.0		8.3		ns
2	$t_{w(CKIH)}$	Pulse duration of CLKIN high	4.2		3.9		ns
3	$t_{w(CKIL)}$	Pulse duration of CLKIN low	4.2		3.9		ns
4	$t_{t(CKI)}$	Transition time of CLKIN †		1.5		1.5	ns
5	F_{LOCK}	Phased-locked loop lock range	40	80	40	80	MHz

† This parameter is verified by computer simulation and is not tested.

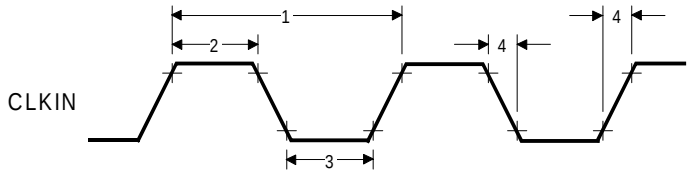


Figure 107. CLKIN Timing

local bus switching characteristics - CLKOUT

CLKOUT may switch at either the CLKIN rate (PLL mode) or ½ the CLKIN rate (non-PLL mode). In either mode, no skew or phase relationship is guaranteed (function of PLL is to allow for higher processor rates with lower frequency oscillator). Each state of a memory access begins on the falling edge of CLKOUT.

NO.	PARAMETER		C82-50		C82-60		UNIT
			MIN	MAX	MIN	MAX	
6	$t_{c(CKO)}$	Period of CLKOUT (non-PLL mode)	$2t_{c(CKI)}$†		$2t_{c(CKI)}$†		ns
6	$t_{c(CKO)}$	Period of CLKOUT (PLL mode)	$t_{c(CKI)}$†		$t_{c(CKI)}$†		ns
7	$t_{w(CKOH)}$	Pulse duration of CLKOUT high	t_H-4.5		t_H-3.7		ns
8	$t_{w(CKOL)}$	Pulse duration of CLKOUT low	t_H-4.5		t_H-3.7		ns
9	$t_{t(CKO)}$	Transition time of CLKOUT		2‡		2‡	ns
10	t_{JITTER}	CLKOUT jitter (PLL mode only)		1.2§		0.8§	ns

† This is a functional minimum and is not tested. This parameter may also be specified as $2t_H$.
‡ This parameter is specified by computer simulation and is not tested.
§ This parameter is characterized and is not tested.

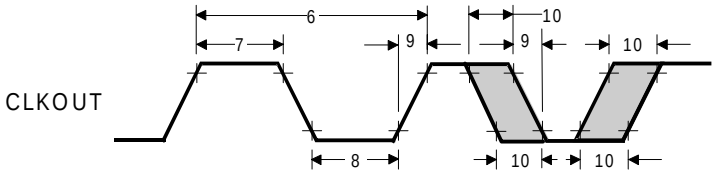


Figure 108. CLKOUT Timing

device reset timing requirements

NO.	PARAMETER			MIN	MAX	UNIT
11	$t_{w(RSTL)}$	Duration of /RESET low	Initial reset during power-up (non-PLL mode)	$6t_H$		ns
			Initial reset during power-up (PLL mode)	2		µs
			Reset during active operation	$6t_H$		
12	$t_{su(HRQL-RSTH)}$	Setup time of /HREQ low to /RESET high to configure self-bootstrap mode		$4t_H$		ns
13	$t_{h(RSTH-HRQL)}$	Hold time, /HREQ low after /RESET high to configure self-bootstrap mode		0		ns
14	$t_{su(RDYL-RSTH)}$	Setup time of READY low to /RESET high to configure big-endian operation		$4t_H$		ns
15	$t_{h(RSTH-RDYL)}$	Hold time, READY low after /RESET high to configure big-endian operation		0		ns

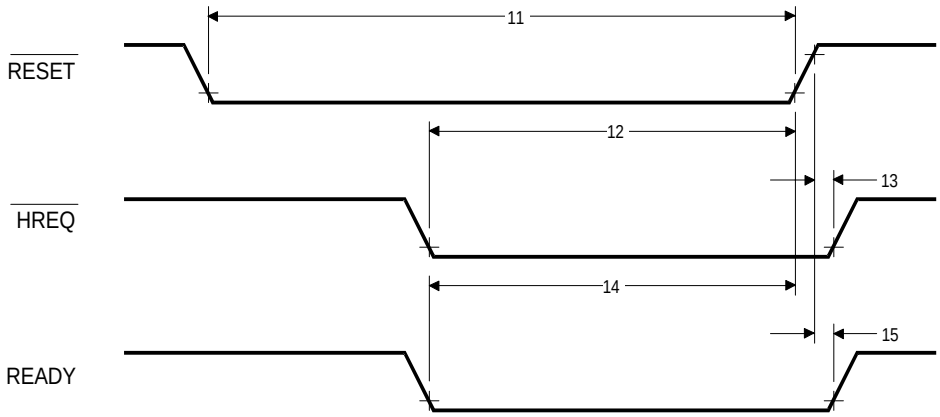


Figure 109. Device Reset Timing

local bus timing: memory exceptions

Memory exceptions are signaled to the ‘C82 via the /EXCEPT[1:0] inputs. The /EXCEPT[1:0] inputs are sampled at the beginning of each row access during the ad2 state, and on each CLKOUT falling edge following rl1. The READY input is also sampled during the ad2 state and during each column access (2 and 3 cyc/col accesses only). The value *n* as used in the parameters represents the integral number of half-cycles between the transitions of the two signals in question.

NO.	PARAMETER		C82-50		C82-60		UNIT
			MIN	MAX	MIN	MAX	
16	$t_{a(MIDV-CMPV)}$	Access time, /EXCEPT[1:0] †, READY valid after memory identification (address, status) valid		nt_H-8		nt_H-8	ns
17	$t_{su(CMPV-CKO)}$	Setup time, /EXCEPT[1:0] †, READY valid to CLKOUT no longer high/low	7.5		7.5		ns
18	$t_{h(CKO-CMPV)}$	Hold time, /EXCEPT[1:0] †, READY valid after CLKOUT no longer high/low	2.0		2.0		ns
19	$t_{a(RASL-RRV)}$	Access time /EXCEPT[1:0], READY valid from /RAS low		nt_H-7.5		nt_H-7.5	ns
20	$t_{a(RLL-RRV)}$	Access time, /EXCEPT[1:0], READY valid from /RL low		nt_H-7.5		nt_H-7.5	ns
21	$t_{a(CASL-RDYV)}$	Access time, READY valid from /CAS low		nt_H-8		nt_H-8	ns

† This parameter also applies to refresh cycles, wherein cycle timing is determined via the /EXCEPT[1:0] pin codings.

local bus timing: memory exceptions (continued)

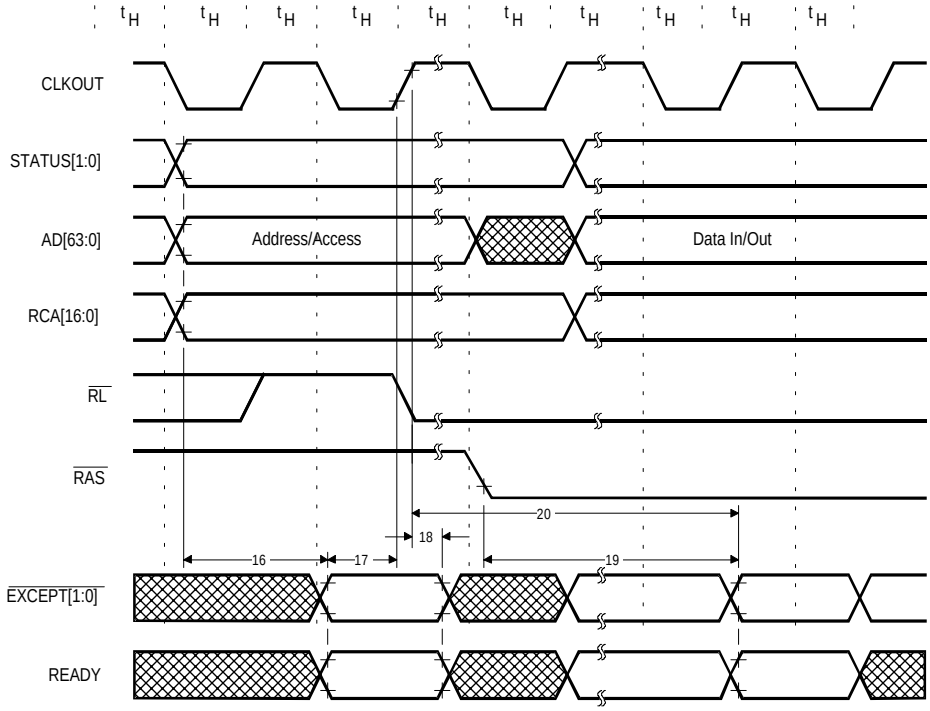


Figure 110. Row Time Cycle Completion Input Timing

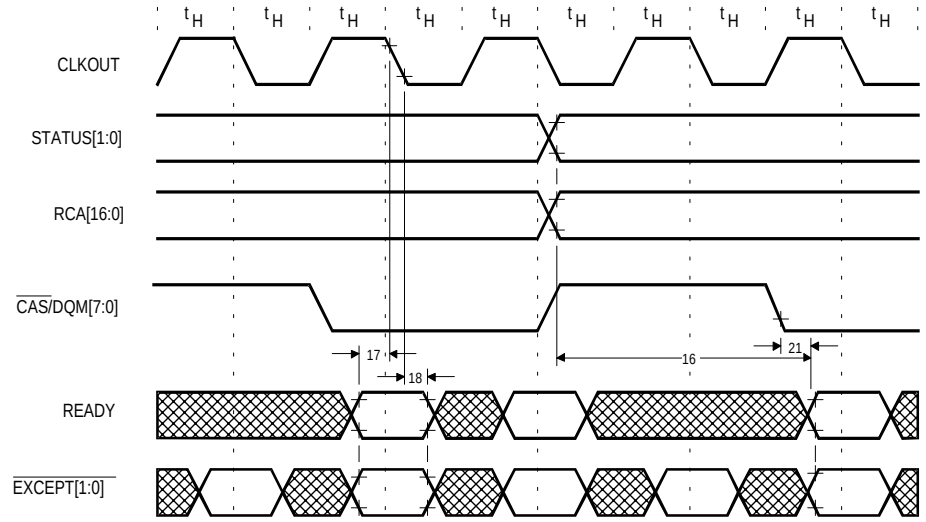


Figure 111. Column Time Cycle Completion Input Timing

general output signal characteristics

The following general timing parameters apply to all TMS320C82 output signals unless otherwise specifically given. The value *n* as used in the parameters represents the integral number of half-cycles between the transitions of the two outputs in question. For timing purposes, outputs fall into one of three groups: the address/data bus (AD[63:0]); the other output busses (RCA[16:0], STATUS[1:0], /CAS/DQM[7:0]); and non-bus outputs (/DBEN, /DDIN, DSF, /RAS, /RL, /TRG/CAS, /W). When measuring output to output, the named group refers to the first output to transition (Output A), and the second output (Output B) refers to any output group.

general output signal characteristics (continued)

NO.	PARAMETER		C82-50		C82-60		UNIT
			MIN	MAX	MIN	MAX	
22	$t_{h(OUTV-CKOL)}$	Hold time, CLKOUT high after output valid					ns
		AD[63:0]	nt_H-5.3		nt_H-5.3		
		RCA[16:0], STATUS[1:0],	nt_H-4.5		nt_H-4.4		
		/CAS/DQM[7:0] (CT = 0X0X)	nt_H-4.8		nt_H-4.7		
		/CAS/DQM[7:0] (CT != 0X0X)	nt_H-4.4		nt_H-4.4		
		/DBEN, /DDIN, DSF, /RAS, /RL, /TRG/CAS, /W	nt_H-3.9		nt_H-3.9		
23	$t_{h(OUTV-CKOH)}$	Hold time, CLKOUT low after output valid					ns
		AD[63:0]	nt_H-5.5		nt_H-5.3		
		RCA[16:0], STATUS[1:0],	nt_H-4.5		nt_H-4.5		
		/CAS/DQM[7:0] (CT = 0X0X)	nt_H-5.0		nt_H-4.7		
		/CAS/DQM[7:0] (CT != 0X0X)	nt_H-4.6		nt_H-4.5		
		/DBEN, /DDIN, DSF, /RAS, /RL, /TRG/CAS, /W	nt_H-4.1		nt_H-4.1		
24	$t_{h(CKOL-OUTV)}$	Hold time, output valid after CLKOUT low	nt_H-5		nt_H-5		ns
25	$t_{h(CKOH-OUTV)}$	Hold time, output valid after CLKOUT high	nt_H-5.4		nt_H-5		ns
26	$t_{h(OUTV-OUTV)}$	Hold time, output valid after output valid					ns
		AD[63:0]	nt_H-6.5		nt_H-6.3		
		RCA[16:0], STATUS[1:0],	nt_H-5.5		nt_H-5.5		
		/CAS/DQM[7:0] (CT = 0X0X)	nt_H-5.9		nt_H-5.7		
		/CAS/DQM[7:0] (CT != 0X0X)	nt_H-5.5		nt_H-5.5		
		/DBEN, /DDIN, DSF, /RAS, /RL, /TRG/CAS, /W	nt_H-5		nt_H-4.8		
27	$t_{d(CKOL-OUTV)}$	Delay time, CLKOUT no longer high to output valid					ns
		AD[63:0]		nt_H+6.5		nt_H+5.9	
		RCA[16:0], STATUS[1:0], /CAS/DQM[7:0]		nt_H+5.5		nt_H+5.5	
		/DBEN, /DDIN, DSF, /RAS, /RL, /TRG/CAS, /W		nt_H+5		nt_H+4.7	
28	$t_{d(CKOH-OUTV)}$	Delay time, CLKOUT no longer low to output valid					ns
		AD[63:0]		nt_H+6.5		nt_H+5.9	
		RCA[16:0], STATUS[1:0], /CAS/DQM[7:0]		nt_H+5.5		nt_H+5.5	
		/DBEN, /DDIN, DSF, /RAS, /RL, /TRG/CAS, /W		nt_H+5		nt_H+4.7	
29	$t_{d(OUTV-OUTV)}$	Delay time, output no longer valid to output valid					ns
		AD[63:0]		nt_H+6.5		nt_H+6.3	
		RCA[16:0], STATUS[1:0],		nt_H+5.5		nt_H+5.5	
		/CAS/DQM[7:0] (CT = 0X0X)		nt_H+5.9		nt_H+5.7	
		/CAS/DQM[7:0] (CT != 0X0X)		nt_H+5.6		nt_H+5.5	
		/DBEN, /DDIN, DSF, /RAS, /RL, /TRG/CAS, /W		nt_H+5		nt_H+4.8	
30	$t_{d(OUTV-CKOL)}$	Delay time, output no longer valid to CLKOUT low		nt_H+5		nt_H+5	ns
31	$t_{d(OUTV-CKOH)}$	Delay time, output no longer valid to CLKOUT high		nt_H+5.4		nt_H+5	ns
32	$t_{w(OUTV)}$	Pulse width, output valid					ns
		AD[63:0]	nt_H-6.5		nt_H-6.3		
		RCA[16:0], STATUS[1:0],	nt_H-5.5		nt_H-5.5		
		/CAS/DQM[7:0] (CT = 0X0X)	nt_H-5.9		nt_H-5.6		
		/CAS/DQM[7:0] (CT != 0X0X)	nt_H-5.6		nt_H-5.5		
		/DBEN, /DDIN, DSF, /RAS, /RL, /TRG/CAS, /W	nt_H-5		nt_H-5		
33	$t_{h(CKOL-ADZ)}$	Hold time, AD[63:0] driven after CLKOUT low†	nt_H-5		nt_H-5		ns
34	$t_{h(CKOH-ADZ)}$	Hold time, AD[63:0] driven after CLKOUT high†	nt_H-5		nt_H-5		ns
35	$t_{d(CKOL-ADZ)}$	Delay time, CLKOUT no longer high to AD[63:0] Hi-Z‡		nt_H+5		nt_H+5	ns
36	$t_{d(CKOH-ADZ)}$	Delay time, CLKOUT no longer low to AD[63:0] Hi-Z‡		nt_H+5		nt_H+5	ns

† This parameter is a functional minimum specified by logic and is not tested.
‡ This parameter is specified by characterization and is not tested.

general output signal characteristics (continued)

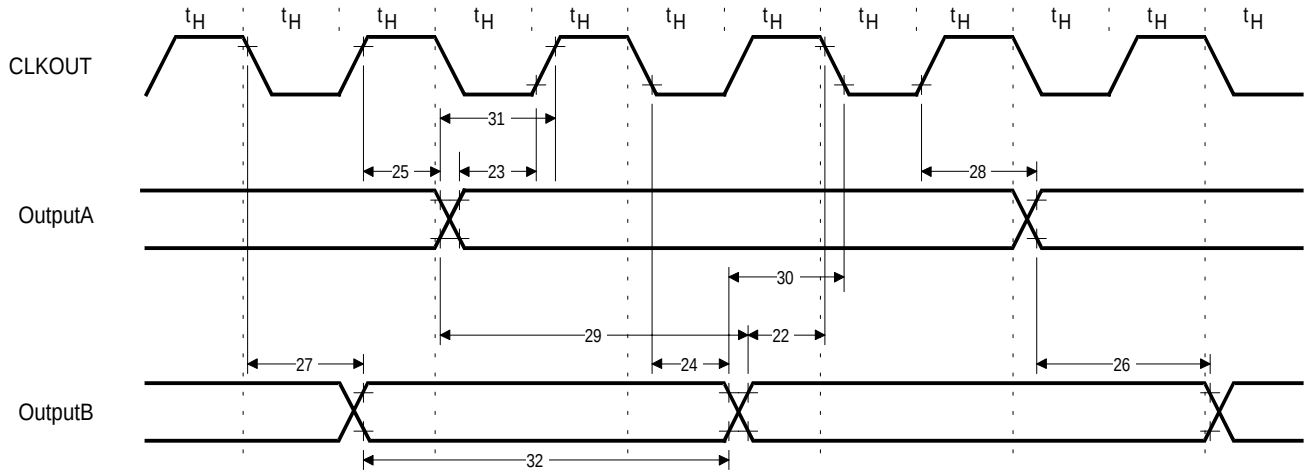


Figure 112. General Output Timing

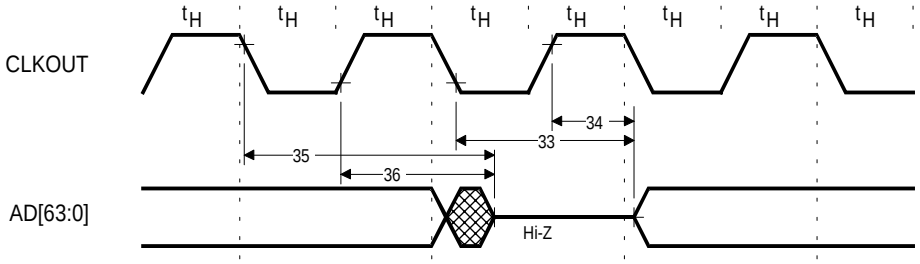


Figure 113. AD[63:0] Turnoff/Turn-Around Timing

data input signal characteristics

The following general timing parameters apply to the AD[63:0] inputs unless otherwise specifically given. The value *n* as used in the parameters represents the integral number of half cycles between the transitions of the output and input in question.

NO.	PARAMETER		C82-50		C82-60		UNIT
			MIN	MAX	MIN	MAX	
37	$t_{a(CKOH-DV)}$	Access time, CLKOUT high to AD[63:0] valid		nt_H-5.3		nt_H-5.3	ns
38	$t_{a(CKOL-DV)}$	Access time, CLKOUT low to AD[63:0] valid		nt_H-6.5		nt_H-6.5	ns
39	$t_{su(DV-CKOH)}$	Setup time, AD[63:0] valid to CLKOUT no longer low	6.5		6.1		ns
40	$t_{su(DV-CKOL)}$	Setup time, AD[63:0] valid to CLKOUT no longer high	6.1		6.1		ns
41	$t_{h(CKOL-DV)}$	Hold time, AD[63:0] valid after CLKOUT low	2.5		2.5		ns
42	$t_{h(CKOH-DV)}$	Hold time, AD[63:0] valid after CLKOUT high	2.5		2.5		ns
43	$t_{a(OUTV-DV)}$	Access time, output valid to AD[63:0] inputs valid AD[39:0] RCA[16:0], /CAS/DQM[7:0], STATUS[1:0] /DBEN, /DDIN, DSF, /RAS, /RL, /TRG/CAS, /W		 nt_H-7.6 nt_H-7 nt_H-6.5		 nt_H-7.6 nt_H-7 nt_H-6.5	ns
44	$t_{h(OUTV-DV)}$	Hold time, AD[63:0] valid after output valid RCA[16:0], /CAS/DQM[7:0] (CT = 00xx) RCA[16:0], /CAS/DQM[7:0] (CT = 01xx) /RAS	 2.5 2 2		 2.5 2 2		ns

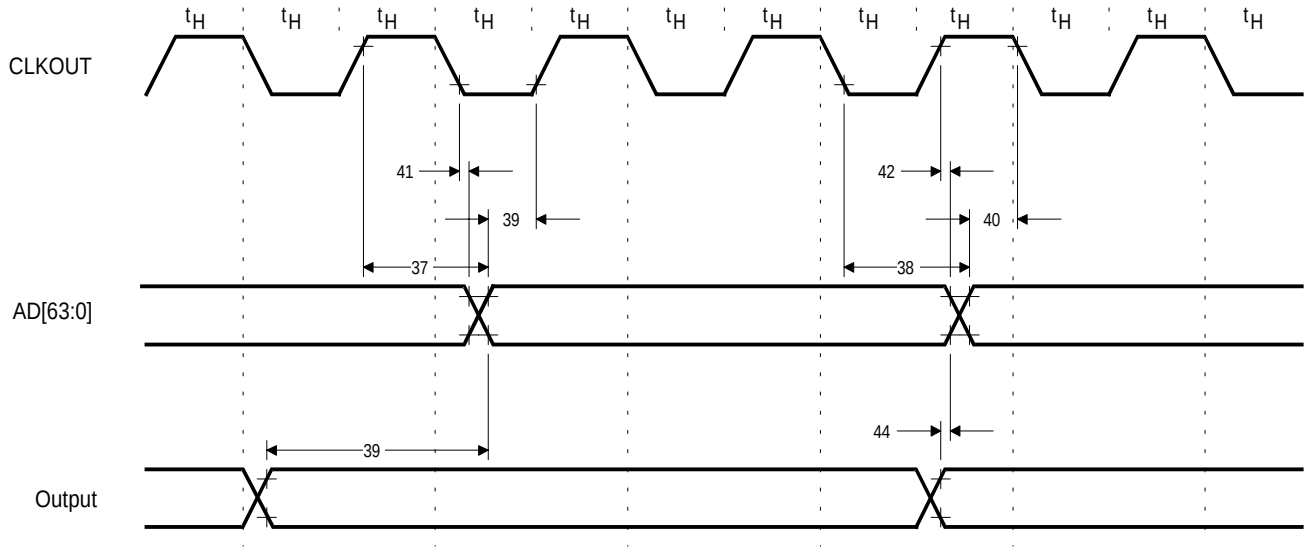


Figure 114. Data Input Timing

external interrupt timing

The following description defines the timing of the edge-triggered interrupts /EINT1 - /EINT3 and the level-triggered interrupt /LINT4 (see Note 5).

NO.	PARAMETER		C82-50		C82-60		UNIT
			MIN	MAX	MIN	MAX	
45	$t_{w(EINL)}$	Pulse duration, /EINTx low†	6		6		ns
46	$t_{su(EINH-CKOH)}$	Setup time, /EINTx high before CLKOUT no longer low‡	10		10		ns
47	$t_{w(EINH)}$	Pulse duration, /EINTx high†	6		6		ns
48	$t_{su(LINL-CKOH)}$	Setup time, /LINT4 low before CLKOUT no longer low‡	9.5		9.5		ns

† This parameter is specified by characterization and is not tested.
‡ This parameter must only be met to ensure that the interrupt is recognized on the indicated cycle.
NOTE 5: In order to ensure recognition, /LINT4 must remain low until cleared by the interrupt service routine.

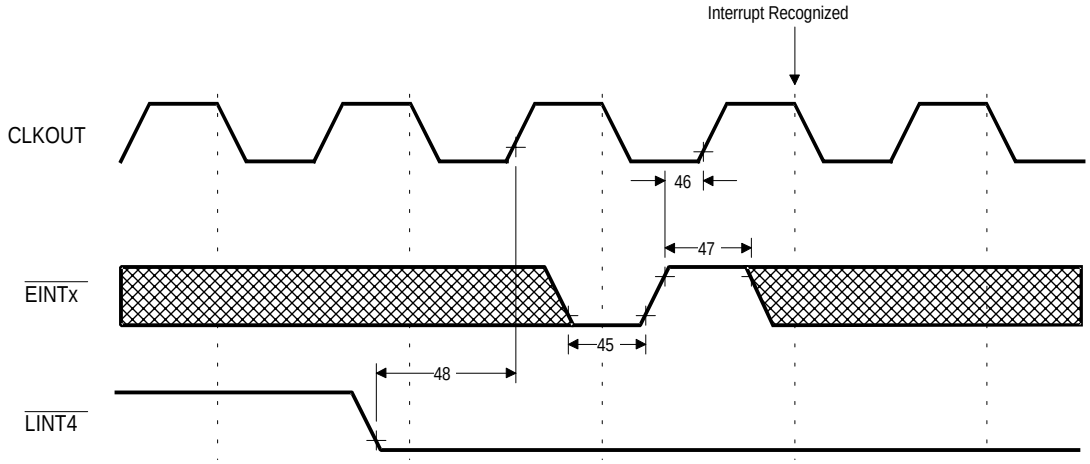


Figure 115. External Interrupt Timing

XPT input timing

The following description defines the sampling of the /XPT[3:0] inputs. The value encoded on the /XPT[3:0] inputs is synchronized over multiple cycles to ensure that a stable value is present.

NO.	PARAMETER		C82-50		C82-60		UNIT
			MIN	MAX	MIN	MAX	
49	$t_{w(XPTV)}$	Pulse duration, /XPTx valid¶	$12t_H$		$12t_H$		ns
50	$t_{su(XPTV-CKOH)}$	Setup time, /XPT[3:0] valid before CLKOUT no longer low†	12		12		ns
51	$t_{h(CKOH-XPTV)}$	Hold time, /XPT[3:0] valid after CLKOUT high	5		5		ns
52	$t_{h(RLL-XPTV)}$	Hold time, /XPT[3:0] valid after /RL low‡		$6t_H$		$6t_H$	ns

† This parameter must only be met to ensure that the XPT input is recognized on the indicated cycle.
‡ This parameter must be met to ensure that a second XPT request does not occur. This parameter is a functional maximum specified by logic and is not tested.
¶ This parameter is a functional minimum specified by logic and is not tested.

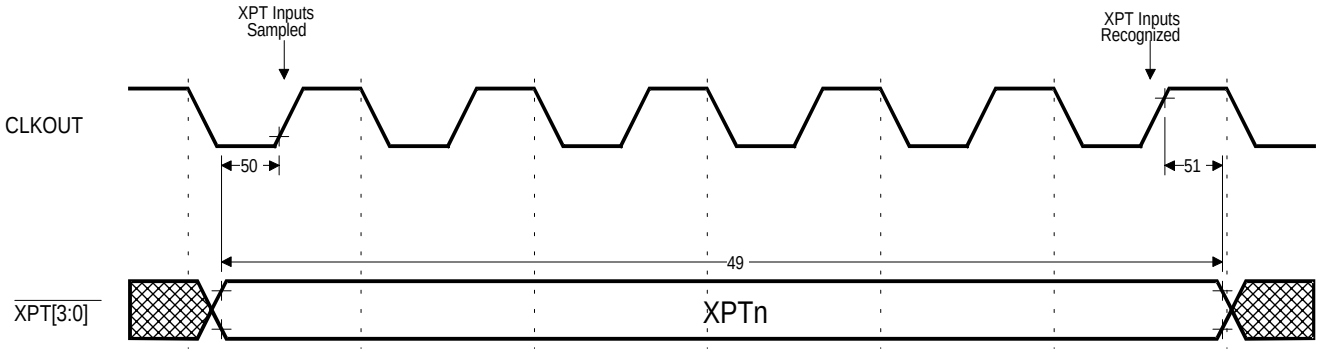


Figure 116. XPT Input Timing - XPT Recognition

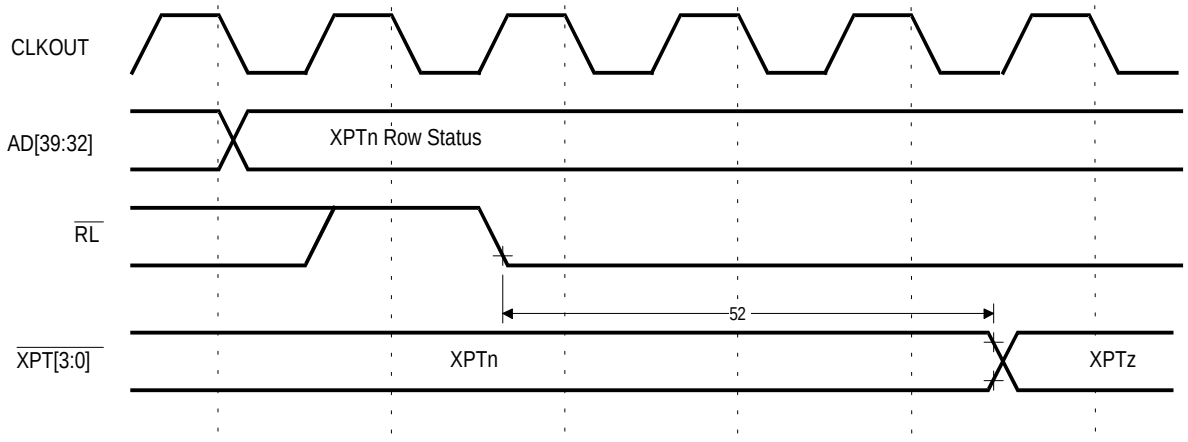


Figure 117. XPT Input Timing - XPT Service

host interface timing

NO.	PARAMETER		C82-50		C82-60		UNIT
			MIN	MAX	MIN	MAX	
53	$t_{h(REQV-CKOH)}$	Hold time, CLKOUT low after REQ valid	t_H-7		t_H-5.5		ns
54	$t_{h(CKOH-REQV)}$	Hold time, REQ valid after CLKOUT high	t_H-7		t_H-5.5		ns
55	$t_{h(HRQL-HAKL)}$	Hold time, /HACK high after /HREQ low‡	$4t_H$-12		$4t_H$-12		ns
56	$t_{d(HAKL-OUTZ)}$	Delay time, /HACK low to output Hi-Z†					
		All signals except AD[63:0]		0		0	ns
		AD[63:0]		1		1	
57	$t_{d(HRQH-HAKH)}$	Delay time, /HREQ high to /HACK no longer low		10		10	ns
58	$t_{d(HAKH-OUTD)}$	Delay time, /HACK high to outputs driven‡	$6t_H$		$6t_H$		ns
59	$t_{su(HRQL-CKOH)}$	Setup time, /HREQ low to CLKOUT no longer low (see Note 6)	8.5		8.5		ns

† This parameter is specified by characterization and is not tested.
‡ This parameter is a functional minimum and is not tested.
Note 6: This parameter must be met only to ensure /HREQ is sampled low on the indicated clock cycle.

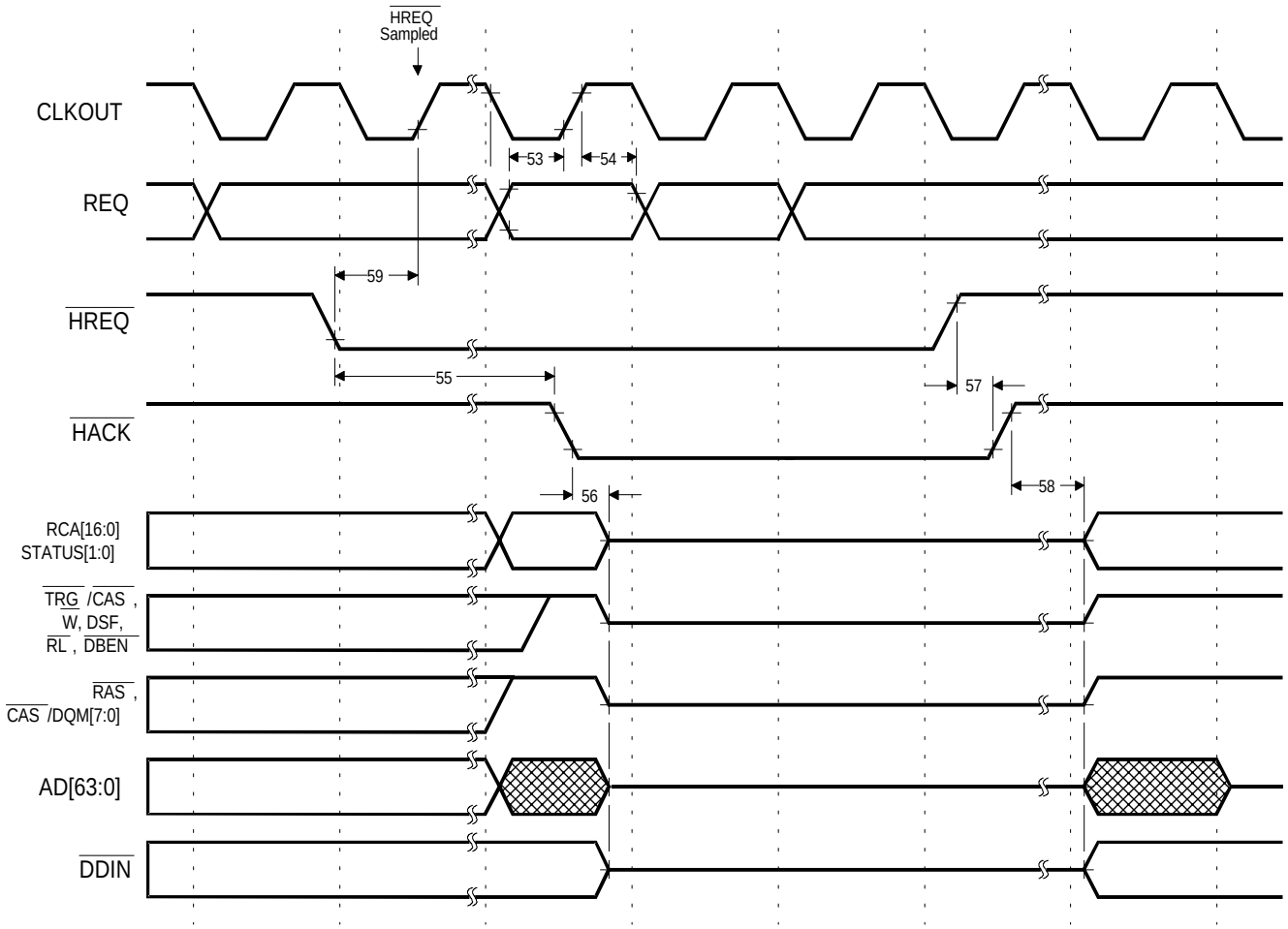


Figure 118. Host Interface Timing

thermal resistance

The following graph illustrates the maximum ambient temperature allowed for various air flow rates across the TMS320C82 to ensure that the case temperature is kept below the maximum operating temperature (85°C). (See Note 7.)

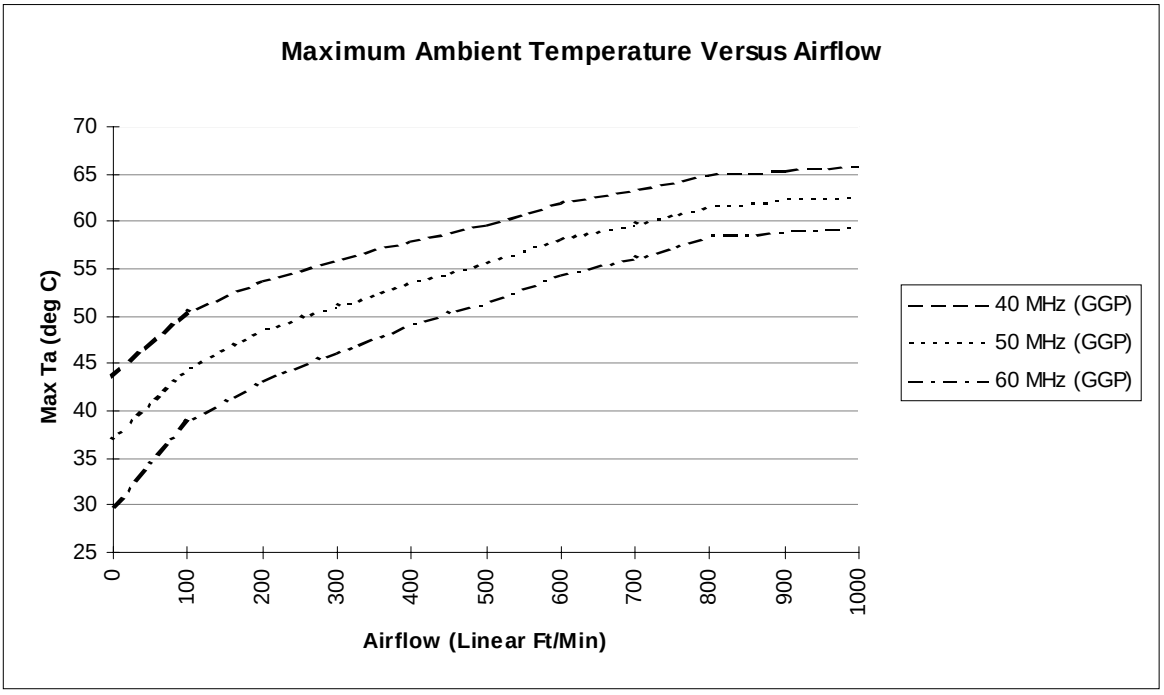
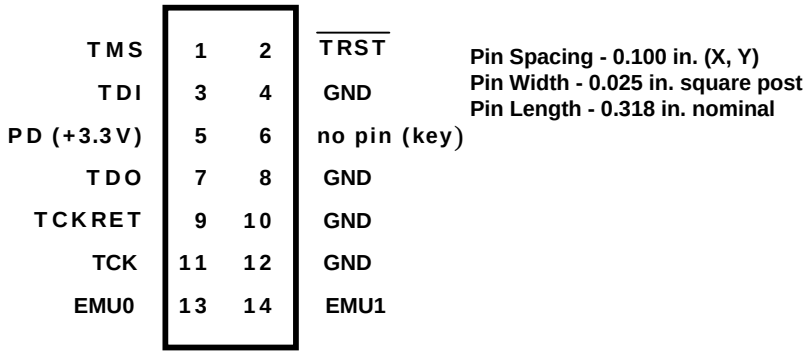


Figure 119. TMS320C82 Airflow Recommendations

Note 7: TMS320C82 power consumption is based on the "typical" values of I_{DD} measured at V_{DD} = 3.3 V. Power consumption will vary by application based on TMS320C82 processor activity and I/O pin loadings. Users must ensure that the case temperature (T_C) specifications are met when defining airflow and other thermal constraints of their system.

emulator interface connection

The TMS320C82 supports emulation through a dedicated emulation port which is a superset of the IEEE 1149.1 (JTAG) standard. To support the TMS320C82 emulator, a target system must include a 14-pin header (2 rows of 7 pins) with the connections shown below.



XDS 510™ SIGNAL	XDS 510 STATE	TARGET STATE	DESCRIPTION
TMS	O	I	JTAG test mode select
TDI	O	I	JTAG test data input
TDO	I	O	JTAG test data output
TCK	O	I	JTAG test clock - 10 MHz clock source from emulator. Can be used to drive system test clock.
/TRST	O	I	JTAG test reset
EMU0	I	I/O	Emulation pin 0
EMU1	I	I/O	Emulation pin 1
PD (+3.3 V)	I	O	Presence detect - Indicates that the target is connected and powered-up. Should be tied to +3.3 V on target system
TCKRET	I	O	JTAG test clock return - Test clock input to the XDS510 emulator. May be buffered or unbuffered version of TCK.

For best results, the emulation header should be located as close as possible to the TMS320C82. If the distance exceeds 6 inches, the emulation signals should be buffered.

XDS510 is a trademark of Texas Instruments Incorporated.

TEXAS INSTRUMENTS
POST OFFICE BOX 1443 • HOUSTON, TEXAS 77251-1443

emulator interface connection (continued)

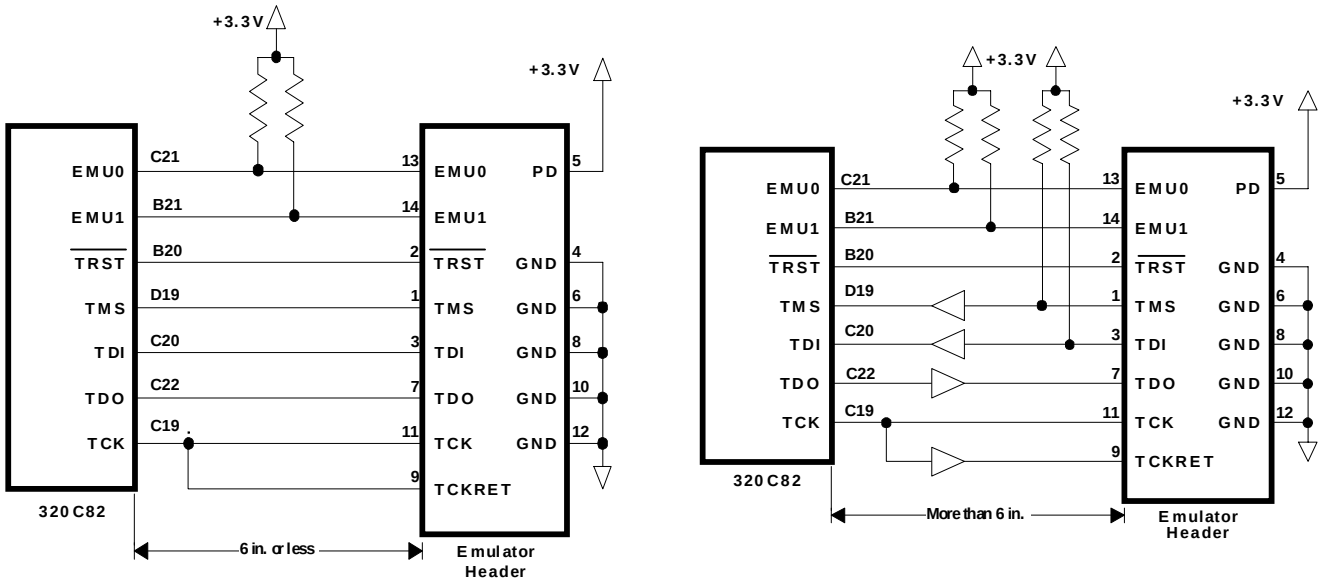


Figure 120. Emulation Header Connections - Emulator-Driven Test Clock

The target system may also generate the test clock. This allows the user to:

- Set test clock frequency to match system requirements. (The emulator provides a 10-MHz test clock)
- Have other devices in the system that require a test clock when the emulator is not connected.

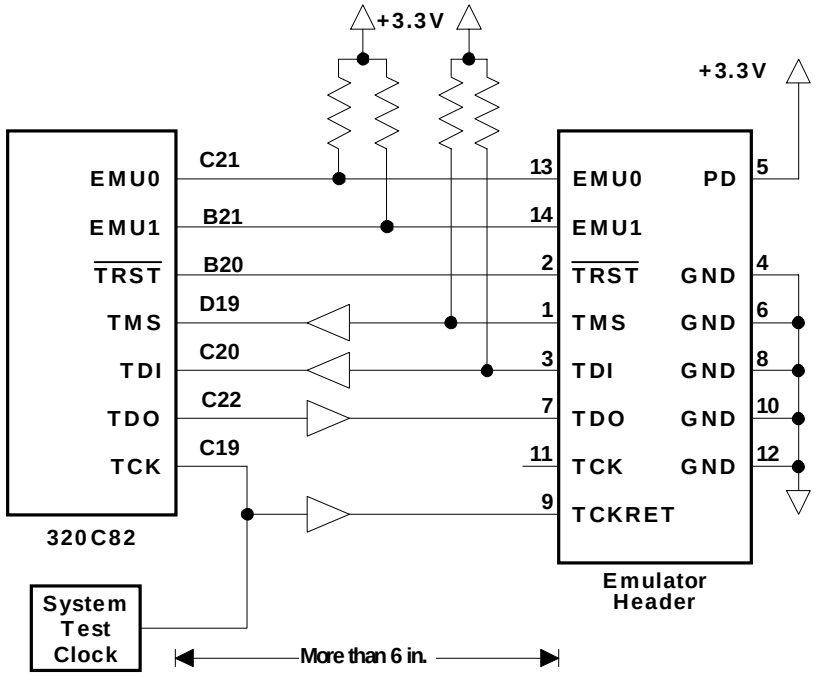


Figure 121. Emulation Header Connections - System-Driven Test Clock

emulator interface connection (continued)

For multiprocessor applications, the following are recommended:

- Buffer TMS, TDI, TDO, and TCK through the same physical package to reduce timing skew.
- Buffer inputs for TMS, TDI, and TCK should be pulled high (3.3 V). A pullup resistor of 4.7 kΩ or greater is suggested.
- Buffering EMU0 and EMU1 is highly recommended to provide isolation. The buffers need not be in the same physical package as TMS, TCK, TDI, or TDO. Pullups to 3.3 V are required and should provide a signal rise time of less than 10 μs. A 4.7-kΩ resistor is suggested for most applications.
- To ensure high quality signals, special PWB routing and use of termination resistors may be required. The emulator provides fixed series termination (33 Ω) on TMS and TDI and optional parallel terminators (180 Ω pullup and 270 Ω pulldown) on TCKRET and TDO.

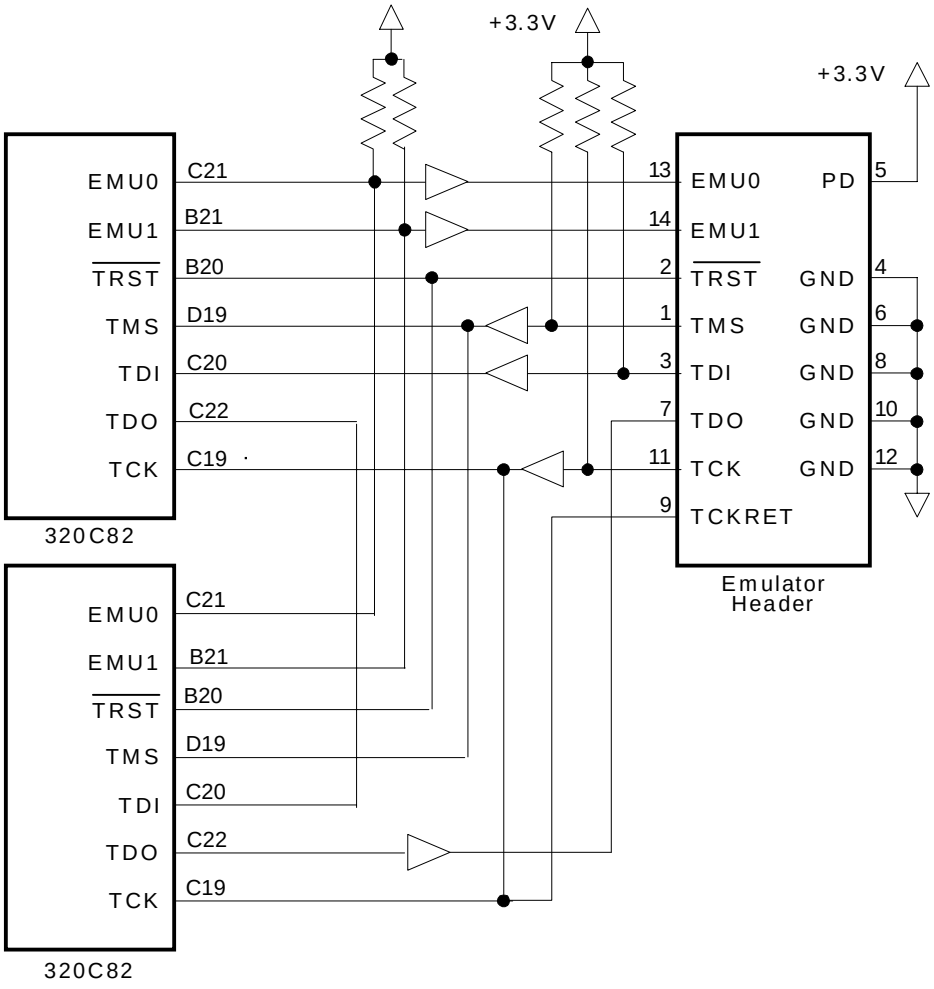
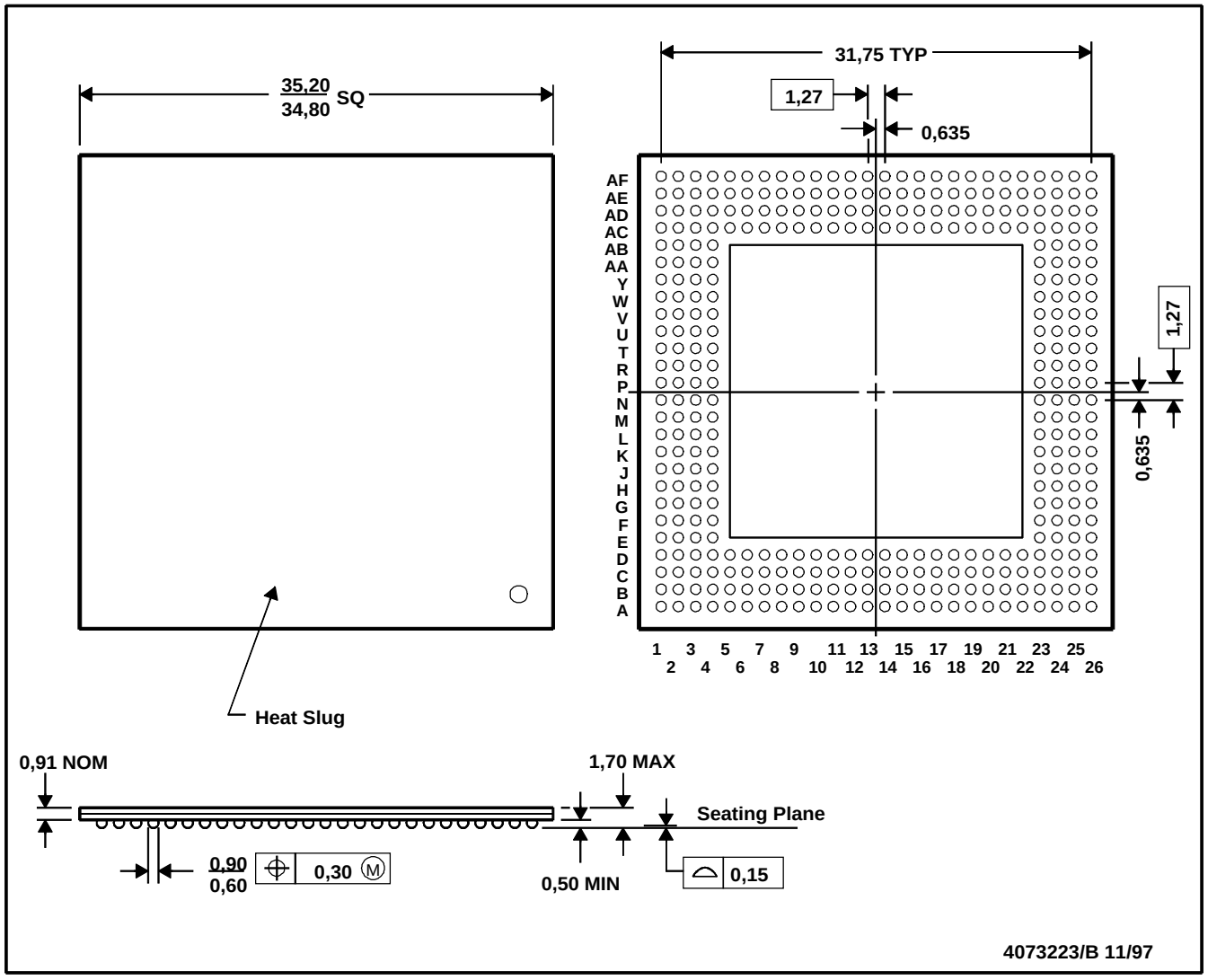


Figure 122. Emulation Header Connections - Multiprocessor Applications

MECHANICAL DATA

GGP (S-PBGA-N352) **PLASTIC BALL GRID ARRAY (CAVITY DOWN)**



NOTES: 1. All linear dimensions are in millimeters
2. This drawing is subject to change without notice.
3. Thermally enhanced plastic package with metal heat slug (HSL)

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish	MSL Peak Temp (3)
TMS320C82GGP50	NRND	BGA	GGP	352	1	TBD	SNPB	Level-4-220C-72HR
TMS320C82GGP60	NRND	BGA	GGP	352	24	TBD	SNPB	Level-4-220C-72HR
TMX320C82GGP	OBSOLETE	BGA	GGP	352		TBD	Call TI	Call TI
TMX320C82GGP50	OBSOLETE	BGA	GGP	352		TBD	Call TI	Call TI
TMX320C82GGP60	OBSOLETE	BGA	GGP	352		TBD	Call TI	Call TI

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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