TEXAS INSTRUMENTS

Interfacing SDRAM to the TMS320C80

Application Report

1996

Digital Signal Processing Solutions

Printed in U.S.A., August 1996

SPRA055

Interfacing SDRAM to the TMS320C80

Application Report

Dave Comisky
Systems Engineer — 'C8x Applications

SPRA055
August 1996

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

Appendixes

List of Illustrations

List of Tables

INTRODUCTION

The TMS320C80 offers incredible horsepower through its vastly parallel architecture. As is often the case though, algorithms, regardless of how rigorously they are coded, often achieve only average performance without a high-speed memory interface. The ’C80 can support burst transfer rates up to 400 MB/s at 50 MHz using a single-cycle memory interface. While several choices for single-cycle memory exist, one of the most economical is SDRAM. Beginning with Revision 3.0 of the ’C80, SDRAM support is built directly into the input/output (I/O) interface, facilitated by the on-chip transfer controller (TC). This application report illustrates the operation of the SDRAM interface and provides a reference design (schematics) for a baseline SDRAM interface to the TMS320C80.

This application report discusses an interface between the ’C80 and the TMS626802-10 SDRAM(s). The TMS626802 is organized as two 1M × 8-bit banks, features programmable burst length and CAS latency, and runs from a 3.3-V supply. Timing and loading considerations also are included, for both a 40- and 50-MHz interface.

SDRAM SUPPORT

The direct interface support for SDRAM provides systems that are based on the TMS320C80 with the opportunity for a low-cost, single-cycle-access memory bank. In order to support the unique commands of SDRAM, several changes were made to the ’C80. Most notably is the addition of several status codes and an extra column timing (CT) input which is used to select SDRAM cycles.

Row-Time Status

Several status codes were added to the Revision 3.0 silicon to support SDRAM. The new codes occupy previously reserved spaces in the status code table. The row-time status codes are shown in Table 1.

Table 1. Row-Time Status Codes

STATUS[5:0]	CYCLE TYPE
0 0 0 0 0 0	Normal Read
0 0 0 0 0 1	Normal Write
0 0 0 0 1 0	Refresh
0 0 0 0 1 1	SDRAM DCAB
0 0 0 1 0 0	Peripheral Device PT Read
0 0 0 1 0 1	Peripheral Device PT Write
0 0 0 1 1 0	Reserved
0 0 0 1 1 1	Reserved
0 0 1 0 0 0	Reserved
0 0 1 0 0 1	Block Write PT
0 0 1 0 1 0	Reserved
0 0 1 0 1 1	Reserved
0 0 1 1 0 0	SDRAM MRS
0 0 1 1 0 1	Load Color Register
0 0 1 1 1 0	Reserved
0 0 1 1 1 1	Reserved
0 1 0 0 0 0	Frame 0 Read Transfer
0 1 0 0 0 1	Frame 0 Write Transfer
0 1 0 0 1 0	Frame 0 Split-Read Transfer
0 1 0 0 1 1	Frame 0 Split-Write Transfer
0 1 0 1 0 0	Frame 1 Read Transfer
0 1 0 1 0 1	Frame 1 Write Transfer
0 1 0 1 1 0	Frame 1 Split-Read Transfer
0 1 0 1 1 1	Frame 1 Split-Write Transfer
0 1 1 0 0 0	Reserved
0 1 1 0 0 1	Reserved
0 1 1 0 1 0	Reserved
0 1 1 0 1 1	Reserved
0 1 1 1 0 0	PT Read Transfer
0 1 1 1 0 1	PT Write Transfer
0 1 1 1 1 0	Reserved
0 1 1 1 1 1	Idle

STATUS[5:0]	CYCLE TYPE
1 0 0 0 0 0	Reserved
1 0 0 0 0 1	Reserved
1 0 0 0 1 0	Reserved
1 0 0 0 1 1	Reserved
1 0 0 1 0 0	XPT1 Read
1 0 0 1 0 1	XPT1 Write
1 0 0 1 1 0	XPT1 PDPT Read
1 0 0 1 1 1	XPT1 PDPT Write
1 0 1 0 0 0	XPT2 Read
1 0 1 0 0 1	XPT2 Write
1 0 1 0 1 0	XPT2 PDPT Read
1 0 1 0 1 1	XPT2 PDPT Write
1 0 1 1 0 0	XPT3 Read
1 0 1 1 0 1	XPT3 Write
1 0 1 1 1 0	XPT3 PDPT Read
1 0 1 1 1 1	XPT3 PDPT Write
1 1 0 0 0 0	XPT4/SAM1 Read
1 1 0 0 0 1	XPT4/SAM1 Write
1 1 0 0 1 0	XPT4/SAM1 PDPT Read
1 1 0 0 1 1	XPT4/SAM1 PDPT Write
1 1 0 1 0 0	XPT5/SOF1 Read
1 1 0 1 0 1	XPT5/SOF1 Write
1 1 0 1 1 0	XPT5/SOF1 PDPT Read
1 1 0 1 1 1	XPT5/SOF1 PDPT Write
1 1 1 0 0 0	XPT6/SAM0 Read
1 1 1 0 0 1	XPT6/SAM0 Write
1 1 1 0 1 0	XPT6/SAM0 PDPT Read
1 1 1 0 1 1	XPT6/SAM0 PDPT Write
1 1 1 1 0 0	XPT7/SOF0 Read
1 1 1 1 0 1	XPT7/SOF0 Write
1 1 1 1 1 0	XPT7/SOF0 PDPT Read
1 1 1 1 1 1	XPT7/SOF0 PDPT Write

The row-time SDRAM DCAB status code (000011) is output during SDRAM deactivate cycles, which occur after reset as part of the power-up sequence. Deactivation cycles are performed to ensure that all SDRAM banks are deactivated before the power-up refresh and mode-initialization sequences are performed. The deactivate cycle is performed only if SDRAM is present in the system (indicated on CT[2:0]).

The SDRAM MRS status code (001100) indicates that a mode register set command is being performed. The MRS cycle is performed only after all power-up refresh cycles are completed; it is performed to initialize the SDRAM operation mode. Like the DCAB cycle, the MRS cycle is performed if SDRAM is present in the system.

Column-Time Status

With the addition of SDRAM support, column-time status codes are identical to those in previous versions of the 'C80, with two exceptions. The column-time SDRAM DCAB status code (111111) is used to indicate the deactivate cycle that is performed at the end of each page of SDRAM accesses. This status code is not required by SDRAM, but is used primarily during peripheral-device transfers to aid in peripheral-address increment control. Similarly, an idle status code (011111) is added during column time. The idle status code is especially useful for detecting bubbles in the pipeline. Again, SDRAM does not require this code; idle is used primarily for peripheral-address increment control. The 'C80 column-time status codes output on STATUS[5:0] are shown in Table 2.

Table 2. Column-Time Status Codes

STATUS[5:0]	CYCLE TYPE
0 0 0 0 0 0	PP0 Low-Priority Packet Transfer
0 0 0 0 0 1	PP0 High-Priority Packet Transfer
0 0 0 0 1 0	PP0 Instruction Cache
0 0 0 0 1 1	PP0 DEA
0 0 0 1 0 0	PP1 Low-Priority Packet Transfer
0 0 0 1 0 1	PP1 High-Priority Packet Transfer
0 0 0 1 1 0	PP1 Instruction Cache
0 0 0 1 1 1	PP1 DEA
0 0 1 0 0 0	PP2 Low-Priority Packet Transfer
0 0 1 0 0 1	PP2 High-Priority Packet Transfer
0 0 1 0 1 0	PP2 Instruction Cache
0 0 1 0 1 1	PP2 DEA
0 0 1 1 0 0	PP3 Low-Priority Packet Transfer
0 0 1 1 0 1	PP3 High-Priority Packet Transfer
0 0 1 1 1 0	PP3 Instruction Cache
0 0 1 1 1 1	PP3 DEA
0 1 0 0 0 0	MP Low-Priority Packet Transfer
0 1 0 0 0 1	MP High-Priority Packet Transfer
0 1 0 0 1 0	MP Urgent Packet Transfer (Low)
0 1 0 0 1 1	MP Urgent Packet Transfer (High)
0 1 0 1 0 0	XPT/VCPT in Progress
0 1 0 1 0 1	XPT/VCPT Complete
0 1 0 1 1 0	MP Instruction Cache (Low)
0 1 0 1 1 1	MP Instruction Cache (High)
0 1 1 0 0 0	MP DEA (Low)
0 1 1 0 0 1	MP DEA (High)
0 1 1 0 1 0	MP Data Cache (Low)
0 1 1 0 1 1	MP Data Cache (High)
0 1 1 1 0 0	Frame 0
0 1 1 1 0 1	Frame 1
0 1 1 1 1 0	Refresh
0 1 1 1 1 1	Idle

STATUS[5:0]	CYCLE TYPE
1 0 0 0 0 0	Reserved
1 0 0 0 0 1	Reserved
1 0 0 0 1 0	Reserved
1 0 0 0 1 1	Reserved
1 0 0 1 0 0	Reserved
1 0 0 1 0 1	Reserved
1 0 0 1 1 0	Reserved
1 0 0 1 1 1	Reserved
1 0 1 0 0 0	Reserved
1 0 1 0 0 1	Reserved
1 0 1 0 1 0	Reserved
1 0 1 0 1 1	Reserved
1 0 1 1 0 0	Reserved
1 0 1 1 0 1	Reserved
1 0 1 1 1 0	Reserved
1 0 1 1 1 1	Reserved
1 1 0 0 0 0	Reserved
1 1 0 0 0 1	Reserved
1 1 0 0 1 0	Reserved
1 1 0 0 1 1	Reserved
1 1 0 1 0 0	Reserved
1 1 0 1 0 1	Reserved
1 1 0 1 1 0	Reserved
1 1 0 1 1 1	Reserved
1 1 1 0 0 0	Reserved
1 1 1 0 0 1	Reserved
1 1 1 0 1 0	Reserved
1 1 1 0 1 1	Reserved
1 1 1 1 0 0	Reserved
1 1 1 1 0 1	Reserved
1 1 1 1 1 0	Reserved
1 1 1 1 1 1	Write Drain / SDRAM DCAB

Low – MP operating in low- (normal-) priority mode
High – MP operating in high-priority mode

Cycle Timing

To accommodate the use of SDRAMs, the ’C80’s CT input is expanded with CT2 (added to CT[1:0]), which is sampled at row time to select SDRAM cycles. The cycle timing codes are shown in Table 3.

Table 3. Cycle Timing Codes

CT2	CT1	CT0	CYCLE TIMING
0	0	0	Pipelined (Burst Length 1) SDRAM CAS Latency of 2
0	0	1	Pipelined (Burst Length 1) SDRAM CAS Latency of 3
0	1	0	Interleaved (Burst Length 2) SDRAM CAS Latency of 2
0	1	1	Interleaved (Burst Length 2) SDRAM CAS Latency of 3
1	0	0	Pipelined 1 Cycle/Column
1	0	1	Nonpipelined 1 Cycle/Column
1	1	0	2 Cycles/Column
1	1	1	3 Cycles/Column

The cycle timing information must be input at the beginning of each memory access (during the R2 state) to indicate to the ’C80 the type of memory cycle timing that should be used for the rest of the access. When an SDRAM CT code is detected, the TC then operates at the requested CAS latency and burst length for all subsequent commands until the row is deactivated.

SDRAM PINS

SDRAMs have pin names which, for historic reasons, duplicate the pin names on standard DRAM devices. For the most part, the ’C80 uses its pins to implement the SDRAM function for the pin of the same name. The following are the SDRAM control pins:

- $\overline{\text{RAS}}$ – row-address strobe/control input
- $\overline{\text{CAS}}$ – column-address strobe/control input
- $\overline{\text{W}}$ – write enable/control input
- DQM – data/output mask
- $\overline{\text{CS}}$ – chip select (command enable)
- CKE – clock enable
- CLK – clock input

$\overline{\text{RAS}}$, $\overline{\text{CAS}}$, and $\overline{\text{W}}$ are all command lines. They are latched on the rising CLK edge by the SDRAM to determine the current operation. These signals are considered valid only if $\overline{\text{CS}}$ is low during the rising edge of CLK.

The $\overline{\text{CS}}$ input to the SDRAM(s) can be used to select or deselect the device for command entry, as might be required for multiple memory-device decoding. Device select is performed by holding $\overline{\text{CS}}$ low on the rising edge of CLK. The device does not respond to $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, or $\overline{\text{W}}$ until the device is selected. The device can be selected or deselected on a cycle-by-cycle basis; however, the $\overline{\text{CS}}$ level does not affect an access burst that is in progress. The SDRAM $\overline{\text{CS}}$ input must be implemented in external logic.

DQM is an I/O buffer control signal. It disables writes and tristate outputs during reads when it is at a logic-high level. DQM has a *fixed* 2-CLK latency for reads and a *fixed* 0-CLK latency for writes.

Clock-enable (CKE) input suspends data (that is, read data remains valid and write data is inhibited) during an active read or write. CKE forces the SDRAM to enter into power-down mode if all banks are deactivated. The ’C80 does not support these operations; therefore, no CKE is provided.

Since the DQM signals serve as byte strobes, their function is implemented on the ’C80’s $\overline{\text{CAS}}$[7:0] outputs. The SDRAM $\overline{\text{CAS}}$ signal is remapped to the $\overline{\text{TRG}}$ signal of the ’C80. The SDRAM control pins map to the ’C80 pins as shown in Table 4.

Table 4. SDRAM Control Pins

’C80 Pin Number	’C80 Pin Name	SDRAM Signal
A21	$\overline{\text{RAS}}$	$\overline{\text{RAS}}$
B22	$\overline{\text{TRG}}$	$\overline{\text{CAS}}$
C23	$\overline{\text{W}}$	$\overline{\text{W}}$
A13	$\overline{\text{CAS7}}$	DQM7
B14	$\overline{\text{CAS6}}$	DQM6
A15	$\overline{\text{CAS5}}$	DQM5
E17	$\overline{\text{CAS4}}$	DQM4
B16	$\overline{\text{CAS3}}$	DQM3
C19	$\overline{\text{CAS2}}$	DQM2
B20	$\overline{\text{CAS1}}$	DQM1
D20	$\overline{\text{CAS0}}$	DQM0
E25	CLKOUT	CLK

SDRAM OPERATION

SDRAM control is maintained by issuing various commands, defined by the state of the control inputs. Although other commands exist, there are five basic commands which control most operations of SDRAM. They are:

- Bank activate/row-address entry (ACTV)
- Column-address entry/write (WRT)
- Column-address entry/read (READ)
- Bank deactivate (DCAB)
- $\overline{\text{CAS}}$-before-$\overline{\text{RAS}}$ self-refresh entry (REFR)

Table 5 illustrates the commands that are supported by the ’C80 and are used by the SDRAM.

Table 5. SDRAM Commands

COMMAND	STATE OF BANKS	$\overline{CS}$	$\overline{RAS}$	$\overline{CAS}$	$\overline{W}$	A11	A10	A9–A0	MNEMONIC
Mode register set	T = deac B = deac	L	L	L	L	X	X	A9 = X A8 = 0 A7 = 0 A6–A0 = V	MRS
Deactivate all banks	X	L	L	H	L	X	H	X	DCAB
Bank deactivate	X	L	L	H	L	BS	L	X	DEAC†
Bank activate/ Row-address entry	SB = deac	L	L	H	H	BS	V	V	ACTV
Column-address entry/ Write operation	SB = actv	L	H	L	L	BS	L	V	WRT
Column-address entry/ Read operation	SB = actv	L	H	L	H	BS	L	V	READ
No operation	X	L	H	H	H	X	X	X	NOOP
Control-input inhibit	X	H	X	X	X	X	X	X	DESL
CBR refresh	T = B = deac	L	L	L	H	X	X	X	REFR

Legend:
L = logic low
H = logic high
X = don’t care
V = valid
T = bank T
B = bank B
actv = activated
deac = deactivated
BS = bank select; 1 = bank T; 0 = bank B
SB = bank selected by A11
† This command is implemented with external logic.

CAS Latency

Most SDRAMs allow the user to program the beginning data-output cycle of a read burst to occur 1, 2, or 3 CLK cycles after the read command. This feature allows the user to adjust the SDRAM to operate in accordance with the system’s capability to latch the data output from the device. The delay between the read command and the beginning of the output burst is known as *CAS latency*. The ’C80 supports SDRAMs with CAS latencies of two or three cycles. The latency to be used is determined by the CT code input to the TC at row time during an MRS cycle.

There is no latency for write cycles. The first data-in cycle of a write burst is entered on the same rising CLK edge on which the write command is entered. This latency is fixed and is *not* determined by the contents of the mode register.

CAS latency is determined at initialization time, during an MRS cycle.

Burst Length

All data for SDRAMs is written or read in a burst fashion; that is, a single starting address is latched into the device and then the SDRAM internally accesses a sequence of locations based on that starting address.

Subsequent accesses can be to preceding as well as succeeding column addresses, depending on the starting address, and whether the SDRAM is in serial or interleave mode. The ’C80 supports burst lengths of 1 and 2. The order of accesses for burst length 2 is shown in Table 6 (note that A0 refers to pin A0 of the SDRAM).

Table 6. SDRAM 2-Bit Burst Sequences

	INTERNAL COLUMN ADDRESS A0 START 2ND	
serial	0 1	1 0
interleave	0 1	1 0

Because the TC can change the column address on every cycle, a burst length of 1 is the most attractive. However, some SDRAMs (including the TMS626802 used in this report) operate in prefetch mode, and the column address can change only on every other cycle. There is no advantage to a burst length of 1 for these devices.

The ’C80 actually performs all accesses in serial mode; if a burst length of 2 is specified and the second access is not needed, it is disabled through the DQM pin(s). Burst length is determined at initialization time during an MRS cycle.

Multiple Banks

SDRAMs consist of two banks, which can be accessed independently or in an interleaved fashion. SDRAMs also support operations involving both banks simultaneously; however, in order for the TC to take advantage of this feature, it must be able to determine if the next access is for the other bank. In order to determine this crossover point, the TC must know which address line is physically connected to the bank-select input of the SDRAM. Since this is system-dependent, it is impossible for the TC to know if the other bank should be activated on the next access; therefore, the TC cannot support multiple active-bank operations.

Bank Deactivation

Bank deactivation can be accomplished in one of two ways: by issuing either a DCAB (deactivate both banks) command or a DEAC (bank deactivate) command. Although the ’C80 does not support the DEAC command, external logic can be used to produce the command. The DEAC command is discussed in the Timing Evaluation section.

Enabling SDRAM Support

In order to support SDRAM, the system must signal the ’C80 that SDRAM is present. This is accomplished by presenting one of the SDRAM CT codes during the initial power-up DRAM refresh sequence. The ’C80 performs 32 refresh cycles at reset to ensure that all DRAMs and SDRAMs (if present) are initialized properly. The system designer should use one of the five least significant bits (LSBs) of the refresh pseudo-address (output on A[31:16]) to select between SDRAM and standard DRAM/VRAM banks, if both are present.

The first refresh which receives an SDRAM CT code is abandoned and an internal latch is set to indicate the presence of SDRAM. The TC then executes a DCAB command to deactive all SDRAM banks and then continues with the refresh sequence. When initial refresh cycles are completed, the TC performs an MRS cycle to initialize the SDRAM mode register.

MRS Cycle

After reset initialization cycles are performed on SDRAM, the ’C80 performs an MRS cycle which configures the SDRAM for operation. Read latency and burst length are programmed by executing an MRS

command with information being entered on address lines A0–A11 of the SDRAM. The ’C80 programs the SDRAM according to the CT information input at the beginning of the MRS cycle, as shown in Table 7. The burst-length-of-1 code should be specified only for pipeline architecture SDRAMs which support changing the column address on every cycle.

Table 7. MRS Value

SDRAM Address Pin	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
Meaning	reserved			0	0	read latency			S/I	burst length		
’C80 Output Value	0	0	0	0	0	0	1	CT0	0	0	0	CT1

Because the MRS register is programmed through the SDRAM’s address inputs, the location of the MRS value on the ’C80’s logical address bits varies according to the bus size of the addressed bank of SDRAM. The alignment of the MRS value is shown in Table 8. The location on the *physical* address bus is dependent on the address shift.

Table 8. MRS Value Alignment

Bus Size		Logical Address Bits															
BS1	BS0	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	X	X	X	X	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	1	X	X	X	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	X
1	0	X	X	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	X	X
1	1	X	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	X	X	X

It is assumed that all SDRAM banks will have their mode registers set at the same time. Although the TC cannot perform MRS cycles to different banks, external hardware can be added to support multiple banks of SDRAM with different CAS latencies and/or burst lengths. The system designer must supply the CT code for the first MRS cycle when it occurs, and abort that cycle with a retry ($\overline{\text{RETRY}}$ low at the end of R2). The MRS cycle still runs to completion for the first bank, but the TC does not recognize that it is finished and therefore attempts another MRS cycle. During this second MRS cycle, a different CT code can be specified for a different bank. This requires that external logic must:

- Track which MRS cycle is the current one
- Decode the control signals to each SDRAM bank and assert them during the correct MRS cycle
- Recognize the difference between this *intended* retry and a retry that results from another condition, such as an error or arbitration conflict in the system

SDRAM SYSTEM OVERVIEW

In this application report, the system shown in Figure 1 is considered. The glue logic associated with this design is implemented in programmable array logic / gate array logic (PAL/GALs), although the equations presented are also valid for other implementations.

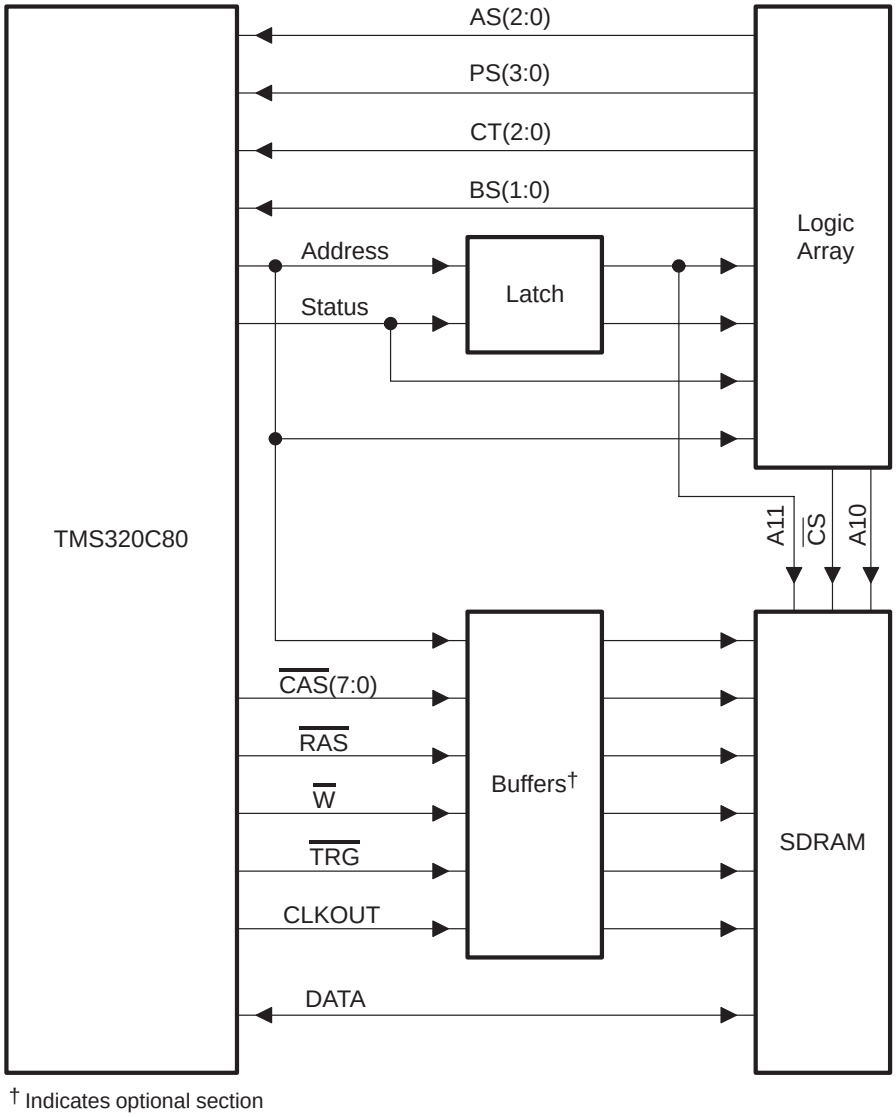

Figure 1. SDRAM Interface Block Diagram

Data Buffers/Transceivers

Because the ’C80 is a 3.3-V part, all inputs to the processor must not exceed V_{CC3} + 0.3 V. For this reason, the ’C80’s data lines D[63:0] must be buffered to 3.3 V using bus transceivers, such as the SN74LVT16245A (X4), when interfacing with 5-V memories and peripherals. The TMS626802 SDRAMs do not require such transceivers for interfacing with the ’C80.

Control Lines

The ’C80’s CLKOUT output, which is used to drive the CLK inputs of eight SDRAMs, is the first signal to be considered here. The TMS626802 specifies a load impedance of 7 pF for the CLK input, or a total of a 56-pF (7 pF × 8 devices) load to the ’C80. Since this can pose a considerable timing constraint to the rest of the system, it is advised that the CLKOUT output of the ’C80 be buffered, using an SN74LVT16244. Multiple copies of CLKOUT (BCLKOUTn) allow other memories and peripherals to be interfaced to the ’C80 without the constraints imposed by the loading of CLKOUT by the SDRAMs. In this example, two copies (BCLKOUT1 and BCLKOUT2) are used to interface with the SDRAM bank.

For the same reason, it is also a good idea to buffer the other signals which interface to all eight SDRAMs — which include $\overline{\text{RAS}}$, $\overline{\text{W}}$, and $\overline{\text{TRG}}$. As each $\overline{\text{CAS}}$ signal interfaces directly to only one SDRAM (5-pF load), it is not necessary to buffer these signals. However, since $\overline{\text{CAS}}$ lines are typically heavily loaded, it is advantageous to use buffered $\overline{\text{CAS}}$ ($\overline{\text{BCAS}}$) lines as the DQM inputs. Each $\overline{\text{CAS}}$ line maps to the appropriate SDRAM DQM input as shown in Table 9.

Table 9. $\overline{\text{CAS}}$-to-DQM Mappings

BUFFERED ’C80 $\overline{\text{CAS}}$ SIGNAL	CONNECT TO DQM OF SDRAM CONNECTED TO ....
$\overline{\text{BCAS7}}$	D63:D56
$\overline{\text{BCAS6}}$	D55:D48
$\overline{\text{BCAS5}}$	D47:D40
$\overline{\text{BCAS4}}$	D39:D32
$\overline{\text{BCAS3}}$	D31:D24
$\overline{\text{BCAS2}}$	D23:D16
$\overline{\text{BCAS1}}$	D15:D8
$\overline{\text{BCAS0}}$	D7:D0

NOTE: $\overline{\text{CAS0}}$ always controls bits 0–7 of the data bus, so the above is valid for both endian modes.

Address Bus

Because SDRAM is a paged memory type, it requires row/column address multiplexing. The ’C80 provides multiplexed addresses directly, controlled by the AS (address shift) inputs. To determine the amount of address shift required and address bus connections, both the memory architecture and the system configuration should be evaluated.

The TMS626802s used in this report have nine column address bits (A0–A8). On a 64-bit bus, the DQM signals serve as byte strobes, so the LS 3 bits of column address can be ignored (in a 32-bit system, the LS 2 bits would be ignored). Thus, in this system, the LS ’C80 address line is A12 (9+3). To line up *logical* address bit 3 with *physical* bit 12, an address-shift amount of AS = 010 is used, as shown in Figure 7–2 of the *TMS320C80 (MVP) Transfer Controller User’s Guide* (literature number SPRU105A). The nine column-address bits (C80A[20:12]) connect to SDRAM address lines A0–A8; additionally, A21 is connected to A9 (A9 is a *don’t care* at column time). The two remaining address lines on the TMS626802s have special meanings and must be generated in external logic.

Like the data lines, the ’C80’s address bus may be connected directly to the SDRAMs. As the address bus is output only, C80A[31:0] may be connected directly to other memories and peripherals, provided that the V_{IH} specification of these devices is met. External pullups (to 3.3 V) or buffers (SN74LVT16244-type ×2) may be required if the memory interface presents a particularly large load to the ’C80. Since portions of the address bus are typically the most heavily loaded signals in a design, it is advantageous to include buffers on the address bus.

A10, A11 and $\overline{CS}$

In order to complete the SDRAM interface, it is necessary to develop some glue logic to generate the $\overline{CS}$ and the remaining two address lines, A10 and A11. Because the ’C80 uses a multiplexed address scheme, a simple address decode is not sufficient for the $\overline{CS}$ signal. The glue logic must generate the $\overline{CS}$ input from both address and status codes output from the ’C80. Also, since STATUS[5:0] and A[31:0] are subject to change from cycle to cycle, it is required that external hardware latch the address and status information at row time. This can be accomplished by using a transparent latch, such as the SN74LVT16373 and the ’C80’s row latch ($\overline{RL}$) signal.

The number of address lines used to generate the decode is system dependent. In this example, four lines, C80A[31:28], are used; this designates sixteen 256M slots in the memory map. The SDRAM bank is located at 0xC0000000. Additionally,’C80A[17:16] are latched for use in the decode logic; this designates four areas of memory that require refreshing (dynamic memory). ’C80A[17:16] represent the 2 LS bits of the refresh pseudo-address output during refreshes. In this report, SDRAM occupies bank 3 of the “refresh map” (LC80A16=LC80A17=1). The signals LSTAT[5:0] and LC80A[x:y] refer to the latched versions of STATUS[5:0] and C80A[x:y], respectively.

Chip Select

The $\overline{CS}$ signal for all the SDRAMs in the bank is the same. It must be decoded so that the device is selected ($\overline{CS}$ low) for all of the following:

- READ and WRITE commands
- MRS cycle(s)
- SDRAM refresh commands
- Deactivation (DCAB) cycles

It is not necessary that $\overline{CS}$ be low during read or write bursts, only that it be low during the command. However, the logic is easier to design if $\overline{CS}$ is low for all cycles that interface with the SDRAM bank.

Row-time status codes that affect the SDRAM bank are:

- 000000 – Normal read
- 000001 – Normal write
- 000010 – Refresh†
- 000011 – SDRAM DCAB
- 001100 – SDRAM MRS

† The same status code is output for all refresh cycles; bank decoding must be done using the refresh pseudo-address output on C80A[31:16] during the refresh cycle. SDRAM occupies bank 3 of the refresh map (C80A16 = C80A17 = 1 during refresh).

Other status codes can be used by systems designed for specific applications. For example, you may wish to add one of the XPT status codes to the SDRAM $\overline{CS}$ decoding, if external packet requests are used by the system. A pseudo-code expression for $\overline{CS}$ might look like the following:

```
!CS = (MRS_cycle) # (DCAB_cycle) # (SDRAM_read) # (SDRAM_write)
    # (SDRAM_refresh) # (APPLICATION_SPECIFIC_CS) ;
```

where

```
MRS_cycle     =  (!LSTAT5 & !LSTAT4 & LSTAT3 & LSTAT2 & !LSTAT1 & !LSTAT0);
DCAB_cycle    =  (!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & LSTAT1 & LSTAT0);
SDRAM_read    =  (!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & !LSTAT1 & !LSTAT0)
                 & (LC80A31 & LC80A30 & !LC80A29 & !LC80A28);
SDRAM_write   =  (!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & !LSTAT1 & LSTAT0)
                 & (LC80A31 & LC80A30 & !LC80A29 & !LC80A28);
SDRAM_refresh =  (!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & LSTAT1 & !LSTAT0)
                 & (LC80A17 & LC80A16);
REFRESH       =  (!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & LSTAT1 & !LSTAT0);
```

A10 and A11

Address lines A11 and A10 are defined a little differently than the other address lines of SDRAM. Address line A11 is defined as the bank select and, therefore, must remain at a constant level during an entire row of SDRAM accesses. Address line A11 should be connected to a latched address bit. As discussed previously, address bits A0–A9 connect to 'C80 address lines A12–A21. A10, therefore, is generated with 'C80 address line A22, and thus A11 should be latched address line A23 (LC80A23).

A10 of the SDRAM bank has multiple functions. When a bank-activate (ACTV) command is issued, it must be valid as part of the row address. During read and write commands, the system must force A10 low to prevent auto-deactivation. During a bank-deactive (DCAB) command, A10 must be high (the 'C80 drives C80A[31:0] to 1s during a DCAB cycle in order to force C80A22 to a 1). There are several ways to generate the A10 signal, but the simplest is:

```
A10 = C80A22 & !RAS ;
```

While simple, this implementation results in a timing constraint during ACTV commands. The section on A10 addresses this issue and presents an alternative method for generating A10.

'C80 Cycle Configuration Inputs

In addition to providing the SDRAM with various signals, external logic also must provide the 'C80 with some control signals to describe the types of cycles that the 'C80 should generate. These signals are sampled at row time during the R2 state; namely, AS[2:0], BS[1:0], CT[2:0], and PS[3:0]. The logic that creates these signals also must be responsible for decoding this information for other memory types and peripherals in the system as well. The decoding of MRS cycles, DCAB cycles, reads, writes, and refreshes is identical to those presented in the section on chip select; however, address and status lines are *unlatched* in the following equations, due to timing constraints.

```
AS2 =(SYSTEM_SPECIFIC_REQUIRING_AS2=1)
AS1 = (C80A31 & C80A30 & !C80A29 & !C80A28);
   # (MRS_cycle)
   # (SYSTEM_SPECIFIC_REQUIRING_AS1=1);
AS0 =(SYSTEM_SPECIFIC_REQUIRING_AS1=1);
BS1 = (C80A31 & C80A30 & !C80A29 & !C80A28)
   # MRS_cycle
   #(SYSTEM_SPECIFIC_REQUIRING_BS1=1);
BS0 = (C80A31 & C80A30 & !C80A29 & !C80A28)
   # (MRS_cycle)
   #(SYSTEM_SPECIFIC_REQUIRING_BS0=1);
CT2 =(SYSTEM_SPECIFIC_REQUIRING_CT2=1);
CT1 =(C80A31 & C80A30 & !C80A29 & !C80A28)
   & (!REFRESH)
   # (MRS_cycle)
   # (SDRAM_refresh)
   # (DCAB_cycle)
   #(SYSTEM_SPECIFIC_REQUIRING_CT1=1);
CT0 = (C80A31 & C80A30 & !C80A29 & !C80A28)
   & (!REFRESH)
   # (MRS_cycle)
   # (SDRAM_refresh)
   # (DCAB_cycle)
   #(SYSTEM_SPECIFIC_REQUIRING_CT0=1);
PS3 = (C80A31 & C80A30 & !C80A29 & !C80A28)
   #(SYSTEM_SPECIFIC_REQUIRING_PS3=1);
PS2 =(SYSTEM_SPECIFIC_REQUIRING_PS2=1);
PS1 =(C80A31 & C80A30 & !C80A29 & !C80A28)
   #(SYSTEM_SPECIFIC_REQUIRING_PS1=1);
PS0 =(SYSTEM_SPECIFIC_REQUIRING_PS0=1);
```

Cycle Code	Value
AS[2:0]	010
BS[1:0]	11 (64-bit bus)
CT[2:0]	011 (burst length 2, CAS latency 3)
PS[3:0]	1010 (4K)

NOTE: SYSTEM_SPECIFIC_REQUIRING_xxx conditions must mask out the SDRAM cycles for proper operation.

TIMING EVALUATION

A handful of timing parameters need to be evaluated in order to design the SDRAM interface; these include:

- Setup and hold times for data relative to CLKOUT – 'C80-driven (writes)
- Setup and hold times for address relative to CLKOUT
- Access time – SDRAM drives data bus (reads)
- Setup and hold times for A10, A11, and $\overline{CS}$
- Setup and hold times for $\overline{W}$, $\overline{RAS}$, $\overline{CAS}$, DQM inputs

Furthermore, the setup and hold times for the cycle configuration inputs to the 'C80 should be evaluated.

Data Setup and Hold Times

For SDRAM writes, the 'C80 drives the data bus with the characteristics shown in Table 10.

Table 10. Data-Bus Characteristics (SDRAM Writes)

PARAMETER	'C80 SPEED		UNITS
	40	50	MHz
$t_{h(OUTV-CKOL)}$	5.5	4.5	ns
$t_{h(CKOH-OUTV)}$	7	5	ns

Data setup and hold times are calculated as follows:

<u>50 MHz</u>

$t_{(data\ setup)} = t_{(data\ setup\ from\ 'C80)} + t_{(CLK\ delay\ min)}$

$t_{DS} = t_{h(OUTV-CKOL)} + t_{PLHMIN244}$

$= (t_H - 5.5) + t_{PLHMIN244}$

$= 4.5 + 1$

$= 5.5$ ns

$t_{(data\ hold)} = t_{(data\ hold\ from\ 'C80)} - t_{(CLK\ delay\ max)}$

$t_{DH} = t_{h(CKOH-OUTV)} - t_{PLHMAX244}$

$= (t_H - 5) - t_{PLHMIN244}$

$= 5 - 4.1$

$= .9$ ns

<u>40 MHz</u>

$t_{(data\ setup)} = t_{(data\ setup\ from\ 'C80)} + t_{(CLK\ delay\ min)}$

$t_{DS} = t_{h(OUTV-CKOL)} + t_{PLHMIN244}$

$= (t_H - 7) + t_{PLHMIN244}$

$= 5.5 + 1$

$= 6.5$ ns

$t_{(data\ hold)} = t_{(data\ hold\ from\ 'C80)} - t_{(CLK\ delay\ max)}$
$t_{DH} = t_{h(CKOH-OUTV)} - t_{PLHMAX244}$
$= (t_H - 5.5) - t_{PLHMAX244}$
$= 7 - 4.1$
$= 2.9$ ns

The TMS626802-10 requires 3-ns setup and 1-ns hold time for data relative to CLK.

Address Setup and Hold Times

The address setup and hold times are the shown in Table 11.

Table 11. Address-Bus Characteristics (SDRAM)

PARAMETER	'C80 SPEED		UNITS
	40	50	MHz
$t_{h(OUTV-CKOL)}$	6.5	5.5	ns
$t_{h(CKOH-OUTV)}$	7	5	ns

Setup and hold times for the address inputs to the SDRAM bank are calculated as follows:

50 MHz

$t_{(addr\ setup)} = t_{h(OUTV-CKOL)} + t_{(CLK\ delay\ min)} - t_{(addr\ delay)}$
$t_{AS} = t_{su(AV-CKOH)} + t_{PLHMIN244} - t_{PxxMAX244}$
$= (t_H - 4.5) + t_{PLHMIN244} - t_{PxxMAX244}$
$= 5.5 + 1 - 4.1$
$= 2.4$ ns

$t_{(addr\ hold)} = t_{(addr\ hold\ from\ 'C80)} - t_{(CLK\ delay\ max)} + t_{(addr\ delay)}$
$t_{AH} = t_{h(CKOH-OUTV)} - t_{PLHMAX244} + t_{PxxMIN244}$
$= (t_H - 5) - t_{PLHMAX244} + t_{PxxMIN244}$
$= 5 - 4.1 + 1$
$= 1.9$ ns

40 MHz

$t_{(addr\ setup)} = t_{h(OUTV-CKOL)} + t_{(CLK\ delay\ min)} - t_{(addr\ delay)}$
$t_{AS} = t_{su(AV-CKOH)} + t_{PLHMIN244} - t_{PxxMAX244}$
$= (t_H - 6.5) + t_{PLHMIN244} - t_{PxxMAX245}$
$= 6 + 1 - 4.1$
$= 2.9$ ns

$t_{(addr\ hold)} = t_{(addr\ hold\ from\ 'C80)} - t_{(CLK\ delay\ max)} + t_{(addr\ delay)}$
$t_{AH} = t_{h(CKOH-OUTV)} - t_{PLHMAX244} + t_{PxxMIN244}$
$= (t_H - 5.5) - t_{PLHMAX244} + t_{PxxMIN244}$
$= 7 - 4.1 + 1$
$= 3.9$ ns

Like the data setup and hold requirements, the setup and hold times for addresses relative to CLK are 3 ns and 1 ns respectively.

Access Time

Access time for SDRAM reads for the TMS626802-10 device is specified at a maximum of 8 ns for CAS-latency-3 accesses. The ’C80 requires 14.7-ns access time at 50 MHz, and 17-ns at 40 MHz.

$t_{access} = t_{(SDRAM\ access)} + t_{(CLK\ delay)}$
$t_a = t_{AC(latency\ 3)} + t_{PxxMAX245}$
$= 9 + 4.1$
$= 13.1$ ns

$t_{DH} = t_{OH} + t_{(CLK\ delay)}$
$= t_{OH} + t_{PLHMIN244}$
$= 3 + 1$
$= 4$ ns

The ’C80 requires 2 ns of hold for data.

$\overline{CS}$ and A11

The $\overline{CS}$ input generated in external logic is activated or deactivated a maximum delay equal to the propagation delay of the logic array after the rise of $\overline{RL}$. The minimum time between the rise of $\overline{RL}$ and the entry of a command to the SDRAM is 53.5 ns ($6t_H$–6.5). In this report, PAL22LVT10-7s are used; therefore, a maximum logic delay of 7.5 ns is considered in the following analysis.

50 MHz

$t_{(CS\ setup)} = t_{(setup\ from\ 'C80)} - t_{prop\ latch} + t_{(CLK\ delay)} - t_{(decode\ prop)}$
$t_{CS} = t_{h(OUTV-OUTV)} - t_{PLHMAX373} + t_{PLHMIN244} - t_{PROPPAL}$
$= (6t_H - 6.5) - t_{PLHMAX373} + t_{PLHMIN244} - t_{PROPPAL}$
$= 53.5 - 6.9 + 1 - 7.5$
$= 40.1$ ns

40 MHz

$t_{(CS\ setup)} = t_{(setup\ from\ 'C80)} - t_{prop\ latch} + t_{(CLK\ delay)} - t_{(decode\ prop)}$
$t_{CS} = t_{h(OUTV-OUTV)} - t_{PLHMAX373} + t_{PLHMIN244} - t_{PROPPAL}$
$= (6t_H - 7) - t_{PLHMAX373} + t_{PLHMIN244} - t_{PROPPAL}$
$= 75 - 6.9 + 1 - 7.5$
$= 61.6$ ns

The A11 input (latched C80A23) setup time is calculated below:

50 MHz

$t_{(A11\ setup)} = t_{(setup\ from\ 'C80)} - t_{prop\ latch} + t_{(CLK\ delay)}$
$t_{AS} = t_{h(OUTV-OUTV)} - t_{PHLMAX373} + t_{PLHMIN244}$
$= (6t_H - 6.5) - t_{PHLMAX373} + t_{PLHMIN244}$
$= 53.5 - 6.9 + 1$
$= 47.6$ ns

40 MHz

$t_{(A11\ setup)} = t_{(setup\ from\ 'C80)} - t_{prop\ latch} + t_{(CLK\ delay)}$
$t_{AS} = t_{h(OUTV-OUTV)} - t_{PHLMAX373} + t_{PLHMIN244}$
$= (6t_H - 7) - t_{PHLMAX373} + t_{PLHMIN244}$
$= 75 - 6.9 + 1$
$= 69.1$ ns

The TMS626802-10s require 3-ns setup and 1-ns hold time for the $\overline{CS}$ input.

A10

As implemented here, A10 poses a timing constraint. A10 is low when $\overline{RAS}$ is high and is valid otherwise. This means that during command entry at row time, it may take up to the propagation delay of the generating logic (7.5 ns in this example) for A10 to be valid during R5. Since the ’C80 specifies a minimum of $t_{h(OUTV-CKOH)} = t_H - 4.1$ ns (at 50 MHz) for the delay from $\overline{RAS}$ to CLKOUT, the 3-ns setup time required by the SDRAM for A10 may be violated. Note that this exists only for the commands initiated at row time. The DCAB cycle at the end of a row access is not affected.

If A10 is sampled low during the DCAB cycle, then the DEAC command is executed and the bank selected by A11 (which is latched) is deactivated. If A10 is sampled high, the DCAB cycle completes normally. In either case, both banks are deactivated correctly at the end of the access. Since many logic implementations are even slower than the PALs used here, another method may be desirable. The goal is to keep A10 (C80A22) valid during the ACTV (activate) command, and low afterwards (which results in an end-of-row DEAC command, instead of the ’C80-instituted DCAB command). One implementation might be as shown in Figure 2.

Alternate Method for Generating A10

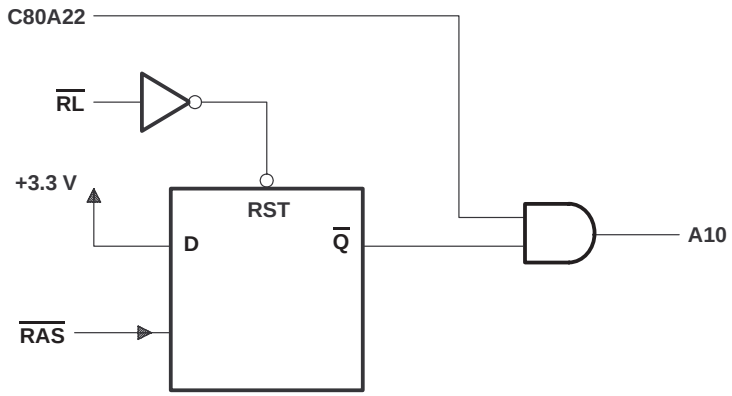

Figure 2. A10 Generator

This implementation is simple and works for normal accesses as well as PDPTs. The resulting output waveform for A10 is shown in Figure 3.

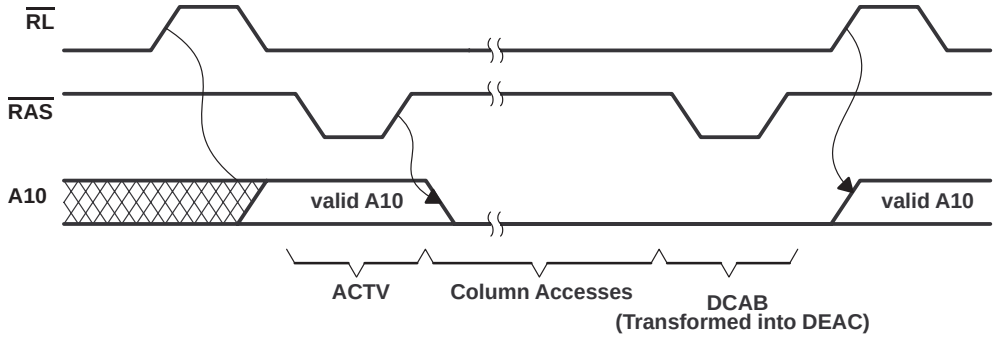

Figure 3. A10 Waveform

$\overline{W}$, $\overline{RAS}$, $\overline{CAS}$, and DQM

Setup and hold times for the SDRAM $\overline{W}$, $\overline{RAS}$, $\overline{CAS}$, and DQM signals can be calculated as follows. Recall that all four of these signals, as well as the CLK input to which they are referenced, are buffered signals, and therefore, the minimum propagation delay of the buffer is included. The TMS626802-10 specifies setup and hold times of 3 ns and 1 ns respectively.

50 MHz

$t_{(RAS\ setup)} = t_{(setup\ from\ 'C80)} + t_{(CLK\ delay)} - t_{(RAS\ prop)}$
$t_{CS} = t_{(OUTV\text{-}CKOL)} + t_{PLHMIN244} - t_{PHLMAX244}$
$= (t_H - 4) + t_{PLHMIN244} - t_{PHLMAX244}$
$= 6 + 1 - 4.1$
$= 2.9$ ns

$t_{(RAS\ hold)} = t_{(hold\ from\ 'C80)} - t_{(CLK\ delay)} + t_{(RAS\ prop)}$
$t_{CH} = t_{(CKOH\text{-}OUTV)} - t_{PLHMAX244} + t_{PLHMIN244}$
$= (t_H - 5.5) - t_{PLHMAX244} + t_{PLHMIN244}$
$= 5 - 4.1 + 1$
$= 1.9$ ns

40 MHz

$t_{(RAS\ setup)} = t_{(setup\ from\ 'C80)} + t_{(CLK\ delay)} - t_{(RAS\ prop)}$
$t_{CS} = t_{(OUTV\text{-}CKOL)} + t_{PLHMIN244} - t_{PHLMAX244}$
$= (t_H - 5.5) + t_{PLHMIN244} - t_{PHLMAX244}$
$= 7 + 1 - 4.1$
$= 3.9$ ns

$t_{(RAS\ hold)} = t_{(hold\ from\ 'C80)} - t_{(CLK\ delay)} + t_{(RAS\ prop)}$
$t_{CH} = t_{(CKOH\text{-}OUTV)} - t_{PLHMAX244} + t_{PLHMIN244}$
$= (t_H - 5.5) - t_{PLHMAX244} + t_{PLHMIN244}$
$= 7 - 4.1 + 1$
$= 3.9$ ns

The above is valid for $\overline{RAS}$, $\overline{TRG}$ (SDRAM $\overline{CAS}$) and $\overline{W}$.

Setup and hold times for the DQM signals ('C80 $\overline{CAS}$) are as follows:

50 MHz

$t_{(DQM\ setup)} = t_{(setup\ from\ 'C80)} + t_{(CLK\ delay)} - t_{(RAS\ prop)}$
$t_{CS} = t_{(OUTV\text{-}CKOL)} + t_{PLHMIN244} - t_{PHLMAX244}$
$= (t_H - 4.5) + t_{PLHMIN244} - t_{PHLMAX244}$
$= 5.5 + 1 - 4.1$
$= 2.4$ ns

$t_{(DQM\ hold)} = t_{(hold\ from\ 'C80)} - t_{(CLK\ delay)} + t_{(RAS\ prop)}$
$t_{CH} = t_{(CKOH\text{-}OUTV)} - t_{PLHMAX244} + t_{PLHMIN244}$
$= (t_H - 5) - t_{PLHMAX244} + t_{PLHMIN244}$
$= 5 - 4.1 + 1$
$= 1.9$ ns

40 MHz

$t_{(DQM\ setup)} = t_{(setup\ from\ 'C80)} + t_{(CLK\ delay)} - t_{(RAS\ prop)}$
$t_{CS} = t_{(OUTV\text{-}CKOL)} + t_{PLHMIN244} - t_{PHLMAX244}$
$= (t_H - 6.5) + t_{PLHMIN244} - t_{PHLMAX244}$
$= 6 + 1 - 4.1$
$= 2.9$ ns

$t_{(DQM\ hold)} = t_{(hold\ from\ C80)} - t_{(CLK\ delay)} + t_{(RAS\ prop)}$
$t_{CH} = t_{(CKOH\text{-}OUTV)} - t_{PLHMAX244} + t_{PLHMIN244}$
$= (t_H - 5.5) - t_{PLHMAX244} + t_{PLHMIN244}$
$= 7 - 4.1 + 1$
$= 3.9$ ns

'C80 Cycle Configuration Inputs

The 'C80's cycle configuration inputs AS[2:0], PS[3:0], CT[2:0], and BS[1:0] are sampled during the R2 state, and determine the types of cycles that the 'C80 generates. These inputs can be decoded from address and status information output by the 'C80 during row time. The generating logic should use unlatched versions of the address and status; otherwise, access and setup times for the 'C80 may not be met.

$t_{a(MIDV\text{-}CFGV)} = t_{PROPPAL}$
$= 7.5$ ns

The 'C80 only requires that $t_{a(MIDV\text{-}CFGV)}$ be less than 20 ns ($3t_H - 10$) at 50 MHz.

APPENDIX A: BILL OF MATERIALS

Table 12. Bill of Materials

QUANTITY USED	PART TYPE	DESIGNATORS
55	0.1 μF	C1A C1B C2A C2B C3A C3B C4A C4B C5A C5B C6A C6B C7A C7B C8A C8B C15A C15B C15C C15D C15E C15F C15G C15H C15I C15J C15K C15L C15M C15N C15P C15Q C17 C18A C18B C19A C19B C20A C20B C21A C21B C22C C22D C22A C23 C24 C25 C27A C27B C28A C28B C29A C29B C30A C30B
3	4.7-μF electrolytic cap	C15R C15S C22B
6	10-kΩ resistor	R1 R2 R3 R4 R5 R6 R7 R8 R22
6	22-Ω resistor pack (8)	RP1 RP2 RP3 RP4 RP5 RP6
2	22 μF	C59 C60
3	SN74LVT16244	U18 U19 U20
1	SN74LVT16373	U21
1	7 × 2 HEADER	U16
1	OSC14	U17
3	PAL22LV10PLCC	U23 U24 U25
1	SW SPST	S1
1	TL7705A	U22
1	TMS320C80-GF	U15
8	TMS626802	U1 U2 U3 U4 U5 U6 U7 U8

APPENDIX B: SCHEMATICS

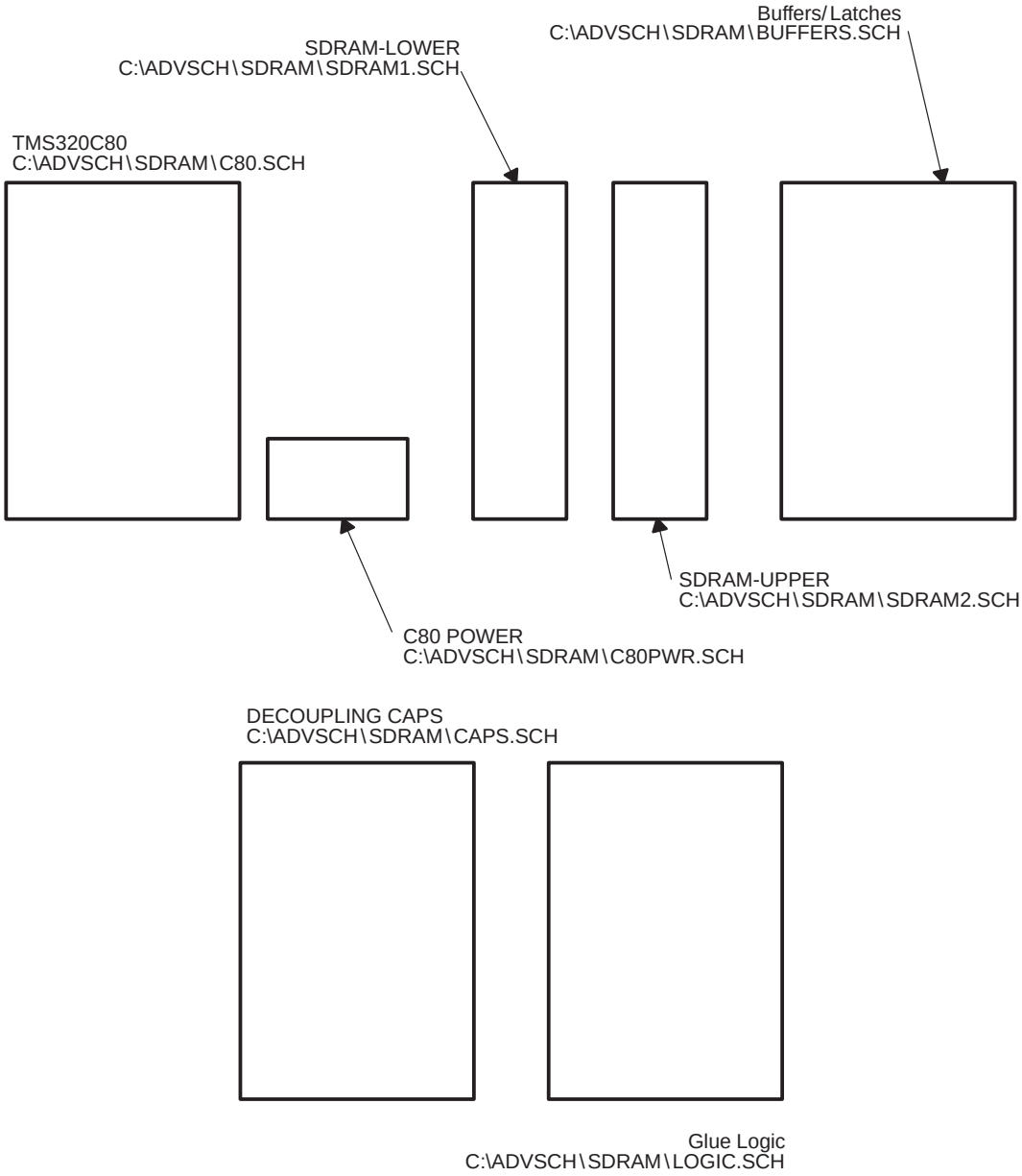

Figure 4. SDRAM Application Report

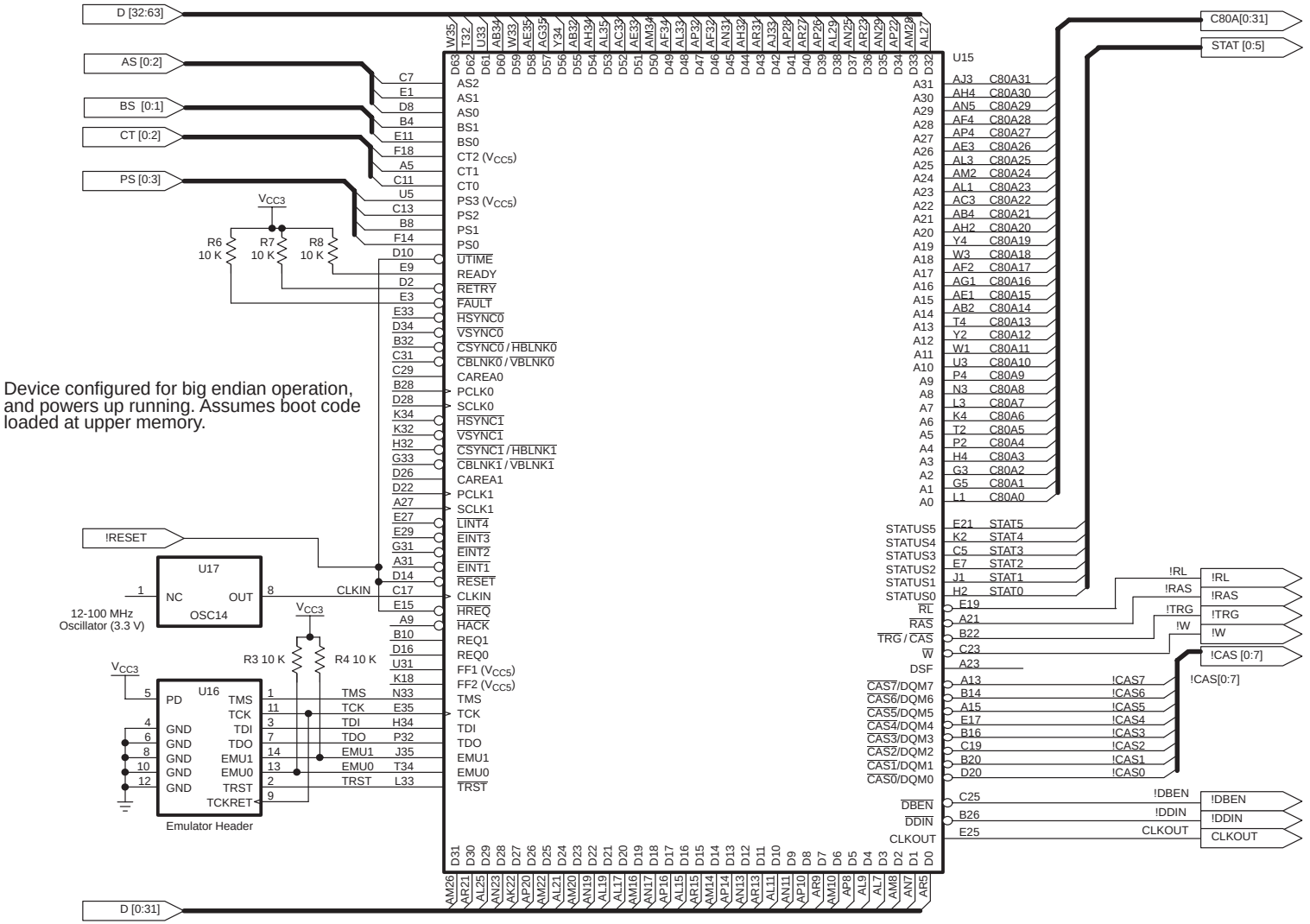

Figure 5. TMS320C80-GF

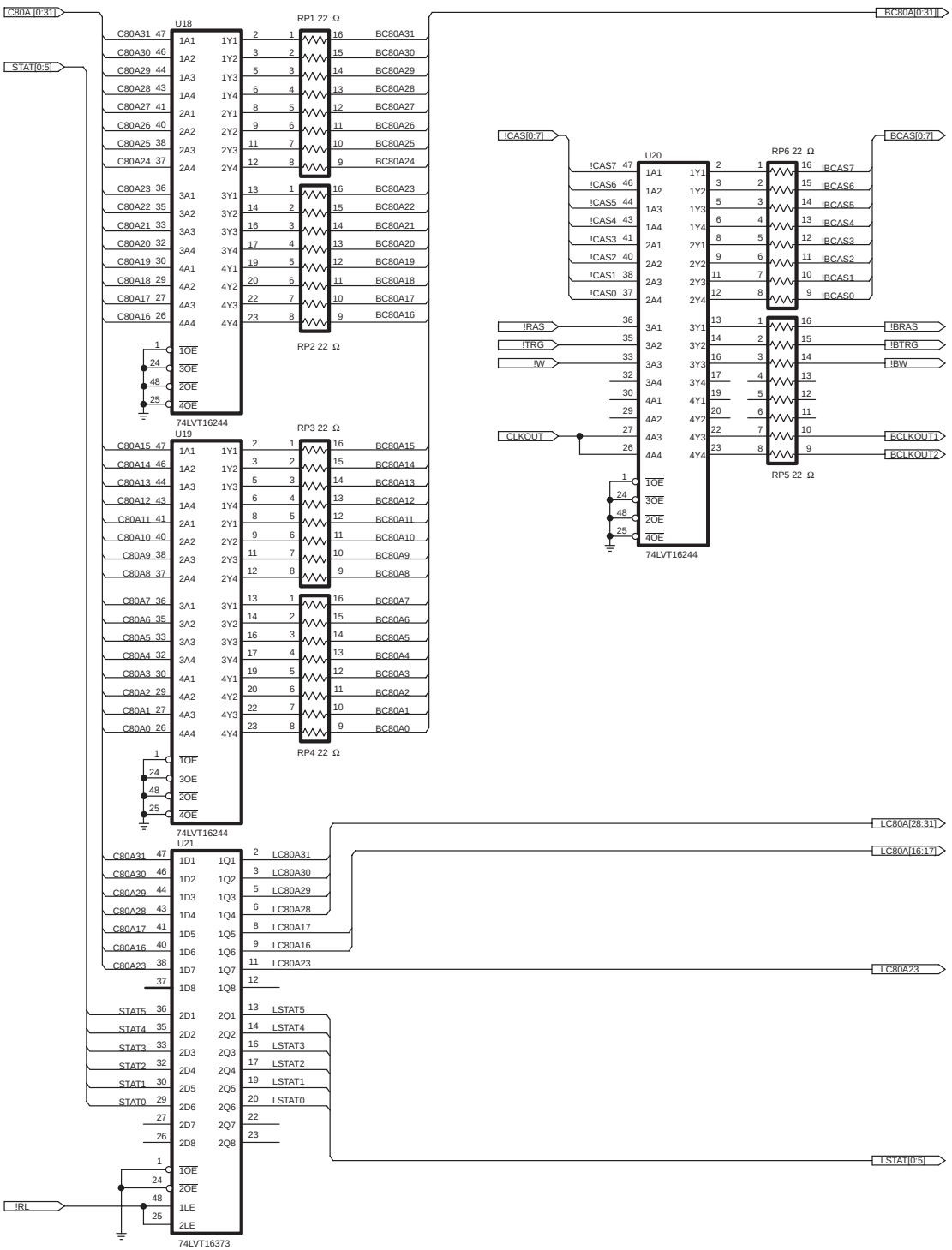

Figure 6. Address Buffers and Latches

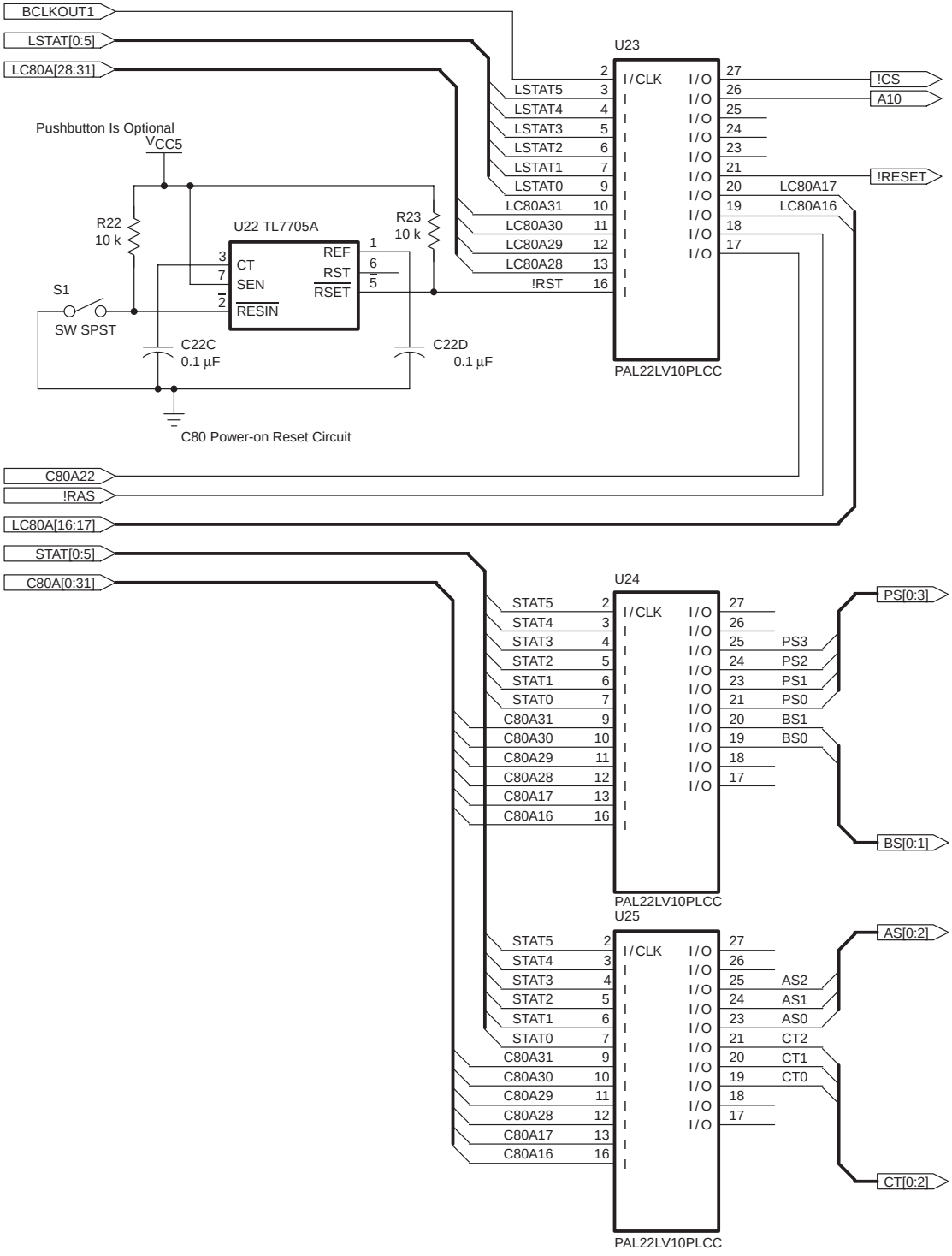

Figure 7. Glue Logic

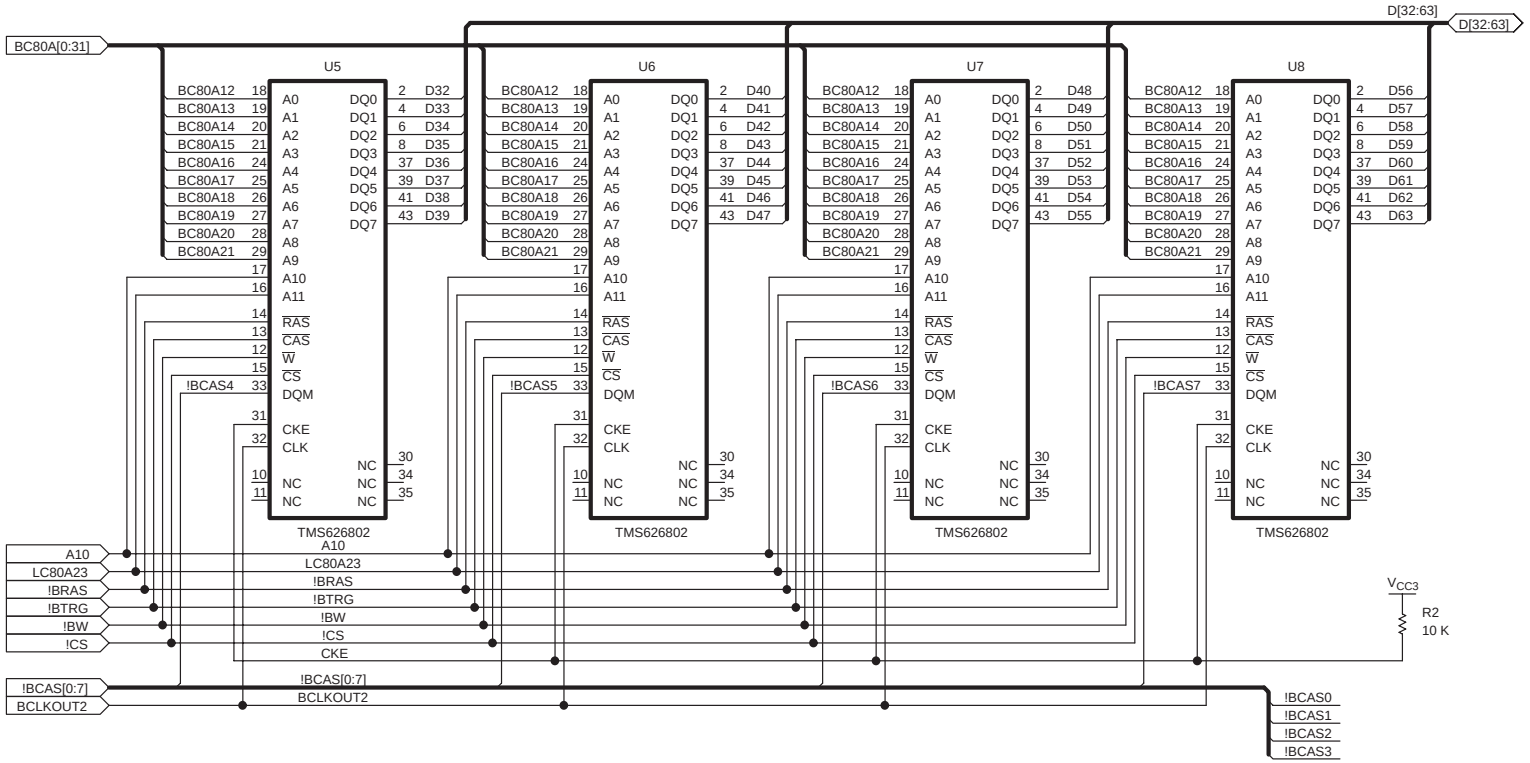

Figure 8. SDRAM Bank (Upper Half)

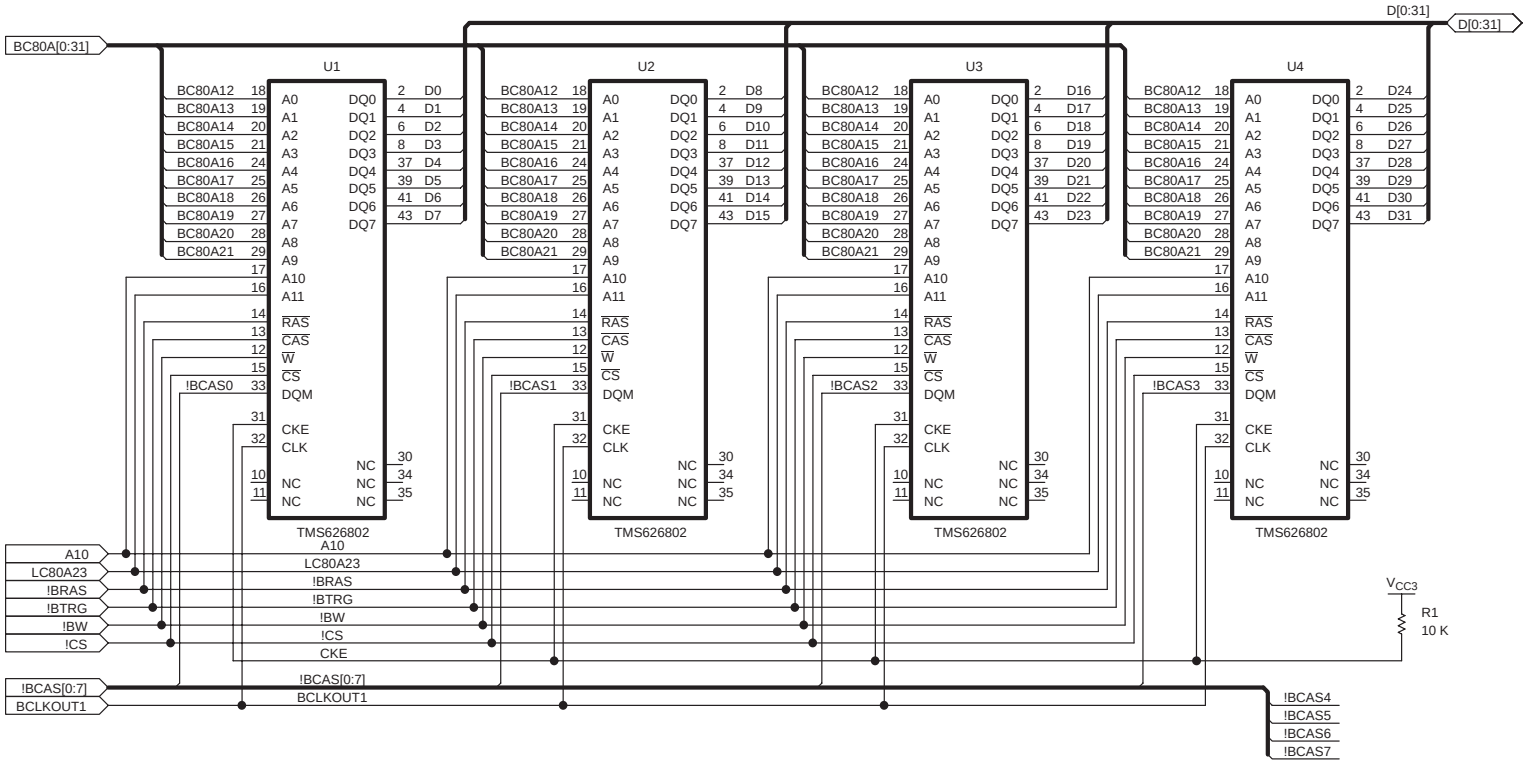

Figure 9. SDRAM Bank (Lower Half)

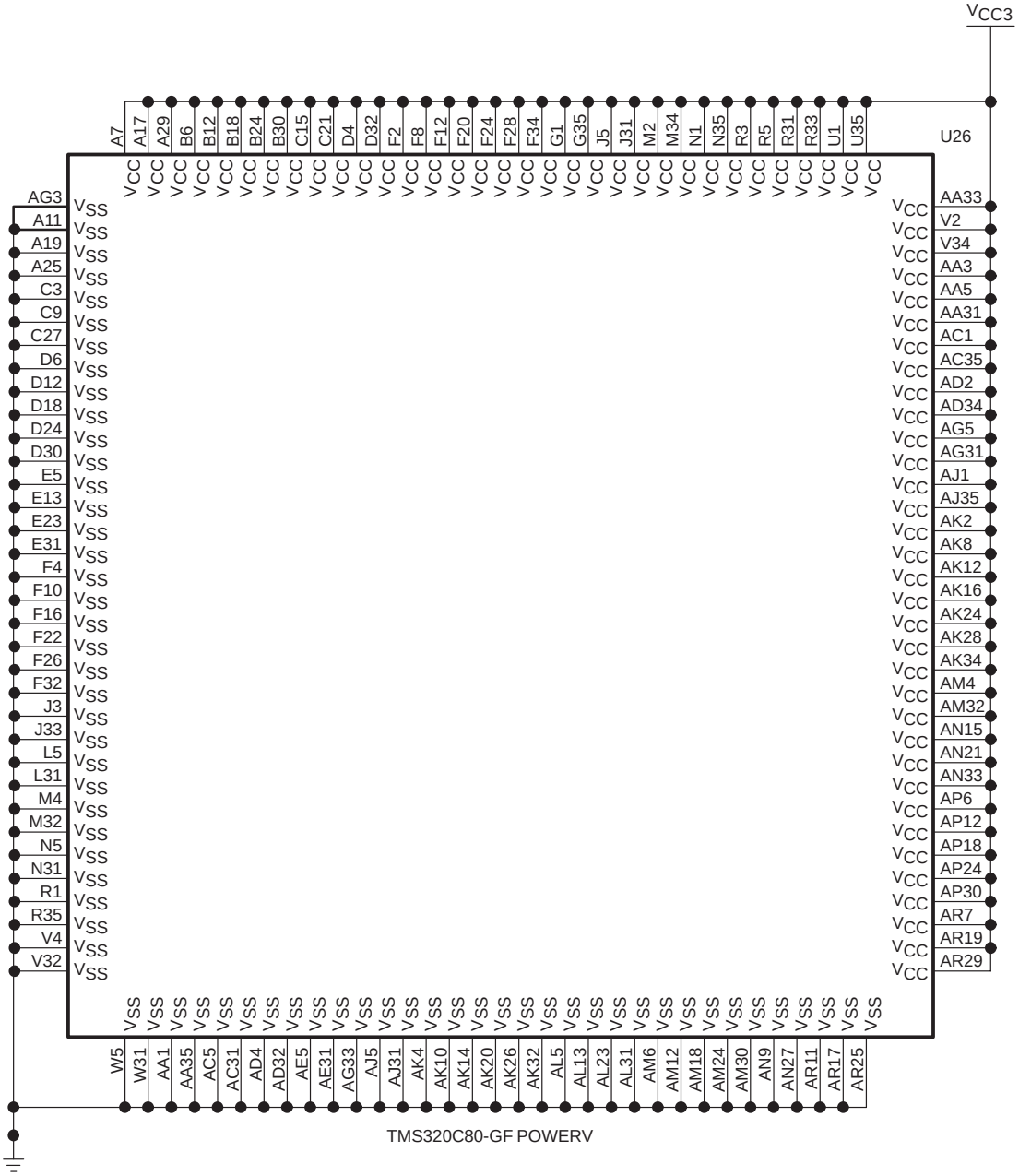

Figure 10. Power and Ground Connections

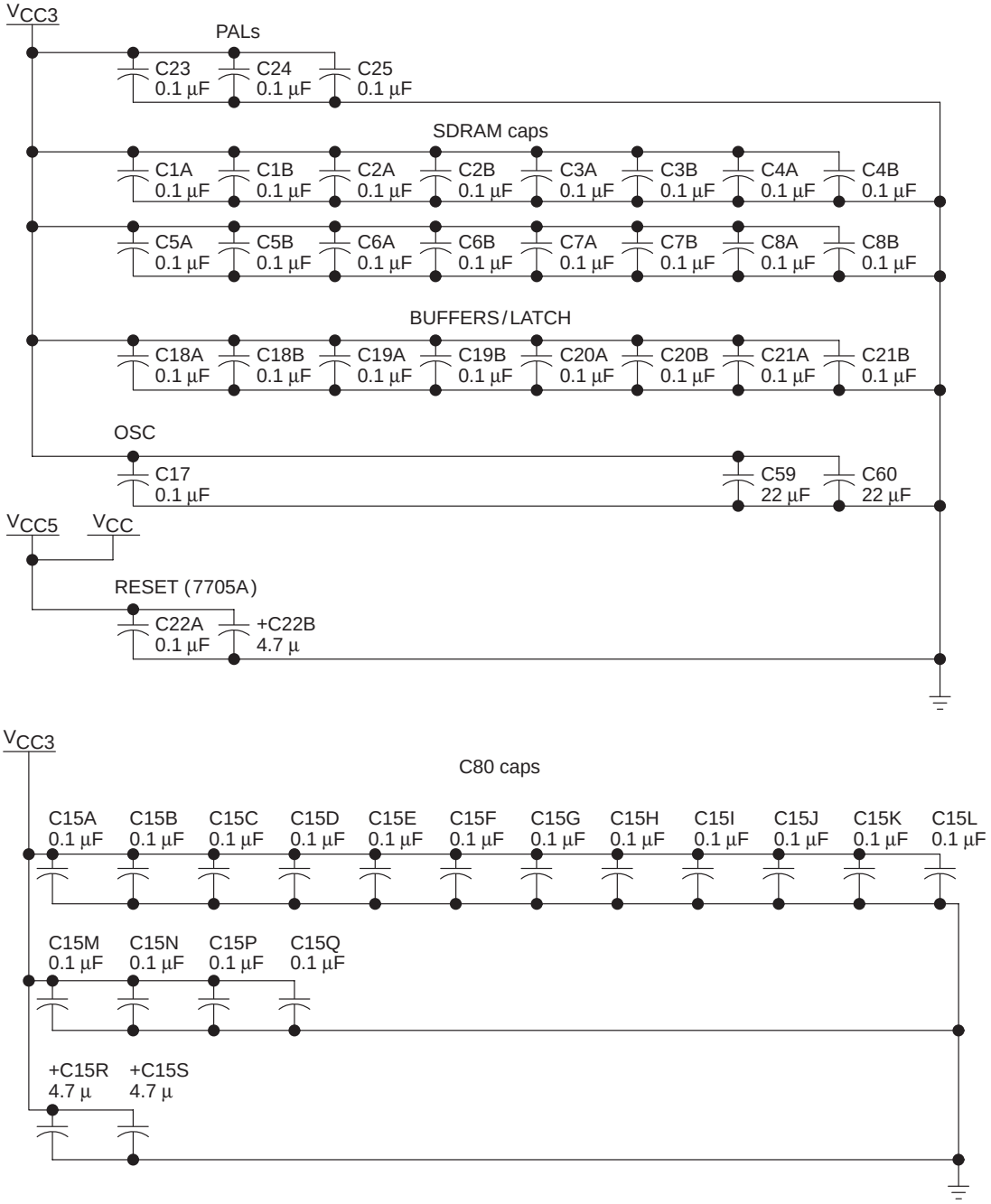

Figure 11. Decoupling Caps

APPENDIX C: ABEL™ FILES

```
module    SDRAMCONTROL
title'
DWG NAME  LOGIC.SCH
PAL #     U23
COMPANY   TEXAS INSTRUMENTS INCORPORATED
ENGINEER  C80 APPLICATIONS
DATE      05_08_95'
   xx_001    device 'P22V10C'; " LVT22V10-7C PLCC
   BCLKOUT1  Pin 2; "C80 CLKOUT
   LSTAT5    Pin 3; "C80 STATUS[5] --EXTERNALLY LATCHED BY _RL
   LSTAT4    Pin 4; "C80 STATUS[4] --EXTERNALLY LATCHED BY _RL
   LSTAT3    Pin 5; "C80 STATUS[3] --EXTERNALLY LATCHED BY _RL
   LSTAT2    Pin 6; "C80 STATUS[2] --EXTERNALLY LATCHED BY _RL
   LSTAT1    Pin 7; "C80 STATUS[1] --EXTERNALLY LATCHED BY _RL
   LSTAT0    Pin 9; "C80 STATUS[0] --EXTERNALLY LATCHED BY _RL
   LA31      Pin 10; "C80 address line 31 --EXTERNALLY LATCHED BY _RL
   LA30      Pin 11; "C80 address line 30 --EXTERNALLY LATCHED BY _RL
   LA29      Pin 12; "C80 address line 29 --EXTERNALLY LATCHED BY _RL
   LA28      Pin 13; "C80 address line 28 --EXTERNALLY LATCHED BY _RL
   _RST      Pin 16; "board RESET signal – active low
   vss       Pin 14; "Ground
   vcc       Pin 28; "Power
   _CS       Pin 27; "SDRAM chip select – active low
   A10       Pin 26; "SDRAM A10 signal
   "NC       Pin 25;
   "NC       Pin 24;
   "NC       Pin 23;
   RESET     Pin 21; "C80 _RESET signal – active low (powers up running, big endian)
   LA17      Pin 20; "C80 address line 17 --EXTERNALLY LATCHED BY _RL
   LA16      Pin 19; "C80 address line 16 --EXTERNALLY LATCHED BY _RL
   _RAS      Pin 18; "C80 row address strobe – active low
   C80A22    Pin 17; "C80 address line 22 – unlatched
```

ABEL is a trademark of DATA I/O.

```
”constants and alias names
   ADDR          = [LA31..LA28] ;
   REFADD        = [LA17..LA16] ;
   STAT          = [LSTAT5..LSTAT0];
   addrv         = 12 ;”SDRAM address valid
   refval        = 3 ; ”SDRAM bank refresh pseudo-address
   MRS_cycle     = (!LSTAT5 & !LSTAT4 & LSTAT3 & LSTAT2 & !LSTAT1 & !LSTAT0);
   MRS           = 12 ;
   DCAB_cycle    = (!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & LSTAT1 & LSTAT0);
   DCAB          = 3 ;
   SDRAM_read    = (!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & !LSTAT1 & !LSTAT0)
                 & (LA31 & LA30 & !LA29 & !LA28);
   READ          = 0 ;
   SDRAM_write   =(!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & !LSTAT1 & LSTAT0)
                 &(LA31 & LA30 & !LA29 & !LA28);
   WRITE         = 1 ;
   SDRAM_refresh=(!LSTAT5 & !LSTAT4 & !LSTAT3 & !LSTAT2 & LSTAT1 & !LSTAT0)
                 &(LA17 & LA16);
   REFRESH       = 2 ;
equations
!_CS       = (MRS_cycle) # (DCAB_cycle) # (SDRAM_read) # (SDRAM_write)
           # (SDRAM_refresh) ;
           ”# APPLICATION_SPECIFIC_CS)
A10        = C80A22 & !_RAS ;
_RESET     = _RST ;
test_vectors” SDRAM _CS
([ ADDR , STAT, REFADD ] -> [ _CS ])
 [ 0 , .X. , .X. ]   -> [ 1 ] ; ” SDRAM not addressed
 [ 1 , .X. , .X. ]   -> [ 1 ] ; ” SDRAM not addressed
 [ 2 , .X. , .X. ]   -> [ 1 ] ; ” SDRAM not addressed
 [ 3 , .X. , .X. ]   -> [ 1 ] ; ” SDRAM not addressed
 [ 4 , .X. , .X. ]   -> [ 1 ] ; ” SDRAM not addressed
 [ 5 , .X. , .X. ]   -> [ 1 ] ; ” SDRAM not addressed
 [ 6 , .X. , .X. ]   -> [ 1 ] ; ” SDRAM not addressed
 [ 7 , .X. , .X. ]   -> [ 1 ] ; ” SDRAM not addressed
 [ 8 , .X. , .X. ]   -> [ 1 ] ; ” SDRAM not addressed
 [ 9 , .X. , .X. ]   -> [ 1 ] ; ” SDRAM not addressed
 [ 10 , .X. , .X. ]  -> [ 1 ] ; ” SDRAM not addressed
 [ 11 , .X. , .X. ]  -> [ 1 ] ; ” SDRAM not addressed
```

```
 [ 13 , .X. , .X. ]     -> [ 1 ] ; " SDRAM not addressed
 [ 14 , .X. , .X. ]     -> [ 1 ] ; " SDRAM not addressed
 [ 15 , .X. , .X. ]     -> [ 1 ] ; " SDRAM not addressed
 [ 12 , READ, .X. ]     -> [ 0 ] ; " SDRAM read cycle
 [ 12 ,WRITE, .X. ]     -> [ 0 ] ; " SDRAM write cycle
 [ .X. ,REFRESH,refval] -> [ 0 ] ; " SDRAM refresh
 [ .X. ,REFRESH, 0 ]    -> [ 1 ] ; " refresh- not SDRAM
 [ .X. ,REFRESH, 1 ]    -> [ 1 ] ; " refresh- not SDRAM
 [ .X. ,REFRESH, 2 ]    -> [ 1 ] ; " refresh- not SDRAM
 [ .X. , MRS , .X .]    -> [ 0 ] ; " SDRAM MRS
 [ .X. , DCAB , .X. ]   -> [ 0 ] ; " SDRAM DCAB
test_vectors"A10
([ C80A22, _RAS ] -> [A10])
 [ 0 , 0 ] -> [ 0 ] ;
 [ 0 , 1 ] -> [ 0 ] ;
 [ 1 , 0 ] -> [ 1 ] ;
 [ 1 , 1 ] -> [ 0 ] ;
test_vectors"_RESET
([ _RST ] -> [_RESET ])
 [ 0 ] -> [ 0 ] ;
 [ 1 ] -> [ 1 ] ;
end SDRAMCONTROL
```

```
module    C80codes
title'
DWG NAME  LOGIC.SCH
PAL #     U24
COMPANY   TEXAS INSTRUMENTS INCORPORATED
ENGINEER  C80 APPLICATIONS
DATE      05_08_95'
   xx_001 device 'P22V10C'; " LVT22V10-7C PLCC
   STAT5  Pin 2; "C80 STATUS[5]
   STAT4  Pin 3; "C80 STATUS[4]
   STAT3  Pin 4; "C80 STATUS[3]
   STAT2  Pin 5; "C80 STATUS[2]
   STAT1  Pin 6; "C80 STATUS[1]
   STAT0  Pin 7; "C80 STATUS[0]
   A31    Pin 9; "C80 address line 31
   A30    Pin 10; "C80 address line 30
   A29    Pin 11; "C80 address line 29
   A28    Pin 12; "C80 address line 28
   A17    Pin 13; "C80 address line 17
   A16    Pin 16; "C80 address line 16
   vss    Pin 14; "Ground
   vcc    Pin 28; "Power
   "NC    Pin 27;
   "NC    Pin 26;
   PS3    Pin 25; "C80 PS[3] input (page size)
   PS2    Pin 24; "C80 PS[2] input (page size)
   PS1    Pin 23; "C80 PS[1] input (page size)
   PS0    Pin 21; "C80 PS[0] input (page size)
   BS1    Pin 20; "C80 BS[2] input (bus size)
   BS0    Pin 19; "C80 BS[1] input (bus size)
   "NC    Pin 18;
   "NC    Pin 17;
"constants and alias names
   ADDR      = [A31..A28] ;
   REFADD    = [A17..A16] ;
   STAT      = [STAT5..STAT0];
   addrv     = 12 ; "SDRAM address valid
   refval    = 3 ; "SDRAM bank refresh pseudo-address
   MRS_cycle = (!STAT5 & !STAT4 & STAT3 & STAT2 & !STAT1 & !STAT0);
```

```
    MRS          = 12 ;
    DCAB_cycle   =(!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & STAT0);
    DCAB         = 3 ;
    SDRAM_read   =(!STAT5 & !STAT4 & !STAT3 & !STAT2 & !STAT1 & !STAT0)
                 & (A31 & A30 & !A29 & !A28);
    READ         = 0 ;
    SDRAM_write  =(!STAT5 & !STAT4 & !STAT3 & !STAT2 & !STAT1 & STAT0)
                 &(A31 & A30 & !A29 & !A28);
    WRITE        = 1 ;
    SDRAM_refresh=(!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & !STAT0)
                 &(A17 & A16);
    REFH         = (!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & !STAT0);
    REFRESH      = 2 ;
equations
    PS3    = (A31 & A30 & !A29 & !A28) ;
    "#(SYSTEM_SPECIFIC_REQUIRING_PS3=1)
    PS2    = 0 ;
    "#(SYSTEM_SPECIFIC_REQUIRING_PS2=1)
    PS1    = (A31 & A30 & !A29 & !A28) ;
    "#(SYSTEM_SPECIFIC_REQUIRING_PS1=1)
    PS0    = 0;
    "#(SYSTEM_SPECIFIC_REQUIRING_PS0=1)
    BS1= (A31 & A30 & !A29 & !A28)
    #MRS_cycle ;
    "#(SYSTEM_SPECIFIC_REQUIRING_BS1=1);
    BS0    = (A31 & A30 & !A29 & !A28)
    #(MRS_cycle) ;
    "#(SYSTEM_SPECIFIC_REQUIRING_BS0=1);
test_vectors"PS3
([ ADDR , STAT ] -> [PS3])
 [ 0 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 1 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 2 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 3 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 4 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 5 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 6 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 7 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 8 , .X. ] -> [ 0 ] ;" SDRAM not addressed
```

```
 [ 9 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 10 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 11 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 12 , .X. ] -> [ 1 ] ;" SDRAM addressed
 [ 13 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 14 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 15 , .X. ] -> [ 0 ] ;" SDRAM not addressed
test_vectors"PS2
([ ADDR , STAT ] -> [PS2])
 [ 0 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 1 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 2 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 3 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 4 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 5 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 6 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 7 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 8 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 9 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 10 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 11 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 12 , .X. ] -> [ 0 ] ;" SDRAM addressed
 [ 13 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 14 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 15 , .X. ] -> [ 0 ] ;" SDRAM not addressed
test_vectors"PS1
([ ADDR , STAT ] -> [PS1])
 [ 0 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 1 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 2 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 3 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 4 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 5 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 6 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 7 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 8 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 9 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 10 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 11 , .X. ] -> [ 0 ] ;" SDRAM not addressed
```

```
 [ 12 , .X. ] -> [ 1 ] ;" SDRAM addressed
 [ 13 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 14 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 15 , .X. ] -> [ 0 ] ;" SDRAM not addressed
test_vectors"PS0
([ ADDR , STAT ] -> [PS0])
 [ 0 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 1 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 2 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 3 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 4 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 5 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 6 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 7 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 8 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 9 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 10 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 11 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 12 , .X. ] -> [ 0 ] ;" SDRAM addressed
 [ 13 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 14 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 15 , .X. ] -> [ 0 ] ;" SDRAM not addressed
test_vectors"BS1
([ ADDR , STAT ] -> [BS1])
 [ 0 , .X. ]  -> [ 0 ] ; " SDRAM not addressed
 [ 1 , .X. ]  -> [ 0 ] ; " SDRAM not addressed
 [ 2 , .X. ]  -> [ 0 ] ; " SDRAM not addressed
 [ 3 , .X. ]  -> [ 0 ] ; " SDRAM not addressed
 [ 4 , .X. ]  -> [ 0 ] ; " SDRAM not addressed
 [ 5 , .X. ]  -> [ 0 ] ; " SDRAM not addressed
 [ 6 , .X. ]  -> [ 0 ] ; " SDRAM not addressed
 [ 7 , .X. ]  -> [ 0 ] ; " SDRAM not addressed
 [ 8 , .X. ]  -> [ 0 ] ; " SDRAM not addressed
 [ 9 , .X. ]  -> [ 0 ] ; " SDRAM not addressed
 [ 10 , .X. ] -> [ 0 ] ; " SDRAM not addressed
 [ 11 , .X. ] -> [ 0 ] ; " SDRAM not addressed
 [ 13 , .X. ] -> [ 0 ] ; " SDRAM not addressed
 [ 14 , .X. ] -> [ 0 ] ; " SDRAM not addressed
 [ 15 , .X. ] -> [ 0 ] ; " SDRAM not addressed
```

```
 [ 12 , READ ] -> [ 1 ] ; ” SDRAM read cycle
 [ 12 , WRITE] -> [ 1 ] ; ” SDRAM write cycle
 [ .X. , MRS ] -> [ 1 ] ; ” SDRAM MRS
 test_vectors”BS0
([ ADDR , STAT ] -> [BS0])
 [ 0 , .X. ]  -> [ 0 ] ; ” SDRAM not addressed
 [ 1 , .X. ]  -> [ 0 ] ; ” SDRAM not addressed
 [ 2 , .X. ]  -> [ 0 ] ; ” SDRAM not addressed
 [ 3 , .X. ]  -> [ 0 ] ; ” SDRAM not addressed
 [ 4 , .X. ]  -> [ 0 ] ; ” SDRAM not addressed
 [ 5 , .X. ]  -> [ 0 ] ; ” SDRAM not addressed
 [ 6 , .X. ]  -> [ 0 ] ; ” SDRAM not addressed
 [ 7 , .X. ]  -> [ 0 ] ; ” SDRAM not addressed
 [ 8 , .X. ]  -> [ 0 ] ; ” SDRAM not addressed
 [ 9 , .X. ]  -> [ 0 ] ; ” SDRAM not addressed
 [ 10 , .X. ] -> [ 0 ] ; ” SDRAM not addressed
 [ 11 , .X. ] -> [ 0 ] ; ” SDRAM not addressed
 [ 13 , .X. ] -> [ 0 ] ; ” SDRAM not addressed
 [ 14 , .X. ] -> [ 0 ] ; ” SDRAM not addressed
 [ 15 , .X. ] -> [ 0 ] ; ” SDRAM not addressed
 [ 12 , READ ] -> [ 1 ] ; ” SDRAM read cycle
 [ 12 ,WRITE] -> [ 1 ] ; ” SDRAM write cycle
 [ .X. , MRS ] -> [ 1 ] ; ” SDRAM MRS
end C80codes
```

```
module    C80codes
title'
DWG NAME  LOGIC.SCH
PAL #     U25
COMPANY   TEXAS INSTRUMENTS INCORPORATED
ENGINEER  C80 APPLICATIONS
DATE      05_08_95'
   xx_001    device 'P22V10C'; " PAL22LV10-7C PLCC
   STAT5  Pin 2; "C80 STATUS[5]
   STAT4  Pin 3; "C80 STATUS[4]
   STAT3  Pin 4; "C80 STATUS[3]
   STAT2  Pin 5; "C80 STATUS[2]
   STAT1  Pin 6; "C80 STATUS[1]
   STAT0  Pin 7; "C80 STATUS[0]
   A31    Pin 9; "C80 address line 31
   A30    Pin 10; "C80 address line 30
   A29    Pin 11; "C80 address line 29
   A28    Pin 12; "C80 address line 28
   A17    Pin 13; "C80 address line 17
   A16    Pin 16; "C80 address line 16
   vss    Pin 14; "Ground
   vcc    Pin 28; "Power
   "NC    Pin 27;
   "NC    Pin 26;
   AS2    Pin 25; "C80 AS[2] input (address shift)
   AS1    Pin 24; "C80 AS[1] input (address shift)
   AS0    Pin 23; "C80 AS[0] input (address shift)
   CT2    Pin 21; "C80 CT[2] input (column timing)
   CT1    Pin 20; "C80 CT[1] input (column timing)
   CT0    Pin 19; "C80 CT[0] input (column timing)
   "NC    Pin 18;
   "NC    Pin 17;
"constants and alias names
   ADDR      = [A31..A28] ;
   REFADD    = [A17..A16] ;
   STAT      = [STAT5..STAT0];
   addrv     = 12 ; "SDRAM address valid
   refval    = 3 ; "SDRAM bank refresh pseudo-address
   MRS_cycle = (!STAT5 & !STAT4 & STAT3 & STAT2 & !STAT1 & !STAT0);
```

```
   MRS = 12 ;
   DCAB_cycle   =(!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & STAT0);
   DCAB   = 3 ;
   SDRAM_read   =(!STAT5 & !STAT4 & !STAT3 & !STAT2 & !STAT1 & !STAT0)
      & (A31 & A30 & !A29 & !A28);
   READ   = 0 ;
   SDRAM_write  =(!STAT5 & !STAT4 & !STAT3 & !STAT2 & !STAT1 & STAT0)
      &(A31 & A30 & !A29 & !A28);
   WRITE  = 1 ;
   SDRAM_refresh=(!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & !STAT0)
      &(A17 & A16);
   REFH   = (!STAT5 & !STAT4 & !STAT3 & !STAT2 & STAT1 & !STAT0);
   REFRESH   = 2 ;

equations
AS2  = 0;
     "# (SYSTEM_SPECIFIC_REQUIRING_AS2=1)
AS1  = (A31 & A30 & !A29 & !A28)
     #(MRS_cycle);
     "# (SYSTEM_SPECIFIC_REQUIRING_AS1=1)
AS0  = 0;
     "# (SYSTEM_SPECIFIC_REQUIRING_AS0=1)

CT2  = 0;
     "# (SYSTEM_SPECIFIC_REQUIRING_CT2=1)
CT1  = (A31 & A30 & !A29 & !A28) & ! (REFH)
     #(MRS_cycle)
     #(SDRAM_refresh)
     #(DCAB_cycle);
     "# (SYSTEM_SPECIFIC_REQUIRING_CT1=1)
CT0  = (A31 & A30 & !A29 & !A28) & ! (REFH)
     #(MRS_cycle)
     #(SDRAM_refresh)
     #(DCAB_cycle);
     "# (SYSTEM_SPECIFIC_REQUIRING_CT0=1)
test_vectors"AS2
([ ADDR , STAT ] -> [AS2])
 [ 0 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 1 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
```

```
 [ 2 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 3 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 4 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 5 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 6 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 7 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 8 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 9 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 10 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 11 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 12 , .X. ] -> [ 1 ] ;" SDRAM addressed
 [ 13 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 14 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 15 , .X. ] -> [ 0 ] ;" SDRAM not addressed
test_vectors"AS1
([ ADDR , STAT ] -> [AS1])
 [ 0 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 1 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 2 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 3 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 4 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 5 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 6 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 7 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 8 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 9 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 10 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 11 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 12 , .X. ] -> [ 0 ] ;" SDRAM addressed
 [ 13 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 14 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 15 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ .X. , MRS] -> [ 1 ] ;" SDRAM MRS
test_vectors"AS0
([ ADDR , STAT ] -> [AS0])
 [ 0 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 1 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 2 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 3 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
```

```
 [ 4 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 5 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 6 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 7 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 8 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 9 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 10 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 11 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 12 , .X. ] -> [ 1 ] ;" SDRAM addressed
 [ 13 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 14 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 15 , .X. ] -> [ 0 ] ;" SDRAM not addressed
test_vectors"CT2
([ ADDR , STAT ] -> [CT2])
 [ 0 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 1 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 2 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 3 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 4 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 5 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 6 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 7 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 8 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 9 , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 10 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 11 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 12 , .X. ] -> [ 0 ] ;" SDRAM addressed
 [ 13 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 14 , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 15 , .X. ] -> [ 0 ] ;" SDRAM not addressed
test_vectors"CT1
([ ADDR , STAT, REFADD ] -> [ CT1 ])
 [ 0 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 1 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 2 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 3 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 4 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 5 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 6 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
```

```
 [ 7 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 8 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 9 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 10 , .X. , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 11 , .X. , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 12 , READ, .X. ] -> [ 1 ] ;" SDRAM read
 [ 12 ,WRITE,.X. ]  -> [ 1 ] ;" SDRAM write
 [ 13 , .X. , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 14 , .X. , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 15 , .X. , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ .X. ,REFRESH,refval] -> [ 1 ] ;" SDRAM refresh
 [ .X. ,REFRESH,0]   -> [ 0 ] ;" refresh- not SDRAM
 [ .X. ,REFRESH,1]   -> [ 0 ] ;" refresh- not SDRAM
 [ .X. ,REFRESH,2]   -> [ 0 ] ;" refresh- not SDRAM
 [ .X. , MRS , .X. ] -> [ 1 ] ;" SDRAM MRS
 [ .X. , DCAB, .X. ] -> [ 1 ] ;" SDRAM DCAB
test_vectors"CT0
([ ADDR , STAT, REFADD ] -> [ CT0 ])
 [ 0 , .X. , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 1 , .X. , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 2 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 3 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 4 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 5 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 6 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 7 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 8 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 9 , .X. , .X. ]  -> [ 0 ] ;" SDRAM not addressed
 [ 10 , .X. , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 11 , .X. , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 12 , READ, .X. ] -> [ 1 ] ;" SDRAM read
 [ 12 ,WRITE, .X. ] -> [ 1 ] ;" SDRAM write
 [ 13 , .X. , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 14 , .X. , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ 15 , .X. , .X. ] -> [ 0 ] ;" SDRAM not addressed
 [ .X. ,REFRESH,refval] -> [ 1 ] ;" SDRAM refresh
 [ .X. ,REFRESH,0 ]  -> [ 0 ] ;" refresh- not SDRAM
 [ .X. ,REFRESH, 1]  -> [ 0 ] ;" refresh- not SDRAM
 [ .X. ,REFRESH,2 ]  -> [ 0 ] ;" refresh- not SDRAM
```

```
 [ .X. , MRS , .X. ] -> [ 1 ] ;” SDRAM MRS
 [ .X. , DCAB, .X. ] -> [ 1 ] ;” SDRAM DCAB
end C80codes
```