The TMS320C30 Applications Board Functional Description

APPLICATION REPORT: SPRA403

Tony Coomes
Software Development Systems
Nat Seshan
Digital Signal Processor Products
Semiconductor Group
Texas Instruments

Digital Signal Processing Solutions

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 application 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.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

The TMS320C30 Applications Board Functional Description

Abstract

This book describes the architecture of the TMS320C30 APPB (applications board), part of the TSM320C30 XDS1000 Development System. The APPB was designed to provide a basic platform for software development and a variety of interfaces to the TMS320C30. The four key interfaces used on the APPB are:

- SRAM
- EPROM
- Dual-port SRAM
- DRAM

The book provides basic functional details of the TMS320C30 APPB. Since the SRAM and EPROM interfaces on the APPB are simple, the book's discussion centers on the dual-port SRAM and RAM interfaces and includes the following topics:

- Discussion of the APPB features
- Host/TMS320C30 Interface
- Expansion interface

Supporting figures include:

- Host interface block diagramTMS320C30 bank addressing
- Timing diagrams
- TMS320C30 applications
- TMS320C30 Applications board

Tables included cover:

- Host I/O Memory locations for control registers
- APPB general-purpose control register bits and bit definitions

The book concludes with a series of appendices that contain source code for routines written in C. The contents of these appendices include:

- TMS320C30 applications board routines for both the PC side and the TMS320C30 side
- Memory map and description (TMS320C30 view)
- TMS320C30 software development board
- Various modules
- TMS320C30 software development schematics
- TMS320C30 SWDS DRAM module schematics

Product Support

World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. New users must register with TI&ME before they can access the data sheet archive. TI&ME allows users to build custom information pages and receive new product updates automatically via email.

Email

For technical issues or clarification on switching products, please send a detailed email to dsph@ti.com. Questions receive prompt attention and are usually answered within one business day.

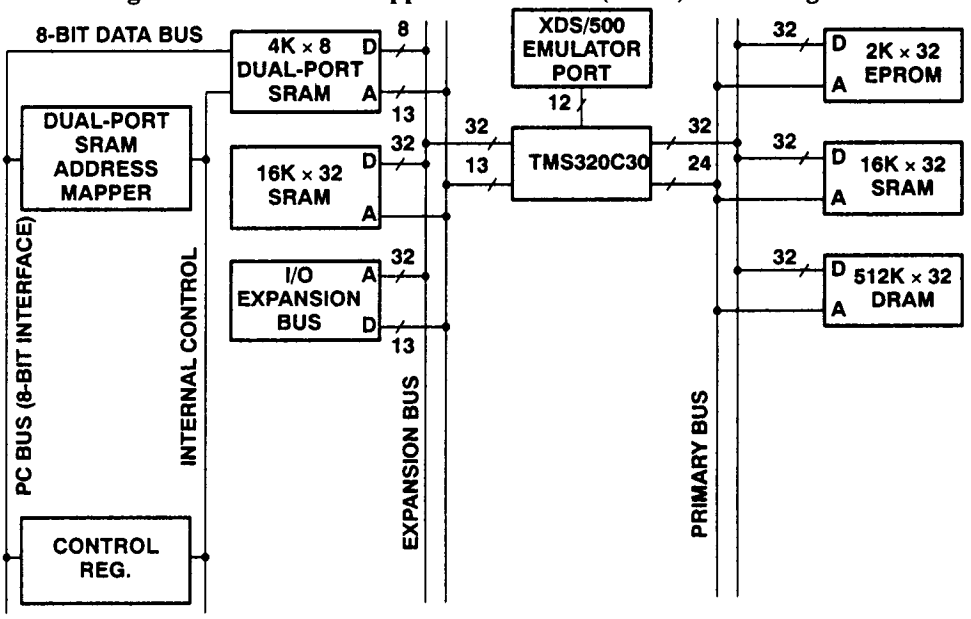
Introduction

This report describes the architecture of the TMS320C30 Applications Board (APPB), which is part of the TMS320C30 XDS1000 Development System. The XDS1000 is an in-circuit emulation tool for TMS320C30 hardware/software system development. The APPB was designed with two goals: to provide a basic platform for software development and to provide a variety of interfaces to the TMS32C30. There are four key interfaces used on the APPB:

1) SRAM
2) EPROM
3) Dual-port SRAM
4) DRAM

The SRAM and EPROM interfaces on the APPB are quite simple; thus, this report focuses on the dual-port SRAM and the DRAM interfaces. Figure 1 shows a basic block diagram of the APPB.

Figure 1. TMS320C30 Applications Board (APPB) Block Diagram



The APPB features include the following:

- TMS320C30/host communications via a designated, relocatable 4K-byte dual-bus SRAM memory block.
- 16K-words (64K-bytes) zero wait-state SRAM on the TMS320C30 primary bus (STRB).
- 2K-words of one wait-state EPROM for interrupt and reset vectors on the TMS320C30 primary bus.
- 16K-words (64K-bytes) zero wait-state SRAM on the TMS320C30 expansion bus (MSTRB). The SRAM can be selected in either one of two 8K-word banks.

- I/O expansion bus.
- 512K-words of DRAM on the TMS320C30 primary bus.
- Emulation port.
- IBM PC, PC/XT, PC/AT support.

The remainder of this document describes each interface in more detail.

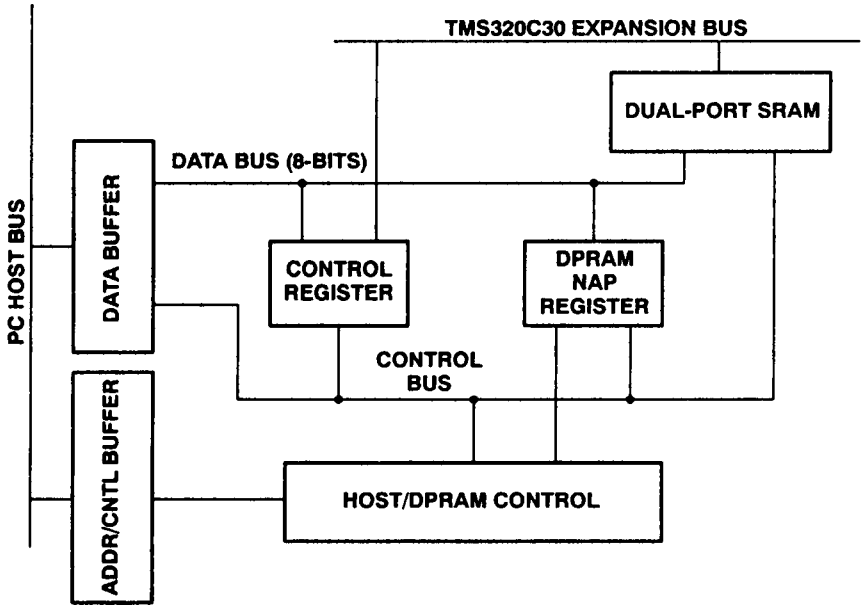
Host/TMS320C30 Interface

The host/TMS320C30 interface is composed of two basic blocks, the dual-port SRAM and the control logic. The control logic consists of address decoding, a read/write control register, and a write-only mapping register. The control registers are mapped into the host I/O space as shown in Table 1. Figure 2 is a block diagram of the host interface.

Table 1. Host I/O Memory Locations for Control Registers

Host I/O Memory Locations	Contents
0330 – 0337	Semaphores (LSB is the only valid bit)
0338	Dual-port SRAM mapping register Q
0339	Control register R

Figure 2. Host Interface Block Diagram



One of the major problems in developing an application for a PC is finding a block of memory that does not conflict with other memory-mapped cards. To ease this problem, the dual port SRAM interface has been designed to be relocatable on 4K-byte boundries throughout the lower 1M-bytes of host memory space. A software example of how to map the dual-port SRAM into this space is given later in this report.

Writing a value to a hardware mapping register on the APPB relocates the dual-port SRAM. When a host memory access is generated, the value in the mapping register is compared to host address bits A12–A19. If they match, a dual-port SRAM access is allowed. To ensure PC and PC/XT compatibility, the dual-port SRAM can be located only in the lower 1M-bytes of host memory.

The APPB contains one general-purpose control register. This register is broken into two four-bit nibbles. The lower nibble can be read from and written to by the host and read by the TMS320C30. The upper nibble can be read from and written to by the TMS320C30 and read by the host. The lower nibble of the control register is cleared by any reset to or from the host PC. The upper nibble of the control register is cleared by any reset to the TMS320C30. The names of the APPB control register bits and host/TMS320C30 access capabilities are given in Table 2. Table 3 gives the control register bit definitions.

Table 2. APPB General-Purpose Control Register Bits

Bit	Name	Host Access	C30 Access
0	CINT	Write/Read	Read only
1	XINTCLR	Write/Read	Read only
2	DPSEL	Write/Read	Read only
3	SWRESET	Write/Read	Read only
4	XINT	Read only	Write/Read
5	CINTCLR	Read only	Write/Read
6	MBANK	Read only	Write/Read
7	MSWAP	Read only	Write/Read

Table 3. APPB General-Purpose Control Register Bit Definitions

Bit	Name	Function
0	CINT	Clears and disables interrupts from the TMS320C30 to the host (XINT). XINTCLR must be set to 1 before the TMS320C30 can generate an interrupt to the host. The host clears and reenables XINT by writing 0, then 1 to XINTCLR. On reset, XINTCLR is read as a 0.
1	XINTCLR	Interrupt (INT0) to the TMS320C30. The host may interrupt the TMS320C30 by setting this bit to 1. The TMS320C30 clears and re-enables the CINT by writing 0, then 1 to CINTCLR. The host cannot generate an interrupt to the TMS320C30 while CINTCLR = 0. On reset, CINT is read as a 0.
2	DPSEL	Dual-port SRAM select. When this bit is set to 1, the dual-port SRAM is memory-mapped in the 4K-byte space of the host PC specified by the 8-bit value in register Q. When DPSEL = 0, the dual-port SRAM will not be mapped in the host PC's address space. On reset, DPSEL is read as a 0.
3	SWRESET	TMS320C30 SWDS soft reset. SWRESET = 0 resets the TMS320C30 SWDS. SWRESET must be set to 1 to take the SWDS out of the reset state. On reset (power on), SWRESET is read as a 0.
4	XINT	Interrupt to the host PC. The TMS320C30 may interrupt the host by setting this bit to 1. The host clears and re-enables XINT by writing 0, then 1 to XINTCLR. The TMS320C30 cannot generate an interrupt to the host while XINTCLR = 0. On reset, XINT is read as a 0.
5	CINTCLR	Clears and disables interrupts from the the host to the TMS320C30 (CINT). CINTCLR must be set to 1 before the host can generate an interrupt to the TMS320C30. The TMS320C30 clears and re-enables CINT by writing 0, then 1 to CINTCLR. On reset, CINTCLR is read as a 0.
6	MBANK	Memory bank select. The 16K-word bank of memory on the TMS320C30 parallel I/O Bus (SRAM space 1) is mapped as two overlapping banks of 8K-words each. MBANK = 0 selects the lower 8K-words, MBANK = 1 selects the upper 8K-words. On reset, MBANK is read as a 0.
7	MSWAP	Memory Swap. The MSWAP bit is used to swap the address map for EPROM and SRAM space 0. MSWAP = 0 maps the EPROM at 000000h–003FFFh and SRAM space 0 at F00000h–F03FFFh. MSWAP = 1 maps the EPROM at F00000h–F03FFFh and SRAM space 0 at 00000h–003FFFh. On reset, MSWAP is read as a 0.

The last portion of the control section contains the dual-port SRAM semaphore registers. Semaphore registers are used to coordinate communications between the host and the TMS320C30. Note that these semaphores do not provide hardware protection of the memory array. Instead, they provide a basic means (via software control) to ensure that data can be accessed from both sides of the dual-port SRAM without being corrupted. A software example that uses the semaphores is presented later in this report.

SRAM and EPROM Interfaces

There are two SRAM interfaces on the APPB: one on the primary bus and one on the expansion bus. Both are implemented with eight 16K-bit × 4, 25-ns SRAMs that provide zero wait-state TMS320C30 operation at 32 MHz. The interfaces are quite simple and consist of a set of address buffers, termination resisters, and a PAL for address decode on the primary bus. Note that the TMS320C30 address lines are routed to various components scattered around the board and then to the primary bus expansion. To prevent line reflections on the SRAM addresses, buffers have been used to isolate the SRAM.

There are two special features on the APPB that apply to the SRAM:

1) You can swap the memory address ranges of the EPROM and the SRAM on the primary bus by setting or clearing the MSWAP bit previously described in Table 3.
2) There are two 8K-word pages of memory on the expansion bus.

By swapping the EPROM and SRAM, you can load in your own interrupt and reset vectors. Otherwise, you would have to remove the EPROMs and reprogram them with your own defined interrupt/reset vectors. The following code segment sets/clears the MSWAP bit.

```
#define EPROM            0              /* select EPROM */
#define SRAM             1              /* select SRAM */

sel_mswap(mem_type)
int mem_type;
{
   char *cntlreg = (char *)0x00805FF7; /* pointer to control reg */

  if (mem_type)     *cntlreg |= 0x80;   /* set MSWAP to 1 select SRAM */
     else           *cntlreg &= 0x7F;   /* set MSWAP to 0 select EPROM */
}
```

There are 16K-words of SRAM on the expansion bus; however, the TMS320C30 can directly access only 8K-words. Instead of wasting the unaddressable 8K-words, you can use a bank addressing bit (MBANK) in the APPB control register to select between the lower and upper 8K-word segments.

The following code segment selects the current bank of memory.

```
#define BANK0            0              /* select lower 8K */
#define BANK1            1              /* select upper8K */

sel_mbank(bank)
int bank;
{
   char *cntlreg = (char *)0x00805FF7; /* pointer to control reg */

    if (bank)       *cntlreg |= 0x40;    /* select bank 1 */
      else          *cntlreg &= 0xBF;    /* select bank 0 */
}
```

The APPB supports 2K-words of one wait-state EPROM on the primary bus for a boot loader and operating system support. As stated earlier, this EPROM is remappable.

DRAM Interface

The APPB provides a DRAM expansion module that is connected to the TMS320C30 primary bus. Historically, DRAM interfaces to DSP devices have not been popular because of interface

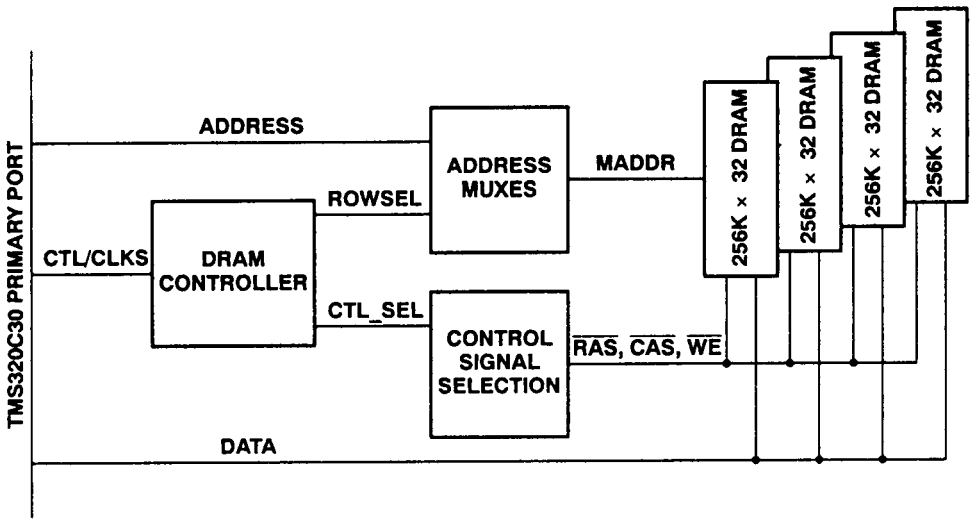
difficulty and limited processor address space. The TMS320C30 supplies solutions to both of those issues with its memory interface and 16M-words address space. Two areas of the TMS320C30 memory interface are most useful for DRAM design:

- Use of bank mode
- The ability to do continous reads while in a bank without deasserting the $\overline{\text{STRB}}$ signal

When you use these two features, it is quite simple to design a medium-speed interface to page-mode DRAMs.

The TMS320C30 DRAM module consists of four banks of memory, each bank 256K × 32 bits, that provide 1M-word (4M-bytes) of medium speed storage for the TMS320C30 (see Figure 3). The bank-switch function on the TMS320C30 provides fast page-mode access on back-to-back read cycles within a DRAM page. All address and control lines to the memory array are buffered and series-terminated for good signal quality. The memory array uses CAS-before-RAS refresh to reduce component count. There is no onboard refresh timer; instead, SDACK0 from the host PC provides a refresh request every 12–16 μs. The DRAM access/cycle times are summarized in Table 4.

Figure 3. TMS320C30 Bank Addressing



In Table 4, these definitions are assumed:

Access Time – Number of clocks from $\overline{\text{STRB}}$ active to data clocked into the TMS320C30.
Cycle time – Number of clocks between two back-to-back cycles (includes DRAM $\overline{\text{RAS}}$ precharge on non-page-mode cycles).

Table 4. TMS320C30 DRAM Access and Cycle Times

Mode	Access Time (clks)	Cycle Time (clks)
Read	3	5
Read (page mode)	3/2†	2
Write	3	4

† First page-mode access takes 3 clocks; the following accesses take 2 clocks each.

The four banks of DRAM are mapped into the TMS320C30 memory space at the address locations shown in Table 5.

Table 5. DRAM Bank Memory Locations in the TMS320C30 Memory Space

DRAM Memory Bank No.	TMS320C30 Memory Location
0 ($\overline{RAS0}$,$\overline{CAS0}$)	400000H–43FFFFH
1 ($\overline{RAS1}$,$\overline{CAS1}$)	440000H–47FFFFH
2 ($\overline{RAS2}$,$\overline{CAS2}$)	480000H–4BFFFFH
3 ($\overline{RAS3}$,$\overline{CAS3}$)	4C0000H–4FFFFFH

Memory decode for the DRAM module is performed in two steps:

1) The APPB main card provides a memory select to decode the board range of 400000H–4FFFFFh.
2) Bank decode is then provided on the DRAM module through TMS320C30 address bits A18 and A19.

The DRAM controller consists of a pair of registered PALs, several SSI gates, and a delay line (used to time DRAM row/column address multiplexing). DRAM timing is generated from PAL UE5 (see schematics in Appendix C), while address decoding and special refresh control are provided by PAL UD5. Both PALs are clocked off of a delayed H1 clock. The DRAM controller looks for every opportunity to generate page-mode cycles to the DRAM. The TMS320C30 leaves $\overline{STRB}$ low for back-to-back reads; the DRAM controller looks for this condition and cycles CAS while holding RAS low (i.e., DRAM page-mode access). When $\overline{STRB}$ goes high, the DRAM controller will take both RAS and CAS high to prepare for a new access. For proper operation, the TMS320C30 primary bus control register (refer to the Primary Bus Control Register subsection in the *Third-Generation TMS320 User's Guide*) must be set to operate off of the external ready signal and use a maximum bank size of 512 words (refer to the the Programmable Bank Switching subsection of the *Third-Generation TMS320 User's Guide*).

Figures 4 through 6 show the timing for the various DRAM cycles.

Figure 4. Page-Mode Read-Cycle Timing Diagram

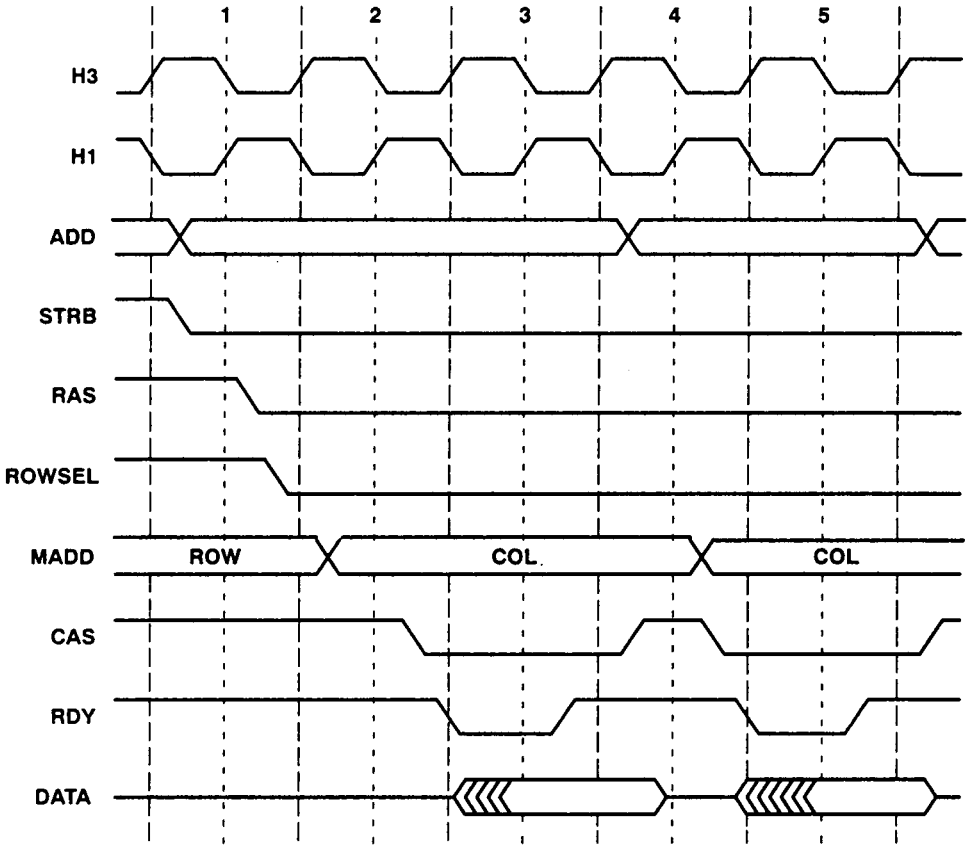


Figure 5. Single Write-Cycle Timing Diagram

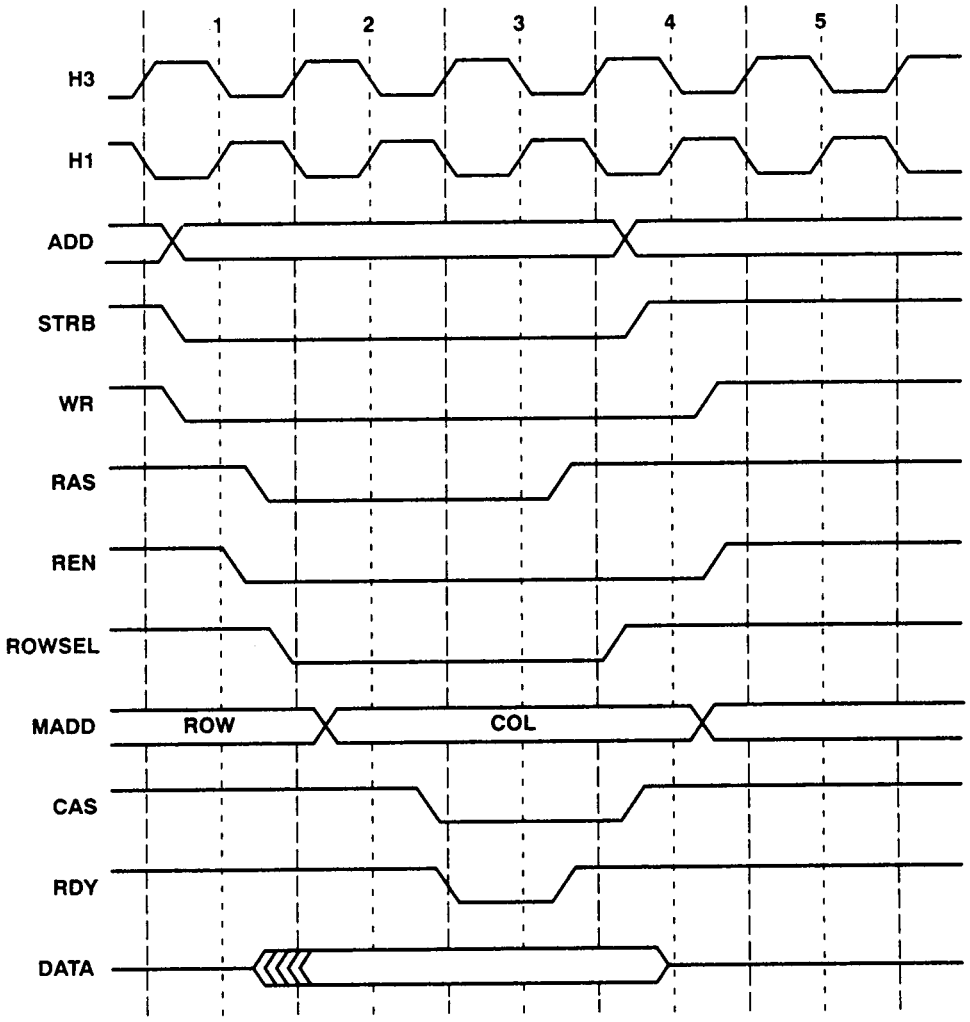
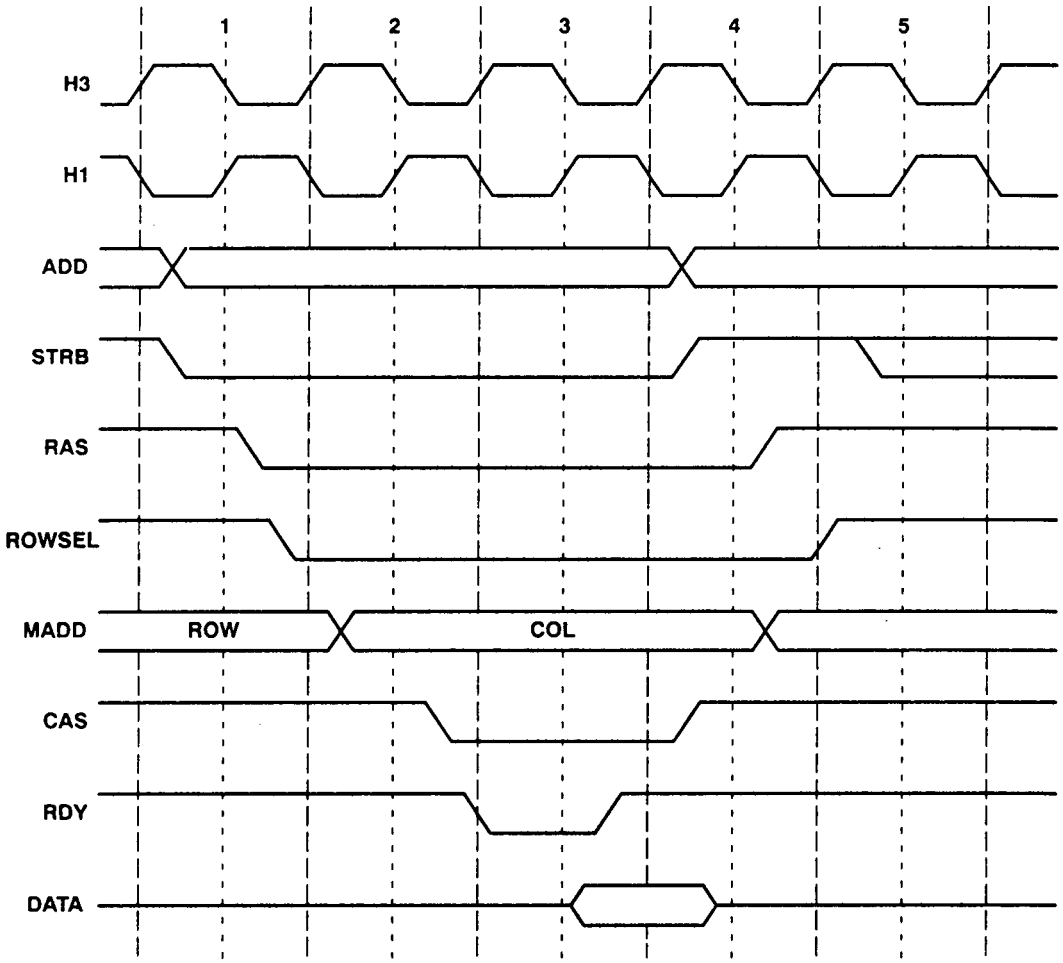


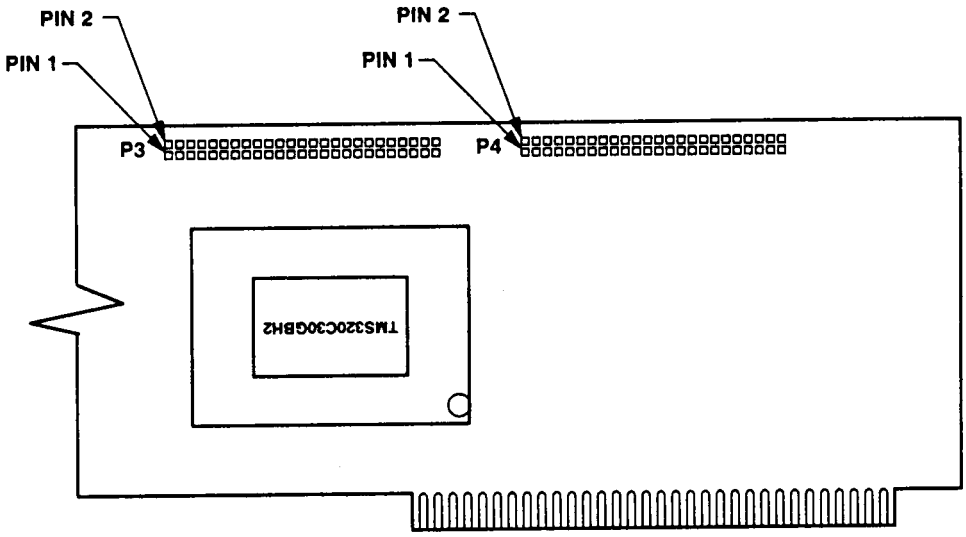
Figure 6. Single Read-Cycle Timing Diagram



Expansion Interface

The APPB's two expansion connectors contain the signals from the TMS320C30 expansion port, serial ports, flag pins, etc. Each 50-pin connector (P3 and P4 of Figure 7) is composed of a dual row of 25 pins located on 0.1-inch centers. These expansion connectors provide easy connection to other hardware via standard 50-wire flat ribbon cable. Figure 6 shows the orientation of the connectors. See schematic sheet 7 of Appendix C for pinout details.

Figure 7. TMS320C30 Applications Board



Dual-Port SRAM Interface

All communications between the TMS320C30 and the host occur through the dual-port SRAM, which is 4K-bytes deep, with 8 dedicated semaphore registers. On the host side, the dual-port memory array is memory-mapped, while the semaphores are I/O-mapped. On the TMS320C30 side, the dual-port SRAM is located on the expansion bus with the memory array mapped from 0x00804000–0x00804FFF and the semaphores mapped from 0x00805FF8–0x00805FFF. The host can directly access the dual-port SRAM without having to compensate for byte-wide access limitations. However, as the TMS320C30 can do only 32-bit accesses, the upper 24 bits of a data word are undefined. The TMS320C30 must therefore format data written to and read from the dual-port SRAM. A software example is given later in this report.

While dual-port SRAMs provide an excellent means for multiprocessor communications, a certain amount of software overhead is required to coordinate data flow. As might be expected, there are numerous methods for coordinating data flow. This application report presents a set of primitives that have been developed to form a basic communications protocol. The primitives are written entirely in C and have been tested on the XDS1000 with the simple test routine provided. Remember that there are numerous ways to do a communications protocol. The method shown in this report is not the best for all applications; it is simply a method that makes good use of the capability of the dual-port SRAM.

The following are basic ideas of the communications protocol developed for this applications report.

1) The dual-port memory is broken into eight equal segments. The first segment is used only for control structures and command passing. The remaining seven segments are used entirely for data passing. Segment size is set to 512 bytes. The number and size of segments can be changed at compile time if desired.

2) Each of the seven data segments is totally independent from any other data segment. However, only one processor can own a particular segment at any given time. The TMS320C30 and host can simultanously access the dual-port SRAM as long as both are not trying to access the same segment.
3) The host is the master; the TMS320C30 is the slave. The TMS320C20 polls the dual-port control segment to determine if the host has deposited a command. If a command is present, the TMS320C30 executes the command and then returns to polling.
4) Only the first semaphore register is used in the dual-port. Each processor uses this semaphore to gain access to the control segment. Access to the seven data memory segments are coordinated via the control structures, not the semaphores.
5) There are seven control structures in the control segment, one for each data segment. Each control structure consists of 22 bytes and are defined as follows:

Byte	Name	Definition
0	pflag	Buffer present (i.e., being used)
1	command	Command to execute
2	buf_stat	Status of the data buffer
3	nc	Reserved
4–7	count	Number of 32–bit words to transfer
8–11	addr	TMS320C30 to read/write data
12–21	message	Ten bytes reserved for message passing

Appendix A contains routines for the communication primitives used by the host and the TMS320C30. Appendix A1 contains routines for the PC side, Appendix A2 routines for the TMS320C30 side. Note that the routines on both sides have the same names and perform essentially the same function. Appendix A3 contains a memory map and description (TMS320C30 view). After the code has been compiled, use the following sequence to execute the test program:

1) Reset the XDS/1000:

```
 xreset   [RETURN]
c30reset  [RETURN]
```

2) Get into the emulator and load the TMS320C30 dual-port code.

```
emu30          [RETURN]       ; load emulator
xr                            ; reset the c30
lo             'file name'    ; load the object file
xd                            ; execute disconnect
[esc]                         ; escape to main menu
q 'yes'                       ; quit emulator
```

At this point, your dual bus code should be executing and waiting for a host input.

3) Execute host dual-port code.

```
'file name'
```

The host code will then print the numbers 0 through 25 to the screen.

Conclusion

This report has provided basic functional details of the TMS320C30 APPB. Because of their complexity, the DRAM and dual-port SRAM interfaces have been discussed. The features of the TMS320C30 allow it to encompass a wide range of interfaces. The TMS320C30 bank-switch mode and continuous strobe signal on back-to-back read cycles overcome traditional DSP/DRAM problems of interface difficulty and limited processor address space. A set of communications primitives routines to use with dual-port SRAM have been provided in Appendix A. These routines are written in C for ease of understanding and modification to meet individual needs.

Appendix A

TMS320C30 Application Board Routines, Memory Map and Description

Appendix A1. TMS320C30 Applications Board Routines–PC Side

```
/*****************************************************************/
/*                                                               */
/*  APPENDIX A1                                                  */
/*                                                               */
/*  TMS320C30 APPLICATION BOARD ROUTINES - PC SIDE               */
/*                                                               */
/*  Texas Instruments Inc.                                       */
/*  10/25/89                                                     */
/*                                                               */
/*  Functions:                                                   */
/*                                                               */
/*      int APPB_reset()        Reset APPB                       */
/*      int APPB_dpinit()       Intialize APPB.                  */
/*      int APPB_getsem()       Get access to semaphore bit N    */
/*      int APPB_relsem()       Release access to semaphore bit N*/
/*      int APPB_getctlblk()    Get a control block in DPRAM     */
/*      int APPB_relctlblk()    Release control block in DPRAM   */
/*      int APPB_getmemblk()    Get a block of memory from DPRAM */
/*      int APPB_putmemblk()    Put a block of memory to DPRAM   */
/*                                                               */
/*  All code was compiled with Microsoft C compiler version 5.1 using the */
/*  large model. If small model is used, then pointers used to access the */
/*  dual port SRAM would have to be declared and used as 'far' pointers */
/*  (i.e. 32-bit pointer). Under the large model, all pointers are */
/*  defaulted to 32 bits.                                        */
/*                                                               */
/*****************************************************************/

#include <stdio.h>

/*****************************************************************/
/*                                                               */
/*  Constant definitions for the TMS320C30 Applications Board.   */
/*                                                               */
/*****************************************************************/

#define     outport     outp
#define     inport      inp
#define     SEM_BASE    0x0330
#define     MAP_REG     0x0338
#define     CTL_REG     0x0339

#define     CINT        0x01
#define     XINTCLR_    0x02
#define     DPSEL       0x04
#define     SMRESET_    0x08
#define     XINT        0x10
#define     CINTCLR_    0x20
#define     MSWAK       0x40
#define     MSHAP       0x80

#define     DPRAM_CTL   0xC9000000
#define     DPRAM_SEG   0xC9
#define     DPRAM_MEMBASE 0xC9000200

#define     DPRAM_SIZE      0x1000
#define     DPRAM_BLKS      7
#define     DPRAM_BLK_SIZE  512
#define     NUM_SEMS        8
#define     MAX_SEM_TIME    10000

#define     BUF_EMPTY       0
#define     BUF_FULL        1

#define     MCP             0x00
#define     HOST_MEM_WR     0x80
#define     HOST_MEM_RD     0x81

typedef unsigned char  UCHAR;
typedef unsigned short UINT;
typedef unsigned long  ULONG;

typedef struct
{
        UCHAR   pflag;
        UCHAR   command;
        UCHAR   buf_stat;
        UCHAR   nc;
        ULONG   count;
        ULONG   addr;
        UCHAR   message[10];
}DPCNTL;
```

```
/*************************************************************************/
/*                                                                       */
/*   Test program.                                                       */
/*                                                                       */
/*   Sequence:                                                           */
/*                                                                       */
/*   1)  Write a block of memory to the dual port.                       */
/*   2)  Read back the block of data from the dual port.                 */
/*                                                                       */
/*************************************************************************/

main()
{
   UINT  semnum[DPRAM_BLKS];
   int i;
   ULONG   memarray[25],mem2array[25];

   APPB_dpint();


   for(i=0;i<25;i++) {memarray[i] = (ULONG)i; mem2array[i] = 0UL;}

   if(APPB_putmemblk(25UL,memarray,0x00809900))
       printf("failed memory write\n");

   if(APPB_getmemblk(25UL,0x00809900,mem2array))
       printf("failed memory read\n");

   for(i=0;i<25;i++)  printf("value read %d\n",mem2array[i]);

   exit(0);
}
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_reset(),PC side                                                */
/*                                                                       */
/*   Reset APPB.                                                         */
/*                                                                       */
/*   Sequence:                                                           */
/*                                                                       */
/*   1)  Clear control register.                                         */
/*   2)  Set SWRESET_ to 1.                                              */
/*                                                                       */
/*************************************************************************/
int APPB_reset()
    {
       outport(CTL_REG,0);
       outport(CTL_REG,SWRESET_);
       return(0);
    }

/*************************************************************************/
/*                                                                       */
/*   APPB_dpint(), PC side                                               */
/*                                                                       */
/*   Sequence:                                                           */
/*                                                                       */
/*   1)  Set DPRAM semaphores to 1 (free).                               */
/*   2)  Set DPRAM mapping register.                                     */
/*   3)  Set DPRAM global enable bit to 1.                               */
/*                                                                       */
/*************************************************************************/
int APPB_dpint()
    {
       int i;
       UINT  semaddr = SEM_BASE;
       UCHAR *dpram  = (UCHAR *)DPRAM_CTL;

       for(i=0;i<8;i++)    outport(semaddr++,1);
       outport(MAP_REG,DPRAM_SEG);
       outport(CTL_REG,DPSEL | SWRESET_);
       return(0);
    }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_getsem(), PC side                                              */
/*                                                                       */
/*   Attempts to gain access of semaphore 'semnum'.                      */
/*   Return a 0 if successful, a -1 if failed.                           */
/*                                                                       */
/*   Sequence                                                            */
/*                                                                       */
/*   1)  Write 0 to semaphore.                                           */
/*   2)  Decrement timoute, check for timeout = 0, or semaphore = 0.     */
/*   3)  Return pass/fail.                                               */
/*                                                                       */
/*************************************************************************/

int APPB_getsem(semnum)
      UINT  semnum;
   {
      UINT  semaddr = SEM_BASE + semnum;
      UINT  timeout = MAX_SEM_TIME;

      outport(semaddr,0);
      while( --timeout && (inport(semaddr) & 1));

      if(timeout) return(0);
      else return(-1);
   }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_relsem(), PC side                                              */
/*                                                                       */
/*   Release semaphore at 'semnum'.                                      */
/*   Return a 0 if successful, a -1 if failed.                           */
/*                                                                       */
/*   Sequence                                                            */
/*                                                                       */
/*   1)  Write 1 to semaphore.                                           */
/*   2)  Decrement timeout, check for timeout = 0, or semaphore = 1.     */
/*   3)  Return pass/fail.                                               */
/*                                                                       */
/*************************************************************************/

int APPB_relsem(semnum)
      UINT  semnum;
   {
      UINT  semaddr = SEM_BASE + semnum;
      UINT  timeout = MAX_SEM_TIME;

      outport(semaddr,1);
      while( --timeout && !(inport(semaddr) & 1));

      if(timeout) return(0);
      else return(-1);
   }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_getctlblk(), PC side                                           */
/*                                                                       */
/*   Find unused block of memory in the dual port.                       */
/*   Return a 0 if successful, a -1 if failed.                           */
/*                                                                       */
/*   Sequence                                                            */
/*                                                                       */
/*   1)  Search control structures for free block of memory.             */
/*   2)  If block free, set semnum to block index, return 0.             */
/*   3)  Else, return -1 (failed to find block).                         */
/*                                                                       */
/*************************************************************************/

int APPB_getctlblk(semnum)
      UINT  *semnum;
   {
      int i;
      DPCNTL *dpctl = (DPCNTL *)DPRAM_CTL;

      if(APPB_getsem(0))  return(-1);

      for(i=0;i<DPRAM_BLKS;i++)
          if(!dpctl[i].pflag)
          {
            dpctl[i].pflag   = 1;
            dpctl[i].command = NOP;
            dpctl[i].buf_stat = BUF_EMPTY;
            *semnum = i;
            if(APPB_relsem(0)) return(-1);
            else return(0);
          }

      APPB_relsem(0); return(-1);
   }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_relctlblk(), PC side                                           */
/*                                                                       */
/*   Release block of memory in the dual port.                           */
/*   Return a 0 if successful, a -1 if failed.                           */
/*                                                                       */
/*   Sequence                                                            */
/*                                                                       */
/*   1)  Null out the control structure.                                 */
/*   2)  Return.                                                         */
/*                                                                       */
/*************************************************************************/
int APPB_relctlblk(semnum)
      UINT  semnum;
   {
      int i;
      DPCNTL *dpctl = (DPCNTL *)DPRAM_CTL;

      if(APPB_getsem(0))  return(-1);
      dpctl[semnum].pflag    = 0;
      dpctl[semnum].command  = NOP;
      dpctl[semnum].buf_stat = BUF_EMPTY;
      if(APPB_relsem(0)) return(-1);
      else return(0);
   }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_putmemblk(), PC side                                           */
/*                                                                       */
/*   Write block of memory to the dual port.                             */
/*   Return a 0 if successful, a -1 if failed.                           */
/*                                                                       */
/*   Sequence                                                            */
/*                                                                       */
/*   1)  Find free block of dual port to write memory.                   */
/*   2)  Write the memory.                                               */
/*   3)  Write memory parameters to control block.                       */
/*                                                                       */
/*************************************************************************/

int APPB_putmemblk(cnt,src,dst)
      ULONG cnt;
      ULONG *src;
      ULONG dst;
   {

      DPCNTL *dpctl = (DPCNTL *)DPRAM_CTL;
      ULONG  *dpram;
      UINT    dpblk;
      int     i;

      if(APPB_getctlblk(&dpblk))  return(-1);

      dpram = (ULONG*)(DPRAM_MEMBASE + (dpblk * DPRAM_BLK_SIZE));

      for(i=0;i<cnt;i++)
      *dpram++ = *src++;

      if(APPB_getsem(0))  return(-1);

      dpctl[dpblk].command  = HOST_MEM_WR;
      dpctl[dpblk].buf_stat = BUF_FULL;
      dpctl[dpblk].count    = cnt;
      dpctl[dpblk].addr     = dst;

      if(APPB_relsem(0))  return(-1);
   }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_getmemblk(), PC side                                           */
/*                                                                       */
/*   Read block of memory to the dual port.                              */
/*   Return a 0 if successful, a -1 if failed.                           */
/*                                                                       */
/*   Sequence                                                            */
/*                                                                       */
/*   1)  Find free block of dual port for memory.                        */
/*   2)  Write memory parameters to control block.                       */
/*   3)  Wait for TMS320C30 to put requested memory into the dual port.  */
/*   4)  Read data from the dual port.                                   */
/*   5)  Release block of dual port memory.                              */
/*                                                                       */
/*************************************************************************/
int APPB_getmemblk(cnt,src,dst)
      ULONG cnt;
      ULONG src;
      ULONG *dst;
   {

      DPCNTL *dpctl = (DPCNTL *)DPRAM_CTL;
      ULONG  *dpram;
      UINT    dpblk;
      int     i;
      UINT  timeout = MAX_SEM_TIME;

      if(APPB_getctlblk(&dpblk))  return(-1);

      dpram = (ULONG*)(DPRAM_MEMBASE + (dpblk * DPRAM_BLK_SIZE));

      if(APPB_getsem(0)) return(-1);

      dpctl[dpblk].command  = HOST_MEM_RD;
      dpctl[dpblk].buf_stat = BUF_EMPTY;
      dpctl[dpblk].count    = cnt;
      dpctl[dpblk].addr     = src;

      while( --timeout )
      {
          if(!APPB_getsem(0) && (dpctl[dpblk].buf_stat == BUF_FULL)) break;
          if(APPB_relsem(0))  return(-1);
      }

      if(APPB_relsem(0) || !timeout) return(-1);

      for(i=0;i<cnt;i++)
          *dst++ = *dpram++;

      if(APPB_relctlblk(dpblk)) return(-1);

   }
```

Appendix A2. TMS320C30 Applications Board Routines–TMS320C30 Side

```
/*****************************************************************/
/*                                                               */
/*  APPENDIX A2                                                  */
/*                                                               */
/*  TMS320C30 APPLICATION BOARD ROUTINES - TMS320C30 SIDE        */
/*                                                               */
/*  Texas Instruments Inc.                                       */
/*  10/20/89                                                     */
/*                                                               */
/*  Functions:                                                   */
/*                                                               */
/*     int APPB_dpinit()      Intialize APPB.                   */
/*     int APPB_getsem()      Get access to semaphore bit N     */
/*     int APPB_relsem()      Release access to semaphore bit N */
/*     int APPB_getctlblk()   Get a control block in DPRAM      */
/*     int APPB_relctlblk()   Release control block in DPRAM    */
/*     int APPB_getmemblk()   Get a block of memory from DPRAM  */
/*     int APPB_putmemblk()   Put a block or memory to DPRAM    */
/*     int APPB_getlong()     Read a long int from the DPRAM    */
/*     int APPB_getcommand()  Read a command and parameters from DPRAM */
/*                                                               */
/*  All code was compiled with TMS320C30 C compiler version 2.1, using the */
/*  small model.                                                 */
/*                                                               */
/*****************************************************************/

/*****************************************************************/
/*                                                               */
/*  Constant definitions for the TMS320C30 Applications Board.   */
/*                                                               */
/*****************************************************************/
#define     SEM_BASE        0x00805FF8
#define     CTL_REG         0x00805FF7

#define     CINT            0x01
#define     XINTCLR_        0x02
#define     DFSEL           0x04
#define     SHRESET_        0x08
#define     XINT            0x10
#define     CINTCLR_        0x20
#define     MBANK           0x40
#define     MSHARP          0x80

#define     DPRAM_CTL       0x00804000
#define     DPRAM_MEMBASE   0x00804200
#define     DPRAM_SIZE      0x1000
#define     DPRAM_BLKS      7
#define     DPRAM_BLK_SIZE  512
#define     NUM_SEMS        8
#define     MAX_SEM_TIME    10000

#define     BUF_EMPTY       0
#define     BUF_FULL        1

#define     NOP             0x00
#define     HOST_MEM_WR     0x80
#define     HOST_MEM_RD     0x81

typedef     unsigned char   UCHAR;
typedef     unsigned short  UINT;
typedef     unsigned long   ULONG;

typedef     struct
            {
                UCHAR       pflag;
                UCHAR       command;
                UCHAR       buf_stat;
                UCHAR       nc;
                UCHAR       count[4];
                UCHAR       addr[4];
                UCHAR       message[10];
            }DPCNTL;

typedef     struct
            {
                UCHAR       mblk;
                UCHAR       mcmd;
                ULONG       mcnt;
                ULONG       maddr;
            }MPARMS;
```

```
/*************************************************************************/
/*                                                                       */
/*   Test program, TMS320C30 side.                                       */
/*                                                                       */
/*   Sequence:                                                           */
/*                                                                       */
/*   1)  Initialize the the dual port SRAM.                              */
/*   2)  Poll dual port for commands.                                    */
/*   3)  Execute commands as encountered.                                */
/*                                                                       */
/*************************************************************************/
main()
{
      int i;
      MPARMS  mparms;

      APPB_dpint();

      for(;;)
      {

          APPB_getcommand(&mparms);
          switch(mparms.mcmd)
          {
              case NOP:   break;

              case HOST_MEM_WR:
                  APPB_getmemblk(mparms.mcnt,mparms.maddr,mparms.mblk);
              break;

              case HOST_MEM_RD:
                  APPB_putmemblk(mparms.mcnt,mparms.maddr,mparms.mblk);
              break;

              default:    break;

          }
     }
 }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_dpint(), TMS320C30 side.                                       */
/*                                                                       */
/*   Sequence:                                                           */
/*                                                                       */
/*   1)  Set DRAM semaphores to 1 (free).                                */
/*   2)  Clear entire dual port RAM.                                     */
/*                                                                       */
/*************************************************************************/
int APPB_dpint()
    {
       int i;
       UCHAR *semaddr = (UCHAR *)SEM_BASE;
       UCHAR *dpram   = (UCHAR *)DPRAM_CTL;

       for(i=0;i<8;i++)     *semaddr++ = 1;
       for(i=0;i<DPRAM_SIZE;i++) *dpram++ = 0;

       return(0);
    }
```

```
/************************************************************************/
/*                                                                      */
/*   APPB_getsem(), TMS320C30 side                                      */
/*                                                                      */
/*   Attempts to gain access of semaphore 'semnum'                      */
/*                                                                      */
/*   Sequence                                                           */
/*                                                                      */
/*   1)  Write 0 to semaphore.                                          */
/*   2)  Wait till read a 0.                                            */
/************************************************************************/

int APPB_getsem(semnum)
      UINT  semnum;
   {
      UCHAR *semaddr = (UCHAR *)(SEM_BASE + semnum);

      *semaddr = 0; while(*semaddr & 1UL); return(0);
   }
```

```
/************************************************************************/
/*                                                                      */
/*   APPB_relsem(), TMS320C30 side                                      */
/*                                                                      */
/*   Release semaphore at 'semnum'                                      */
/*                                                                      */
/*   Sequence                                                           */
/*                                                                      */
/*   1)  Write 1 to semaphore.                                          */
/*   2)  Wait till read 1.                                              */
/************************************************************************/

int APPB_relsem(semnum)
      UINT  semnum;
   {
      UCHAR *semaddr = (UCHAR *)(SEM_BASE + semnum);

      *semaddr = 1; while(!(*semaddr & 1UL)); return(0);
   }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_getctlblk(), TMS320C30 side.                                   */
/*                                                                       */
/*   Find unused block of memory in the dual port.                       */
/*   Return a 0 if successful, a -1 if failed.                           */
/*                                                                       */
/*   Sequence                                                            */
/*                                                                       */
/*   1)  Search control structures for free block of memory.             */
/*   2)  If block free, set semnum to block index, return 0.             */
/*   3)  Else,  return -1 (failed to find block).                        */
/*                                                                       */
/*************************************************************************/
int APPB_getctlblk(semnum)
      UINT  *semnum;
    {
      int i;
      DPCNTL *dpctl = (DPCNTL *)DPRAM_CTL;

      APPB_getsem(0);

      for(i=0;i<DPRAM_BLKS;i++)
          if(!(dpctl[i].pflag & 1UL))
          {
            dpctl[i].pflag    = 1;
            dpctl[i].command  = NOP;
            dpctl[i].buf_stat = BUF_EMPTY;
            *semnum = i;
            APPB_relsem(0);   return(0);
          }

       APPB_relsem(0); return(-1);
    }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_relctlblk(), TMS320C30 side.                                   */
/*                                                                       */
/*   Release block of memory in the dual port.                           */
/*   Return a 0 if successful, a -1 if failed.                           */
/*                                                                       */
/*   Sequence                                                            */
/*                                                                       */
/*   1)  Null out the control structure.                                 */
/*   2)  Return.                                                         */
/*                                                                       */
/*************************************************************************/
int APPB_relctlblk(semnum)
      UINT  semnum;
    {
      int i;
      DPCNTL *dpctl = (DPCNTL *)DPRAM_CTL;

      APPB_getsem(0);
      dpctl[semnum].pflag    = 0;
      dpctl[semnum].command  = NOP;
      dpctl[semnum].buf_stat = BUF_EMPTY;
      APPB_relsem(0);  return(0);
    }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_putmemblk(), TMS320C30 side.                                   */
/*                                                                       */
/*   Move block of data to dual port.                                    */
/*                                                                       */
/*   Sequence                                                            */
/*                                                                       */
/*   1)  Move data to the dual port.                                     */
/*   2)  Set dual port buffer status to BUF_FULL.                        */
/*                                                                       */
/*************************************************************************/
int APPB_putmemblk(cnt,src,dpblk)
      ULONG cnt;
      ULONG *src;
      UINT  dpblk;
   {

      DPCNTL *dpctl = (DPCNTL *)DPRAM_CTL;
      UCHAR  *dpram;
      ULONG   temp;
      int     i,j;

      dpram = (UCHAR *)(DPRAM_MEMBASE + (dpblk * DPRAM_BLK_SIZE));

      for(i=0;i<cnt;i++)
      { temp = *src++; for(j=0;j<32;j+=8) *dpram++ = temp >> j; }

      APPB_getsem(0);
      dpctl[dpblk].buf_stat = BUF_FULL;
      APPB_relsem(0); return(0);
  }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_getmemblk(), TMS320C30 side.                                   */
/*                                                                       */
/*   Move block of data from dual port.                                  */
/*                                                                       */
/*   Sequence                                                            */
/*                                                                       */
/*   1)  Move data from the dual port.                                   */
/*   2)  Release block of dual port memory.                              */
/*                                                                       */
/*************************************************************************/
int APPB_getmemblk(cnt,dst,dpblk)
      ULONG cnt;
      ULONG *dst;
      UINT  dpblk;
   {

      DPCNTL *dpctl = (DPCNTL *)DPRAM_CTL;
      UCHAR  *dpram;
      ULONG   temp;
      int     i,j;

      dpram = (UCHAR *)(DPRAM_MEMBASE + (dpblk * DPRAM_BLK_SIZE));

      for(i=0;i<cnt;i++)
      {
          temp = 0UL;
          for(j=0;j<32;j+=8) temp |= ((*dpram++) &0x000000ff) << j;
          *dst++ = temp;
      }

      APPB_relctlblk(dpblk);  return(0);
  }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_getlong(), TMS320C30 side.                                     */
/*                                                                       */
/*   Get a long word of data from the dual port.                         */
/*************************************************************************/
int APPB_getlong(src,dst)
      ULONG *src;
      ULONG *dst;
   {
      int j;
      *dst = 0UL;
      for(j=0;j<32;j+=8) *dst |= ((*src++) & 0x000000ff) << j;
      return(0);
   }
```

```
/*************************************************************************/
/*                                                                       */
/*   APPB_getcommand(), TMS320C30 side.                                  */
/*                                                                       */
/*   Search the dual port control structures for commands.               */
/*                                                                       */
/*   Sequence                                                            */
/*                                                                       */
/*   1)  Get access to dual port semaphore 0.                            */
/*   2)  If at end of control structures, reset current_blk.             */
/*   3)  Search control structures for a command.                        */
/*   4)  If found, format parameters, return.                            */
/*   5)  Else, search to the end of list, return.                        */
/*                                                                       */
/*************************************************************************/
int APPB_getcommand(mparms)
      MPARMS  *mparms;
   {
      DPCNTL *dpctl = (DPCNTL *)DPRAM_CTL;
      static int  current_blk = -1;

      APPB_getsem(0);

      if(current_blk >= DPRAM_BLKS) current_blk = -1;

      while(current_blk++ < DPRAM_BLKS)
      {
          if(dpctl[current_blk].pflag & 1UL)
          {
              mparms->mcmd = dpctl[current_blk].command & 0x000000ff;
              mparms->mblk = current_blk;
              APPB_getlong(&dpctl[current_blk].count,&mparms->mcnt);
              APPB_getlong(&dpctl[current_blk].addr,&mparms->maddr);
              APPB_relsem(0); return(0);
          }
      }

      APPB_relsem(0); mparms->mcmd = NOP; return(0);
   }
```

APPENDIX A3. Memory Map and Description (TMS320C30 View)

Listed below is a summary of the APPB memory map.

000000 –	003FFF	EPROM (Boot EPROM/remappable)
004000 –	3FFFFF	Unused
400000 –	4FFFFF	DRAM space
400000 –	43FFFF	256K-word DRAM minimum configuration
440000 –	47FFFF	256K-word DRAM minimum configuration
480000 –	4BFFFF	256K-word DRAM option bank 2
4C0000 –	4FFFFF	256K-word DRAM option bank 3
500000 –	7FFFFF	Unused
800000 –	801FFF	SRAM space 1 (16K-byte zero wait-state SRAM)
802000 –	805FFF	Reserved by TI
804000 –	805FFF	I/O Devices
804000 –	804FFF	4K-byte dual-port SRAM
805000 –	805FF6	I/O Expansion Bus
805FF7		Control Register R
805FF8 –	805FFF	dual-port RAM Semaphores (D0 only)
806000 –	807FFF	Reserved by TI
808000 –	8097FF	Memory mapped Peripherals
809800 –	809BFF	RAM Block 0
809C00 –	809FFF	RAM Block 1
80A000 –	EFFFFF	Unused
F00000 –	F03FFF	SRAM space 0 (16K-byte zero wait-state SRAM, remappable)
F00800 –	FFFFFF	Unused

Appendix B

Modules

Appendix	Name
B1	Module U5 – TMS320C30 Software Development Board
B2	Module U6 – TMS320C30 Software Development Board
B3	Module RAMDEC – TMS320C30 Software Development Board
B4	Module RDYEN – TMS320C30 Software Development Board
B5	Module RAMCONTROL – TMS320C30 SWDS DRAM Module
B6	Module RAMDEC – TMS320C30 SWDS DRAM Module

Appendix B1. TMS320C30 Software Development Board

```
Module U5
title'
DWG NAME            TMS320C30 SOFTWARE DEVELOPMENT BOARD
DWG #       2554377
COMPANY     TEXAS INSTRUMENTS INCORPORATED
ENGR        NAT SESHAN
DATE        10/01/88'

XSUC8 device 'P2018';

SA0         Pin 1;
SA1         Pin 2;
SA2         Pin 3;
SA3         Pin 4;
SA4         Pin 5;      "PC XT ADDRESS LINES – INPUTS
SA5         Pin 6;
SA6         Pin 7;
SA7         Pin 8;
SA8         Pin 9;
SA9         Pin 10;
NSMEMW      Pin 11;     "PC XT MEMORY WRITE STROBE
GND         Pin 12;
NSMEMR      Pin 13;     "PC XT MEMORY READ STROBE – INPUT
NSIOW       Pin 14;     "PC XT IO WRITE STROBE – INPUT
NSGBA       Pin 15;     "SDB READ STROBE – OUTPUT
NPQ         Pin 16;     "DUAL-PORT ADDRESS RANGE STROBE – INPUT
XAEN        Pin 17;     "PC XT BUS TRANSACTION DISABLE – INPUT
NRG         Pin 18;     "SDB CONTROL REGISTER R ENABLE – OUTPUT
NQG         Pin 19;     "SDB DUAL-PORT ADDRESS LATCH ENABLE – OUTPUT
NDPSEML     Pin 20;     "DUAL-PORT SEMAPHORE SELECT – OUTPUT
NDPCEL      Pin 21;     "DUAL-PORT SRAM CHIP ENABLE – OUTPUT
SGAB        Pin 22;     "HOST DATA BUS INPUT ENABLE – OUTPUT
NSIOR       Pin 23;     "PC XT IO READ STROBE – INPUT
VCC         Pin 24;

SA = [SA9, SA8, SA7, SA6, SA5, SA4, SA3, SA2, SA1 ,SA0];
 X  = .X.;

equations

            !NQG        =  !XAEN & (SA == ^h338);
            !NRG        =  !XAEN & (SA == ^h339);
            !NDPSEML    =  !XAEN & SA9 & SA8 & !SA7 & !SA6 & SA5 & SA4 & !SA3
                           & !NSIOW
                           # !XAEN & SA9 & SA8 & !SA7 & !SA6 & SA5 & SA4 & !SA3
                           & !NSIOR;
```

```
!NDPCEL    = !XAEN & !NPQ;
SGAB       = !NSIOW & !XAEN
             # !NSMEMW & !XAEN ;
!NSGBA     = !XAEN & !NSIOR & (SA == ^h339)
             # !XAEN & !NSIOR & SA9 & SA8 & !SA7 & !SA6 & SA5
             & SA4 & !SA3
             # !XAEN & !NSMEMR & !NPQ;

end U5
```

Appendix B2. Module U6

```
Module U6
title'
DWG NAME    TMS320C30 SOFTWARE DEVELOPMENT BOARD
DWG #       2554377
COMPANY     TEXAS INSTRUMENTS INCORPORATED
ENGR        NAT SESHAN
DATE        10/01/88'

XSUF10      Device    'P20L8';

CIOA0       Pin 1;
CIOA1       Pin 2;
CIOA2       Pin 3;
CIOA3       Pin 4;
CIOA4       Pin 5;
CIOA5       Pin 6;
CIOA6       Pin 7;
CIOA7       Pin 8;
CIOA8       Pin 9;
CIOA9       Pin 10;
CIOA10      Pin 11;
GND         Pin 12;
CIOA11      Pin 13;
CIOA12      Pin 14;
TIOW        Pin 15;
NSRANGE     Pin 16;
CIORNW      Pin 17;
NFR         Pin 18;
NFG         Pin 19;
NDPMEMGR    Pin 20;
NDPSEMGR    Pin 21;
TIOR        Pin 22;
NCIOSTRB    Pin 23;
VCC         Pin 24;

            X = .X.;
            C = .C.;

            CIOA = [CIOA12,CIOA11,CIOA10,CIOA9,CIOA8,
                    CIOA7,CIOA6,CIOA5,CIOA4,CIOA3,CIOA2,CIOA1,CIOA0];

equations

            !NSRANGE    = !NCIOSTRB & !CIOA12
                          # !NCIOSTRB & (CIOA >= ^h1FF7);
            !NDPMEMGR   = !NCIOSTRB & !CIOA12;
            !NDPSEMGR   = !NCIOSTRB & (CIOA >= ^h1FF8);
```

```
!NFG        = !NCIOSTRB & !CIORNW & (CIOA == ^h1FF7);
!NFR        = !NCIOSTRB &  CIORNW & (CIOA == ^h1FF7);
!TIOR       = NCIOSTRB
              # (CIOA >= ^h1FF7)
              # !CIOA12
              # !CIORNW;

!TIOW       = NCIOSTRB
              # (CIOA >= ^h1FF7)
              # !CIOA12
              # CIORNW;
```

test_vectors

([CIOA, NCIOSTRB, CIORNW] ->
 [TIOR, TIOW, NSRANGE, NFG, NFR, NDPMEMGR, NDPSEMGR]);

READ OR WRITE TO A SEMAPHORE

[^h1FF8, 0, X] -> [0, 0, 0, 1, 1, 1, 0];
[^h1FF9, 0, X] -> [0, 0, 0, 1, 1, 1, 0];
[^h1FFA, 0, X] -> [0, 0, 0, 1, 1, 1, 0];
[^h1FFB, 0, X] -> [0, 0, 0, 1, 1, 1, 0];
[^h1FFC, 0, X] -> [0, 0, 0, 1, 1, 1, 0];
[^h1FFD, 0, X] -> [0, 0, 0, 1, 1, 1, 0];
[^h1FFE, 0, X] -> [0, 0, 0, 1, 1, 1, 0];
[^h1FFF, 0, X] -> [0, 0, 0, 1, 1, 1, 0];

WRITE TO F REGISTER

[^h1FF7, 0, 0] -> [0, 0, 0, 0, 1, 1, 1];

READ FROM F REGISTER

[^h1FF7, 0, 1] -> [0, 0, 0, 1, 0, 1, 1];

NCIOSTRB DISABLED

[X , 1, X] -> [0, 0, 1, 1, 1, 1, 1];

EXTERNAL READS

[^b1000000000000, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000000001, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000000010, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000000011, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000000100, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000000101, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000000110, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000000111, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000001000, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000001001, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];

[^b1000000001010, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000001011, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000001100, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000001101, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000001110, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^b1000000001111, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^h1FF0, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^h1FF1, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^h1FF2, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^h1FF3, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^h1FF4, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^h1FF5, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];
[^h1FF6, 0, 1] -> [1, 0, 1, 1, 1, 1, 1];

EXTERNAL IO WRITES

[^b1000000000000, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000000001, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000000010, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000000011, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000000100, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000000101, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000000110, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000000111, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000001000, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000001001, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000001010, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000001011, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000001100, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000001101, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000001110, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^b1000000001111, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^h1FF0, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^h1FF1, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^h1FF2, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^h1FF3, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^h1FF4, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^h1FF5, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];
[^h1FF6, 0, 0] -> [0, 1, 1, 1, 1, 1, 1];

test_vectors

([CIOA12, NCIOSTRB, CIORNW] ->
[TIOR, TIOW, NSRANGE, NFG, NFR, NDPSEMGR, NDPMEMGR]);

DUAL-PORT SRAM READ OR WRITE

[0, 0, X] -> [0, 0, 0, 1, 1, 1 ,0];

end U6

Appendix B3. Module RAMDEC

```
module RAMDEC
title'
DWG NAME            TMS320C30 SOFTWARE DEVELOPMENT BOARD
DWG #       2554377
COMPANY     TEXAS INSTRUMENTS INCORPORATED
ENGR        TONY COOMES
DATE        10/01/88'

XSUB4       device      'P16L8';

a12         Pin 1;      "c30 address inputs
a13         Pin 2;
a14         Pin 3;
a15         Pin 4;
a16         Pin 5;
a17         Pin 6;
a18         Pin 7;
a19         Pin 8;
a20         Pin 9;
a21         Pin 11;
a22         Pin 13;
a23         Pin 14;
m_swap      Pin 15;     "sram/eprom swap bit
vss         Pin 10;

memen       Pin 18;     "dram expansion select
sram        Pin 17;     " sram select
eprom       Pin 16;     "eprom select
busen       Pin 12;     "eprom/dram data buffer select
vcc         Pin 20;

madd = [a23,a22,a21,a20,a19,a18,a17,a16,a15,a14,a13,a12];

equations

            "On reset the eprom and sram maps are swapped
            "           m_swap = 0          m_swap = 1
            "sram       F00000–F03FFF       000000–003FFF
            "eprom      000000–003FFF       F00000–F03FFF

            sram    =  !(((madd >= ^h000) & (madd <= ^h003) & m_swap)
                        # ((madd >= ^hF00) & (madd <= ^hF03) & !m_swap));

            eprom   =  !(((madd >= ^h000) & (madd <= ^h003) & !m_swap)
                       # ((madd >= ^hF00) & (madd <= ^hF03) & m_swap));

            memen   =  !((madd >= ^h400) & (madd <= ^h4FF));

            busen   =  !(!eprom # !memen);
```

```
test_vectors
([madd,  m_swap ] -> [sram, eprom, memen, busen])
 [^h000,  1   ] -> [ 0,   1,  1,   1 ];
 [^h000,  0   ] -> [ 1,   0,  1,   0 ];
 [^h004,  1   ] -> [ 1,   1,  1,   1 ];
 [^hF00,  1   ] -> [ 1,   0,  1,   0 ];
 [^hF00,  0   ] -> [ 0,   1,  1,   1 ];
 [^hFF0,  1   ] -> [ 1,   1,  1,   1 ];
 [^hF00,  1   ] -> [ 1,   0,  1,   0 ];
 [^h400,  0   ] -> [ 1,   1,  0,   0 ];
 [^h4CF,  1   ] -> [ 1,   1,  0,   0 ];
 [^h800,  1   ] -> [ 1,   1,  1,   1 ];

end RAMDEC
```

Appendix B4. Module RDYEN

```
module RDYEN
title'
DWG NAME            TMS320C30 SOFTWARE DEVELOPMENT BOARD
DWG #       2554377
COMPANY     TEXAS INSTRUMENTS INCORPORATED
ENGR        TONY COOMES
DATE        10/01/88'

XSUC3       device      'P16R4';

clk         Pin 1;
busen       Pin 2;      "eprom/dram data bus enable
eprom       Pin 3;      "eprom select
strb        Pin 4;      "c30 strobe
rd_wr       Pin 5;      "c30 read/write
bhiz        Pin 7;      "dram expansion bus hold
oe          Pin 11;
vss         Pin 10;

dat_rd      Pin 19;     "data read enable
dat_wr      Pin 18;     "data write enable
prdy        Pin 17;     "eprom ready
epromcs     Pin 12;     "eprom chip select
vcc         Pin 20;

c = .C.;

equations

"note: bhiz is active for 1 TMS320C30 clock cycle at the end of a dram
"      access. This provides the necessary turn off time between
"      dram/eprom accesses.

dat_rd      =     !(!busen  & !strb & rd_wr & bhiz);

dat_wr      =     (!busen  & !strb & !rd_wr & bhiz);

epromcs     =     !(!busen & rd_wr & !strb & !eprom & bhiz);

prdy        :=    !(!busen & !strb & rd_wr & prdy & !eprom & bhiz);
```

```
test_vectors
([clk, strb, busen, rd_wr, eprom, oe, bhiz  ] -> prdy)
 [ c,  1,   1,   1,   1,   0,   1  ] -> 1;
 [ c,  0,   0,   1,   0,   0,   0  ] -> 1;
 [ c,  0,   0,   1,   0,   0,   1  ] -> 0;
 [ c,  0,   0,   1,   0,   0,   1  ] -> 1;
 [ c,  0,   0,   1,   0,   0,   1  ] -> 0;
 [ c,  1,   0,   1,   0,   0,   1  ] -> 1;
 [ c,  1,   0,   1,   0,   0,   1  ] -> 1;
test_vectors
([strb, busen, rd_wr, eprom, bhiz  ] -> [dat_rd, dat_wr, epromcs])
 [ 1,   1,   1,   1,    1  ] -> [ 1,    0,    1  ];
 [ 0,   0,   1,   1,    1  ] -> [ 0,    0,    1  ];
 [ 0,   0,   0,   1,    1  ] -> [ 1,    1,    1  ];
 [ 0,   1,   1,   1,    1  ] -> [ 1,    0,    1  ];
 [ 1,   0,   1,   1,    1  ] -> [ 1,    0,    1  ];
 check eprom
 [ 1,   0,   1,   0,    1  ] -> [ 1,    0,    1  ];
 [ 0,   0,   1,   0,    1  ] -> [ 0,    0,    0  ];
 [ 0,   0,   1,   0,    0  ] -> [ 1,    0,    1  ];
 [ 0,   0,   0,   0,    1  ] -> [ 1,    1,    1  ];
 [ 0,   1,   1,   0,    1  ] -> [ 1,    0,    1  ];
 [ 1,   0,   1,   1,    1  ] -> [ 1,    0,    1  ];

end RDYEN
```

Appendix B5. Module RAMCONTROL

```
Module RAMCONTROL
title'
DWG NAME    320C30 SWDS DRAM MODULE
DWG #       2554397
COMPANY     TEXAS INSTRUMENTS INCORPORATED
ENGR        TONY COOMES
DATE        10/01/88'

XDUE5       device      'P16R8';

clk         Pin 1;
refreq_     Pin 2;      "refresh request
strb_       Pin 3;      "c30 strobe
rd          Pin 4;      "c30 read/write
memen_      Pin 5;      "memory board chip select
oe_         Pin 11;     "pal output enable
vss         Pin 10;

s0          Pin 19;     "state variable
refclr      Pin 18;     "refresh clear
casen       Pin 17;     "column address strobe
ren         Pin 16;     "write strobe
rasen       Pin 15;     "row address strobe
mrdy        Pin 14;     "dram ready strobe
busact      Pin 13;     "dram bus active
s1          Pin 12;     "state variable
vcc         Pin 20;

"define machine states
"[refclr,rasen,casen,mrdy,busact,s0,s1];

idle    =   ^b1111111;
ras0    =   ^b1011111;
cas0    =   ^b1000111;
cas1    =   ^b1011101;
whld    =   ^b1111110;
trp     =   ^b1111001;
ref1    =   ^b0101111;
ref2    =   ^b0001111;
ref3    =   ^b0011111;
ref4    =   ^b1111101;

refreq  =   !refreq_;   "convert to positive logic
strb    =   !strb_;
memen   =   !memen_;
oe      = !oe_;

c = .C.;
```

```
c = .C.;
output = [refclr,rasen,casen,mrdy,busact,s0,s1];
equations
ren     :=    !(!rd & !strb_);   high on read, low on writes
state_diagram output
state idle:
        case  ( refreq &  strb &  memen)      :ref1;   "ref has 1st priority
              ( refreq &  strb & !memen)     :ref1;
              ( refreq & !strb &  memen)     :ref1;
              ( refreq & !strb & !memen)     :ref1;
              (!refreq &  strb &  memen)     :ras0;
              (!refreq &  strb & !memen)     :idle;
              (!refreq & !strb &  memen)     :idle;
              (!refreq & !strb & !memen)     :idle;
        endcase;
state   ras0:
        goto cas0;
state   cas0:                  "cycle cas on page mode reads
        case  rd                       :cas1;
              !rd                      :whld;
        endcase;
state   cas1:                  "cycle cas on page mode reads
        case  strb & !refreq           :cas0;
              strb &  refreq           :trp ;
              !strb & !refreq          :trp ;
              !strb &  refreq          :trp ;
        endcase;
state   whld:                  "wait for refreq or !strb
        case  strb & !refreq           :whld;
              strb &  refreq           :ref1;
              !strb & !refreq          ;idle;
              !strb &  refreq          :ref1;
              endcase;
state   trp:                   "cas,ras high
        case  refreq                   :ref1;
              !refreq                  :idle;
        endcase;
state   ref1:                  "cas,refclr low
        goto ref2;
state   ref2:                  "ras low
        goto ref3;
```

```
state     ref3:                    "cas high
          goto ref4;

state     ref4:                    "ras high
          goto idle;

test_vectors  "page mode read, ref, page mode read
([clk,refreq ,strb , rd,memen ,  oe ]->[output,ren])
 [ c,  0,    0,   1,   0,   1  ]->[idle ,  1 ];
 [ c,  0,    1,   1,   1,   1  ]->[ras0 ,  1 ];
 [ c,  0,    1,   1,   1,   1  ]->[cas0 ,  1 ];
 [ c,  0,    1,   1,   1,   1  ]->[cas1 ,  1 ];
 [ c,  0,    1,   1,   1,   1  ]->[cas0 ,  1 ];
 [ c,  1,    1,   1,   1,   1  ]->[cas1 ,  1 ];
 [ c,  1,    1,   1,   1,   1  ]->[trp  ,  1 ];
 [ c,  1,    1,   1,   1,   1  ]->[ref1 ,  1 ];
 [ c,  1,    1,   1,   1,   1  ]->[ref2 ,  1 ];
 [ c,  1,    1,   1,   1,   1  ]->[ref3 ,  1 ];
 [ c,  0,    1,   1,   1,   1  ]->[ref4 ,  1 ];
 [ c,  0,    1,   1,   1,   1  ]->[idle ,  1 ];
 [ c,  0,    1,   1,   1,   1  ]->[ras0 ,  1 ];
 [ c,  0,    1,   1,   1,   1  ]->[cas0 ,  1 ];
 [ c,  0,    1,   1,   1,   1  ]->[cas1 ,  1 ];
 [ c,  0,    1,   1,   1,   1  ]->[cas0 ,  1 ];
 [ c,  0,    1,   1,   1,   1  ]->[cas1 ,  1 ];
 [ c,  0,    0,   1,   1,   1  ]->[trp  ,  1 ];
 [ c,  0,    0,   1,   0,   1  ]->[idle ,  1 ];

test_vectors  "write cycle
([clk,refreq ,strb , rd, memen, oe  ]->[output,ren])
 [ c,  0,    0,   0,   0,   1  ]->[idle ,  1 ];
 [ c,  0,    1,   0,   1,   1  ]->[ras0 ,  0 ];
 [ c,  0,    1,   0,   1,   1  ]->[cas0 ,  0 ];
 [ c,  0,    1,   0,   1,   1  ]->[whld ,  0 ];
 [ c,  0,    1,   0,   1,   1  ]->[whld ,  0 ];
 [ c,  0,    1,   0,   1,   1  ]->[whld ,  0 ];
 [ c,  0,    0,   0,   1,   1  ]->[idle ,  1 ];
 [ c,  0,    0,   1,   0,   1  ]->[idle ,  1 ];

"write cycle /ref
 [ c,  0,    0,   0,   0,   1  ]->[idle ,  1 ];
 [ c,  0,    1,   0,   1,   1  ]->[ras0 ,  0 ];
 [ c,  1,    1,   0,   1,   1  ]->[cas0 ,  0 ];
 [ c,  1,    1,   0,   1,   1  ]->[whld ,  0 ];
 [ c,  1,    1,   0,   1,   1  ]->[ref1 ,  0 ];
 [ c,  1,    1,   0,   1,   1  ]->[ref2 ,  0 ];
 [ c,  1,    0,   0,   0,   1  ]->[ref3 ,  1 ];
 [ c,  0,    0,   1,   0,   1  ]->[ref4 ,  1 ];
 [ c,  0,    0,   1,   0,   1  ]->[idle ,  1 ];

end RAMCONTROL
```

Appendix B6. Module RAMDEC

```
module RAMDEC
title'
DWG NAME            320C30 SWDS DRAM MODULE
DWG #       2554397
COMPANY     TEXAS INSTRUMENTS INCORPORATED
ENGR        TONY COOMES
DATE        10/01/88'

XDUD5       device      'P16R4';

clk         Pin 1;
refclr      Pin 2;      "clear refresh stat
a18         Pin 3;      "c30 address 18
a19         Pin 4;      "c30 address 19
memen       Pin 5;      "dram board memory enable
strb        Pin 6;      "c30 strobe
mux         Pin 7;      "address mux
oe          Pin 11;     "pal output enable
vss         Pin 10;

ras0        Pin 17;     "ras select 0
ras1        Pin 16;     "ras select 1
ras2        Pin 15;     "ras select 2
ras3        Pin 14;     "ras select 3
rowsel      Pin 13;     "row address select
vcc         Pin 20;

c = .C.;

equations

ras0 := !(!refclr # (!a19 & !a18 & !memen & !strb));
ras1 := !(!refclr # (!a19 &  a18 & !memen & !strb));
ras2 := !(!refclr # ( a19 & !a18 & !memen & !strb));
ras3 := !(!refclr # ( a19 &  a18 & !memen & !strb));

rowsel = mux;
```

```
test_vectors  "page mode read, ref, page mode read
([clk,refclr, memen, strb, a19, a18, oe]->[ras0, ras1, ras2, ras3])
[ c,  1,    1,  1,   0,  0,  0 ]->[ 1,   1,   1,   1 ];
[ c,  1,    0,  0,   0,  0,  0 ]->[ 0,   1,   1,   1 ];
[ c,  1,    0,  0,   0,  1,  0 ]->[ 1,   0,   1,   1 ];
[ c,  1,    0,  0,   1,  0,  0 ]->[ 1,   1,   0,   1 ];
[ c,  1,    0,  0,   1,  1,  0 ]->[ 1,   1,   1,   0 ];
[ c,  1,    1,  0,   1,  1,  0 ]->[ 1,   1,   1,   1 ];
[ c,  1,    0,  1,   1,  1,  0 ]->[ 1,   1,   1,   1 ];
[ c,  0,    0,  1,   1,  1,  0 ]->[ 0,   0,   0,   0 ];
[ c,  1,    0,  1,   1,  1,  0 ]->[ 1,   1,   1,   1 ];
[ c,  0,    0,  0,   1,  1,  0 ]->[ 0,   0,   0,   0 ];
[ c,  1,    0,  0,   1,  1,  0 ]->[ 1,   1,   1,   0 ];

test_vectors "rowsel
(mux -> rowsel)
1  ->  1;
0  ->  0;

end RAMDEC
```

Appendix C

TMS320C30 Application Board Schematics

Appendix C1. TMS320C30 Software Development Schematics

REVISIONS			
REV	DESCRIPTION	DATE	APPROVED

NOTES: UNLESS OTHERWISE SPECIFIED

1. ALL AS, ALS, LS DEVICES ARE PREFIXED WITH AN SN74
2. VCC IS APPLIED TO PIN 8 OF ALL 8-PIN IC's, PIN 14 OF ALL 14-PIN IC's, PIN 16 OF ALL 16-PIN IC's, PIN 20 OF ALL 20-PIN IC's, ETC.
3. GROUND IS APPLIED TO PIN 4 OF ALL 8-PIN IC's, PIN 7 OF ALL 14-PIN IC's, PIN 8 OF ALL 16-PIN IC's, PIN 10 OF ALL 20-PIN IC's, ETC.
4. DEVICE TYPE, PIN NUMBERS, AND REFERENCE DESIGNATOR OF GATES ARE SHOWN AS FOLLOWS:

00 AND 04 = DEVICE TYPES
1, 2, AND 3 = PIN NUMBERS
U06 AND U07 = REFERENCE DESIGNATORS

5. RESISTANCE VALUES ARE IN OHMS.
6. RESISTORS ARE 1/4 WATT, 5%.
7. CAPACITANCE VALUES ARE IN MICROFARADS.

COMPUTER GENERATED DRAWING : DO NOT REVISE MANUALLY

PARTS LIST

QTY	ITEM NO	PART OR IDENTIFYING NUMBER	NOMENCLATURE OR DESCRIPTION	NOTES

REVISION STATUS OF SHEETS										
REV	*	*	*	*	*	*	*	*	*	*
SH	1	2	3	4	5	6	7	8	9	10
REV	*	*								
SH	11	12	13	14	15	16	17	18	19	20
REV										
SH	21	22	23	24	25	26	27	28	29	30
REV										
SH	31	32	33	34	35	36	37	38	39	40

2554375 | 7119

NEXT ASSY | USED ON

APPLICATION

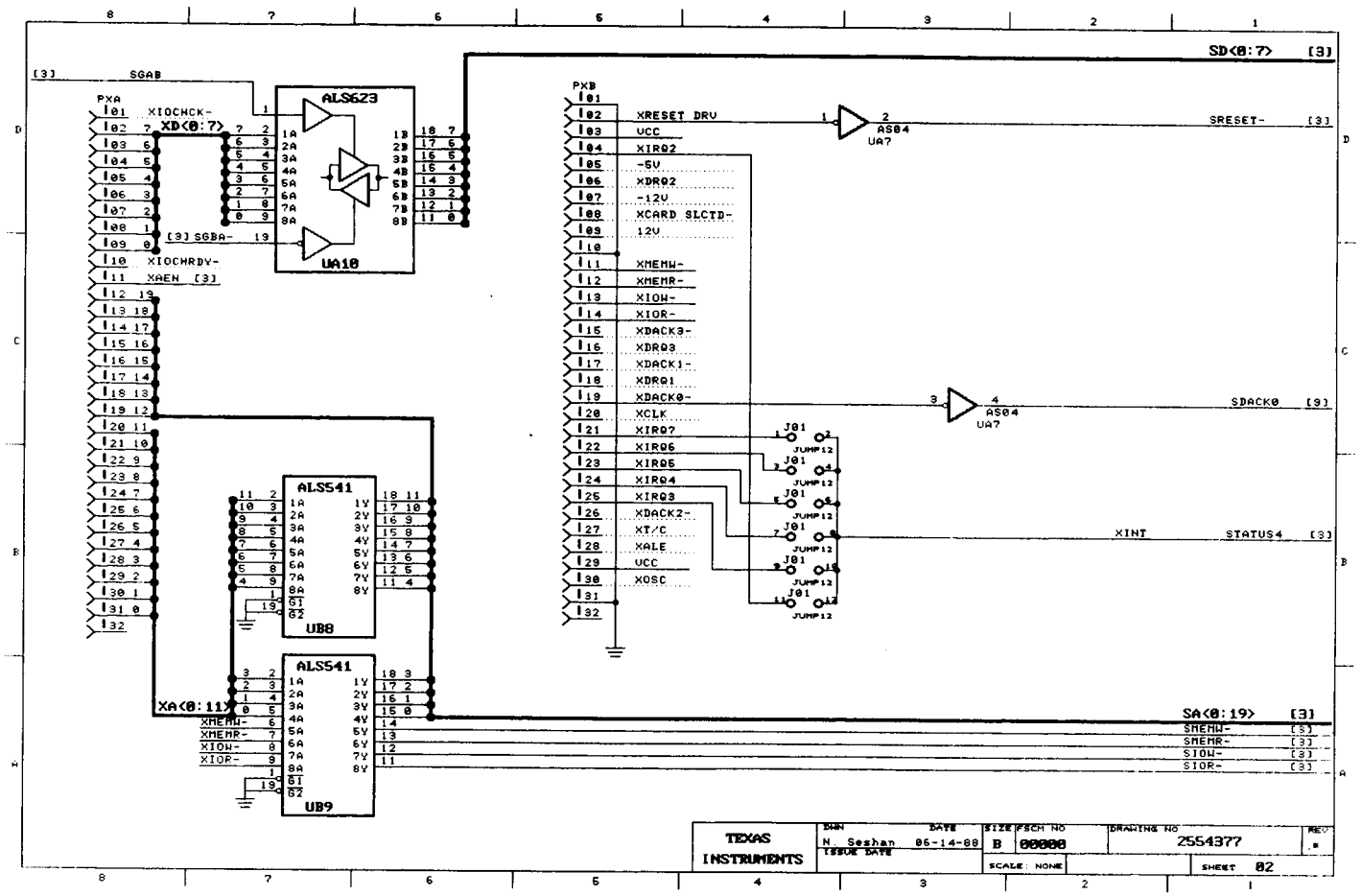
DWN	N. Seshan	06-15-88
CHK		
ENGR	N. Seshan	06-15-88
APVD ENGR	Tony Coomes	
QA		
APVD MFG		
RLSE		

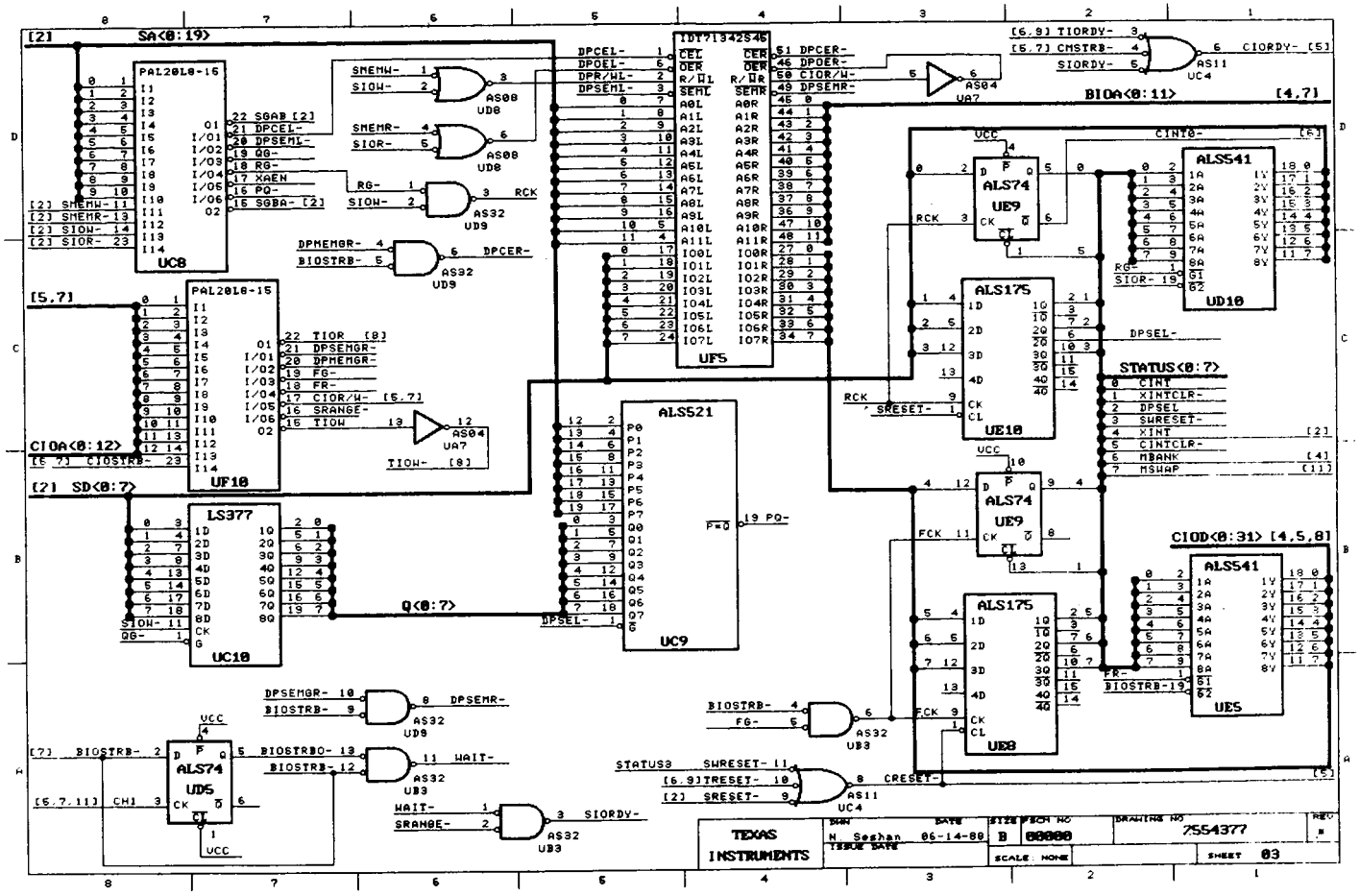
TEXAS INSTRUMENTS
Data Systems Group

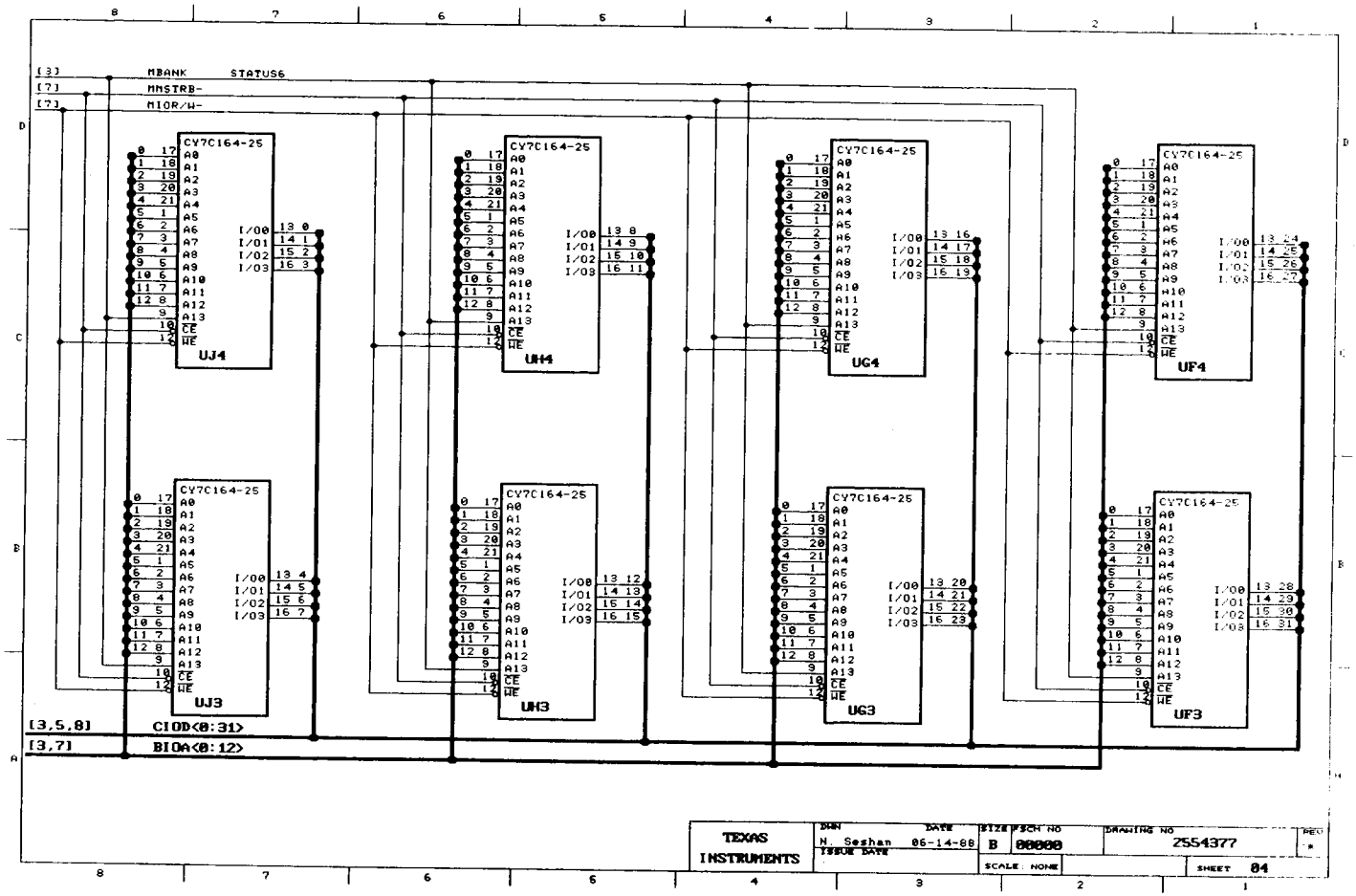
TMS320C30 SOFTWARE DEVELOPMENT BOARD

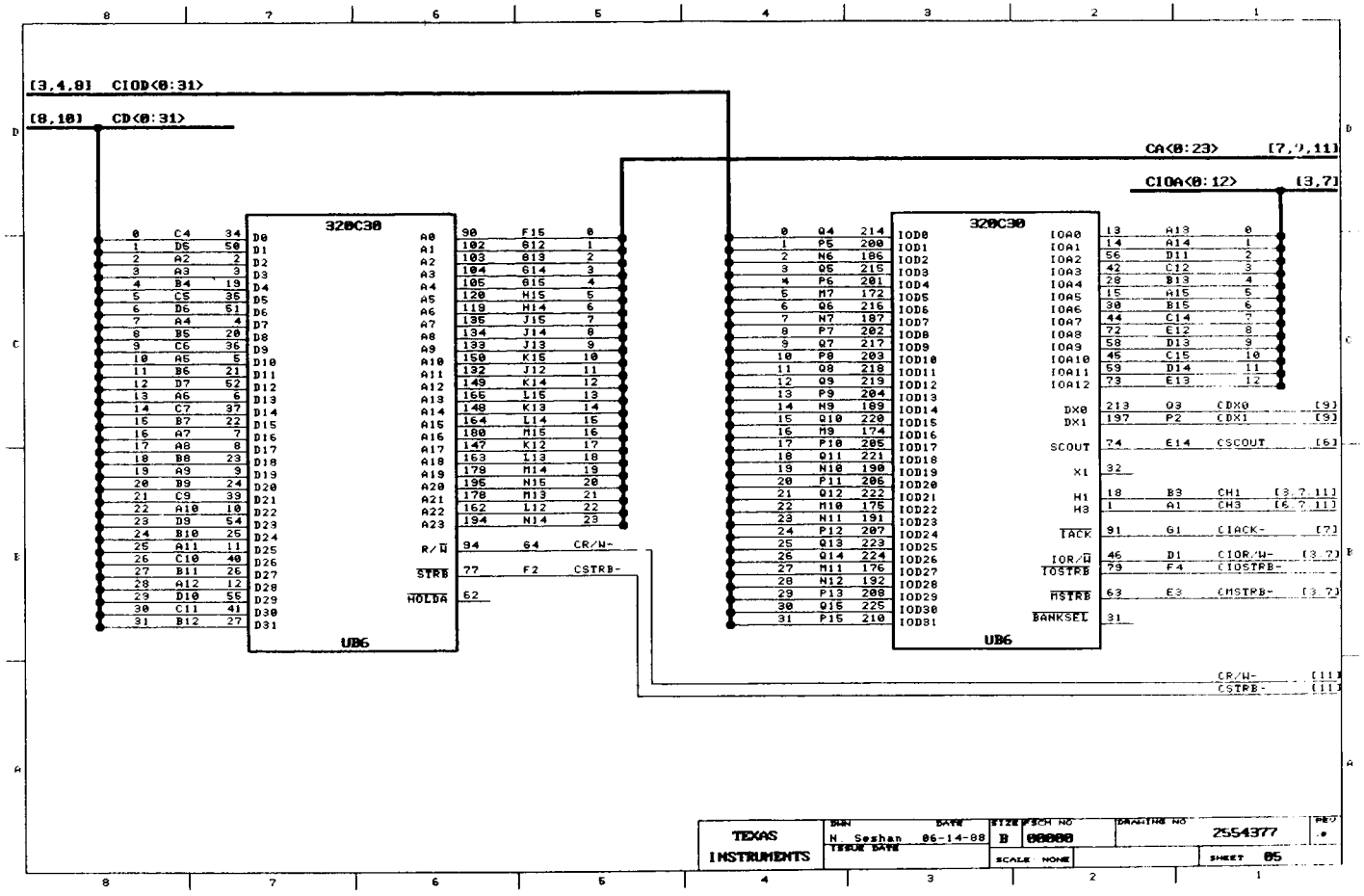
SIZE	FSCM NO	DRAWING NO	REV
B	00000	2554377	*

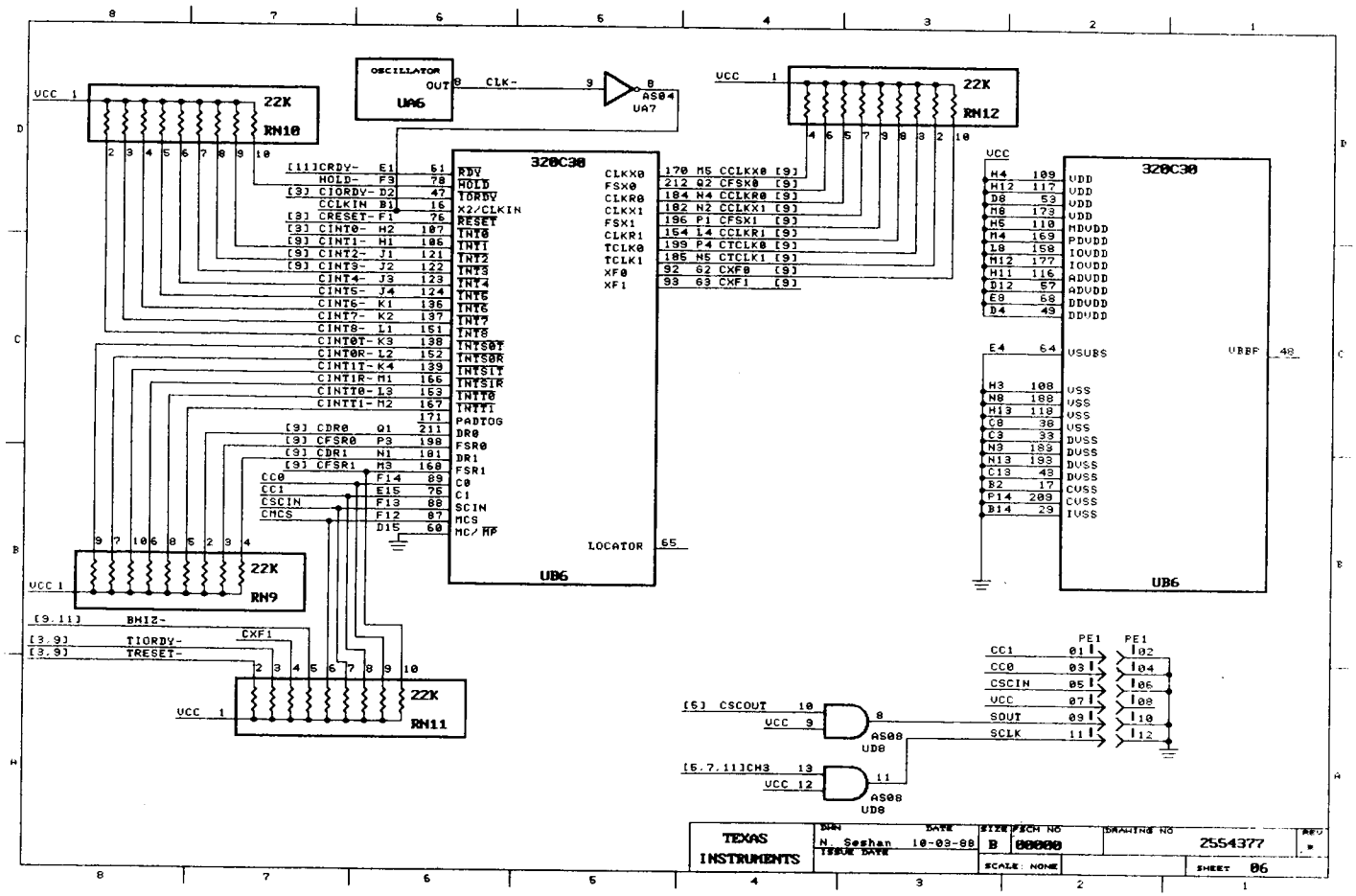
SCALE NONE | SHEET 1 OF 12

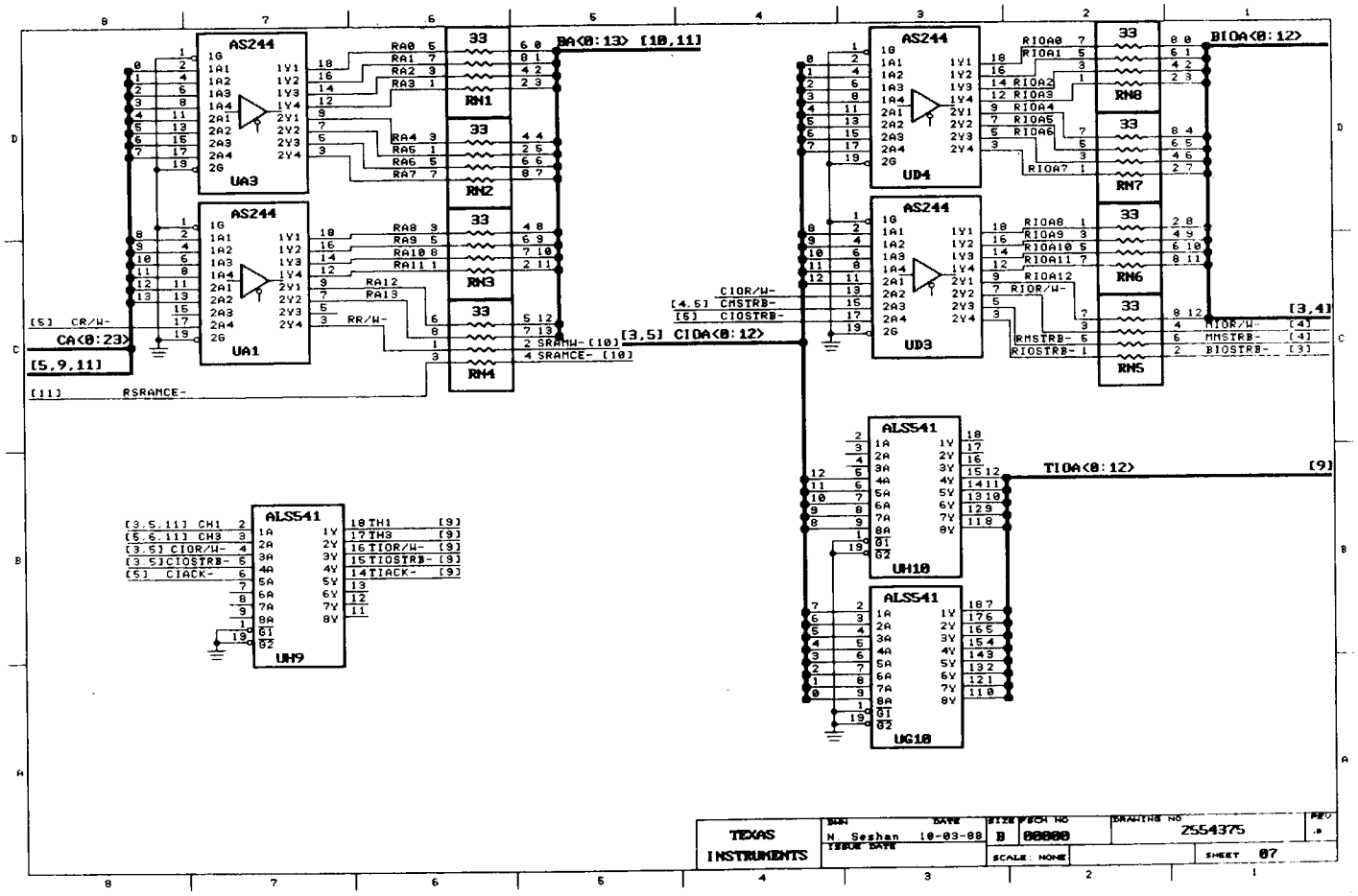

SD<0:7> [3]
[3] SGAB
PXA
XIOCHCK-
XD<0:7>
[3] SGBA-
XIOCHRDY-
XAEN [3]
ALS623
UA10
PXB
XRESET DRV
VCC
XIRQ2
-5V
XDRQ2
-12V
XCARD SLCTD-
12V
XMEMW-
XMEMR-
XIOW-
XIOR-
XDACK3-
XDRQ3
XDACK1-
XDRQ1
XDACK0-
XCLK
XIRQ7
XIRQ6
XIRQ5
XIRQ4
XIRQ3
XDACK2-
XT/C
XALE
VCC
XOSC
AS04
UA7
SRESET- [3]
SDACK0 [9]
XINT
STATUS4 [3]
JUMP12
ALS541
UB8
UB9
XA<0:11>
SA<0:19> [3]
SMEMW- [3]
SMEMR- [3]
SIOW- [3]
SIOR- [3]
TEXAS INSTRUMENTS
DWN N Seshan DATE 06-14-88
ISSUE DATE
SIZE B
FSCM NO 00000
DRAWING NO 2554377
REV
SCALE: NONE
SHEET 02

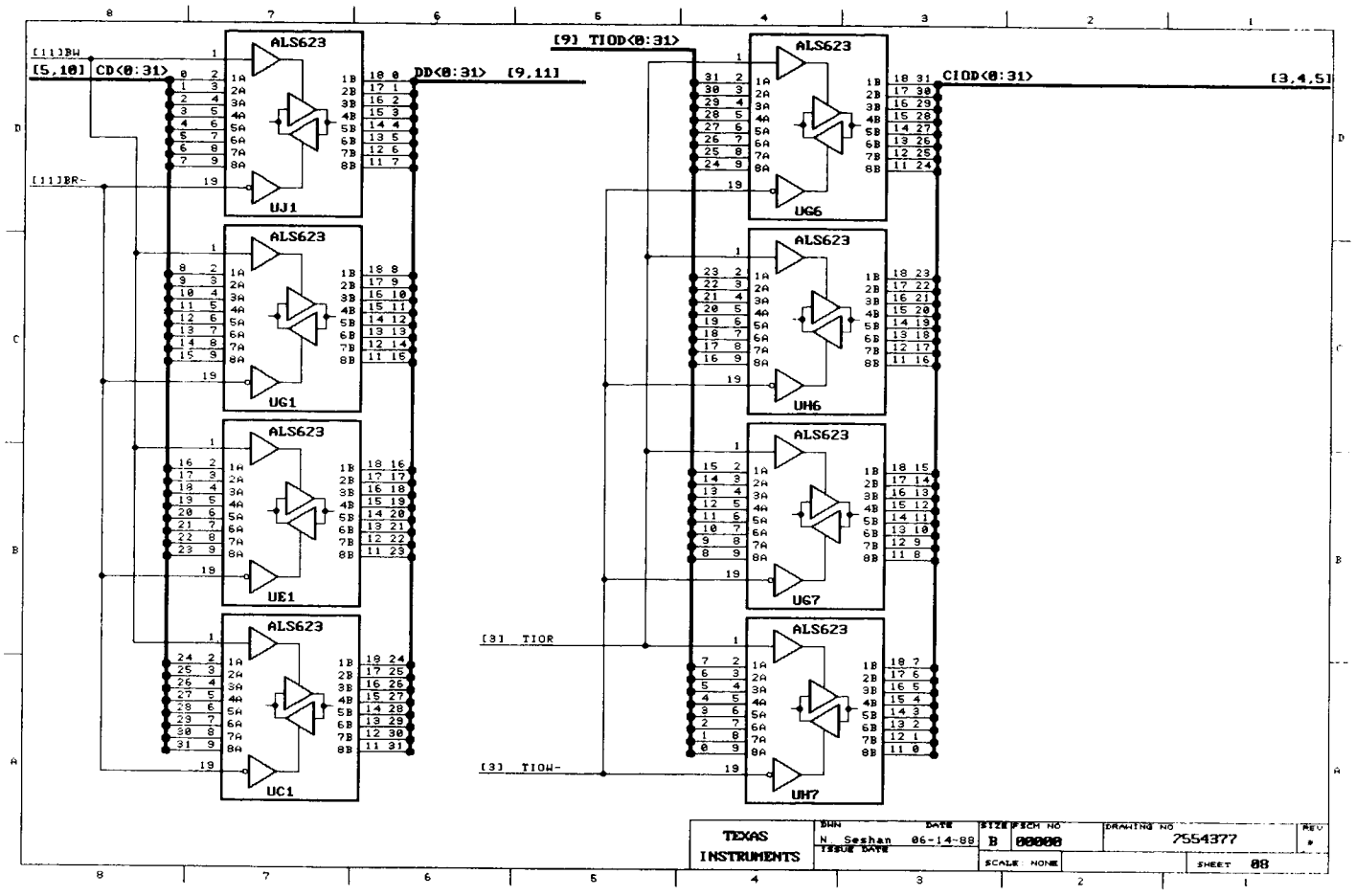

[2] SA<0:19>
PAL20L8-16
UC8
[2] SMEMW- 11
[2] SMEMR- 13
[2] SIOW- 14
[2] SIOR- 23
22 SBAB [2]
21 DPCEL-
20 DPSEML-
19 QB-
18 RB-
17 XAEN
16 PQ-
15 SBBA- [2]
SMEMW-
SIOW-
SMEMR-
SIOR-
AS08 UD8
RB-
RCK
AS32 UD9
DPMEMBR-
BIOSTRB-
DPBER-
[5,7]
PAL20L8-15
UF10
22 TIOR [8]
21 DPSEMBR-
20 DPMEMBR-
19 FB-
18 FR-
17 CIOR/W- [5,7]
16 SRANGE-
15 TIOW
AS04 UA7
TIOW- [8]
CIOA<0:12>
[6,7] CIOSTRB- 23
[2] SD<0:7>
LS377
UC10
SIOW- 11
QB- 1
Q<0:7>
IDT71342S45
UF5
DPCEL-
DPOEL-
DPR/WL-
DPSEML-
SEML
A0L
A1L
A2L
A3L
A4L
A5L
A6L
A7L
A8L
A9L
A10L
A11L
IO0L
IO1L
IO2L
IO3L
IO4L
IO5L
IO6L
IO7L
CEL
OEL
R/WL
CER
OER
R/WR
SEMR
A0R
A1R
A2R
A3R
A4R
A5R
A6R
A7R
A8R
A9R
A10R
A11R
IO0R
IO1R
IO2R
IO3R
IO4R
IO5R
IO6R
IO7R
51 DPCER-
46 DPOER-
50 CIOR/W-
49 DPSEMR-
AS04 UA7
ALS521
UC9
P=Q
19 PQ-
DPSEL-
DPSEMBR- 10
BIOSTRB- 9
AS32 UD9
8 DPSEMR-
[7] BIOSTRB- 2
ALS74 UD5
BIOSTRBO- 13
BIOSTRB- 12
11 WAIT-
AS32 UB3
WAIT- 1
SRANGE- 2
3 SIORDY-
[6,7,11] CMJ 3
[6.9] TIORDY- 3
[6.7] CMSTRB- 4
SIORDY- 5
AS11 UC4
6 CIORDY- [5]
BIOA<0:11> [4,7]
CINTB- [6]
ALS74 UD9
RCK 3
ALS541 UD10
RB- 1
SIOR- 19
STATUS<0:7>
0 CINT
1 XINTCLR-
2 DPSEL
3 SWRESET-
4 XINT [2]
5 CINTCLR-
6 MBANK [4]
7 MSWAP [1]
ALS175 UE10
RCK
SRESET- 1
DPSEL-
CIOD<0:31> [4,5,8]
FCK 11
UE9
UE8
UE5
FR-
BIOSTRB-19
BIOSTRB- 4
FB- 5
AS32 UB3
STATUS3 SWRESET- 11
[6.9] TRESET- 10
[2] SRESET- 9
AS11 UC4
8 CRESET-
[5]
TEXAS INSTRUMENTS
DRAWN N Seshan DATE 06-14-88
ISSUE DATE
SIZE B
PROJ NO 00000
DRAWING NO 7554377
REV M
SCALE NONE
SHEET 03

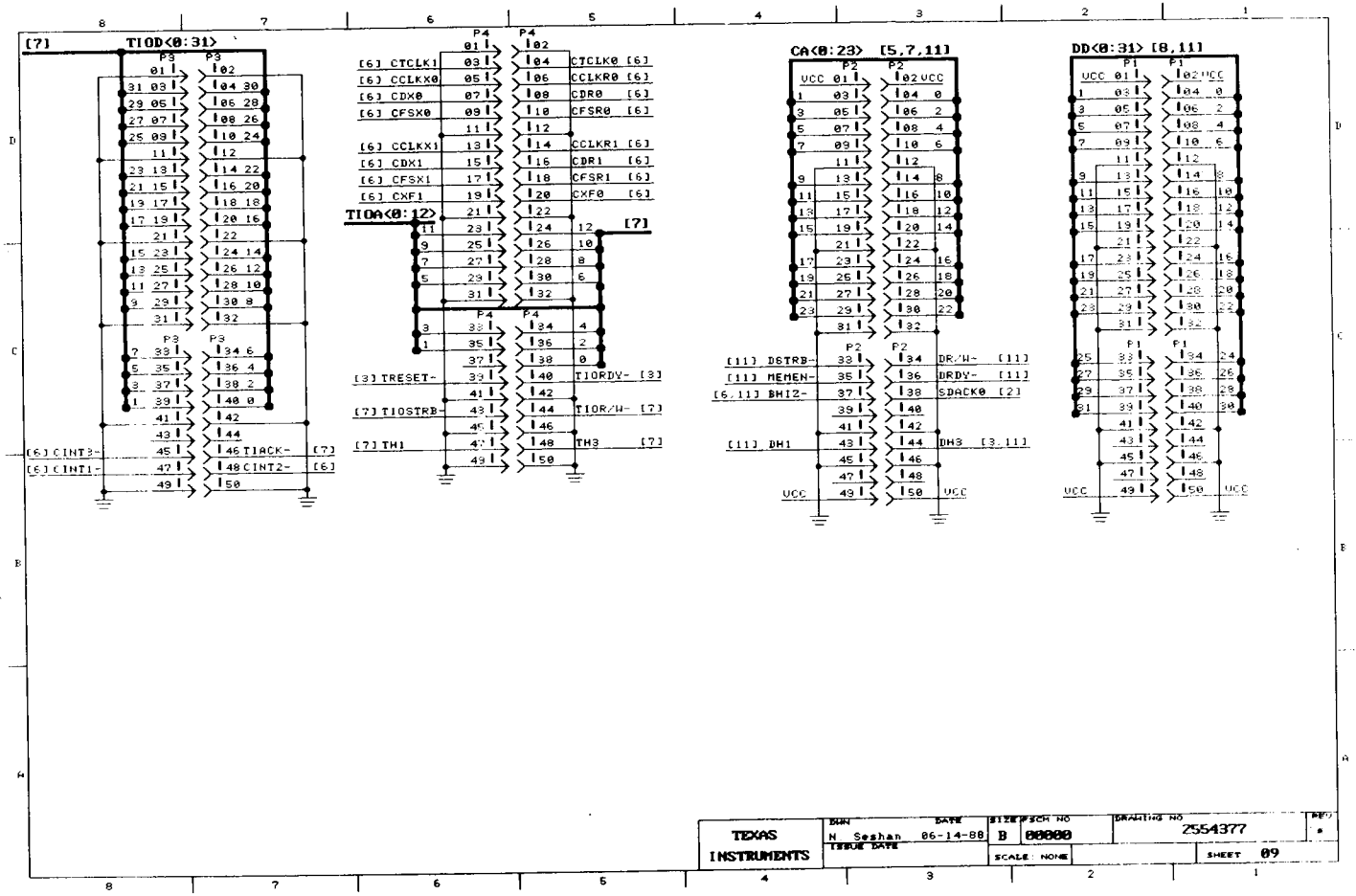

[3] MBANK STATUS6
[7] MMSTRB-
[7] MIOR/W-
CY7C164-25
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13
CE WE
I/O0 I/O1 I/O2 I/O3
UJ4
UH4
UG4
UF4
UJ3
UH3
UG3
UF3
[3,5,8] CIOD<0:31>
[3,7] BIOA<0:12>
TEXAS INSTRUMENTS
DWN N. Seshan DATE 06-14-88
ISSUE DATE
SIZE B
PSCH NO 00000
DRAWING NO 2554377
REV *
SCALE: NONE
SHEET 04


[3,4,8] CIOD<0:31>
[8,10] CD<0:31>
CA<0:23> [7,9,11]
CIOA<0:12> [3,7]
320C30
UB6
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15 D16 D17 D18 D19 D20 D21 D22 D23 D24 D25 D26 D27 D28 D29 D30 D31
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A16 A17 A18 A19 A20 A21 A22 A23
R/W
STRB
HOLDA
IOD0 IOD1 IOD2 IOD3 IOD4 IOD5 IOD6 IOD7 IOD8 IOD9 IOD10 IOD11 IOD12 IOD13 IOD14 IOD15 IOD16 IOD17 IOD18 IOD19 IOD20 IOD21 IOD22 IOD23 IOD24 IOD25 IOD26 IOD27 IOD28 IOD29 IOD30 IOD31
IOA0 IOA1 IOA2 IOA3 IOA4 IOA5 IOA6 IOA7 IOA8 IOA9 IOA10 IOA11 IOA12
DX0 DX1
SCOUT
X1
H1 H3
IACK
IOR/W
IOSTRB
MSTRB
BANKSEL
CR/W-
CSTRB-
CDX0 [9]
CDX1 [9]
CSCOUT [6]
CH1 [3,7,11]
CH3 [6,7,11]
CIACK- [7]
CIOR/W- [3,7]
CIOSTRB-
CMSTRB- [3,7]
TEXAS INSTRUMENTS
DRN N Seshan
DATE 06-14-88
ISSUE DATE
SIZE B
SCH NO 00000
DRAWING NO 2554377
SCALE NONE
SHEET 05

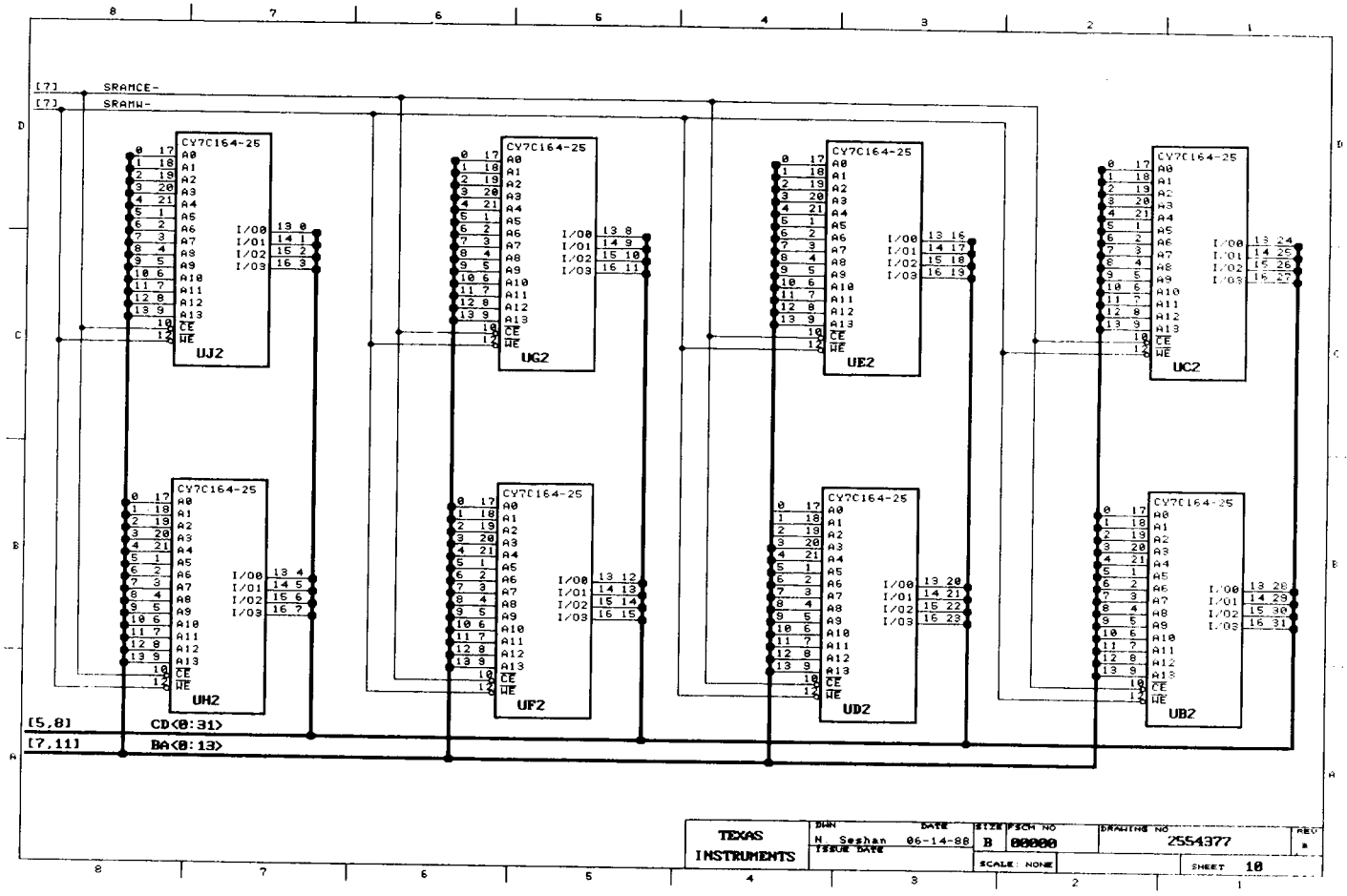

OSCILLATOR
OUT
UA6
CLK-
AS04
UA7
VCC
22K
RN10
RN12
RN9
RN11
320C30
UB6
CLKX0
FSX0
CLKR0
CLKX1
FSX1
CLKR1
TCLK0
TCLK1
XF0
XF1
LOCATOR
RDY
HOLD
TORDY
X2/CLKIN
RESET
INT0
INT1
INT2
INT3
INT4
INT5
INT6
INT7
INT8
INTSDT
INTSDR
INTSIT
INTSIR
INTT0
INTT1
PADIOG
DR0
FSR0
DR1
FSR1
C0
C1
SCIN
MCS
MC/MP
VUBB
VSUBS
VDD
VSS
DUSS
CUSS
IUSS
[11]CRDY-
HOLD-
[3] CIORDY-
CCLKIN
[3] CRESET-
[3] CINT0-
[9] CINT1-
[9] CINT2-
[9] CINT3-
CINT4-
CINT5-
CINT6-
CINT7-
CINT8-
CINT0T-
CINT0R-
CINT1T-
CINT1R-
CINTT0-
CINTT1-
[9] CDR0
[9] CFSR0
[9] CDR1
[9] CFSR1
CC0
CC1
CSCIN
CMCS
[9.11] BHIZ-
[3.9] TIORDY-
[3.9] TRESET-
CXF1
[6] CSCOUT
[6.7.11]CH3
AS08
UD8
PE1
CC1
CC0
CSCIN
VCC
SOUT
SCLK
TEXAS INSTRUMENTS
N. Seshan
10-03-88
B
00000
2554377
SCALE: NONE
SHEET 06

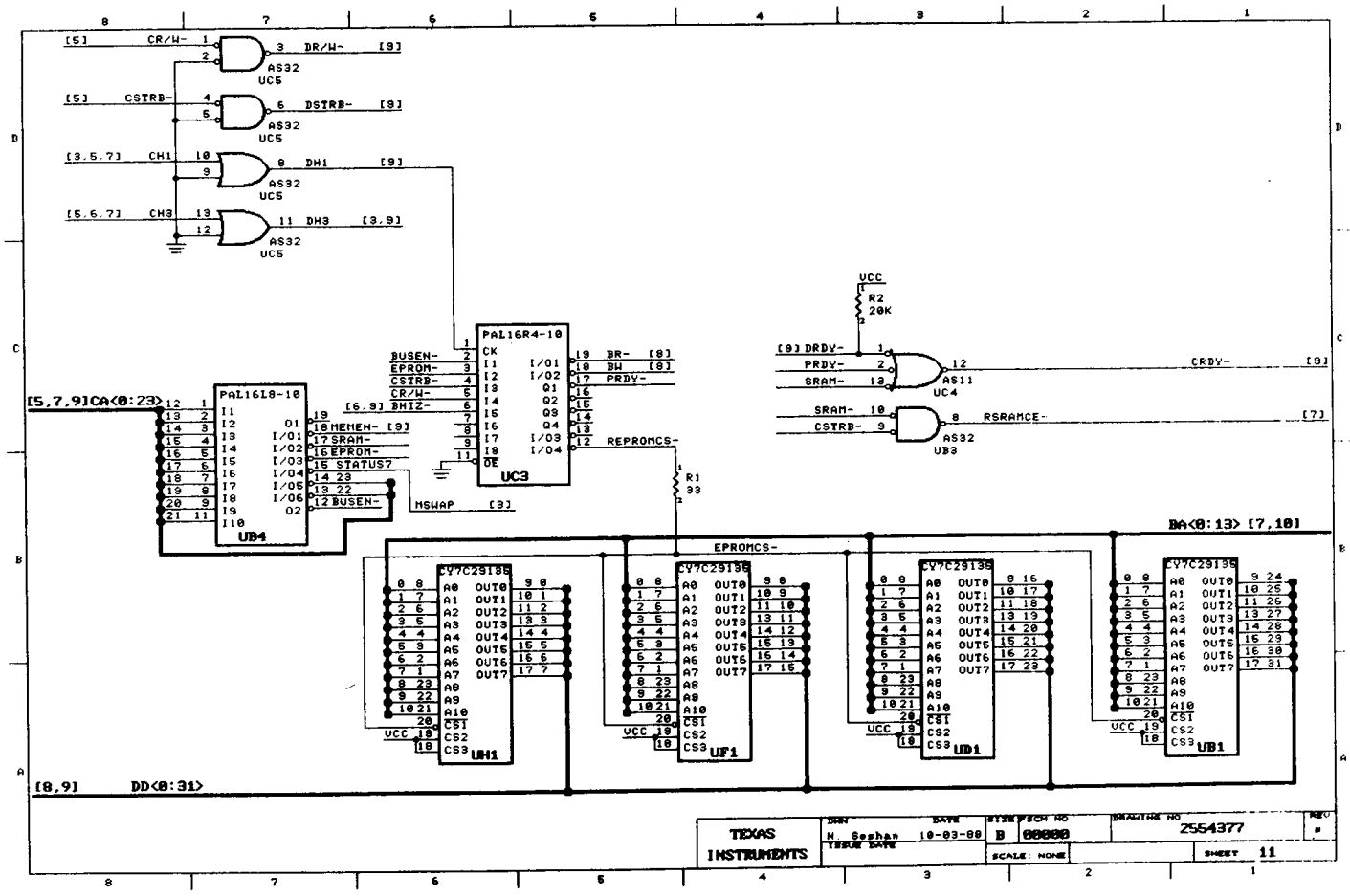

AS244
UA3
UA1
UD4
UD3
RN1
RN2
RN3
RN4
RN5
RN6
RN7
RN8
33
BA<0:13> [10,11]
BIOA<0:12>
[3,4]
CA<0:23>
[5,9,11]
[5] CR/W-
[11] RSRAMCE-
[3,5] CIOA<0:12>
CIOR/W-
[4,5] CMSTRB-
[5] CIOSTRB-
RA/W-
SRAMW- [10]
SRAMCE- [10]
RIOR/W-
RMSTRB- [5]
RIOSTRB- 1
MIOR/W- [4]
MMSTRB- [4]
BIOSTRB- [3]
ALS541
UH9
UH10
UG10
TIOA<0:12> [9]
[3,5,11] CH1
[5,6,11] CH3
[3,5] CIOR/W-
[3,5] CIOSTRB-
[5] CIACK-
TH1 [9]
TH3 [9]
TIOR/W- [9]
TIOSTRB- [9]
TIACK- [9]
TEXAS INSTRUMENTS
N Seshan 10-03-88
B
00000
2554375
SCALE: NONE
SHEET 07

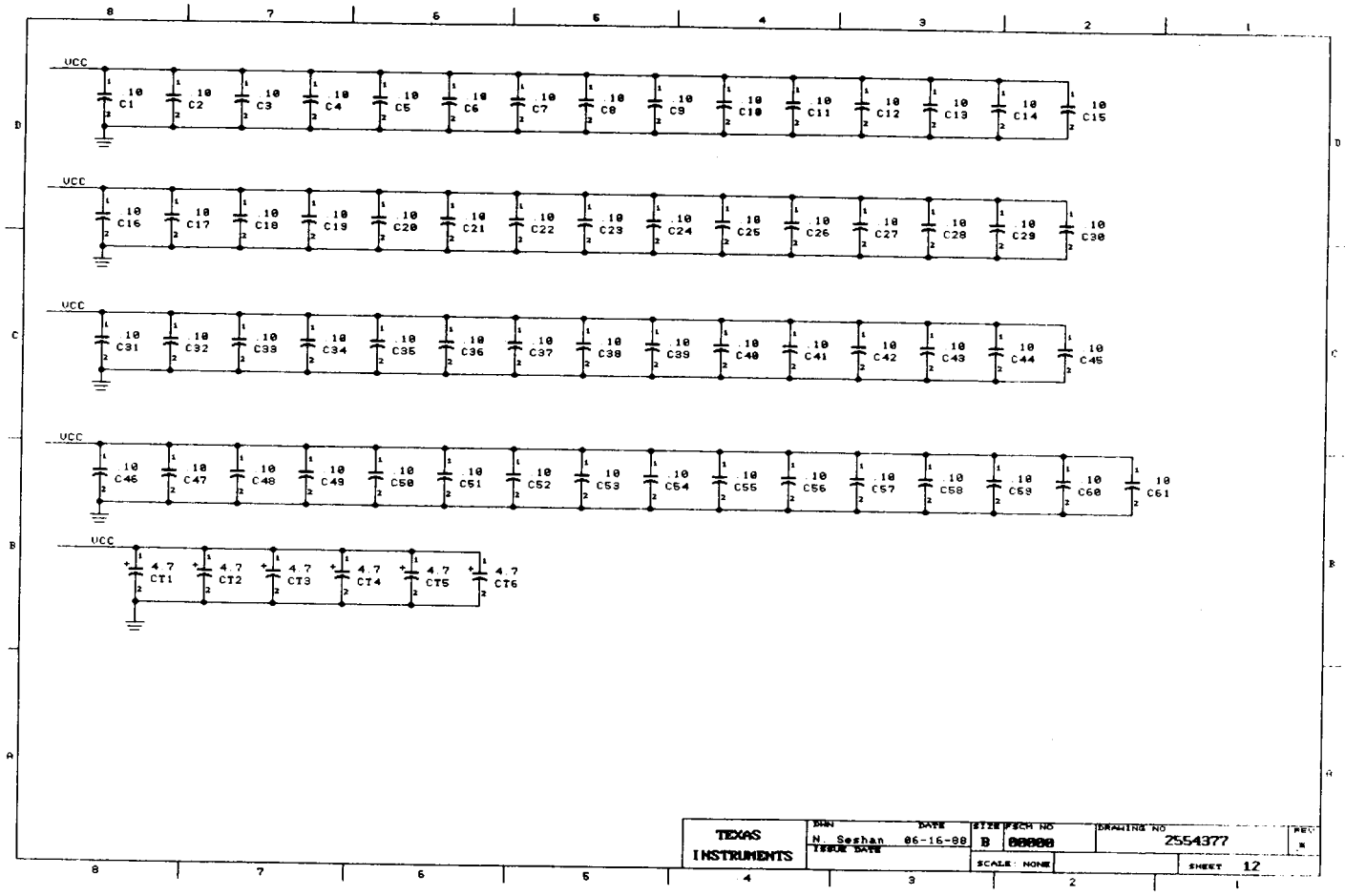

[11]BW
[5,10] CD<0:31>
[11]BR-
ALS623
UJ1
UG1
UE1
UC1
DD<0:31> [9,11]
[9] TIOD<0:31>
UG6
UH6
UG7
UH7
[3] TIOR
[3] TIOW-
CIOD<0:31>
[3,4,5]
TEXAS INSTRUMENTS
DWN
N. Seshan
DATE
06-14-88
ISSUE DATE
SIZE
B
SCH NO
00000
DRAWING NO
7554377
REV
SCALE NONE
SHEET
08



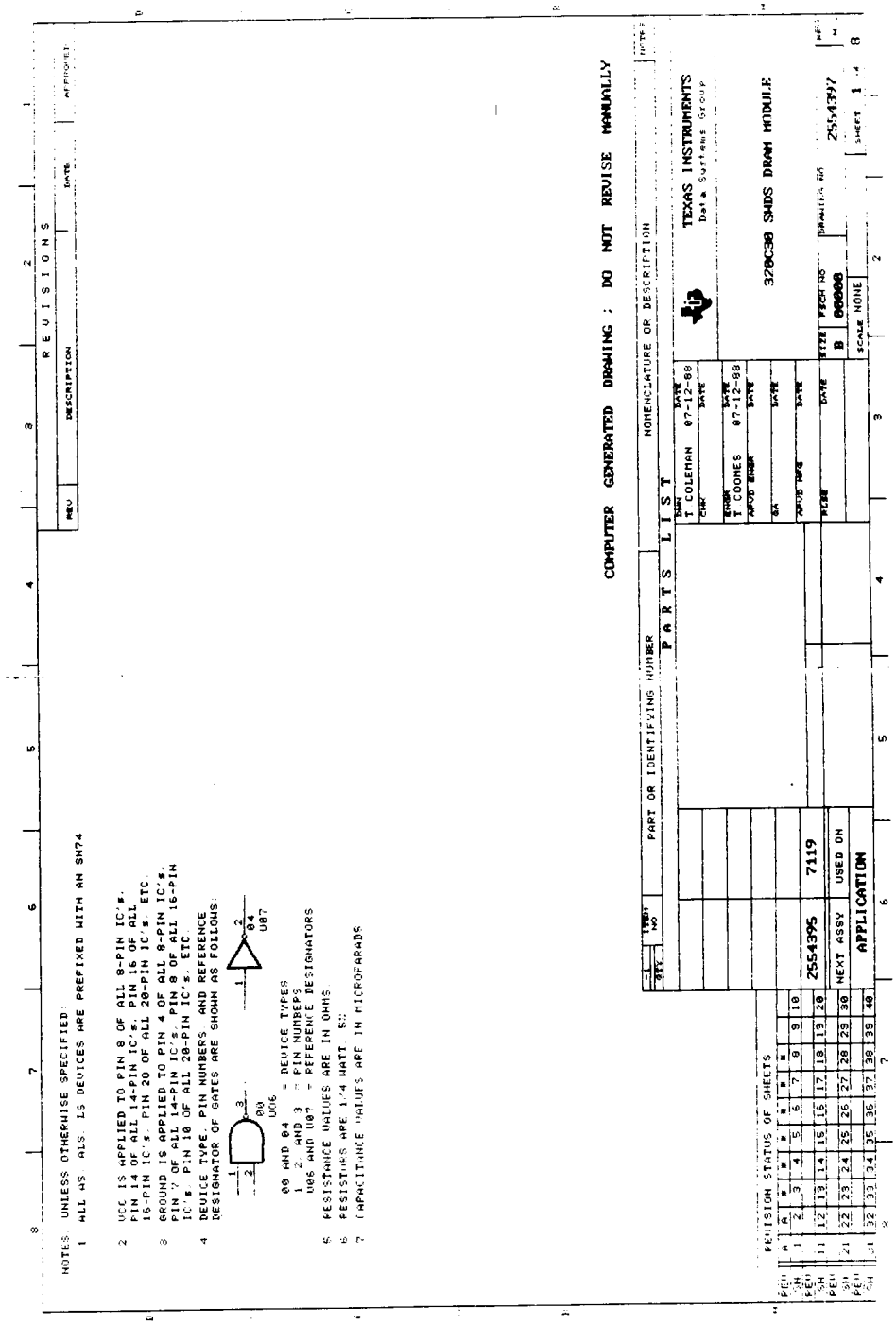
TEXAS INSTRUMENTS	DSN N Seshan DATE 06-14-88	SIZE B	FSCM NO 00000	DRAWING NO 2554377	REV A
	TISSUE DATE	SCALE: NONE		SHEET 09	

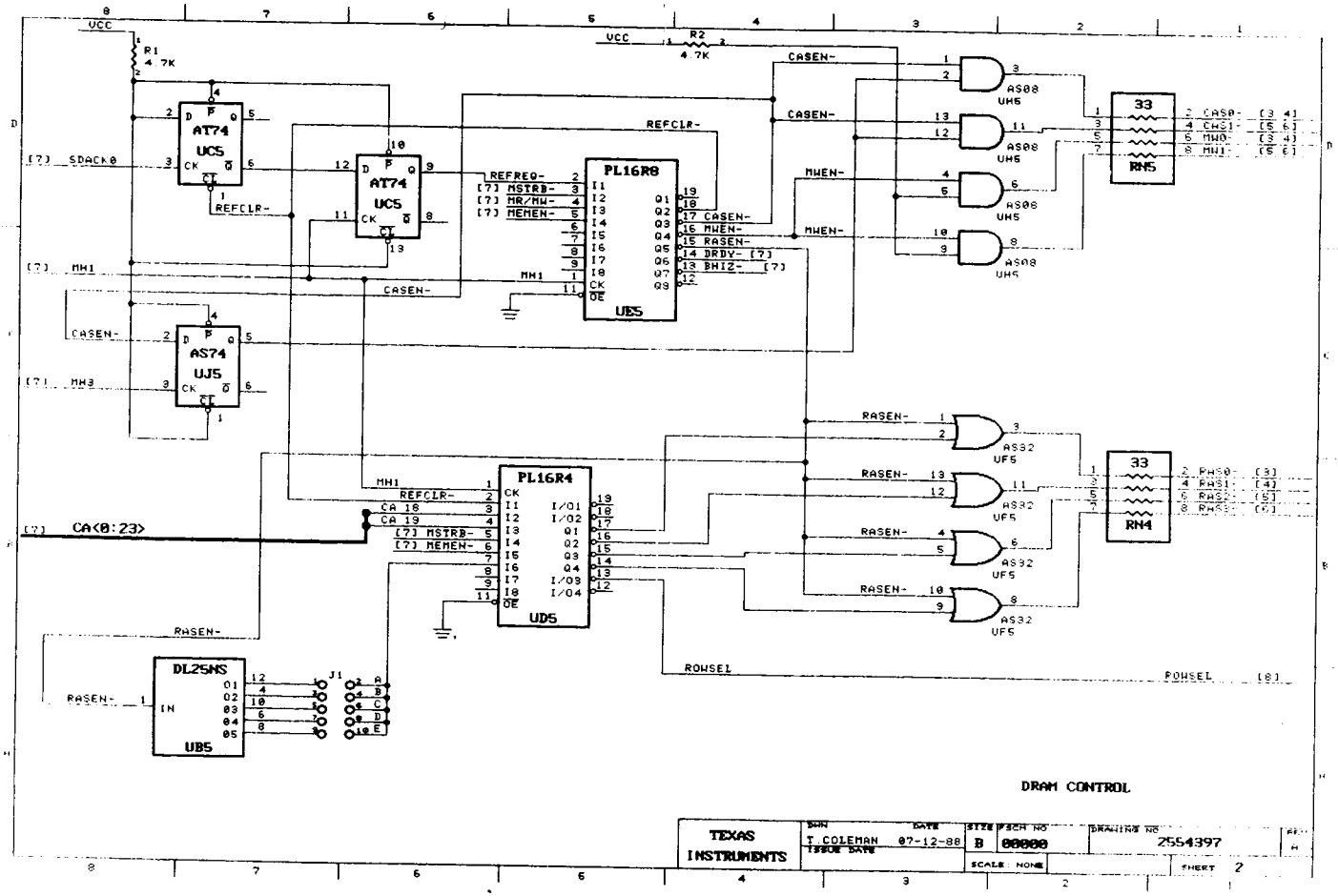

[7] SRAMCE-
[7] SRAMW-
CY7C164-25
UJ2
UG2
UE2
UC2
UH2
UF2
UD2
UB2
I/O0
I/O1
I/O2
I/O3
[5,8] CD<0:31>
[7,11] BA<0:13>
TEXAS INSTRUMENTS
N Seshan 06-14-88
B
00000
2554377
SCALE: NONE
SHEET 10

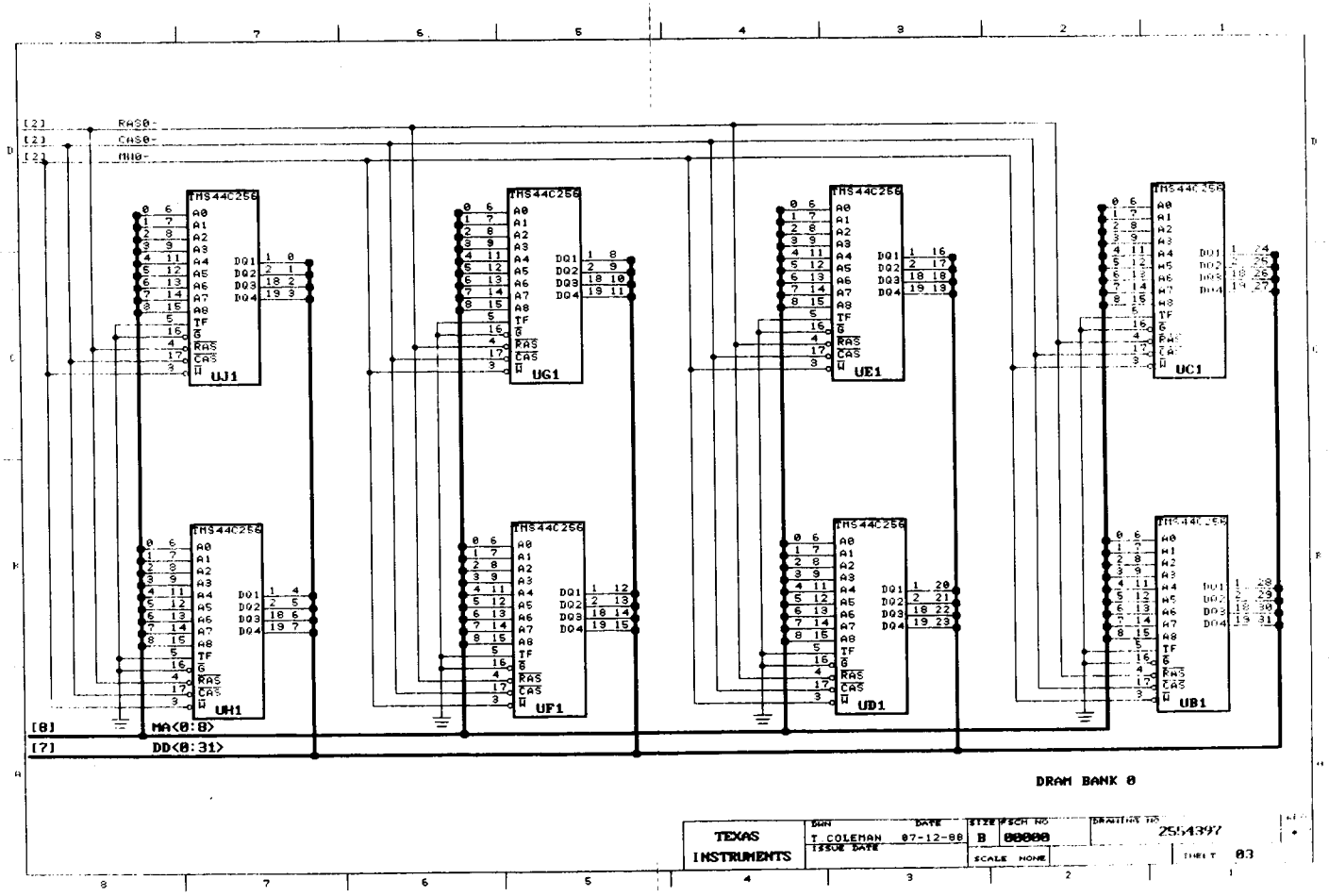

[5] CR/W- DR/W- [9]
AS32 UC5
[5] CSTRB- DSTRB- [9]
AS32 UC5
[3,5,7] CH1 DH1 [9]
AS32 UC5
[5,6,7] CH3 DH3 [3,9]
AS32 UC5
PAL16R4-10
UC3
BUSEN-
EPROM-
CSTRB-
CR/W-
BHIZ- [6,9]
BR- [8]
BW [8]
PRDY-
REPROMCS-
VCC
R2 20K
R1 33
[9] DRDY-
PRDY-
SRAM-
CRDY- [9]
AS11 UC4
SRAM-
CSTRB-
RSRAMCE- [7]
AS32 UB3
[5,7,9]CA<0:23>
PAL16L8-10
UB4
MEMEN- [9]
SRAM-
EPROM-
STATUS7
BUSEN-
MSWAP [9]
EPROMCS-
BA<0:13> [7,10]
CY7C291B6
UH1
UF1
UD1
UB1
[8,9] DD<0:31>
TEXAS INSTRUMENTS
N. Seshan
10-03-88
B
2554377
SCALE: NONE
SHEET 11

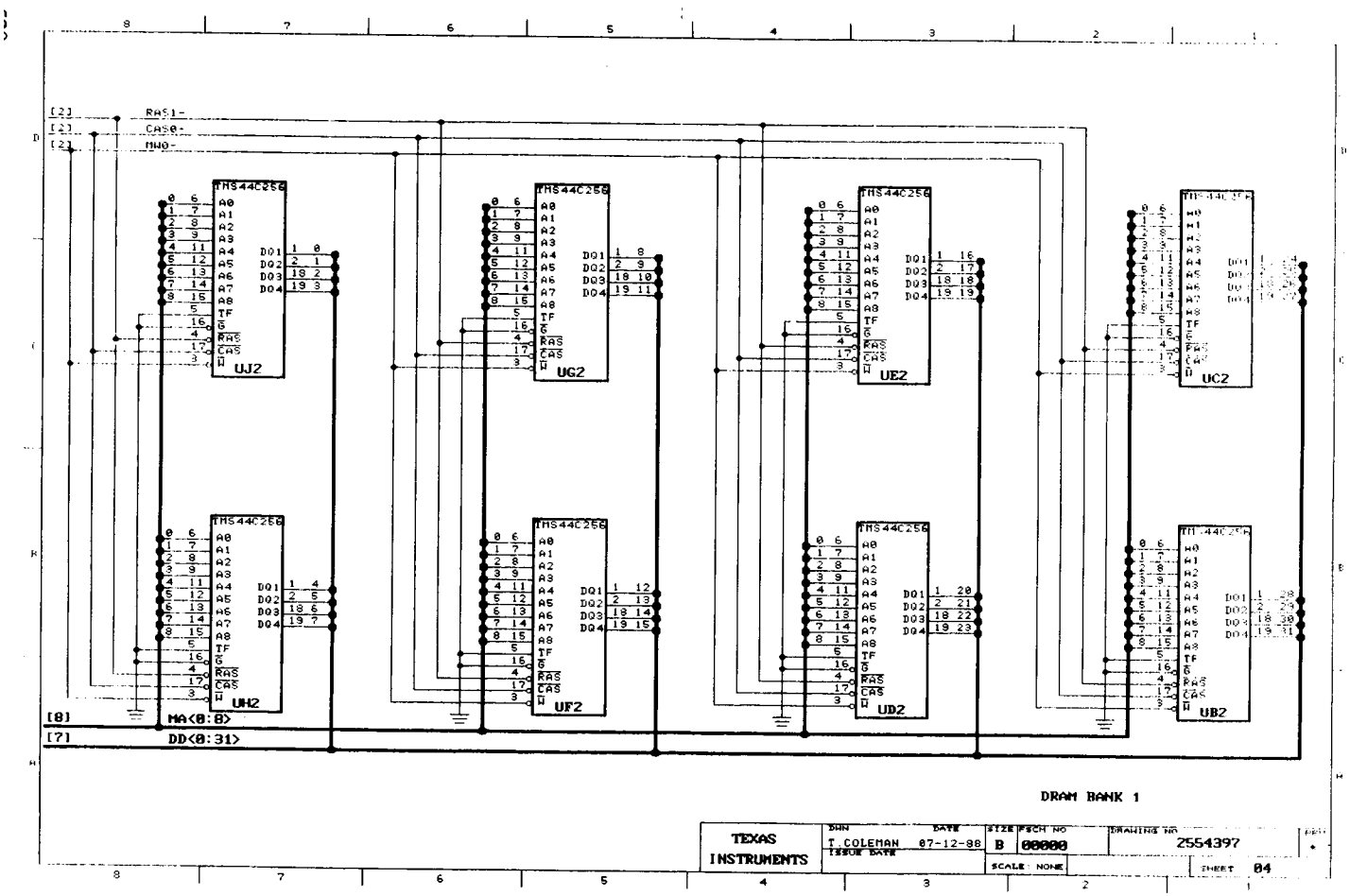

VCC
.10 C1
.10 C2
.10 C3
.10 C4
.10 C5
.10 C6
.10 C7
.10 C8
.10 C9
.10 C10
.10 C11
.10 C12
.10 C13
.10 C14
.10 C15
VCC
.10 C16
.10 C17
.10 C18
.10 C19
.10 C20
.10 C21
.10 C22
.10 C23
.10 C24
.10 C25
.10 C26
.10 C27
.10 C28
.10 C29
.10 C30
VCC
.10 C31
.10 C32
.10 C33
.10 C34
.10 C35
.10 C36
.10 C37
.10 C38
.10 C39
.10 C40
.10 C41
.10 C42
.10 C43
.10 C44
.10 C45
VCC
.10 C46
.10 C47
.10 C48
.10 C49
.10 C50
.10 C51
.10 C52
.10 C53
.10 C54
.10 C55
.10 C56
.10 C57
.10 C58
.10 C59
.10 C60
.10 C61
VCC
4.7 CT1
4.7 CT2
4.7 CT3
4.7 CT4
4.7 CT5
4.7 CT6
TEXAS INSTRUMENTS
DRN
N. Seshan
DATE
06-16-98
ISSUE DATE
SIZE
B
FSCH NO
88888
DRAWING NO
2554377
REV
A
SCALE: NONE
SHEET
12

Appendix C2. TMS320C30 SWDS DRAM Module Schematics



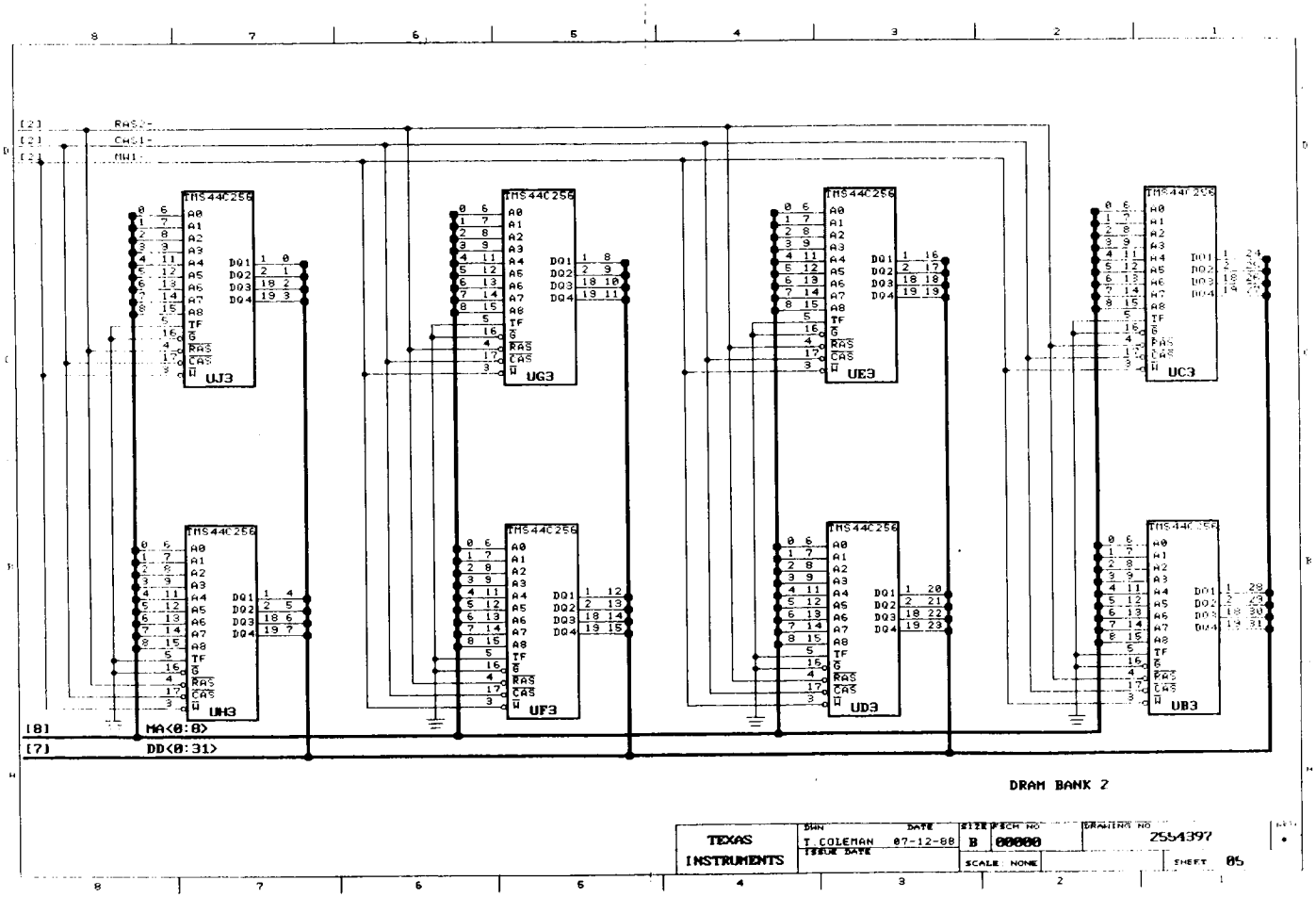

VCC
R1
4.7K
R2
4.7K
AT74
UC5
REFCLR-
[7] SDACK0
REFREQ-
[7] MSTRB-
[7] MR/MW-
[7] MEMEN-
PL16R8
UB5
CASEN-
MWEN-
RASEN-
AS08
UH5
33
RN5
CAS0-
CAS1-
MW0-
MW1-
MH1
[7] MH3
AS74
UJ5
AS32
UF5
RN4
RAS0-
RAS1-
RAS2-
RAS3-
PL16R4
UD5
CA 18
CA 19
[7] CA<0:23>
DL25NS
UB5
J1
ROWSEL
[8]
DRAM CONTROL
TEXAS INSTRUMENTS
T COLEMAN
07-12-88
B
00000
2554397
SCALE: NONE
SHEET 2

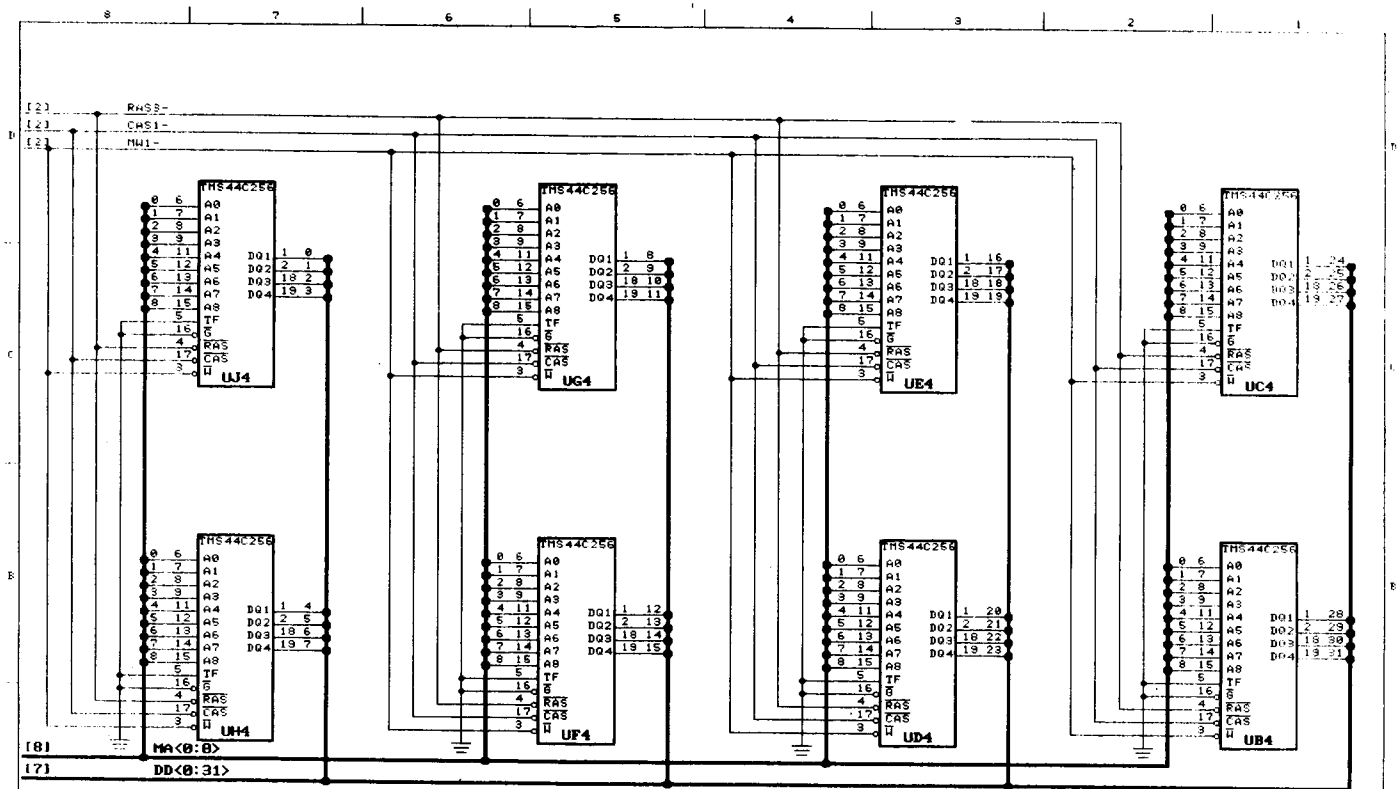

[2] RAS0-
[2] CAS0-
[2] MW0-
THS44C256
UJ1
UG1
UE1
UC1
UH1
UF1
UD1
UB1
[8] MA<0:8>
[7] DD<0:31>
DRAM BANK 0
TEXAS INSTRUMENTS
T COLEMAN 07-12-88
B
2554397
SCALE NONE
SHEET 03

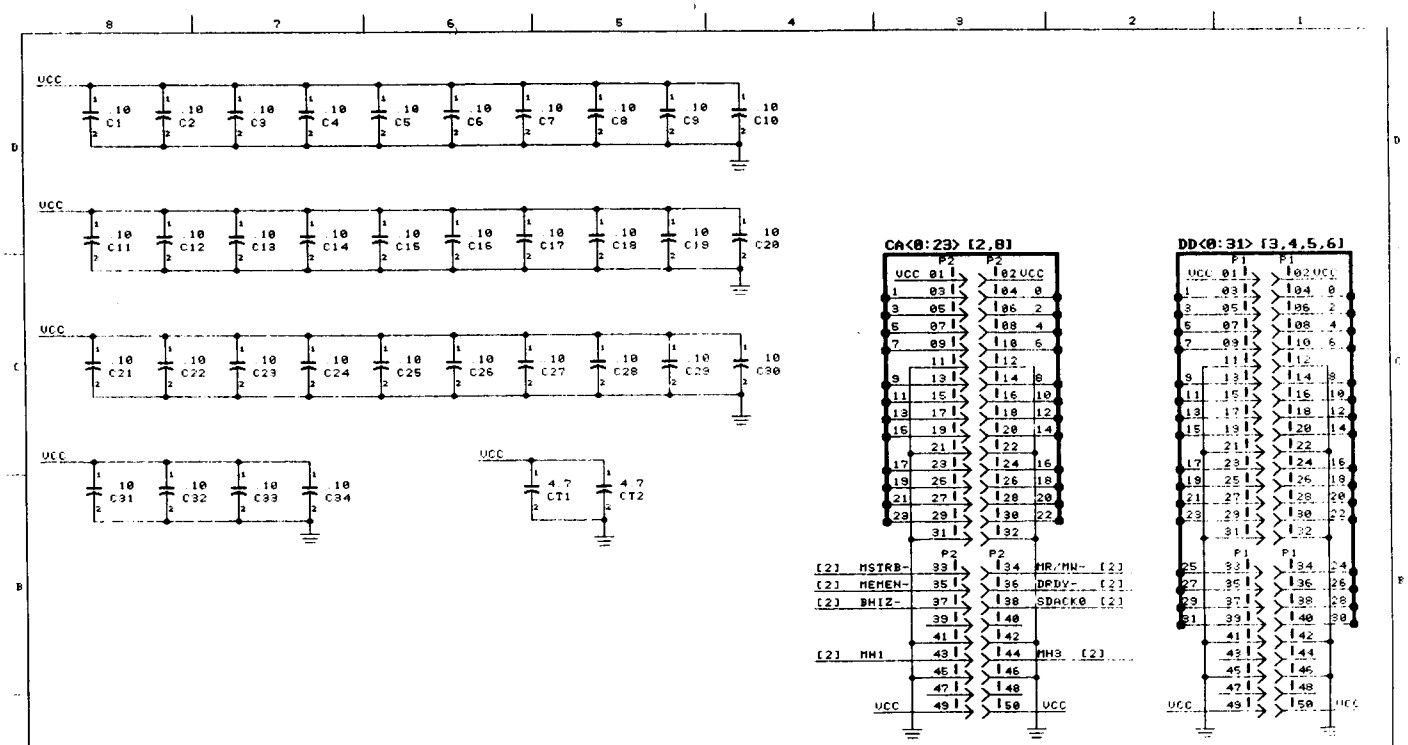


DRAM BANK 1

TEXAS INSTRUMENTS	DRN	DATE	SIZE	FSCH NO	DRAWING NO
	T. COLEMAN	07-12-88	B	00000	2554397
	ISSUE DATE		SCALE: NONE		SHEET 04

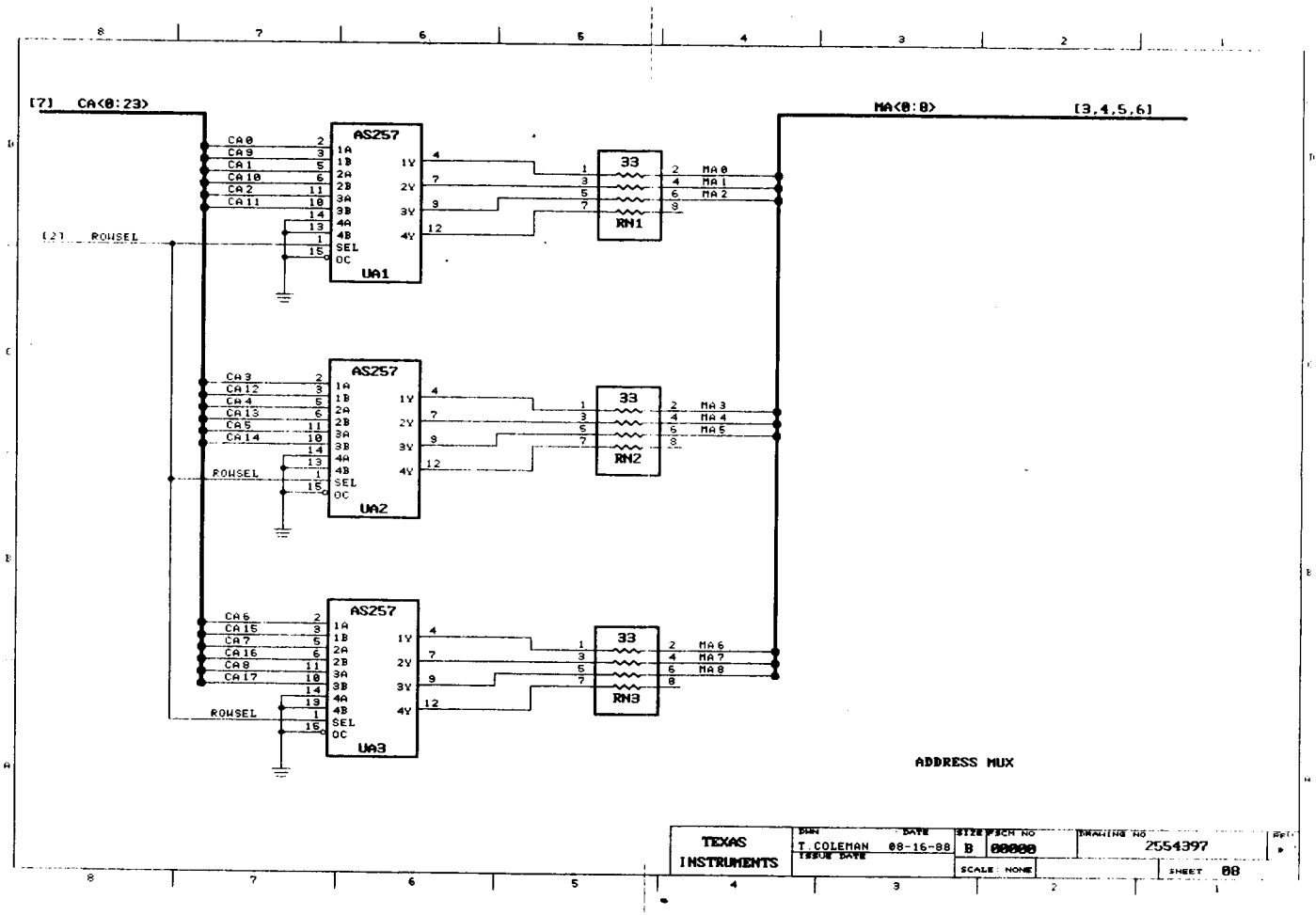

[2] RAS2
[2] CAS1
[2] WE1
TMS44C256
UJ3
UG3
UE3
UC3
UH3
UF3
UD3
UB3
[8] MA<0:8>
[7] DD<0:31>
DRAM BANK 2
TEXAS INSTRUMENTS
DWN T COLEMAN
DATE 07-12-88
ISSUE DATE
SIZE B
SCH NO 00000
DRAWING NO 2554397
SCALE NONE
SHEET 05


[2] RHS3-
[2] CHS1-
[2] MW1-
THS44C256
UJ4
UG4
UE4
UC4
UH4
UF4
UD4
UB4
[8] MA<0:8>
[7] DD<0:31>
DRAM BANK 3
TEXAS INSTRUMENTS
T COLEMAN 07-12-88
SIZE B
00000
2554397
SCALE: NONE
SHEET 06



EXPANSION HEADERS

TEXAS INSTRUMENTS	DRN T. COLEMAN DATE 07-12-88	SIZE B	DWG NO 2554397	REV	
	ISSUE DATE	SCALE: NONE	SHEET 07		


[7] CA<0:23>
MA<0:8>
[3,4,5,6]
[2] ROWSEL
AS257
UA1
UA2
UA3
33
RN1
RN2
RN3
CA0
CA9
CA1
CA10
CA2
CA11
CA3
CA12
CA4
CA13
CA5
CA14
CA6
CA15
CA7
CA16
CA8
CA17
MA0
MA1
MA2
MA3
MA4
MA5
MA6
MA7
MA8
ROWSEL
ADDRESS MUX
TEXAS INSTRUMENTS
T COLEMAN
08-16-88
B
2554397
SCALE: NONE
SHEET 00