

02/03/00  
 JCS17 U.S. PTO

2-4-00

APPROV PTO/SB/16 (2-98)

Approved for use through 01/31/2001. OMB 0651-0037  
 Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

PROVISIONAL APPLICATION COVER SHEET


This is a request for filing a PROVISIONAL APPLICATION under 37 C.F.R. §1.53(c).

Docket Number		8011-7	Type a plus sign (+) inside this box -	+
INVENTOR(S)/APPLICANT(S)				
LAST NAME	FIRST NAME	MIDDLE INITIAL	RESIDENCE (CITY AND EITHER STATE OR FOREIGN COUNTRY)	
Fallon	James	J.	11 Wamous Close, Armonk, New York 10504	
Buck	John		362 Christopher Street, Oceanside, New York 11572	
Pickel	Paul	F.	225 Stewart Avenue, Bethpage, New York 11714	
McErlain	Stephen	J.	325 East 17 <sup>th</sup> Street, New York, New York 10003	
Wolf-Sonkin	Yury		160 Kings Point Road, Kings Point, New York 11024	
TITLE OF THE INVENTION (280 characters max)				
DATA STORAGE AND RETRIEVAL ACCELERATOR				
CORRESPONDENCE ADDRESS				
F. CHAU & ASSOCIATES, LLP 1900 Hempstead Turnpike, Suite 501				
STATE	New York	ZIP CODE	11554	COUNTRY USA
ENCLOSED APPLICATION PARTS (check all that apply)				
<input checked="" type="checkbox"/>	Specification & Drawings	Number of Pages	<u>108</u>	<input checked="" type="checkbox"/> Small Entity Statement
Other (specify) _____				
METHOD OF PAYMENT (check one)				
<input checked="" type="checkbox"/>	A check in the amount of \$ <u>75.00</u> to cover the filing fee is enclosed.		PROVISIONAL FILING FEE AMOUNT(S)	
<input type="checkbox"/>	Charge fee to Deposit Account Number 50-0679. TWO COPIES OF THIS SHEET ARE ENCLOSED.			
<input checked="" type="checkbox"/>	Please charge any deficiency as well as any other fee(s) which may become due under 37 C.F.R. § 1.16 and/or 1.17 at any time during the pendency of this application, or credit any overpayment of such fee(s) to Deposit Account No. 50-0679. TWO (2) COPIES OF THIS SHEET ARE ENCLOSED.		\$75.00	

CERTIFICATION UNDER 37 C.F.R. § 1.10

I hereby certify that this Provisional Application Cover Sheet and the documents referred to as enclosed therein are being deposited with the United States Postal Service on this date February 3, 2000 in an envelope as "Express Mail Post Office to Addressee" Mail Label Number EL433927955US addressed to: Commissioner of Patents and Trademarks, Box Provisional Application, Washington, D.C. 20231.

Frank V. DeRosa  
 (Type or print name of person mailing paper)

  
 (Signature of person mailing paper)

The invention was made by an agency of the United States Government or under a contract with an agency of the United States Government.

No.  
 Yes, the name of the U.S. Government agency and the Government contract number are: \_\_\_\_\_  
\_\_\_\_\_

Respectfully submitted,

SIGNATURE *Frank V. DeRosa*

Date 2/3/00

TYPED or PRINTED NAME Frank V. DeRosa

Registration No. 43,584  
(if appropriate)

Additional inventors are being named on separately numbered sheets attached hereto.

RECEIVED FEBRUARY 3 2000

Applicant or Patentee: Realtime Data, LLC

Serial or Patent No.: 0 / Unassigned

Filed or Issued: Concurrently herewith

For: DATA STORAGE AND RETRIEVAL ACCELERATOR

**VERIFIED STATEMENT CLAIMING SMALL ENTITY  
STATUS (37 CFR 1.9(f) and 1.27(c)) - SMALL BUSINESS CONCERN**

I hereby declare that I am

- the owner of the small business concern identified below:
- an official of the small business concern empowered to act on behalf of the concern identified below:

NAME OF SMALL BUSINESS CONCERN Realtime Data, LLC

ADDRESS OF SMALL BUSINESS CONCERN 206 East 63rd Street, New York, New York 10021

I hereby declare that the above identified small business concern qualifies as a small business concern as defined in 13 CFR 121.12, and reproduced in 37 CFR 1.9(d), for purposes of paying reduced fees to the United States Patent and Trademark Office under Section 41(a) and (b) of Title 35, United States Code, in that the number of employees of the concern, including those of its affiliates, does not exceed 500 persons. For purposes of this statement, (1) the number of employees of the business concern is the average over the previous fiscal year of the concern of the persons employed on a full-time, part-time or temporary basis during each of the pay periods of the fiscal year, and (2) concerns are affiliates of each other when either, directly or indirectly, one concern controls or has the power to control the other, or a third-party or parties controls or has the power to control both.

I hereby declare that rights under contract or law have been conveyed to, and remain with, the small business concern identified above, with regard to the invention described in

- the specification filed herewith.
- application serial no. 0 / \_\_\_\_\_, filed \_\_\_\_\_.
- patent no. \_\_\_\_\_, issued \_\_\_\_\_.

If the rights held by the above identified small business concern are not exclusive, each individual, concern or organization having rights in the invention is listed below\* and no rights to the invention are held by any person, other than the inventor, who would not qualify as an independent inventor under 37 CFR 1.9(c) if that person made the invention, or by any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e).

\*NOTE: Separate verified statements are required from each named person, concern or organization having rights to the invention averring to their status as small entities. (37 CFR 1.27).

(Small Entity/Small Business [7-4]-page 1 of 2)

Each such person, concern or organization having any rights in the invention is listed below:

- No such person, concern, or organization exists.  
 Each such person, concern or organization is listed below.

FULL NAME \_\_\_\_\_

ADDRESS \_\_\_\_\_  
 Individual  Small Business Concern  Nonprofit Organization

FULL NAME \_\_\_\_\_

ADDRESS \_\_\_\_\_  
 Individual  Small Business Concern  Nonprofit Organization

I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small business entity is no longer appropriate. (37 CFR 1.28(b)).

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

NAME OF PERSON SIGNING James J. Fallon

TITLE OF PERSON IF OTHER THAN OWNER Chairman

ADDRESS OF PERSON SIGNING 11 Wamous Close, Armonk, New York 10504

SIGNATURE 

Date 2/02/00

00000044700000



## PREFACE

This manual describes the principles of operation, performance specifications, and detailed design of the Realtime Data, LLC UltraDMA Data Storage and Retrieval Accelerator (hereinafter affectionately referred to as the DSRA). This document begins with a brief introduction to fundamental performance limitations of current disk drives and the dramatic benefits provided by our DSRA. This is followed by a detailed DSRA performance specification. Next, a system overview is presented from a logical (not necessarily physical) functional partitioning with an emphasis on intrafunction dataflow and system level dynamic bandwidth allocation. A detailed block diagram is then presented and the function of each component is discussed in detail. Address maps are presented from the perspective of the onboard digital signal processor, disk controller, and PCI controller along with a view from the host PCI Bus. Available Interrupts and their allocation are discussed. Separate sections describe the internal architecture of the onboard field programmable gate array, programming the field programmable gate array, and the DSRA's reset and initialization methodology. Finally a software guide to the DSRA's command protocol is covered with a detailed specification of each DSRA command. Appendices list reference information including board component placement, a numerical listing of jumpers, external connectors for the UltraDMA disk interface and the PCI Bus, detailed schematics and a fully cross referenced parts list.

The DSRA is the represents Realtime's first product in our comprehensive family of data storage controllers, network data storage, and data centric secure networking. Our technology represents the next logical step in the evolution of high performance data storage and completely secure high bandwidth data transmission. Employing industry standard interfaces and protocols that seamlessly integrate with existing media devices, our product offers a many-fold increase in data storage density, access speed, and security. This approach overcomes the traditional bottlenecks associated with local and network disk data accesses. In order to achieve this level of performance our proprietary data compression and encryption engine reads one byte of data stored on disk and decodes this information into multiple bytes of information for the computer. By implementing this process in a combination of dedicated hardware and an ultra high speed digital signal processor the translation takes place in "real-time". Thus, rather than the traditional delays normally associated with software data compression, our hardware approach creates a many-fold performance improvement. Our technology is designed to be *Scalable* through each successive computer generation; *Adaptable* to serve multiple functions concurrently, increasing performance and enhancing value; *Insertable* to seamlessly integrate into existing marketplaces – modifications to existing standards are not required. Our *Value Proposition* is Product Excellence through Innovation. Our technology creates the state-of-the-art; is easy to install and use, and is flawlessly reliable.

## NOTATIONAL CONVENTIONS

This manual utilizes the following conventions and nomenclature:

### Register Diagrams

Register diagrams are presented as follows:

#### *External Interrupt Polarity Register*

31	4	3	2	1	0
Reserved		XIP7	XIP6	XIP5	XIP4
R, +0		R, +0	RW, +0	RW, +0	RW, +0

All register diagrams, unless otherwise noted, utilize the following notational conventions:

Each rectangle represents a logically related group of bits called a bit field.

Mnemonics for each field name is given within the rectangle.

Numbers directly above the bit field represent starting and ending bit locations (inclusive).

Properties are listed directly below each bit field.

R = Readable by the DSP's CPU

W = Writeable by the DSP's CPU

+x = Value undefined after DSP reset

+0 = Value is 0 after DSP reset

+1 = Value is 1 after DSP reset

C = Clearable by the DSP's CPU

## Software Notation

Program Listings are in courier font

```
LDW .D1    *A0,A1
ADD  .L1    A1,A2,A3
```

All data is presented and utilized in little endian notation.

In syntax descriptions, the command is in **bold face**, and parameters are in *italics*. Portions of a syntax that are in **bold** should be entered as shown; portions of a syntax that are in *italics* describe the type of information that should be entered. Here is an example of a command syntax:

### READ DISK DATA

31	16	15	8	7	0	
<i>Command Packet Number</i> <i>0000h to FFFFh</i>		<i>Command Type</i> <i>00h</i>		<i>Command Parameters (00h)</i>		00h
<i>Starting Block Address (Least Significant Word)</i>						04h
<i>Starting Block Address (Most Significant Word)</i>						08h
<i>Number of Blocks (Least Significant Word)</i>						0Ch
<i>Number of Blocks (Most Significant Word)</i>						10h
<i>Destination Address (Least Significant Word)</i>						14h
<i>Destination Address (Most Significant Word)</i>						18h
<i>[Checksum]</i>				<i>Reserved</i>		1Ch

Square brackets ( [ and ] ) identify an optional parameter. If you use an optional parameter, you specify the information within the brackets; you do not enter the brackets themselves. When there is a choice amongst multiple items they are separated by a |, for example a | b is a selection of either a or b, but not both.

4



## Hardware Notation

DSRA busses and integrated circuit pins often are represented in groups. Device pin group notation consists of the pin name followed by brackets containing the range of pins included in the group. A colon separates the numbers in the range. For example, ED[31:0] represents the 32 bit DSP external data bus.

## Caution Statements

A caution statement describes a situation that could potentially damage the DSRA, host computer, or software.

**CAUTION**

## Warning Statements

A warning statement describes a situation that could potentially cause you injury or death.

**WARNING**

Caution and Warning Statements are provided for your protection. Please read each caution and warning carefully.

# REALTIME DATA, LLC

## UltraDMA/66 Data Storage & Retrieval Accelerator

### TABLE OF CONTENTS

Section	Title	Page
1.0	INTRODUCTION.....	8
2.0	PERFORMANCE SPECIFICATION.....	10
3.0	SYSTEM OVERVIEW.....	11
4.0	DETAILED BLOCK DIAGRAM.....	14
4.1	Master Oscillator.....	14
4.2	Digital Signal Processor.....	14
4.3	OTP EPROM.....	14
4.4	SDRAM.....	16
5.0	MEMORY MAPPING.....	17
6.0	MEMORY BANDWIDTH ALLOCATION.....	19
7.0	INTERRUPTS.....	22
7.1	Interrupt Multiplexer Register.....	24
7.2	External Interrupt Polarity Register.....	24
8.0	ATA AND ULTRA DMA/66 DISK INTERFACE.....	25
8.1	Operational Registers.....	25
8.2	Sample ATA Command Sequence.....	27
8.3	State Diagrams.....	29
8.4	Ultra DMA66 Disk Interface.....	32
8.4.1	Write Transaction.....	34
8.4.2	Read Transaction.....	35
8.5	Timing Diagrams.....	36
8.5.1	Read Timing.....	36
8.5.2	Write Timing.....	39
9.0	XILINX ARCHITECTURE.....	42
9.1	Block Diagram.....	42
9.2	Register Map.....	42

6



## 1.0 INTRODUCTION

### *The Challenge*

Modern Personal Computers have steadily increased in performance through advances in processor, memory and data communications bandwidth. In stark contrast, magnetic hard disk devices have not kept pace with this revolution. In fact, even the highest performance disk storage devices severely limit the capability of consumer, entertainment, office, and workstation personal computers (PCs) for all disk intensive operations. Hard disk storage devices house the computer's operating system, application programs, and data. Rapid access to this information, or bandwidth, is a critical factor in overall computer performance. In normal operation, the computer's operating system issues a request for data. This data might be a new program to execute or data for a program currently running. The disk controller commands the disk to "seek" the requisite data. Three factors then limit disk performance:

1. the time to position the read/write heads over the appropriate disk cylinder (seek time);
2. the time for the disk to rotate under the read/write head to the beginning of data (rotational latency); and
3. the time to read each byte of data from the disk (read/write bandwidth).

Items one and two are typically incurred only once per disk access while read/write bandwidth limitations are encountered for each byte of data requested. Currently the fastest available disk drives (10,200rpm) offer only a 18 Megabyte (MB/sec) per second data access rate. This is in stark contrast to PC's input / output (I/O) bus capability of 264 MB/sec. In short, the best available disks are 15 times too slow to match a PC's capabilities. A typical consumer hard disk drive performance utilizing UltraDMA/66 is often off by a factor of 100 or more.

Industry has generated several inadequate and costly solutions to address disk bandwidth limitations. Emergent high performance disk interface standards such as the Small Computer Systems Interface (SCSI-3) and Fibre Channel offer the ability to communicate at improved data transfer rates. These standards do not solve the hard disk's inherent physical media restriction of 17 MB/sec. Higher disk access data rates have only been achieved simultaneously reading or writing multiple disk drives. This is a costly and complex approach.

### *The Solution*

Realtime's family of storage controllers and network data storage represents the next logical step in the evolution of high performance data storage. Employing industry standard interfaces and protocols that seamlessly integrate with existing media devices, our product offers a three-fold increase in data storage density and access speed. This approach overcomes the traditional bottlenecks associated with local and network disk data accesses.

Our proprietary data compression engine reads one byte of data stored on disk and decodes this information into three bytes of information for the computer. By implementing this process in a combination of dedicated hardware and ultra high speed digital signal processors, the translation takes place in "real-time". Thus, rather than the traditional delays normally associated with software data compression, our hardware approach creates a three-fold performance increase.

Our first product, the Realtime Data Storage and Retrieval Accelerator (affectionately referred to as the DSRA) represents the state of the art in UltraDMA Disk Storage Adapters. Applying new discoveries in information theory, combined with artificial intelligence based application and data management algorithms, Realtime's technology dramatically accelerates computer performance and significantly increases hard disk data storage capacity. For personal computers running standard Microsoft Windows® based business application software:

*Disk storage capacity is typically increased by a factor of 3:1 (for example a 20 gigabyte hard drive effectively becomes a 60 gigabyte hard drive).*

*Computer boot-up time (turn-on and operating system load) is decreased by a factor of two.*

*Application software (for example Microsoft Office Programs) load twice as fast.*

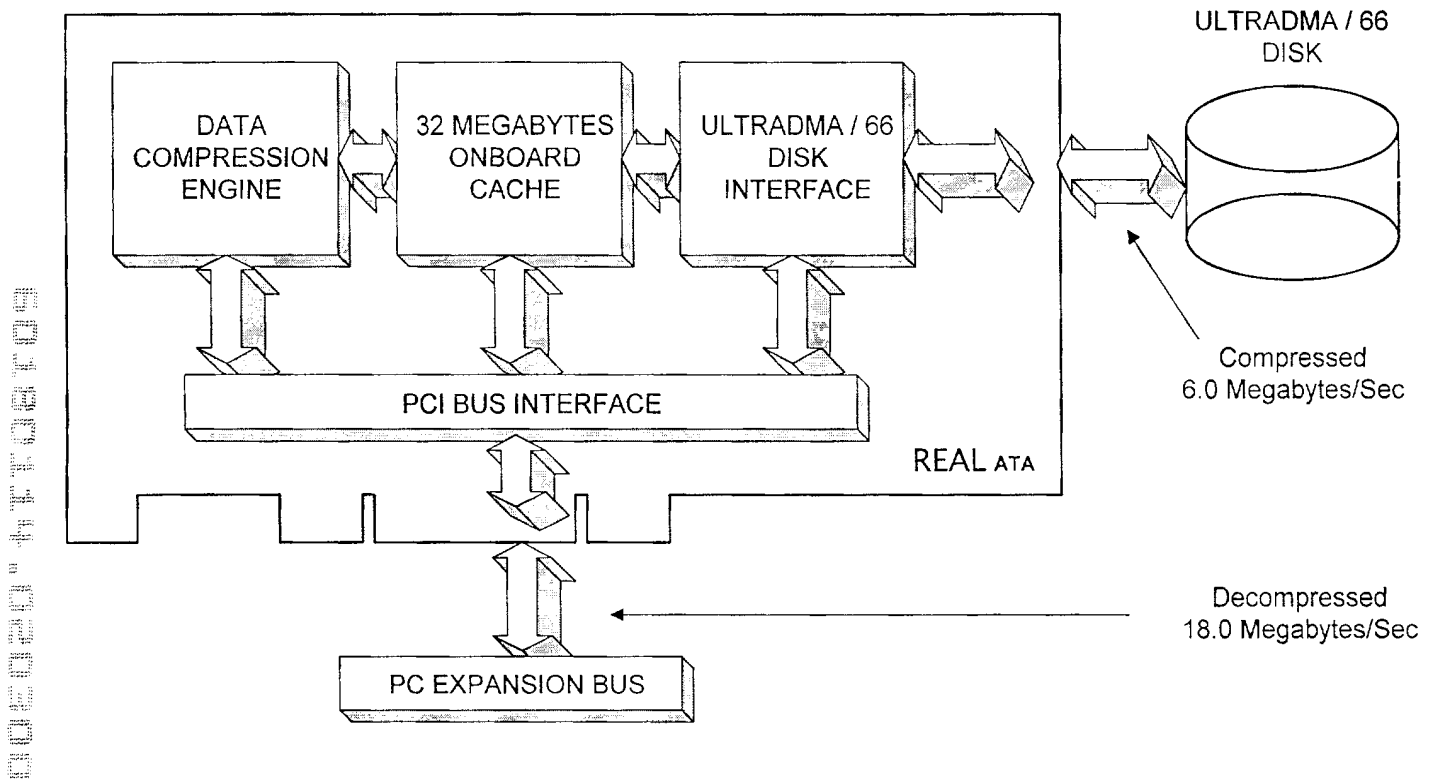
*User data storage and retrieval is increased (on average) by a factor of 3:1.*

## 2.0 PERFORMANCE SPECIFICATION

Storage Device Interface	UltraDMA/66 Implemented in Dynamically Reprogrammable Custom FPGA Full DMA to Local DSP & PCI Busses (\$ gigabyte burst capability)
PCI Bus Interface	PCI Bus Master/Slave Interface, 32 Bit Data, 64 Bit Address Zero Bus Wait States Disk Controller PCI Bus Master Data Transfers Zero Bus Wait State Multi Command Slave Mode Queue Implemented in Dynamically Reprogrammable Custom FPGA Full DMA Capability
Command Set	Fully Transparent PCI Master Data Storage Controller High Level Command Set Facilitates I/O Channel Offload of Host Processor Full Self Diagnostics & SMART Diagnostic Support
OnBoard Memory	32 Megabytes
Data Compression	Embedded 1.2 Billion Instruction per second, 32 Bit Digital Signal Processor Proprietary Information Processing & Artificial Intelligence Data Storage Management
Software Support	Windows 98 & Windows NT 5.0 Utilities for Installation and Data Storage Management
Internal Interface Connectors	One 32 Bit PCI Bus Board Edge Connector One UltraDMA/66 40 Pin Header
External Interface Connectors	None
Dimensions	6.875"(l), 4.187"(w), 0.500"(h)
Power	Less than 10.0 Watts (max)
Mass	6.0 Ounces
Temperature	Operating: 0 to +50°C, Max Slew 15°C per hour NonOperating: -40 to +70°C Max Slew 20°C per hour
Humidity	Operating 10 to 85% Max Slew 10% per hour NonOperating: 5 to 95% Max Slew 10% per hour
MTBF	~500,000 hours @ 40°C

### 3.0 SYSTEM OVERVIEW

Figure 2-1, *UltraDMA Data Storage and Retrieval Accelerator (DSRA) Architecture*, highlights the data internal flows and external interfaces for our storage controller.



The data flow architecture of the DSRA provides maximum system bandwidth by allowing simultaneous data transfers between:

- disk and onboard cache memory,
- DCE and onboard memory,
- PCI Bus and onboard memory.

The DCE, disk interface, and PCI Bus controller have full DMA capability and are able to transfer data without interrupting or interfering with any other ongoing processes. An integral round robin arbitration bandwidth allocation controller allows the disk controller, DCE, and PCI Bus to access the onboard cache with a bandwidth proportional to the overall bandwidth of the respective interface or processing element. Arbitration happens transparently and does not introduce latency in memory accesses. Bandwidth division is performed with a high degree of granularity to minimize the size of requisite onboard buffers to synchronize data from the disk and PCI interfaces. Details of the PCI and Disk Interface are contained within the Xilinx FPGA Design Section contained within this document.

#### Disk Read / Write Data Transfers

When data is read from disk by the host computer data flows from the disk through the DSRA to the host computer. Data is normally stored in one of several proprietary compression formats on the disk. Data blocks are pre-specified in length and are typically handled in fractional or whole equivalents of tracks, e.g. 1/2 track, whole track, multiple tracks... Since tracks may contain variable numbers of sectors this is not always possible. In order to read disk data a DMA transfer is setup from the disk interface to the onboard cache memory. The disk interface has integral DMA control to allow transfers from disk to directly to the onboard 32 megabytes of cache without DCE intervention. It should be noted that the DCE acts as a system level controller and is used to set-up specific registers within both the disk and PCI Interfaces to facilitate DMA transfers to and from DSRA cache memory.

To initiate a transfer from disk to onboard cache, the DMA Transfer is setup via specifying the appropriate command (read disk), the source address (disk logical block number), amount of data to be transferred (number of disk logical blocks), and destination address within the onboard cache memory. Finally the DISKINT# is cleared (if previously set and not cleared) and the command is initiated by writing to the appropriate address space.

Once data has been read from disk and placed into onboard cache memory the DISKINT# is asserted notifying the DCE that requested data is now available in the DSRA's cache memory. Data is then read by the DCE's onboard DMA controller and placed into local memory for subsequent decompression. The decompressed data is then DMA transferred from the DCE's local memory back to the DSRA's 32 megabyte cache memory. Finally, data is DMA transferred via the DSRA's PCI Bus controller from the 32 megabyte cache to the host computer's PCI Bus. In this mode the DSRA acts as a PCI Bus Master. A DSRA PCI DMA transfer is setup via specifying the appropriate command (write to host computer), the source address within the DSRA's cache memory, the quantity of data words to be transferred (transfers are always in 4 byte increments and memory should always be viewed in only 4 byte clusters beginning with address 0x0), and the destination address on the host computer. When a PCI Bus read or write transaction has completed the appropriate PCIRDINT# and PCIWRINT# are asserted to the DCE. Either interrupts is cleared by it's corresponding interrupt service routines through a read or write to the appropriate DCE address.

Similarly, when data is written to disk from the host computer, data flows from the host computer through the DSRA and onto disk. Data is normally received from the host computer in uncompressed (raw) format and must be compressed by the DCE to be stored in one of several proprietary compression formats on the disk. Data blocks from the host are pre-specified in length and are typically handled in blocks that are a fixed multiplier higher than fractional or whole equivalents of tracks, e.g. 1/2 track, whole track, multiple tracks... This multiplier is derived from the expected average compression ratio that is selected when the disk is formatted with the virtual file management system. In order to read host computer data a PCI DMA transfer is setup from the PCI Host Bus to the onboard cache memory. The DSRA's PCI interface controller has integral DMA that allows large block transfers from the host computer directly to the onboard 32 megabytes of cache without DCE intervention. The DSRA's PCI Bus controller acts as a host computer bus master when executing this transfer. Once data has been read from the host and placed into onboard cache memory the data is read by the DCE's onboard DMA controller and placed into local memory for subsequent compression. The compressed data is then DMA transferred from the DCE's local memory back to the DSRA's 32 megabyte cache memory. Finally, data is DMA Transferred via the Disk Controller from the 32 megabyte cache to the disk.

## Commands



Upon host computer power-up or external user reset, the DSRA initializes onboard interfaces prior to release of the external PC expansion Bus from reset. The host computer's processor then requests initial data from the disk to facilitate the computer's boot-up sequence. Disk data is requested over the PCI Expansion Bus via a command packet issued from the host computer. Command packets are eight words long (words are 32 bit in this context). Commands are written from the host computer to the DSRA with the host computer as bus master and the DSRA as slave. The DSRA includes a single PCI Base Address Register (BAR0) for decoding the address of the DSRA command queue.

When a command is received and a PCICMDINT# interrupt is generated to the digital signal processor (DSP) within the data compression engine (DCE). The eight word command is read by the DCE and placed into the command queue that resides either within onboard DCE memory or within the DSRA's 32 megabytes of onboard cache. Because the commands occupy a very small amount of memory the location of the command queue is at the discretion of software and the associated system level performance considerations. Commands may be moved from the PCI bus interface to the command queue by explicit DSP reads and writes or by utilizing programmed DMA from the DSP's Enhanced DMA Controller (EDMA). This second technique may better facilitate system throughput by allowing the EDMA to automatically load commands while the highly pipelined data compression and decompression processing transpires fully undisturbed.

50100114 000300

## 4.0 DETAILED BLOCK DIAGRAM

Figure 2-2 presents a board level overview of our UltraDMA Data Storage and Retrieval Accelerator (DSRA). As shown, the DSRA is designed with a minimum of components for low cost, minimum power consumption, a small printed circuit board footprint, and high reliability.

### 4.1 Master Oscillator

Both the DSP's and DSRA's master clock is generated from an onboard 35 MHz oscillator. The DSP incorporates a by four phase lock loop, yielding a DSP clock of 140MHz. The DSP internally divides this clock by a factor of two down to 70MHz for clocking the DSRA's SDRAM.

### 4.2 Digital Signal Processor

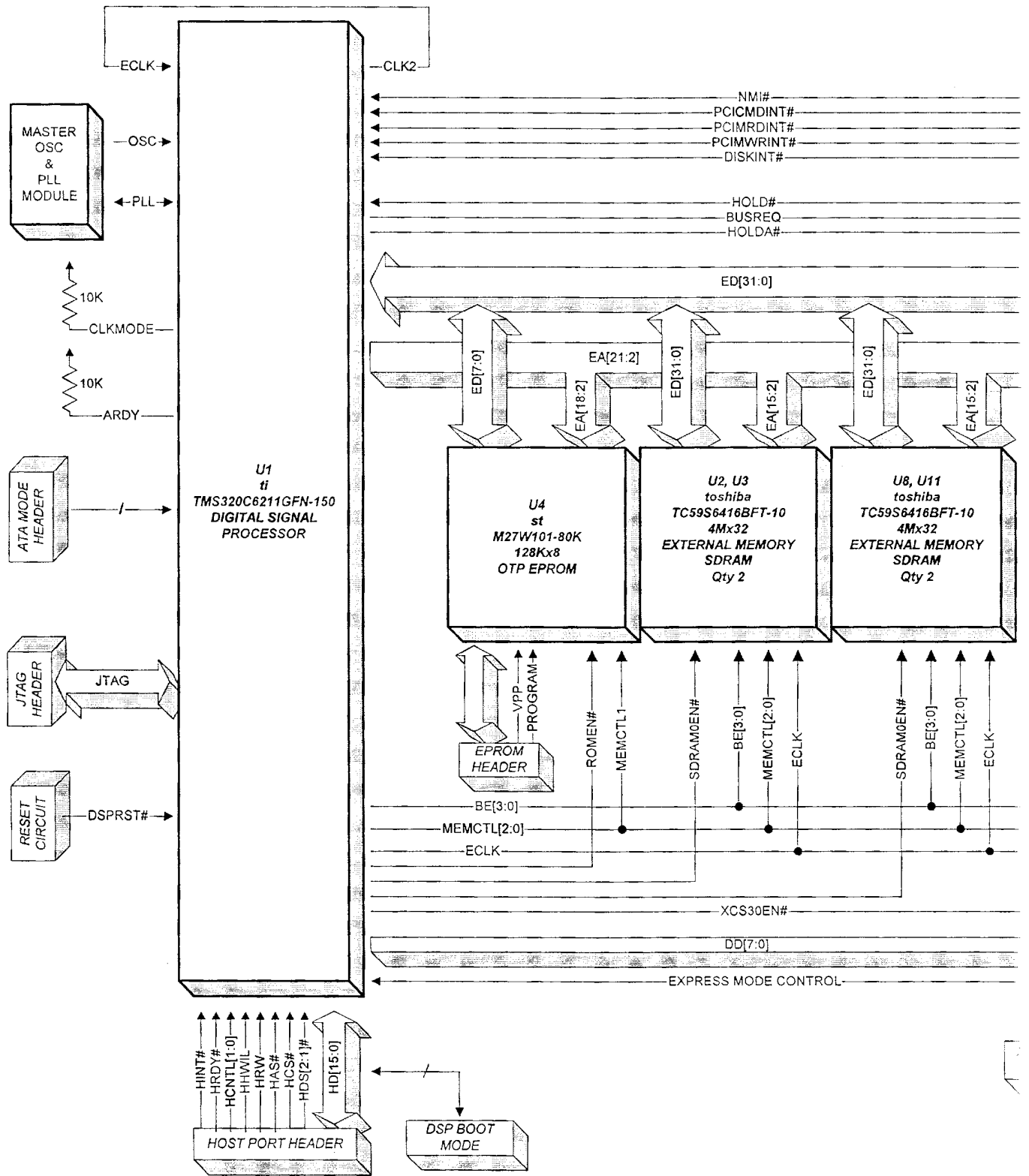
The primary processing element on the DSRA is a Texas Instruments (TI) TMS320C6211GFN-150 Digital Signal Processor (DSP) housed in a Chip Scale 256 pin Ball Grid Array (BGA) package. Utilizing a 5 level metalization 0.18um cmos technology, 3.3volt I/O, and a 1.8volt core, our DSP is capable of up to 1.2 Billion Instructions per Second. Additional features of interest include a highly parallel eight processor single cycle instruction execution, onboard 4K byte L1P Program Cache, 4K L1D Data Cache, and 64K byte Unified L2 Program/Data Cache. A 32 bit External Memory Interface (EMIF), also resident on the DSP, provides for a glueless interface to the two banks of SDRAM along with the non-volatile One Time (Erasable) Programmable Memory (OTP-EPROM). Two Multi-Channel Buffered Serial Ports (McBSPs) and two 32 bit General Purpose Timers are also included within the DSP. The DSRA disables the I/O Capability of these devices and utilizes the I/O ports as general purpose I/O for both programming the FPGA with a strobed eight bit interface and signaling via a Light Emitting Diode (LED). Ancillary DSP features include a 16 bit Host Port Interface and full JTAG emulation capability for development support.

### 4.3 OTP EPROM

Nonvolatile storage is provide by a 128K byte M27W101-80K one time (erasable) programmable memory. This device is decoded at the DSP's Chip Enable CE1 space (see Memory Map Section in this document for further details). The lower 80K bytes are utilized for program storage with the first 1k bytes utilized for the DSP's boot loader. Upon DSP reset the first 1K of OTP EPROM is copied into Internal RAM by the DSP's Enhanced DMA Controller (EDMA). Although the boot process begins when the DSP is released from external reset, the transfer actually occurs while the CPU is internally held in reset. This methodology allows for selection of the boot prom width (in our case 8 bits). After completion of the 1K block transfer, the CPU is removed from reset and execution begins at address 0x0.

The upper 48K bytes of the OTP EPROM is utilized for storage of FPGA Data. Since the DSRA is typically the primary boot storage device on the host computer, the interface to both the PCI Bus and the Disk must be stored on the DSRA and loaded prior to release of the PCI Bus from Reset. Revision 2.2 of the PCI Local Bus Specification is still quite lax in it's specification of a host computer's power on sequencing. The specification calls for a typical delay of 100msec from power-stable before release of PCI Reset. In practice this delay is currently 200msec although this varies amongst computer manufacturers. A detailed discussion of the power-on sequencing and boot operation of the DSRA is contained in the PCI Reset Section of this document.

030200 440200

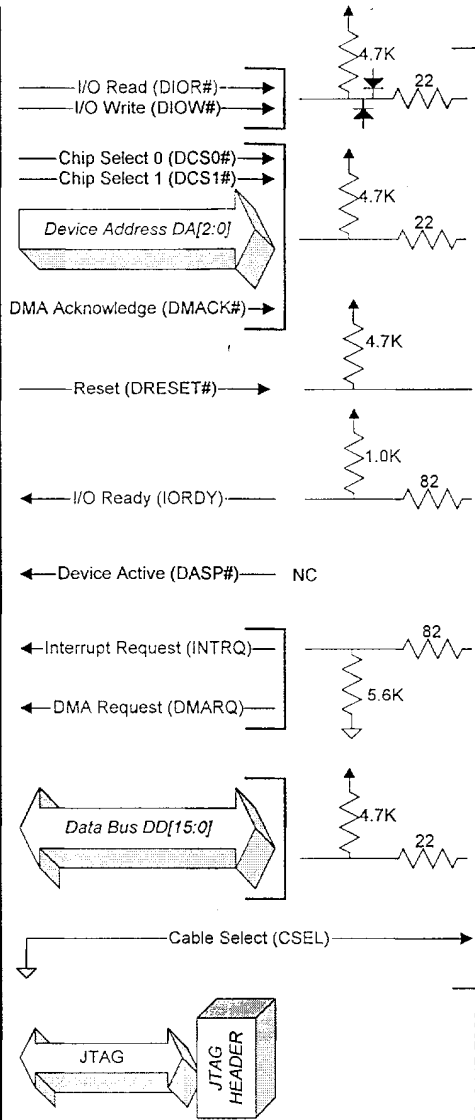


Realtime Data Compression Systems Pro

15

BOOT DE  
 PCIAD[31:0]  
 PCICBE[3:0]  
 PCICLK  
 PAR  
 FRAME#  
 TRDY#  
 IRDY#  
 STOP#  
 DEVSEL#  
 IDSEL  
 INTA#  
 REO#  
 GNT#  
 PERR#  
 SERR#  
 PCIRST#

U5  
 XILINX  
 XCS30XL-PQ240-4  
 FPGA



"REAL" ATA - 5 CONNECTOR

Connector: 3M 3417-7000  
 Strain Relief: 3M 3448-2040

UNIVERSAL PCI BUS INTERFACE

**REALATA™ SYSTEM BLOCK DIAGRAM**  
 Date: June 16, 1999  
 By: James J. Fallon  
 Revision: E



## 5.0 MEMORY MAPPING

Start Address	End Address	External Memory	Size (bytes)	Description
0000 0000	0000 FFFF	N/A	64K	Internal RAM (L2)
0001 0000	017F FFFF	N/A	24M-64K	Reserved
0180 0000	0183 FFFF	N/A	256k	Internal Configuration Bus EMIF Registers
0184 0000	0187 FFFF	N/A	256K	Internal Configuration Bus L2 Control Registers
0188 0000	018B FFFF	N/A	256K	Internal Configuration Bus HPI Register
018C 0000	018F FFFF	N/A	256K	Internal Configuration Bus McBSP 0 Registers
0190 0000	0193 FFFF	N/A	256K	Internal Configuration Bus McBSP 1 Registers
0194 0000	0197 FFFF	N/A	256K	Internal Configuration Bus Timer 0 Registers
0198 0000	019B FFFF	N/A	256K	Internal Configuration Bus Timer 1 Registers
019C 0000	019F FFFF	N/A	256K	Internal Configuration Bus Interrupt Selector Registers
01A0 0000	01A3 FFFF	N/A	256K	Internal Configuration Bus EDMA RAM and Registers
01A4 0000	1FFF FFFF	N/A		Reserved
2000 0000	2FFF FFFF	N/A		McBSP 0/1 Data
3000 0000	7FFF FFFF	N/A		Reserved
8000 0000	80FF FFFF	CE0	16M	SDRAM Bank 0
8100 0000	8FFF FFFF	CE0	240M	Reserved for SDRAM Bank 0 Expansion
9000 0000	9001 FFFF	CE1	128K	ROM
9002 0000	9FFF FFFF	CE1	256M-128K	Reserved for ROM Expansion
A000 0000	A0FF FFFF	CE2	16M	SDRAM Bank 1
A100 0000	AFFF FFFF	CE2	240M	Reserved for SDRAM Bank 1 Expansion

B000 0000	B000 00BF	CE3	64K	Xilinx PCI Interface
B001 0000	B000 00C0	CE3	64K	Xilinx UltraDMA/66 Interface
B002 0000	B002 FFFF	CE3	64K	Reserved for ADSL Modem
B003 0000	B003 FFFF	CE3	64K	Reserved for Secure Network / Data
B004 0000	BFFF FFFF	CE3	256M-256K	Reserved for Future Expansion
C000 0000	FFFF FFFF	N/A	1G	Reserved

00000000-44000000

Specific Allocations of Control Registers, Data Ports, and Memory is detailed in the respective sections of this document. This document has utilized the Texas Instrument's *TMS320C6000 Peripherals Reference Guide*, Literature Number SPRU190C, dated April of 1999 and *TMS320C6211 Fixed-Point Digital Signal Processor*, Literature Number SPRS073A – August 1998 – Revised March 1999 as it's primary reference documents. It is anticipated that there will be changes due to the preliminary nature of the Texas Instruments TMS320C6211GFN1-150 Digital Signal Processor, this document will be updated as and when appropriate to reflect both component and board level changes.

## 6.0 MEMORY BANDWIDTH ALLOCATION

The DSRA onboard cache is shared by the DSP, Disk Interface, and PCI Bus. The best case, maximum bandwidth for the SDRAM memory is 70 megawords per second, or equivalently, 280 megabytes per second.

The DSRA's 32 bit PCI Bus interface has a best case bandwidth of 132 megabytes per second, or equivalently 33 megawords per second. In current practice this bandwidth is only achieved in short bursts. The granularity of PCI data bursts to/from the DSRA is governed by the FPGA's PCI Bus interface data buffer depth of sixteen words (64 bytes). The time division multiplexing nature of the current FPGA PCI Data Transfer Buffering methodology cuts the sustained PCI bandwidth down to 66 megabytes/second.

Data is transferred across the ultraDMA disk interface at a maximum burst rate of 66 megabytes/second. It should be noted that the burst rate is only achieved with disks that contain onboard cache memory. Currently this is becoming more and more popular within the industry. However assuming a disk cache miss, the maximum transfer rates from current disk drives is approximately six megabytes per second. Allotting for technology improvements over time, the DSRA has been designed for a maximum sustained disk data rate of 20 megabytes second (5 megawords/second). A design challenge is created by the need for continuous access to the SDRAM memory. Disks are physical devices and it is necessary to continuously read data from disk and place it into memory, otherwise the disk will incur a full rotational latency prior to continuing the read transaction. The maximum SDRAM access latency that can be incurred is the depth of the each of the two disk FIFO s or sixteen data. Assuming the FIFO is sixteen words deep the maximum latency time for emptying the other disk fifo and restoring it to the disk interface is sixteen words at 5 megawords per second or  $(16 \times 3.2\text{usec}) = 1\text{usec}$ . Each EMIF clock cycle is 14.2857nsec, thus the maximum latency translates to 224 clock cycles. It should be noted that transfers across the disk interface are 16 bits wide, thus the FPGA is required to translate 32 bit memory transfers to 16 bit disk transfers, and vice-versa.

The DSP services request for its external bus from two requestors, the Enhanced Direct Memory Access (EDMA) Controller and an external shared memory device controller. The DSP can typically utilize the full 280 megabytes of bus bandwidth on an 8k through 64K byte (2k word through 16k word) burst basis. It should be noted that the DSRA does not utilize the SDRAM memory for interim processing storage, and as such only utilizes bandwidth in direct proportion to disk read and write commands.

For a single read from disk transaction data is transferred from and DMA transfer into SDRAM memory. This data is then DMA transferred by the DSP into onboard DSP memory, processed, and re transferred



back to SDRAM in decompressed format (3 words for every one word in). Finally the data is read from SDRAM by the DSRA's PCI Bus Controller and placed into host computer memory. This equates to eight SDRAM accesses, one write from disk, one read by the DSP, three writes by the DSP and three by the PCI Bus. Disk write transactions similarly require eight SDRAM accesses, three from the PCI, three DSP reads, one DSP write, and one to the disk.

Neglecting overhead for setting up dma transfers, arbitration latencies, and memory wait states for setting up SDRAM transactions, the maximum DSRA theoretical SDRAM bandwidth limit for disk reads or writes is 280/8 megabytes second or 35 megabytes second. It should be noted that the best case allocation of SDRAM bandwidth would be dynamic dependent upon the data compression and decompression ratios. Future enhancements to the DSRA will utilize a programmable timeslice system to allocate SDRAM bandwidth, however this first embodiment will utilize a fixed allocation ratio as follows:

If all three requestors require SDRAM simultaneously:

PCI Bus Interface	3/8
DSP Accesses	4/8
UltraDMA Disk Interface	1/8

The state machine sequence for SDRAM accesses is:

STATE	000b	001b	010b	011b	100b	101b	100b
SDRAM BUS OWNER	DISK	PCI	DSP	PCI	DSP	PCI	DSP
CLOCK CYCLES	4	16+4 (20)	2	16+4 (20)	4	16+4 (20)	16+4 (20)
Clock Cumul.			4	24	26	46	50
TIME (nsec)	57	286	29	286	57	286	286
%							

If only the PCI Bus and DSP require SDRAM:

PCI Bus Interface	4/8
DSP Accesses	4/8

SDR100114-020300

If only the DSP and Disk require SDRAM:

DSP Accesses                      6/8

UltraDMA Disk Interface        2/8

If only the PCI Bus and Disk require SDRAM:

PCI Bus Interface                6/8

UltraDMA Disk Interface        2/8

If only one device requires SDRAM it receives the full SDRAM bandwidth.

CONFIDENTIAL

## 7.0 INTERRUPTS

As with all C6000 family digital signal processors, the TMS320C6211 DSP has 12 useable interrupts, with provisions for up to 32 interrupt sources. Available DSP Interrupt sources and their allocation on our DSRA are listed below:

Interrupt Selection Number	Interrupt Acronym	Interrupt Description	DSRA Function
00000b	DSPINT	Host Port to DSP Interrupt	Host Port for Development Only
00001b	TINT0	Timer Interrupt 0	Not Used
00010b	TINT1	Timer Interrupt 1	Not Used
00011b	SD_INT	EMIF SDRAM Timer Interrupt	Used for Refresh ?
00100b	EXT_INT4	External Interrupt 4	Spare
00101b	EXT_INT5	External Interrupt 5	PCI Interrupt
00110b	EXT_INT6	External Interrupt 6	PCI EDMA Interrupt
00111b	EXT_INT7	External Interrupt 7	Disk Interrupt
01000b	EDMA_INT	EDMA Channel (0 through 15) Interrupt	DSP DMA Control
01001b	Reserved	Not Used	Not Used
01010b	Reserved	Not Used	Not Used
01011b	Reserved	Not Used	Not Used
01100b	XINT1	McBSP 1 Transmit Interrupt	Not Used
01101b	RINT1	McBSP 1 Receive Interrupt	Not Used
01110b	XINT0	McBSP 0 Transmit Interrupt	Not Used
01110b	RINT0	McBSP 0 Receive Interrupt	Not Used
other	Reserved	Not Used	Not Used

Interrupt selection registers are mapped as follows:

Byte Address	Register Name	Description
019C 0000h	Interrupt Multiplexer High	Selects which interrupts drive CPU interrupts 10-15 (INT10-15)
019C 0004h	Interrupt Multiplexer Low	Selects which interrupts drive CPU interrupts 4-9 (INT4-9)
019C 0008h	External Interrupt Polarity	Selects the polarity of the external interrupts (EXT INT4 – EXT INT7)

The default interrupt mapping after DSP reset is:

CPU Interrupt	Related INTSEL Field	INTSEL Reset Value	Interrupt Acronym	Interrupt Description	DSRA Function
INT4	INTSEL4	00100b	EXT_INT4	External Interrupt Pin 4	Spare
INT5	INTSEL5	00101b	EXT_INT5	External Interrupt Pin 5	PCI Interrupt
INT6	INTSEL6	00110b	EXT_INT6	External Interrupt Pin 6	PCI EDMA Interrupt
INT7	INTSEL7	00111b	EXT_INT7	External Interrupt Pin 7	Disk Interrupt
INT8	INTSEL8	01000b	EDMA_INT	EDMA Interrupt	EDMA Interrupt
INT9	INTSEL9	01001b	Reserved	Reserved	Reserved
INT10	INTSEL10	00011b	SD_INT	EMIF SDRAM Timer Interrupt	EMIF SDRAM Timer Interrupt
INT11	INTSEL11	01010b	Reserved	Reserved	Reserved
INT12	INTSEL12	01011b	Reserved	Reserved	Reserved
INT13	INTSEL13	00000b	DSPINT	Host Port to DSP Interrupt	Host Port to DSP Int - Development Only
INT14	INTSEL14	00001b	TINT0	Timer 0 Interrupt	Not Used
INT15	INTSEL15	00010b	TINT1	Timer 1 Interrupt	Not Used

The interrupt registers are to be programmed only once upon DSP reset prior to enabling interrupts. Once the registers are set the interrupt flag registers should be cleared by DRSA software after a reasonable delay to ensure that any spurious transitions caused by interrupt configuration.

### 7.1 Interrupt Multiplexer Register

Mapping of interrupt sources to CPU interrupts is performed by configuring the INTSEL fields within the Interrupt Multiplexer High and Low Registers. Fields INTSEL4 through INTSEL15 correspond to CPU Interrupts INT4 through INT15. Assigning interrupts is performed by setting the INTSEL fields to the value of the interrupt desired.

#### Interrupt Multiplexer High Register

31	30	26	25	21	20	16
Reserved	INTSEL15		INTSEL14		INTSEL13	
R, +0	RW, +00010		RW, +00001		RW, +00000	
15	14	10	9	5	4	0
Reserved	INTSEL12		INTSEL11		INTSEL10	
R, +0	RW, +01011		RW, +01010		RW, +00011	

#### Interrupt Multiplexer Low Register

31	30	26	25	21	20	16
Reserved	INTSEL9		INTSEL8		INTSEL7	
R, +0	RW, +01001		RW, +01000		RW, +00111	
15	14	10	9	5	4	0
Reserved	INTSEL6		INTSEL5		INTSEL4	
R, +0	RW, +00110		RW, +00101		RW, +00100	

### 7.2 External Interrupt Polarity Register

The external interrupt polarity register sets the polarity of the four external interrupts (EXT\_INT4 to EXT\_INT7). The DSRA asserts all interrupts on a low to high transition. By default the external interrupts are set to this low to high polarity (XIP value of 0). Therefore no additional initialization is required although good design practice suggests stepping on (writing to) bits 3 through 0 of this register during the DSP initialization routine.

Jay, remember to amend block diagram and Xilinx info!

31	4	3	2	1	0
Reserved	XIP7	XIP6	XIP5	XIP4	
R, +0	R, +0	RW, +0	RW, +0	RW, +0	RW, +0

25

## 8.0 ATA AND ULTRA DMA/66 DISK INTERFACE

The Realtime Data Storage and Retrieval Accelerator (DSRA) utilizes the American National Standard for Information Systems (ANSI) AT Attachment Interface (ATA/ATAPI-4) to connect the DSRA (host) and data storage devices. This standard defines the connectors and cables for the physical interconnects between the DSRA and the storage devices, along with the electrical and logical characteristics of the interconnecting signals.

### 8.1 Operational Registers

Operational registers within the storage device along with commands and the protocol for the storage devices are also defined. Xilinx Register definitions for the DSRA PIO Mode 2 Interface are as follows:

#### ATA-4 Reset Register

A valid read or write cycle to DSP Memory map Address 0xB0000D0 issues and holds the Disk Interface in Reset. The minimum reset pulse is 25 usec although it is suggested that a minimum of 50 nsec is utilized. The DSP is now responsible for timing the reset pulse width. All data to or from this location is invalid.

31	0
Reserved	
+XXXXXXXXXXXXXXXXXXXXXXXXXXXX	

#### ATA-4 Set Register

A valid read or write cycle to DSP Memory map Address 0xB0000D4 puts the Disk Interface into set mode. All data to or from this location is invalid.

31	0
Reserved	
+XXXXXXXXXXXXXXXXXXXXXXXXXXXX	

### ATA-4 Address Register

The ATA-4 Address Register, located within the DSP Memory Map at location 0xB00000C0 is utilized to access command, control, and data registers within the Target Disk Drive. The address of the various registers are per the aforementioned ATA-4 specification. It should be noted that -CS0 and -CS1's inverse polarity is accounted for by the Xilinx ATA-4 interface, thus these bits are asserted true by writing 1's to the appropriate bit locations.

31	8	7	5
Reserved		Reserved	
+XXXXXXXXXXXXXXXXXXXXXXXXXX		+000	

4	3	2	1	0
CS1	CS0	DA1	DA1	DA0
W, +0	W, +0	W, +0	W, +0	W, +0

### ATA-4 Data IO Write Command

A valid data write cycle to DSP Memory Map Address 0xB00000C4 issues an ATA-4 Data IO Write Command to the register specified in the target disk device specified by the ATA-4 Address Register. The data word written by the Data IO Write command is the actual data written to target disk. The data IO Write Command typically is utilized to write data to the command, control, and data registers within the Target Disk Drive. Dependent upon the command being executed the Xilinx ATA-4 Interface senses the 8 / 16 bit nature of the requested command by monitoring the disk's -HIOCS16 output line. However it is the responsibility of the DSP's software to understand the nature of the command issued and write the appropriate number of valid bits with the ATA-4 Data IO Write Command. This register is write only.

31	16
Reserved	
+XXXXXXXXXXXXXXXXXXXXXXXXXX	

15	8	7	0
Upper Write Data Byte		Lower Write Data Byte	
W, +0		W, +0	

### ATA-4 Data IO Read Command

A valid read cycle to DSP Memory Map Address 0xB00000C8 issues an ATA-4 Data IO Read Command to the register specified in the ATA-4 Address Register and reads the data in the ATA-4 Data Register (usually 8 bits for command and 16 bits for data). Any data values returned by this command are invalid; the data must be read from the Data IO Read Register specified below.

31	0
Reserved	
+XXXXXXXXXXXXXXXXXXXXXXXXXXXX	

### ATA-4 Data Read Register

The ATA-4 Data Read Register, located within the DSP Memory Map at location 0xB00000CC is utilized to read data from the command, control, and data registers within the Target Disk Drive. Dependent upon the command being executed the Xilinx ATA-4 Interface senses the 8 / 16 bit nature of the requested command by monitoring the disk's -HIOCS16 output line. However it is the responsibility of the DSP's software to understand the nature of the command issued and read only the appropriate number of bits within the ATA-4 Data Read Register. This register is read only.

31	16		
Reserved			
+XXXXXXXXXXXXXXXXXXXXXXXXXXXX			
15	8	7	0
Upper Read Data Byte		Lower Read Data Byte	
R, +0		R, +0	

Note: Because of the ability for the ability of the ATA-4 interface to extend or write cycles by 1.25usec it has been necessary to add the Data Read Register. The DSP must delay at least 2 usec from the issuance of a Data IO Read Command to ensure that valid data is placed in the Data IO Read Register. Subsequent iterations of this interface may utilize an interrupt to avoid this difficulty.

Since write data is latched into an internal Xilinx register immediately upon a Data IO Write Cycle, The use of a Data IO Write Register is unnecessary.

### 8.2 Sample ATA Command Sequence

The proper startup sequence for ATA commands for disk operation is as follows:

1. reset
2. set
3. pause until disk ready
4. idle (spins up disk)

28



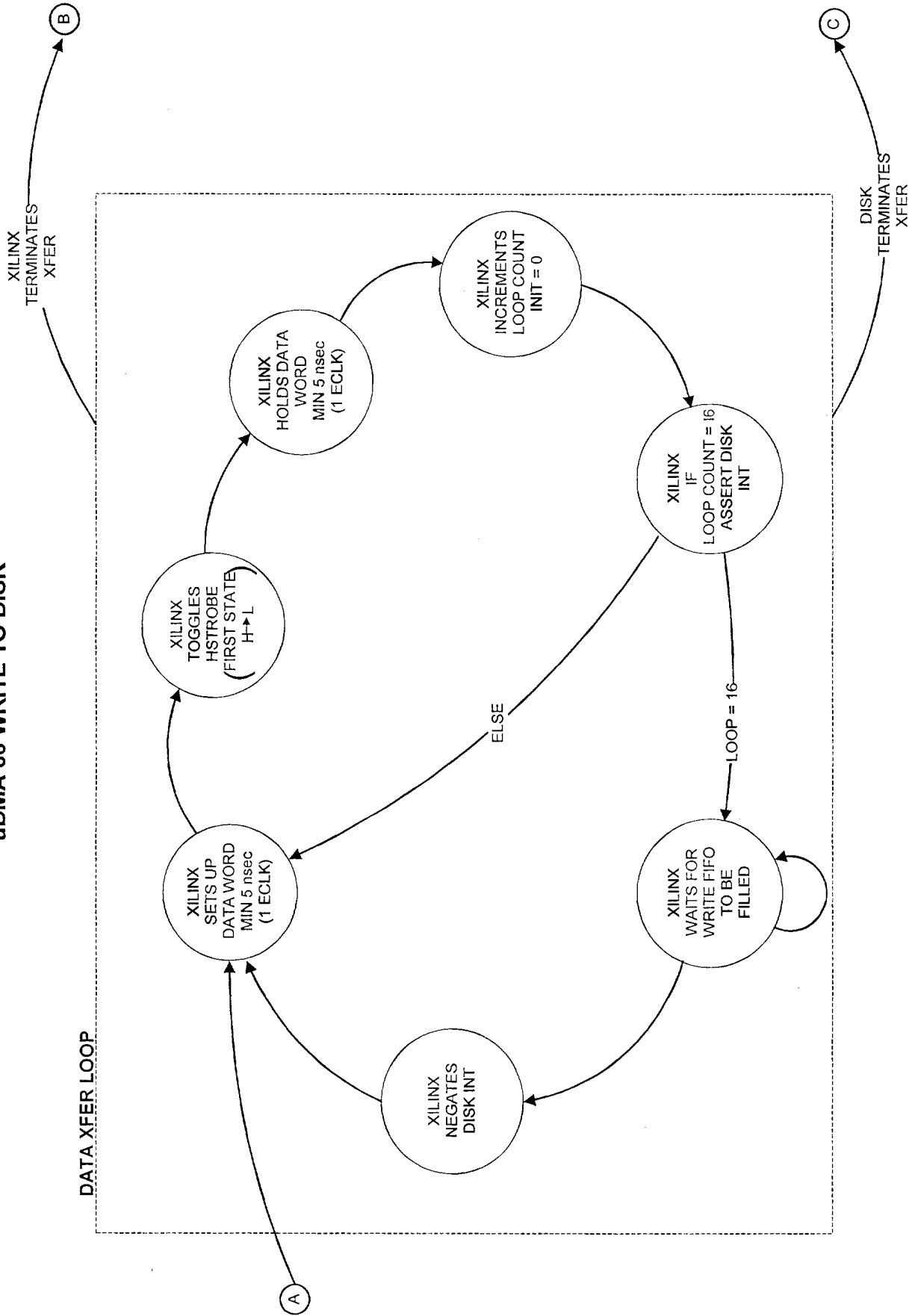
5. pause until disk ready
6. query disk for max LBA
7. set disk to max LBA

A sample set of commands for a disk write and a read is given below:

1. set desired LBA
2. seek
3. write long
4. write data
5. write ECC (error correction code)
6. read long
7. read data
8. read ECC (error correction code)



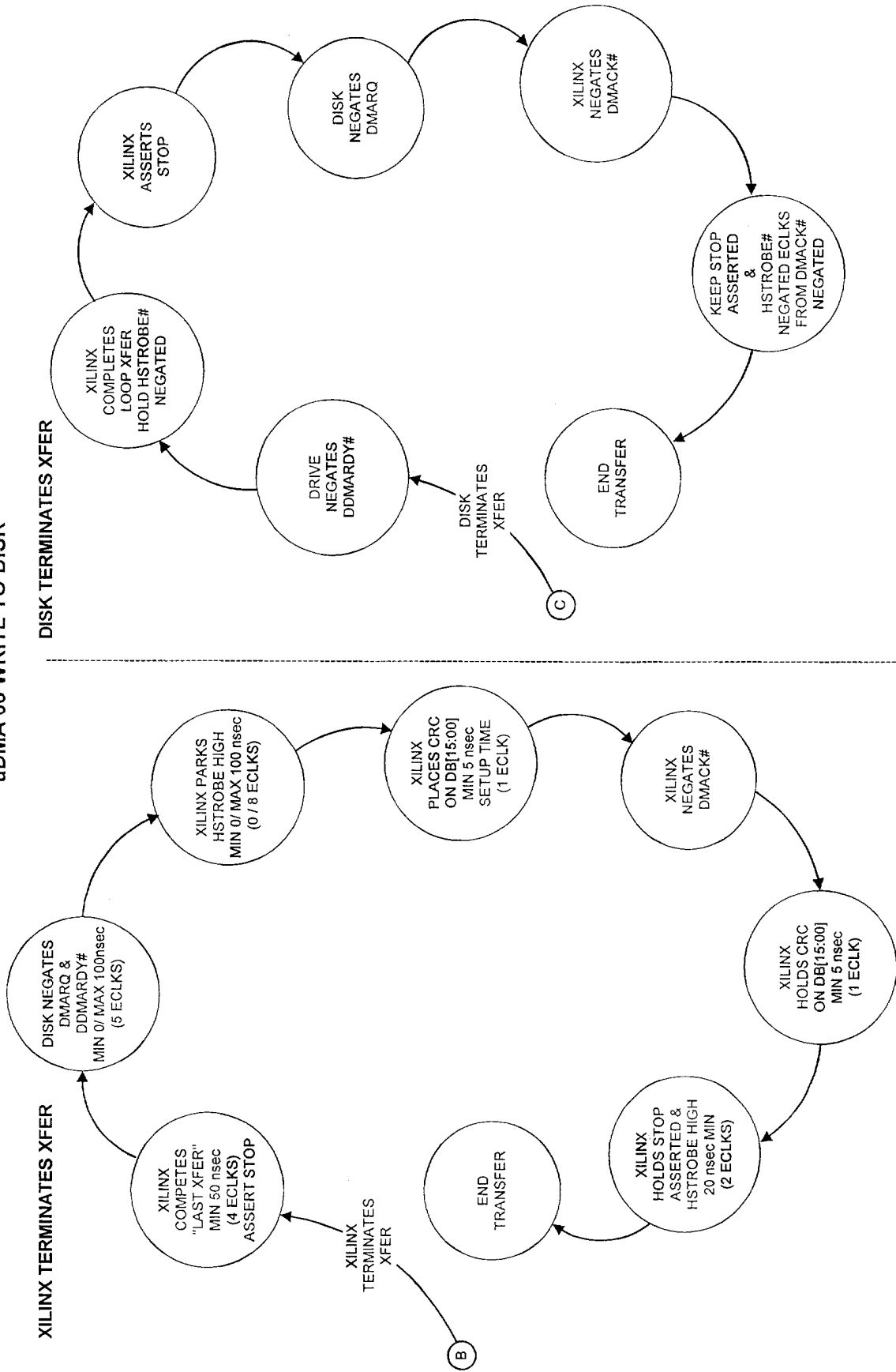
### UDMA 66 WRITE TO DISK



UDMA 66 WRITE TO DISK

XILINX TERMINATES XFER

DISK TERMINATES XFER



## 8.4 Ultra DMA66 Disk Interface

The Data Storage and Retrieval Accelerator (DSRA) transfers both commands and configuration information to the uDMA Disk utilizing a legacy PIO Mode 4 command protocol. Both the IDE and uDMA Disk interfaces are housed within the onboard Xilinx Field Programmable Gate Array (FPGA). The DSRA employs a Xilinx Spartan XCS40XL-4 40,000 field programmable gate array which instantiates a realtime proprietary IDE/uDMA 66 Interface to the following specification:

*Working Document of the Information Technology – AT Attachment with Packet Interface Extension (ATA/ATAPI-4) Revision 17 dated October 30, 1997*

.This specification is implemented subject to the limitations described in Section 8.0 of IBM OEM Hard Disk Drive Specifications fro DPTA-3xxxxx (37.5 – 13.6 GB) 3.5 Inch Hard Disk Drive with ATA Interface Revision 2.1.

When setting up data transfers, the Enhanced Direct Memory Access (EDMA) Controller utilizes two Control Registers, a 16 Word Data Write to Disk FIFO, a 16 Word Data Read From Disk FIFO, and a Disk Data Interrupt (DISKINT).

The 32 Bit uDMA Disk Transfer Count Register holds the Count for the DSRA Disk Reads or Writes. The uDMA Disk Transfer Count Register also clears DISKINT each time it is written to within DSP Memory Space at 0xB000 00D8.

### uDMA Disk Transfer Count Register

31	0
uDMA Disk Transfer Count	
W, +0x0000	

The second Control Register is the uDMA Disk Transfer Command Register that specifies the direction of the data transfer along with and associate Parameters. In order to accommodate EDMA operations, the uDMA Disk Transfer Command Register is located contiguously is DSP Memory Space at address location 0xB000 00DC.

### uDMA Disk Transfer Command

31	0
UDMA Disk Command	
W, +0x0000	

Commands are as follows:

uDMA Mode Disabled	0x0000
uDMA Disk Write	0x0001
uDMA Disk Read	0x0002

The uDMA Disk Transfer Register is also used to assert DISKINT during uDMA Disk Data Write Transactions, however it does not assert DISKINT during uDMA Disk Data Reads. Please refer to the upcoming descriptions of uDMA Disk Data Write and Read Operation for further discussion.

### **uDMA Disk Data Write FIFO**

Data is written to the Disk from the DSP via the 16 Word Disk Data Write FIFO located within address range of 0xB000 0100 through 0xB000 04FF.

Data writes from the DSP to anywhere within this address range place that data word in the next available location within the Disk FIFO. Writes to the Disk Data Write FIFO are also used to clear the DISKINT Interrupt. Please refer to the upcoming descriptions of the uDMA Disk Write Operation for further discussion.

### **uDMA Data Read FIFO**

Data is read from the Disk to the DSP via the 16 Word Disk Data Read FIFO located within address range of 0xB000 0500 through 0xB000 08FF.

Data read by the DSP from anywhere within this address range provides the next data word from the Disk Read FIFO. When any word is read from the Disk Data Read FIFO the DISKINT Interrupt is Cleared. Please refer to the upcoming descriptions of uDMA Disk Read Operation for further discussion.

### 8.4.1 Write Transaction

uDMA Disk Data Write Transactions occur according to the following protocol:

1. The DSP EDMA writes the Word Count to the uDMA Disk Transfer Count Register which also clears DISKINT.
2. The DSP EDMA writes the uDMA Write Command into the Disk Command Register which then asserts DISKINT. This also resets the uDMA Disk Data Write FIFO Count to "zero" words.
3. The DSP EDMA writes the first 16 words into the uDMA Disk Data Write FIFO which also clears DISKINT. (*DISKINT should already be set and it is cleared on each write into the FIFO*)
4. After the 16th Word is written into the FIFO by the DSP EDMA, the Xilinx uDMA Disk Interface writes 16 words held within the FIFO into the Disk Buffer and then subsequently asserts DISKINT. (The Word Count is also decremented with each data word written to the Disk.)
5. The DSP EDMA writes the next 16 words into the uDMA Disk Data Write FIFO which also clears DISKINT. (*DISKINT should already be set and it is cleared on each write into the FIFO*)
6. After the next 16<sup>th</sup> Word is written into the FIFO by the DSP EDMA, the Xilinx uDMA Disk Interface Writes 16 words held within the FIFO into the Disk and then asserts DISKINT. (The Word Count is again decremented with each data word written to the PCI Bus.)
7. Repeat Steps 5 & 6 until the last 16 words.
8. The DSP EDMA writes the last 16 words into the Disk Data Write FIFO which also clears DISKINT. (*DISKINT should already be set and it is cleared on each write into the FIFO*)
9. After the last 16<sup>th</sup> Word is written into the FIFO by the DSP EDMA, the Xilinx uDMA Disk Interface Writes 16 words held within the FIFO into the Disk and then asserts DISKINT. (The Word Count should be decremented with each data word written to the DISK down to zero.)
10. To complete the transfer the DSP EDMA writes a zero count into the uDMA Disk Transfer Count Register which also clears DISKINT. Alternately, the transfer may be chained by writing the next Count to the uDMA Disk Transfer Count Register and proceeding to Step 2. (It should be noted that this also clears DISKINT.)

## 8.4.2 Read Transaction

uDMA Disk Data Read Transactions occur according to the following protocol:

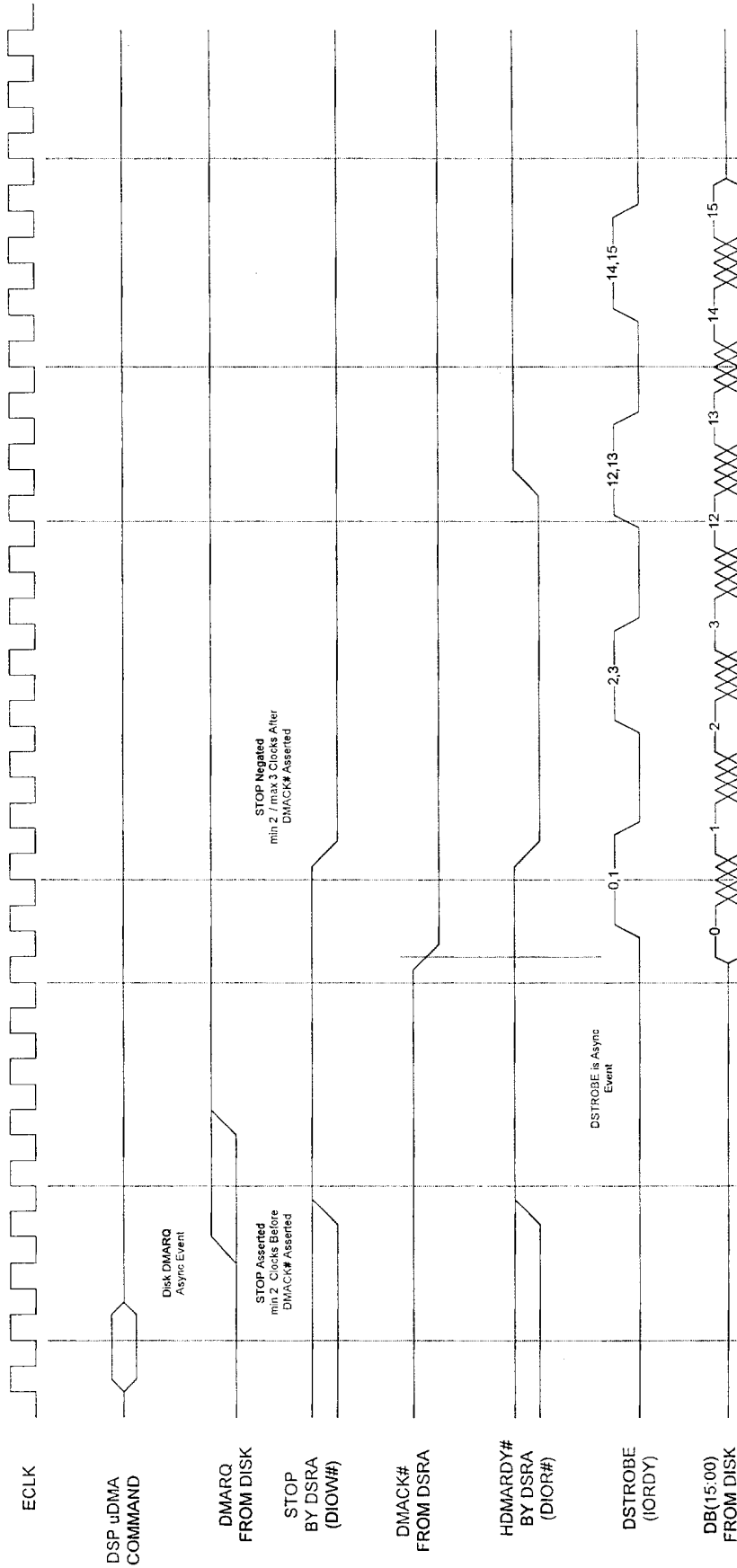
1. The DSP EDMA writes the Word Count to the uDMA Disk Transfer Count Register which also clears DISKINT.
2. The DSP EDMA writes the uDMA Read Command into the Disk Command Register It should be noted that this does not affect or assert DISKINT. This also resets the Disk Data Read FIFO Count to “zero” words.
3. The Xilinx uDMA Disk Interface then reads 16 Words. After the 16<sup>th</sup> Word is read and placed into the FIFO, DISKINT is asserted . (Word Count is also decremented with each data word read from the Disk.)
4. The DSP EDMA then reads the 16 words from the uDMA Disk Data Read FIFO which also clears DISKINT. (*DISKINT should already be set and it is cleared on each read from the FIFO*)
5. After the 16<sup>th</sup> Word is read from the FIFO by the DSP EDMA, the Xilinx uDMA Disk Interface reads the next 16 words from the Disk and places them in the uDMA Disk Data Read FIFO. After the 16<sup>th</sup> Word is read and placed into the FIFO, DISKINT is asserted . (Word Count is also decremented with each data word read from the Disk.)
6. The DSP EDMA then reads the next 16 words from the uDMA Disk Data Read FIFO which also clears DISKINT. (*DISKINT should already be set and it is cleared on each read from the FIFO*)
7. Repeat Steps 5 & 6 until the last 16 words.
8. The Xilinx PCI Bus Interface reads the last 16 words from the Disk and places them in the uDMA Disk Data Read FIFO. After the last 16<sup>th</sup> Word is read and placed into the FIFO, DISKINT is asserted . (Word Count is also decremented with each data word read from the Disk and should be zero.)
9. The DSP EDMA then reads the last 16 words from the uDMA Disk Data Read FIFO which also clears DISKINT. (*DISKINT should already be set and it is cleared on each read from the FIFO except the last word*). When the last word is read, (last read by DSP EDMA & Counter = 0), DISKINT is asserted.
10. To complete the transaction the DSP EDMA writes zero into the uDMA Disk Transfer Count Register which also clears DISKINT. Alternately, the transfer may be chained by writing the next Count into the uDMA Disk Transfer Count Register and proceeding to Step 2. (It should be noted that this also clears DISKINT.)



8.5 Timing Diagrams

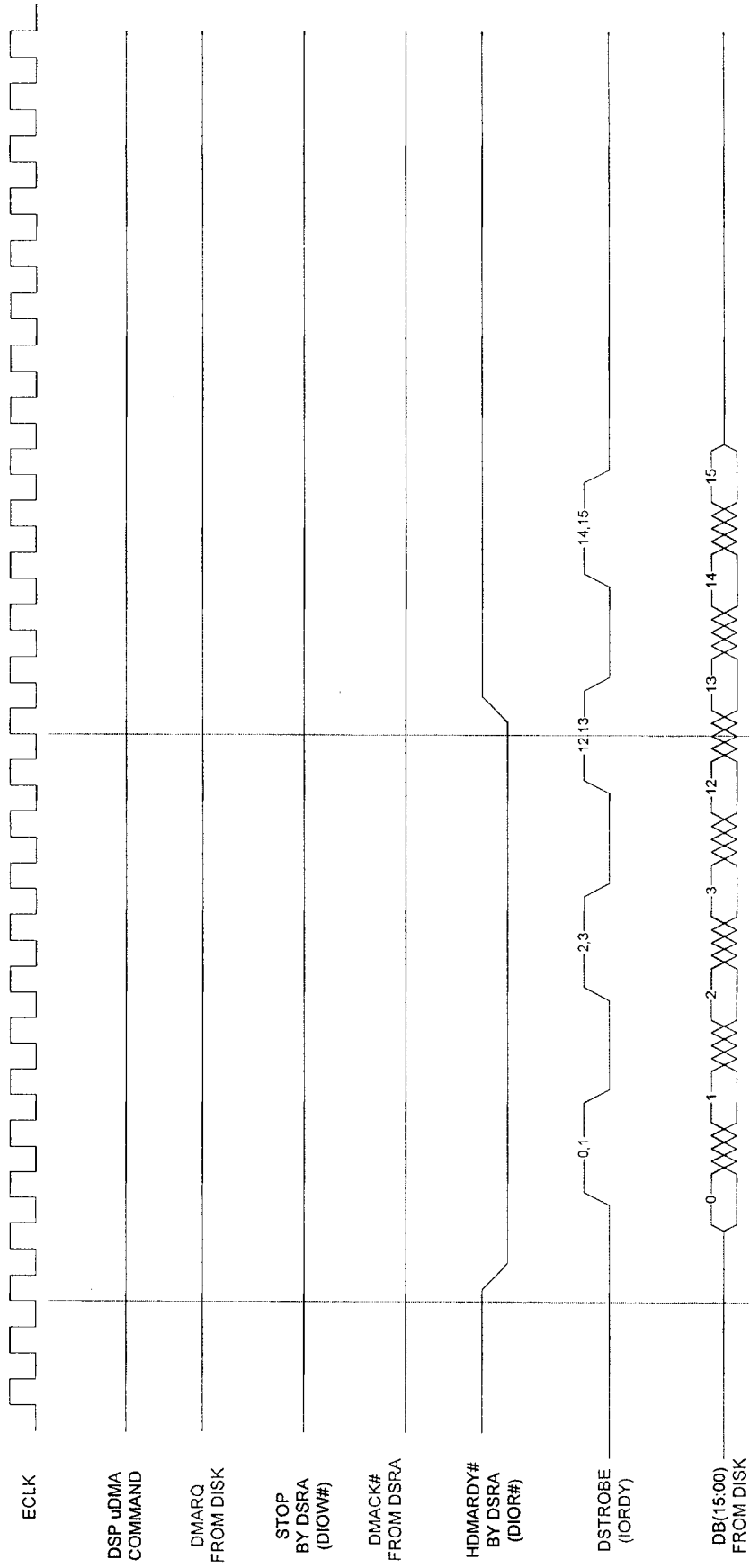
8.5.1 Read Timing

REALata uDMA MODE 4 - READ & PAUSE DATA TRANSFER TIMING

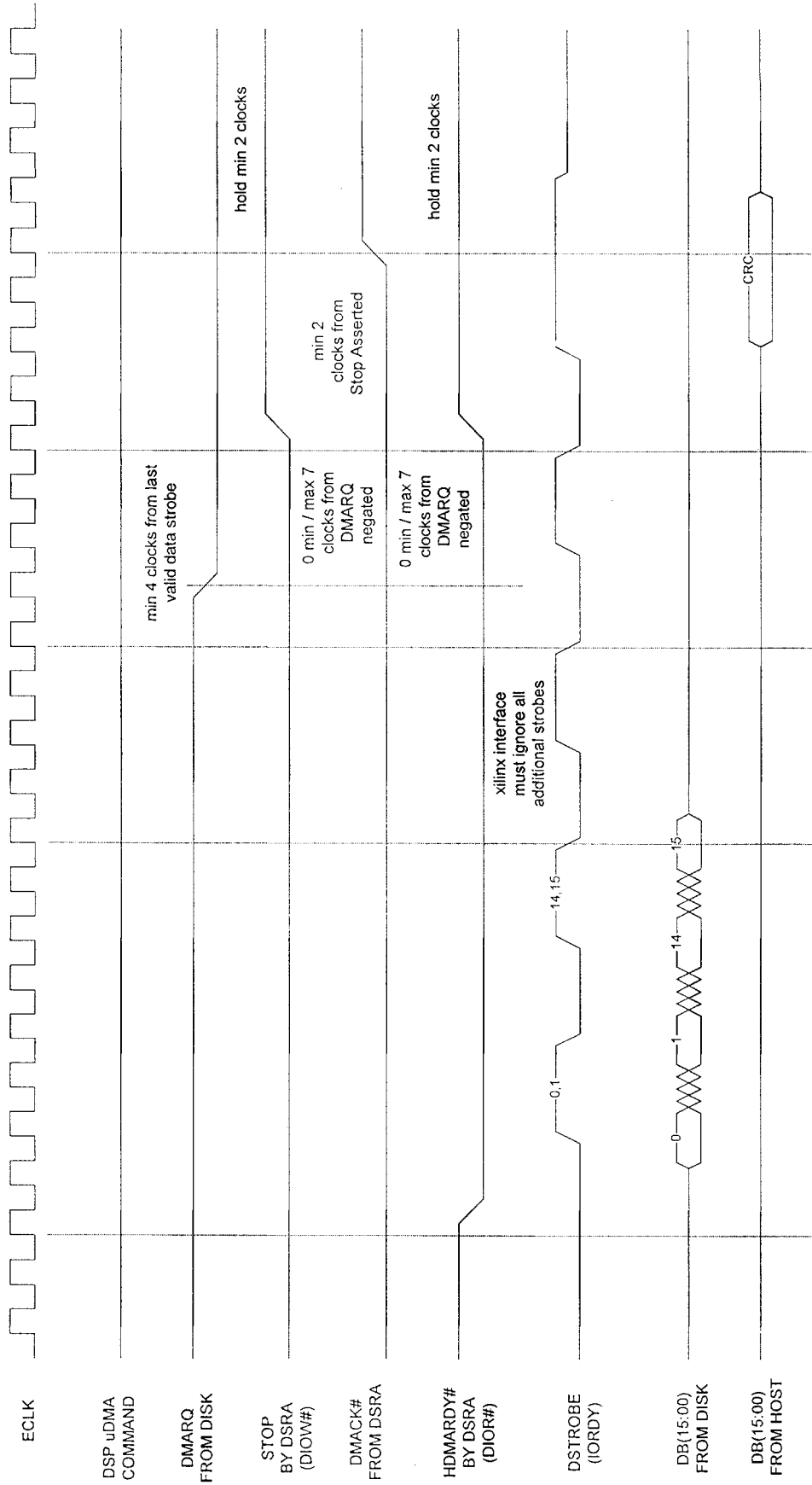


37

### REALata uDMA MODE 4 - READ RESUME TIMING

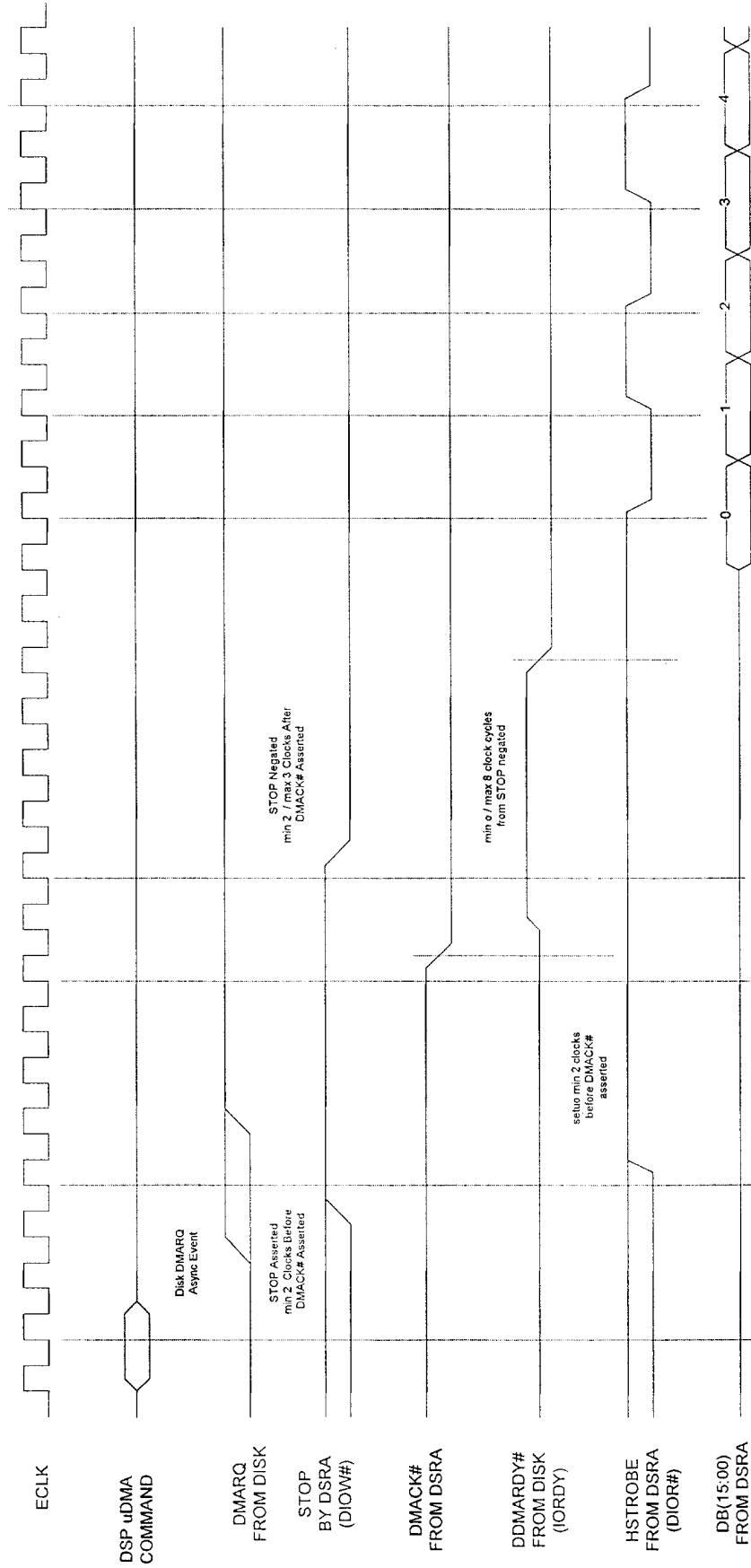


# REALata uDMA MODE 4 - READ DISK TERMINATE TIMING

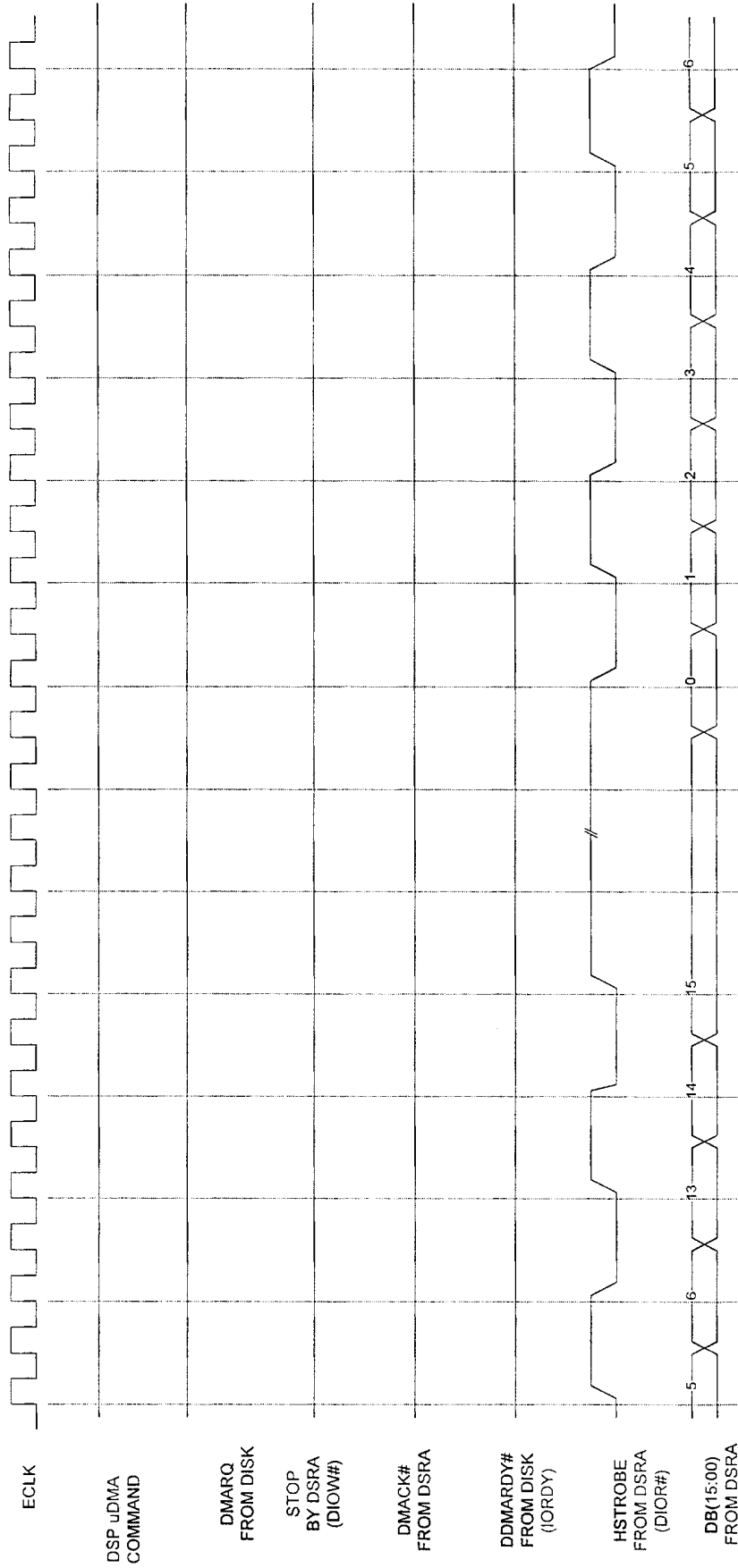


8.5.2 Write Timing

REALata uDMA MODE 4 - WRITE DATA TRANSFER TIMING

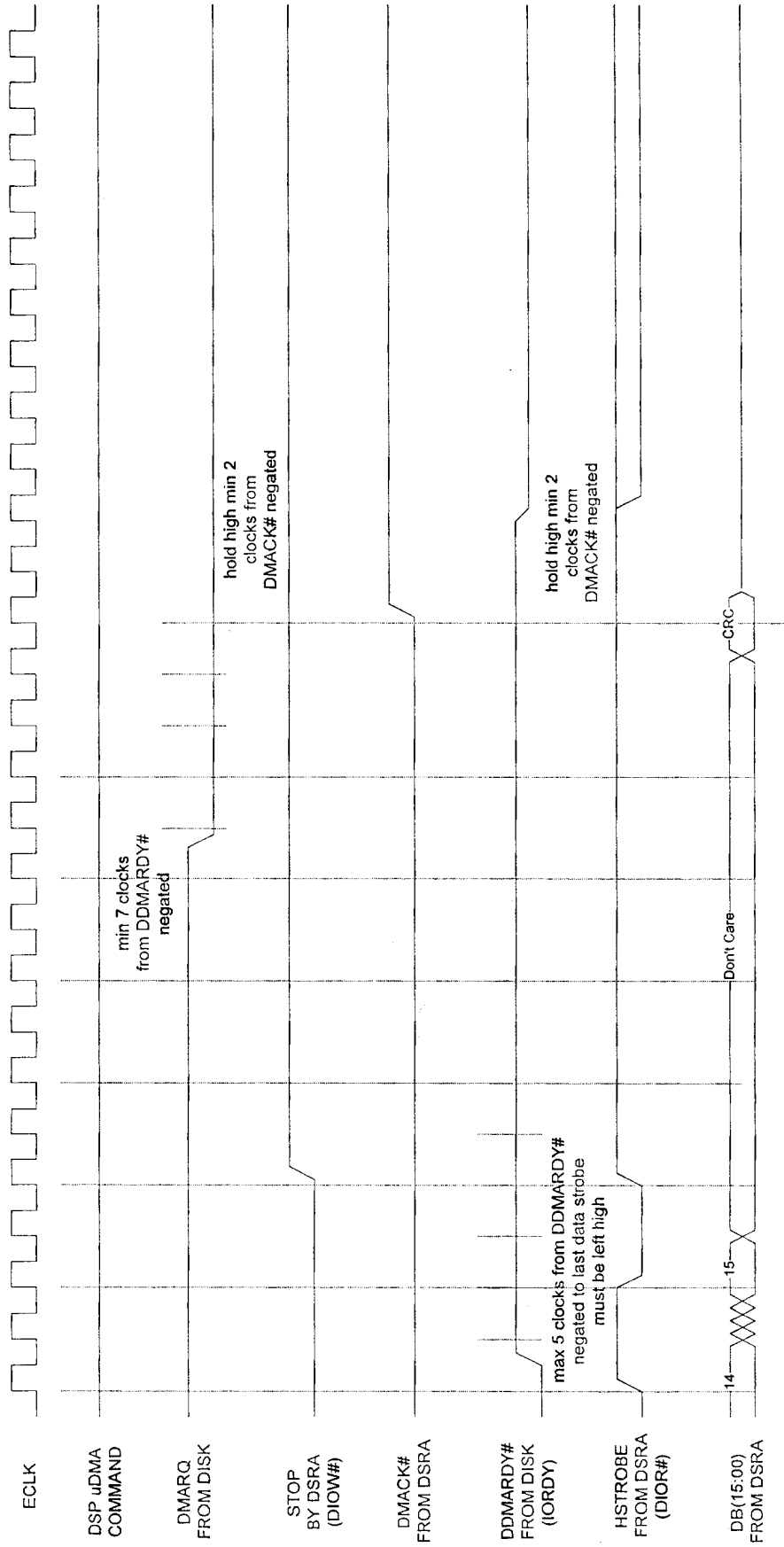


# REALata uDMA MODE 4 - HOST PAUSING WRITE DATA TRANSFER TIMING



41

# REALata uDMA MODE 4 - DISK TERMINATING WRITE DATA TRANSFER TIMING



42

## 9.0 XILINX ARCHITECTURE

### 9.1 Block Diagram

A high level block diagram of the DSRA's onboard XCS40PQ240 Field Programmable Gate Array is shown below:

### 9.2 Register Map

#### PCI Command FIFO

PCI Command FIFO	B000 0000 to B000 0020
PCI Command FIFO Future Expansion	B000 0021 to B000 007F

#### PCI Command Control

PCI Counter FIFO Clear	B000 0080
PCI Command Interrupt Clear	B000 0084
PCI Command Interrupt Set	B000 0088

#### SDRAM Test

SDRAM Write Test	B000 0090
SDRAM Read Test	B000 0094

#### XILINX Master Control

Xilinx State Machine Reset	B000 0098
----------------------------	-----------

#### ATA-4 Disk Interface - PIO Mode 4

ATA Write Adjust Register	B000 00B0
ATA Read Adjust Register	B000 00B4
ATA Reserved	B000 00B8 to B000 00BC

ATA Internal Register Address	B000 00C0
ATA Write Command (incl data)	B000 00C4
ATA Read Command (excl data)	B000 00C8
ATA Read Data Register	B000 00CC
ATA Interface Reset	B000 00D0
ATA Interface Set	B000 00D4

### ATA-4 Disk Interface – uDMA Mode

uDMA Disk Transfer Count Register	B000 00D8
uDMA Disk Transfer Command Register	B000 00DC
uDMA Reserved	B000 00E0 to B000 00FC

### ATA-4 Disk Interface – uDMA FIFOs

ATA-4 Disk Write FIFO	B000 0100 to B000 04FF
ATA-4 Disk Read FIFO	B000 0500 to B000 08FF

### PCI Data Interface Control

PCI Address	B000 0900
PCI Count	B000 0904
PCI Data Interface Reserved	B000 0908 to B000 9FF
PCI IntA Set	B000 00A0
PCI IntA Clear	B000 00A4

The following interface control is for test purposes only and it is to be used from the PC side, *not the DSP* side of the interface.

PCI IntA Toggle	PCI Base Addr + FC
-----------------	--------------------

### PCI Data Interface FIFO

PCI Data FIFO Write Data Address (To PCI)	B000 0A00 to B000 0DFF
PCI Data FIFO Read Address	B000 0E00 to B000 11FF



## 10.0 XILINX PROGRAMMING

The Realtime Data Storage and Retrieval Accelerator (affectionately referred to as the DSRA) employs an onboard Texas Instruments TMS320C6211 Digital Signal Processor (DSP) to program the onboard Xilinx Spartan Series XCS40XL upon power-up or system level PCI reset. An onboard reset circuit ensures that from system power-up and/or the assertion of PCI reset the DSP is allotted a minimum of 10msec to boot the DSP and load the Xilinx XCS40XL. Because of a potential race condition between host computer power-up or assertion of PCI RST# to configuration of the Xilinx for accepting PCI Commands, Express Mode programming mode has been selected as it is the fastest means for configuring the SpartanXL family XCS40XL Device.

In Express Mode the Xilinx XCS40XL requires a Binary Bitstream File (\*.BIT) of 387,824 bits or exactly 48,478 bytes. The partitioning of the EPROM space is as follows:

0x9000 0000 through 0x9001 3FFF EPROM DSP Code Space

0x9001 4000 through 0x9001 FFFF EPROM Xilinx Data

A fall back position would be if the actual DSP Code for the EPROM requires greater than 80K and we were able to use a smaller pin compatible device (XCS30XL which requires 298,664 bits or 37,333 bytes), in which case the following partitioning is suggested:

0x9000 0000 through 0x9001 5FFF EPROM DSP Code Space

0x9001 6000 through 0x9001 FFFF EPROM Xilinx Data

The second limits the design to XCS30XL devices only. At present we are using the XCS40XL.

The XCS40XL is factory set to byte-wide Express-Mode programming by setting both the M1/M0 bits to 0x0. For reference, it should be noted that express mode is selected anytime M1 is set to 0x0 and M0 is actually a "don't care" bit. The DSRA utilizes a PQ240 package to house the XCS40XL. In this package M1 is assigned to pin 58 and M0 to pin 60. M1 is grounded via a 33Ω resistor by a factory configured shorting jumper on J25 (pin 1 to 2), similarly M0 is also grounded via a factory configured shorting jumper on J26 (pin 1 to 2).

### Shared DSP Data Port

In order to accommodate express mode programming of the XILINX, the DSP utilizes its serial ports reconfigured as general purpose I/O. However, after the XILINX is programmed, the DSP may then reconfigure its serial ports for use with other devices. Using the same DSP ports for multiple purposes allows us greater flexibility while minimizing hardware resources and thus reducing product cost.

45

## Re-Programmable Disk Interface

The volatile nature of the XILINX effectively gives us the ability to have an unlimited number of hardware interfaces. Any number of XILINX programs can be kept in storage (EPROM, hard disk, or other storage device). Each program can contain new disk interfaces, interface modes or subsets thereof. Examples of interfaces include Programmed I/O mode and Ultra DMA66. When necessary the DSP can clear the interface currently residing in the XILINX and reprogram it with a new interface. This allows us to have compatibility with a large number of interfaces while minimizing hardware resources and thus reducing product cost.

The following DSP to XCS40XL pin assignments are hardwired on the DSRA to accommodate express mode programming.

DSP Signal Name	DSP Mode	DSP Pin #	to	XCS40XL Signal Name	XCS Mode	XCS Pin#
CLKR0	Output	H3		PROGRAM#	Input	122
FSX0	Output	H1		CCLK	Input	179
CLKX0	Output	G3		D0	Input	177
DX0	Output	H2		D1	Input	173
FSR0	Output	J3		D2	Input	159
CLKR1	Output	M1		D3	Input	152
FSX1	Output	L1		D4	Input	148
CLKX1	Output	L3		D5	Input	141
DX1	Output	L2		D6	Input	129
FSR1	Output	M3		D7	Input	123
DR0	Input	J1		INIT#	Output	89
CLKS0	Input	K3		DONE	Output	120
DR1	Input	M2		HDC	Output	64
CLKS1	Input	E1		LDC	Output	68

The DSP utilizes its serial ports reconfigured as general purpose I/O. Since the DSP imposes specific limitations specific pins (input/output/or both) we have tried to be judicious in our selection to minimize programming effort. The functions of the XCS40XL pins are summarized in the table below:

Signal	Type	Direction	Description
M1	Mode Selection	Input	Set Low for Express Mode
M0	Mode Selection	Input	Don't Care
D0 through D7	Data	Input	Write Configuration Data into XCS40XL
DOUT	Data	Output	Not Used in This Configuration
CCLK	Clock	Input	Loads Configuration Data on Rising Edge
PROGRAM#	Control	Input	Begin Clearing Configuration Memory
INIT#	Control	Output (Open Drain)	Transition from Low to High Indicates Configuration Memory is Clear and Ready to Accept Programming
DONE	Status	Output (Open Drain)	High Indicates Configuration Process Complete
HDC	Status	Output	High During Configuration, Low when Configuration Complete & I/Os Go Active
LDC#	Status	Output	Low During Configuration, High when Configuration Complete & I/Os Go Active
CS1	Control	Input	Pulled High to Signify First & Only FPGA

DSP I/O is utilized as follows:

(Excerpted from TMS320C6000 Peripherals Reference Guide, April 1999, SPRU190C).

TNMS320C6211 McBSP Registers Assignment in DSP Memory Map

Hex Byte Address			Abbreviation	Register Name
McBSP0	McBSP1	McBSP2		
-	-	N/A	RBR	Receive Buffer Register
-	-	N/A	RSR	Receive Shift Register
-	-	N/A	XSR	Transmit Shift Register
018C 0000	0190 0000	N/A	DRR	Data Receive Register (Notes 2 & 3)
018C 0004	0190 0004	N/A	DXR	Data Transmit Register (Note 4)
018C 0008	0190 0008	N/A	SPCR	Serial Port Control Register
018C 000C	0190 000C	N/A	RCR	Receive Control Register
018C 0010	0190 0010	N/A	XCR	Transmit Control Register
018C 0014	0190 0014	N/A	SRGR	Sample Rate Generator Register
018C 0018	0190 0018	N/A	MCR	Multi-channel Control Register
018C 001C	0190 001C	N/A	RCER	Receive Channel Enable Register
018C 0020	0190 0020	N/A	XCER	Transmit Channel Enable Register
018C 0024	0190 0024	N/A	PCR	Pin Control Register

Notes:

1. The Receive Buffer Register (RBR), Receive Shift Register (RSR), and Transmit Shift Register (XSR) are not directly accessible by the CPU, DMA Controller, or Xilinx.
2. Data Receive Register is CPU & DMA Read Only.
3. The McBSP0 Data Receive Register (DRR) is also mapped at 3000 0000 to 33FF FFFF. The McBSP1 Data Receive Register (DRR) is also mapped at 3400 0000 to 3FFF FFFF.
4. The McBSP0 Data Transmit Register (DXR) is also mapped at 3000 0000 to 33FF FFFF. The McBSP1 Data Transmit Register (DXR) is also mapped at 3400 0000 to 3FFF FFFF.

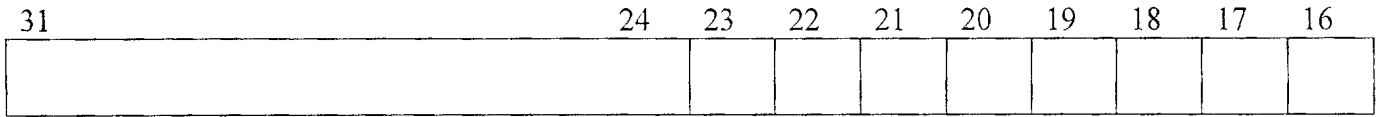
Two conditions allow the McBSP serial ports (CLKX, FSX, DX, CLKR, FSR, DR, and CLKS) to be utilized as general-purpose I/O rather than serial ports:

1. The related portion (transmitter or receiver) of the serial port is in reset: n(R/X)RST = 0 in the Serial Port Control Register (SPCR).
2. General-purpose I/O is enabled for the related portion of the serial port: (R/X)IOEN = 1 in the Pin Control Register (PCR).

48

Each serial port is configured by its 32-bit Serial Port Control Register (SPCR) and Pin Control Register (PCR).

Serial Port Control Register (SPCR)



15 14 13

00000000000000000000000000000000

49

Programming the Xilinx is accomplished in four steps.

1. Clearing the configuration memory
2. Initialization
3. Configuration
4. Start-Up

When either of three events occur: the host computer is first powered-up, a power failure and subsequent recovery occurs, or a front panel computer reset is initiated, the host computer asserts RST# (reset) on the PCI Bus. The DSRA contains a patentable power-on reset circuit that senses initial host computer power turn on along with PCI Bus RST# assertion. It is important to note that the DSRA is utilized in the computer boot-up sequence and as such it must be available exactly 5 clock cycles after the PCI RST# is deasserted, per PCI Bus Specification Revision 2.2.

While exact timings vary from computer to computer, the typical PCI bus reset is 200msec from initial power turn-on. This is in large part due the default standard in power-on reset chips. PCI Bus Specification 2.2 calls for a typical value of 100msec although there is no hard specification. Further compounding this issue is a minimum reset active time specification of 1msec from when power is stable. Practical limitations including Boot ROM code execution probably place a current practical minimum on when the PCI Bus becomes active, we cannot assume that this will hold for the foreseeable future.

In general PCI RST# is asserted as soon as the computer's power exceeds a nominal threshold of about 1 volt (although this varies) and remains asserted for 200msec thereafter. Power failure detection of the 5volt or 3.3 volt bus typically resets the entire computer as if it is an initial power-up event. Front panel resets are more troublesome and are derived from a debounced push-button switch input. Typical front panel reset times are a minimum of 20msec, although again the only governing specification limit is 1msec reset pulse width.

As will be discussed in the following section, it may not be necessary to reload the Xilinx each time the DSP is reset. A state machine input is readable by the DSP to ascertain who initiated the reset. For example, with a front-panel reset, power remains stable on the PCI Bus, thus the Dora's Xilinx should not require reloading. These issues will be discussed at length in the Reset Mode Section.

Assuming a DSP reset and that we are always reloading the Xilinx, the following steps should occur.

1. Upon Deassertion of DSP Reset, the lower 1K bytes of CE1 Space EPROM Memory (EPROM located at 0x9000 0000 through 0x9001 FFFF) is copied to the 1<sup>st</sup> 1k bytes of DSP's low memory (0x0000 0000 through 0x0000 03FF)
2. The DSP begins execution out of the lower 1K bytes of memory.
3. The DSP should initialize with at least the functionality to read EPROM Memory (CE1) space.
4. The DSP should configure the Serial Ports and Timers as General Purpose I/O per the specifications given in this document.
5. Deassert CCLK output (low)
6. The DSP initializes the Xilinx by:
  - Asserting PROGRAM# (low) for 50usec.
  - Polling on the INIT# input until it is deasserted (high)
7. Clear Byte Count
8. Read the 1<sup>st</sup> Xilinx data byte beginning at location 0x9001 4000 (preferred embodiment)
9. Increment Byte Count
10. Load the 1<sup>st</sup> output byte into the DSP's I/O locations with LSB at D0 and MSB at D7
11. Ensure a minimum delay of 5 usec from INIT# deasserted (high) (Xilinx internal setup time from initialization complete)
12. Assert CCLK output (High) for a minimum of 50nsec (this latches in data), then deassert. Steps 13 through 15 may be performed prior to deassertion to "pipeline" Xilinx Programming and minimize programming duration. CCLK must also be deasserted for a duration of not less than 50 nsec, thus maximum programming bandwidth is 10Mbytes/sec or a total duration of approximately 3.8msec excluding epilogs & prologs.
13. Test if Byte Count < 37,333 (0x91D5), if No branch to Step 19, otherwise
14. Read next data byte for Xilinx from EPROM
15. Increment byte count
16. Load into DSP I/O. Delay 20 nsec to ensure minimum data set-up time to Xilinx is observed.
17. Reassert CCLK output (high) for a minimum of 50nsec, then deassert for a minimum of 50nsec(may also be piped).
18. Loop beginning at Step 13  
\*\*\*\*\*
19. Test to ensure DONE is low (negated)
20. Do last byte
21. Poll on DONE
22. When DONE asserted (high) exit to remainder of DSRA Initialization
23. Otherwise timeout (1msec), post error to led (blink) and retry

## 11.0 DSRA INITIALIZATION & RESET

### 11.1 Reset Sequence

The DSP reset uses an active low signal, RESET\*. While RESET\* is low, the device is held in reset and is initialized to the prescribed reset state. All 3-state outputs are placed in the high-impedance state, and all other outputs are returned to their default states. The rising edge of RESET\* starts the processor running with the prescribed boot configuration. The RESET\* pulse may have to be increased if the phase-locked loop (PLL) requires synchronization following power-up or when PLL configuration pins change during reset.

The boot configuration is determined by the BOOTMODE[4:3] external pins. The value of BOOTMODE[4:3] are latched during the low period of RESET\*. The following table lists the possible values for BOOTMODE[4:3]. Note that BOOTMODE[2:0] are used for the 6201 and 6202 but are not used for the 6211.

BOOTMODE[4:3]	Boot Process
00	Host-port interface
01	8 bit ROM with default timings
10	16 bit ROM with default timings
11	32 bit ROM with default timings

The DSRA uses BOOTMODE[4:3] set to 01 for the ROM boot process needed for an 8-bit ROM. See the Memory Map Section in this document for details relating to the memory map.

When using a ROM boot process, the program loaded in external ROM is copied to address 0 by the EDMA controller. The EDMA copies the 1K bytes from the beginning of CE1 to address 0, using default ROM timings. After the transfer the CPU begins executing from address 0.

Although the boot process begins when the device is released from external reset, this transfer occurs while the CPU is internally held in reset. This boot process also lets you choose the width of the ROM. In this case, the EMIF automatically assembles consecutive 8-bit bytes or 16-bit halfwords to form the 32-bit instruction words to be moved. These values are expected to be stored in little endian format in the external memory, typically a ROM device.

### 11.2 Initialization Sequence

The on-chip peripherals as well as the XILINX must be initialized before program operation may begin. The following sections detail each step of the initialization process.

#### 11.2.1 Internal Memory

The L2 memory is configured for 16k bytes of cache and 48k bytes mapped as RAM. The L2 requestor priority is set such that CPU accesses have priority over enhanced DMA accesses. These settings are made in the Cache Configuration Register (CCFG).



To enable the L1 cache to work with data from the top 16MB of SDRAM bank 0 the Memory Attribute Register 0 (MAR 0) must be set to 1.

### 11.2.2 EDMA

The EDMA is initially setup with no events pending or enabled.

### 11.2.3 EMIF

Control of the EMIF and the memory interfaces it supports is maintained through memory-mapped registers within the EMIF. The memory-mapped registers are listed below.

Address	Register Name
0180 0000	EMIF Global Control
0180 0004	EMIF CE1 space control
0180 0008	EMIF CE0 space control
0180 000C	Reserved
0180 0010	EMIF CE2 space control
0180 0014	EMIF CE3 space control
0180 0018	EMIF SDRAM control
0180 001C	EMIF SDRAM Timing
0180 0020	EMIF SDRAM Extension

The EMIF Global Control Register controls the polarity of a number of signals as well as the clock enables. The settings used are given below:

Field	Setting
BUSREQ	BUSREQ output is high
ARDY	ARDY input is high
HOLD	HOLD* input is low
HOLDA	HOLDA* input is low
NOHOLD	Hold enabled
CLK1EN	CLKOUT1 enabled to clock
CLK2EN	CLKOUT2 enabled to clock

The EMIF CE space is arranged as follows:

CE Space #	Device
0	SDRAM0
1	ROM
2	Not used
3	XILINX

There is an EMIF CE space control register for each of the four CE spaces. These registers are used to configure the memory device type and the necessary timing information. The settings are given below.

Parameter	CE0	CE1	CE2	CE3
Memory Type	32 bit	8 bit	Not used	32 bit

	SDRAM	asynchronous (EPROM)		asynchronous (XILINX)
Read Setup	N/A	2	N/A	4 (8 temporary)
Read Strobe	N/A	2	N/A	3 (10 temporary)
Read Hold	N/A	1	N/A	3 (3 temporary)
Write Setup	N/A	N/A	N/A	4 (4 temporary)
Write Strobe	N/A	N/A	N/A	3 (10 temporary)
Write Hold	N/A	N/A	N/A	2 (3 temporary)
TA	N/A	2	N/A	2

The units for all fields (except memory type) are in ECLKOUT clock cycles (ECLKOUT is 75MHz; 1 cycle = 13.3333nsec). Those fields are not applicable for SDRAM. The control information for the SDRAM is in the EMIF SDRAM Control Register.

TA is the turn around time that controls the number of ECLKOUT cycles between reads/writes to different CE spaces for asynchronous memory types.

The EMIF SDRAM Control Register has the following settings.

Field	Description	Setting
TRC	Specifies the $t_{RC}$ (70ns) of the SDRAM $TRC = t_{RC} / (\text{ECLKOUT period}) - 1$	4
TRP	Specifies the $t_{RP}$ (20ns) of the SDRAM $TRC = t_{RP} / (\text{ECLKOUT period}) - 1$	1
TRCD	Specifies the $t_{RCD}$ (20ns) of the SDRAM $TRC = t_{RCD} / (\text{ECLKOUT period}) - 1$	1
INIT	0 = No effect 1 = initializes SDRAM in each CE space configured for SDRAM	1
RFEN	0 = SDRAM refresh disabled 1 = SDRAM refresh enabled	1
SDCSZ	Column size 0 = 9 column address pins 1 = 8 column address pins 2 = 10 column address pins	1
SDRSZ	Row size 0 = 11 row address pins 1 = 12 row address pins 2 = 13 row address pins	2
SDBSZ	Bank size 0 = two banks 1 = four banks	1

The EMIF SDRAM timing register controls the refresh period in terms of ECLKOUT cycles. Optionally, the period field can send an interrupt to the CPU. Thus, this counter can be used as a general purpose timer if it is not being used to refresh the SDRAM. The DSRA uses this timing register to refresh the SDRAM and not as a general purpose timer. Therefore we do not use the interrupt.

54

This register is also used to control the number of refreshes performed when the refresh counter expires. Up to four refreshes can be performed when the counter expires. The DSRA is programmed to perform two refreshes per refresh period. The MIRA 64-bit Synchronous DRAM Data Sheet indicates that the RAM must be refreshed 4096 times every 64msec. Therefore there should be at least one refresh every 16usec. To add a safety margin, it was decided to refresh more often. The EMIF SDRAM timing register period is set to refresh the RAM every 1024 clocks (or 13.65 usec).

Refresh commands (REFR) to the SDRAM enable all CE\* signals for all CE spaces selected to use SDRAM. REFR is automatically preceded by a DCAB (deactivate all banks) command. This ensures that all CE spaces selected with SDRAM are deactivated. The REFR requests are considered high priority. Transfers in progress are allowed to complete.

When the XILINX has control of the bus, the refresh counters within the EMIF continue to log refresh requests; however, no refresh cycles can be performed (by the EMIF) until bus control is again granted to the EMIF. Before giving up the bus the EMIF deactivates and refreshes any memory spaces configured as SDRAM. After that it is up to the XILINX to either give up the bus before the next refresh period or to refresh the SDRAM.

The EMIF SDRAM Extension Register has the following settings.

Field	Description	Setting
TLC	Specifies the Cas latency of the SDRAM in ECLKOUT cycles TCL= 0: Cas latency = 2 cycles TCL= 1: Cas latency = 3 cycles	0
TRAS	Specifies $t_{RAS}$ (50ns) value of the SDRAM in ECLKOUT cycles $TRAS = t_{RAS} - 1$	3
TRRD	Specifies $t_{RRD}$ value of the of the SDRAM in ECLKOUT cycles TRRD = 0, then $t_{RRD} = 2$ cycles TRRD = 1, then $t_{RRD} = 3$ cycles	0
TWR	Specifies $t_{WR}$ (10ns) value of the SDRAM in ECLKOUT cycles $TRAS = t_{WR} - 1$	0
THZP	Specifies $t_{HZP}$ (3 – 6 ns) value of the SDRAM in ECLKOUT cycles $TRAS = t_{HZP} - 1$	0
RD2RD	Specifies the number of cycles between READ to READ command (same CE space) (14nsec) of the SDRAM in ECLKOUT cycles RD2RD= 0: Read to Read = 1 cycle RD2RD= 1: Read to Read = 2 cycles	2
RD2DEAC	Specifies the number of cycles between READ to DEAC/DCAB of the SDRAM in ECLKOUT cycles RD2DEAC = (# of cycles READ to	2

55

	DEAC/DCAB) -1	
RD2WR	Specifies the number of cycles between READ to WRITE command (same CE space) (40nsec) of the SDRAM in ECLKOUT cycles  RD2WR = (# of cycles READ to WRITE) -1	2
R2WDQM	Specifies the number of cycles that BEx signals must be high preceding a WRITE interrupting a READ  R2WDQM = (# of cycles BEx high) -1	2
WR2WR	Specifies the number of cycles between WRITE to WRITE command (same CE space) (14nsec) of the SDRAM in ECLKOUT cycles WR2WR = (# of cycles WRITE to WRITE) -1	1
WR2DEAC	Specifies the minimum number of cycles between WRITE to DEAC/DCAB command of the SDRAM in ECLKOUT cycles WR2DEAC = (# of cycles WRITE to DEAC/DCAB) -1	1
WR2RD	Specifies the minimum number of cycles between WRITE to READ command (same CE space) (14nsec) of the SDRAM in ECLKOUT cycles WR2RD = (# of cycles WRITE to READ) -1	1

### 11.2.4 Interrupts

These are the available interrupts for the 6211.

Interrupt Selection #	Acronym	Description
0	DSPINT	Host port to DSP interrupt
1	TINT0	Timer 0 interrupt
2	TINT1	Timer 1 interrupt
3	SD_INT	EMIF SDRAM timer interrupt
4	EXT_INT4	External interrupt 4 (disk int)
5	EXT_INT5	External interrupt 5 (PCI cmd int)
6	EXT_INT6	External interrupt 6 (PCI data full)
7	EXT_INT7	External interrupt 7 (PCI data empty)

56

8	EDMA_INT	EDMA channel (0 – 15) interrupt
9	Reserved	Not used
A	Reserved	Not used
B	Reserved	Not used
C	XINT0	McBSP 0 transmit interrupt
D	RINT0	McBSP 0 receive interrupt
E	XINT1	McBSP 1 transmit interrupt
F	RINT1	McBSP 1 receive interrupt
other	Reserved	Reserved

These are the interrupt assignments made in the startup code.

CPU Interrupt #	Interrupt Selection #	Acronym	Description
4	4	EXT_INT4	External interrupt 4 (disk int)
5	5	EXT_INT5	External interrupt 5 (PCI cmd int)
6	6	EXT_INT6	External interrupt 6 (PCI data full)
7	7	EXT_INT7	External interrupt 7 (PCI data empty)
8	8	EDMA_INT	EDMA channel (0 – 15) interrupt
9	Not used	Not used	Not used
10	3	SD_INT	EMIF SDRAM timer interrupt
11	Not used	Not used	Not used
12	Not used	Not used	Not used
13	0	DSPINT	Host port to DSP interrupt
14	1	TINT0	Timer 0 interrupt
15	2	TINT1	Timer 1 interrupt

Although the C6000 peripheral set has space reserved for 32 interrupt selection numbers, currently only 13 are available for use by the 6211. However the 6211 only has the capability for 12 interrupt assignments (that is any 12 of the 13 available may be used).

The mapping of CPU interrupts to interrupt selection numbers is performed via the Interrupt multiplexer high and Interrupt multiplexer low registers. The order of the mapping is based on the default settings.

The polarity is currently set so that a high to low transition is recognized as an interrupt. This is controlled via the External interrupt polarity register.

### 11.3 Instant Boot Device / Application Quick Launch

Typically when a boot device controller resets it simply waits for commands over the computer bus (such as PCI). Since the boot device always resets before the computer bus starts sending commands this wait period is unproductive time. The initial commands inevitably instruct it to retrieve data from the boot device (such as a disk) for the operating system. Since most boot devices are relatively slow compared to the speed of most computer busses, a long delay is seen by the computer user. This is evident in the time it takes for a typical computer to boot.

The proposed boot device controller employs a dedicated IO channel processor with or without data compression to pre-load computer operating systems and applications.

Once the boot device controller is reset, it does not need to wait for commands over the computer bus. Instead it can proceed to pre-load the operating system from the boot device into local memory. Since the same parts of the operating system must be loaded upon each boot process there is no reason for the boot device controller to wait until it is commanded to load the operating system.

Using this method, when the computer system bus issues its first commands to the boot device controller asking for operating system data, the controller will already have the data in its local memory. The controller will then be able to instantly start transmitting the data to the computer system bus. There will be no need to wait for the relatively slow boot device. This practice will significantly reduce the time required during the computer boot process.

In addition to preloading operating system data the DSRA could also preload other data that the user would likely want to use at startup. An example of this would be a frequently used application such as a word processor and any number of document files.

There are two methods by the DSRA would know what data to preload from the boot device. The first method would allow the user to indicate what applications/data should be preloaded. The user would do this using a custom utility program.

The second method requires no input from the user. The DSRA would remember the first series of data requests it received after a power-on / reset. It would store this list of data to preload on the hard disk (or other storage device). Then upon each subsequent power-on / reset the DSRA would read that list and proceed to preload that data into its cache memory.

During the computer's boot up if a request for data is made which the boot device controller does not already have in local memory it will retrieve it from the boot device and deliver it to the computer bus. In addition, it will record what data was requested. Upon the next boot sequence, the boot device controller will pre-load that data into it's local memory along with everything it previously loaded.

During the computer's boot up if no request is made for a data block that was pre-loaded into the boot device controller's local memory then that data block will be removed from the list of data which is pre-loaded. Upon the next boot sequence, the boot device controller will not pre-load that data into it's local memory.

## 12.0 COMMAND PROTOCOL

### 12.1 Command List

The Realtime Data DSRA utilizes a set of six commands to implement all data storage, retrieval, and disk maintenance functions. Each command consists of eight thirty-two bit data words stored and transmitted in little endian format.

The commands are:

- Read Disk Data
- Write Disk Data
- Copy Disk Data
- Format Virtual Disk
- Disk Recovery
- Read Disk Status

The host computer commands the DSRA over the PCI Bus. Upon computer power-up or reset the host computer issues a PCI Bus Reset with a minimum pulse width of 100msec (See PCI Bus Specification Revision 2.2).

Upon completion of the PCI Bus reset the DSRA is fully initialized and waiting for the normal PCI configuration cycle. Upon completion of the configuration cycles the DSRA waits in an idle state awaiting the first disk command.

In operation, the operating system issues a command to the DSRA device driver to store, retrieve, or copy specific logical data blocks. Each command is transmitted over the PCI Data Bus at the Address assigned to the DSRA's Base Address Register.

Commands are issued to the DSRA with the Host Computer as the Bus Master and the DSRA as slave. When the DSRA is in the idle state it waits in slave mode for a command via the PCI Bus. Once a full command is received and stored in the DSRA's PCI Bus input FIFO (eight thirty-two bit data words) the DSRA does not accept additional command data and aborts any further bus cycles (without storing data) by deasserting target ready. It should be noted that the DSRA's PCI Bus interface housed in the Xilinx XCS40XL gate array.

Once a full command is received the Xilinx Interrupts the C6211 DSP by asserting the Data Command Available Interrupt. The FIFO is read across the DSP's external memory interface and is stored within the command queue located at TBD by DSP software. The interrupt is cleared by writing to the Xilinx at memory location TBD. Once the Command Available Interrupt is cleared the PCI Bus asserts Target Ready and awaits the next command.

The general format for each command is:

31	16	15	0	
Command Packet Number		Command Type		00h
Command Parameters				04h
Command Specific		Command Specific		08h
Command Specific		Command Specific		0Ch
Command Specific		Command Specific		10h
Command Specific		Command Specific		14h
Command Specific		Command Specific		18h
Checksum		Reserved		1Ch

00000000000000000000000000000000



### 12.1.1 Read Disk Data

31	16	15	8	7	0	
Command Packet Number 0000h to FFFFh		Command Type 00h		Command Parameters (00h)		00h
Starting Block Address (Least Significant Word)						04h
Starting Block Address (Most Significant Word)						08h
Number of Blocks (Least Significant Word)						0Ch
Number of Blocks (Most Significant Word)						10h
Destination Address (Least Significant Word)						14h
Destination Address (Most Significant Word)						18h
Checksum			Reserved			1Ch

00000000000000000000000000000000

61

### 12.1.2 Write Disk Data

31	16	15	8	7	0	
Command Packet Number 0000h to FFFFh		Command Type 01h		Command Parameters (00h)		00h
Starting Block Address (Least Significant Word)						04h
Starting Block Address (Most Significant Word)						08h
Source Address (Least Significant Word)						0Ch
Source Address (Most Significant Word)						10h
Source Length (Least Significant Word)						14h
Source Length (Most Significant Word)						18h
Checksum			Reserved			1Ch

00E0E0"4TT0A0S

62

### 12.1.3 Copy Disk Data

31	16	15	8	7	0	
Command Packet Number 0000h to FFFFh		Command Type 02h		Command Parameters (00h)		00h
Starting Block Address (Least Significant Word)						04h
Starting Block Address (Most Significant Word)						08h
Number of Blocks (Least Significant Word)						0Ch
Number of Blocks (Most Significant Word)						10h
Destination Block Address (Least Significant Word)						14h
Destination Block Address (Most Significant Word)						18h
Checksum			Reserved			1Ch

00000000"FFFFF0

63

### 12.1.4 Format Virtual Disk

31	16	15	8	7	0	
Command Packet Number 0000h to FFFFh		Command Type 10h		Command Parameters (00h)		00h
						04h
						08h
						0Ch
						10h
						14h
						18h
Checksum			Reserved			1Ch

0000h-FFFFh

64

### 12.1.5 Disk Recovery

31	16	15	8	7	0	
Command Packet Number 0000h to FFFFh		Command Type 20h		Command Parameters (00h)		00h
						04h
						08h
						0Ch
						10h
						14h
						18h
Checksum			Reserved			1Ch

65

### 12.1.6 Read Disk Status

31	16	15	8	7	0	
Command Packet Number 0000h to FFFFh		Command Type 30h		Command Parameters (00h)		00h
						04h
						08h
						0Ch
						10h
						14h
						18h
Checksum			Reserved			1Ch

00000000000000000000000000000000

66

## 12.2 Command and Data Interface

Both Commands to Data Storage and Retrieval Accelerator (DSRA) along with data to and from the DSRA are communicated via the 32 bit, 33MHz, PCI Data Bus. The DSRA's PCI Interface is housed within the onboard Xilinx Field Programmable Gate Array (FPGA). The DSRA employs a Xilinx Spartan XCS40XL-4 40,000 field programmable gate array which instantiates a Xilinx PCI 32, 32 Bit, 33MHz PCI Bus Interface (PCI Bus Revision 2.2).

The PCI Bus interface operates in Slave Mode when receiving Commands and as a Bus Master when reading or writing data. The source and destination for all data is specified within each command packet.

When setting up data transfers, the Enhanced Direct Memory Access (EDMA) Controller utilizes two Control Registers, a 16 Word Data Write to PCI Bus FIFO, a 16 Word Data Read From PCI Bus FIFO, and a PCI Data Interrupt (PCIDATINT). The 32 Bit PCI Address Register holds either the starting Source Address for DSRA Disk Writes where data is read from the PCI Bus, or the starting Destination Address for DSRA Disk Reads where data is written to the PCI Bus. The PCI Address Register also clears PCIDATINT each time it is written to within DSP Memory Space at 0xB000 0900.

### PCI Address Register

31	0
PCI Bus Source or Destination Address	
W, +0x0000	

The second Control Register is the PCI Count Register that specifies the direction of the data transfer along with the number of 32 Bit Data Words to be written to or from the PCI Bus. In order to accommodate EDMA operations, the PCI Count Register is located contiguously in DSP Memory Space at address location 0xB000 0904.

### PCI Count Register

31	30	24	23	0
Dir	Reserved		Word Transfer Count	
W, +0x0000				

The Most Significant Bit (Bit 31) specifies the direction of the PCI Bus Transfer.

Bit 31 = 0      Read from the PCI Bus

Bit 31 = 1      Write to the PCI Bus

The PCI Count Register is also used to assert PCIDAT INT during PCI Bus Data Write Transactions, however it does not assert PCIDATINT during PCI Bus Data Reads. Please refer to the upcoming descriptions of PCI Bus Write and Read Operation for further discussion.

## PCI Data Write FIFO

Data is written to the PCI Bus from the DSP via the 16 Word PCI Data Write FIFO located within address range of 0xB000 0A00 through 0xB000 0DFC.

Data writes from the DSP to anywhere within this address range place that data word in the next available location within the FIFO. Writes to the PCI Data Write FIFO are also used to clear the PCIDATINT Interrupt. Please refer to the upcoming descriptions of PCI Bus Write Operation for further discussion.

## PCI Data Read FIFO

Data is read from the PCI Bus to the DSP via the 16 Word PCI Data Read FIFO located within address range of 0xB000 0E00 through 0xB000 11FC.

Data read by the DSP from anywhere within this address range provides the next data word from the FIFO. When any word is read from the PCI Data Read FIFO the PCIDATINT Interrupt is Cleared. Please refer to the upcoming descriptions of PCI Bus Read Operation for further discussion.



### 12.2.1 Data Write Transaction

PCI Bus Data Write Transactions occur according to the following protocol:

1. The DSP EDMA writes the start address for the PCI Bus into the PCI Address Register which also clears PCIDATINT.
3. The DSP EDMA writes the direction bit (MSB write equal to 1 ) concurrent with the word count (bits 23 through 0) into the PCI Count Register which then asserts PCIDATINT. This also resets the PCI Data Write FIFO Count to “zero” words.
3. The DSP EDMA writes the first 16 words into the PCI Data Write FIFO which also clears PCIDATINT. (*PCIDATINT should already be set and it is cleared on each write into the FIFO*)
5. After the 16th Word is written into the FIFO by the DSP EDMA, the Xilinx PCI Bus Interface writes 16 words held within the FIFO into the PCI Bus and then subsequently asserts PCIDATINT. (The Word Count is also decremented with each data word written to the PCI Bus.)
5. The DSP EDMA writes the next 16 words into the PCI Data Write FIFO which also clears PCIDATINT. (*PCIDATINT should already be set and it is cleared on each write into the FIFO*)
8. After the next 16<sup>th</sup> Word is written into the FIFO by the DSP EDMA, the Xilinx PCI Bus Interface Writes 16 words held within the FIFO into the PCI Bus and then asserts PCIDATINT. (The Word Count is again decremented with each data word written to the PCI Bus.)
9. Repeat Steps 5 & 6 until the last 16 words.
8. The DSP EDMA writes the last 16 words into the PCI Data Write FIFO which also clears PCIDATINT. (*PCIDATINT should already be set and it is cleared on each write into the FIFO*)
9. After the last 16<sup>th</sup> Word is written into the FIFO by the DSP EDMA, the Xilinx PCI Bus Interface Writes 16 words held within the FIFO into the PCI Bus and then asserts PCIDATINT. (The Word Count should be decremented with each data word written to the PCI Bus down to zero.)
10. To complete the transfer the DSP EDMA writes any start address into the PCI Address Register which also clears PCIDATINT. Alternately, the transfer may be chained by writing the next Start Address to the PCI Address Register and proceeding to Step 2. (It should be noted that this also clears PCIDATINT.)

### 12.2.1 Data Read Transaction

PCI Bus Data Read Transactions occur according to the following protocol:

1. The DSP EDMA writes the start address for the PCI Bus into the PCI Address Register which also clears PCIDATINT.
2. The DSP EDMA writes the direction bit (MSB write equal to 0 ) concurrent with the word count (bits 23 through 0) into the PCI Count Register. It should be noted that this does not affect or assert PCIDATINT. This also resets the PCI Data Read FIFO Count to “zero” words.
3. The Xilinx PCI Bus Interface then reads 16 Words starting at the address specified in the PCI Address Register. After the 16<sup>th</sup> Word is read and placed into the FIFO, PCIDATINT is asserted. (Word Count is also decremented with each data word read from the PCI Bus.)
4. The DSP EDMA then reads the 16 words from the PCI Data Read FIFO which also clears PCIDATINT. (*PCIDATINT should already be set and it is cleared on each read from the FIFO*)
5. After the 16<sup>th</sup> Word is read from the FIFO by the DSP EDMA, the Xilinx PCI Bus Interface reads the next 16 words from the PCI Bus and places them in the PCI Data Read FIFO. After the 16<sup>th</sup> Word is read and placed into the FIFO, PCIDATINT is asserted. (Word Count is also decremented with each data word read from the PCI Bus.)
6. The DSP EDMA then reads the next 16 words from the PCI Data Read FIFO which also clears PCIDATINT. (*PCIDATINT should already be set and it is cleared on each read from the FIFO*)
7. Repeat Steps 5 & 6 until the last 16 words.
8. The Xilinx PCI Bus Interface reads the last 16 words from the PCI Bus and places them in the PCI Data Read FIFO. After the last 16<sup>th</sup> Word is read and placed into the FIFO, PCIDATINT is asserted. (Word Count is also decremented with each data word read from the PCI Bus and should be zero.)
10. The DSP EDMA then reads the last 16 words from the PCI Data Read FIFO which also clears PCIDATINT. (*PCIDATINT should already be set and it is cleared on each read from the FIFO except the last word*). When the last word is read, (last read by DSP EDMA & Counter = 0), PCIDATINT is asserted.
10. To complete the transaction the DSP EDMA writes any start address into the PCI Address Register which also clears PCIDATINT. Alternately, the transfer may be chained by writing the next Start Address to the PCI Address Register and proceeding to Step 2. (It should be noted that this also clears PCIDATINT.)

### 12.3 Command Protocol Errors

Command Errors are detected by the DSP when each queued command is parsed and syntactically and lexically analyzed. It is up to the DSP Disk Command software and Host Device Driver / File System to develop a protocol for handling Command Protocol Errors.

This section is TBD

## 12.4 Command Acquisition

After completion of the Xilinx initialization by the DSP and subsequent negation of the PCI Bus Reset signal (RST#) by the host computer's PCI Bridge, the DSRA is ready to accept commands from the host computer via the PCI Bus. When accepting commands it should be noted that the DSRA is a PCI Target (Slave) Device. Commands are fixed in length at exactly 8 (thirty-two bit) words long. Commands are written from the host computer to the DSRA via the PCI Bus utilizing the DSRA's Base Address Register 0 (BAR0). The PCI Bus Reset initially sets the Command FIFO's Counter to zero and also signals the Xilinx's PCI Bus State Controller that the Command FIFO is empty and enable to accept a command. To handle error conditions, the command FIFO may also be cleared by reading or writing to location B000 0080 within the PCI Control Register Space mapped into the DSP's CE3 (Xilinx) memory space. Additionally, it should be noted that the PCI Reset ensures that the Command Available Interrupt to the DSP is negated.

Whenever a data write occurs within the valid data range of BAR0, the data word is accepted from PCI Bus and placed in the next available memory position within the Command FIFO. When the last of the 8 thirty-two bit data words is accepted by the PCI Bus (thus completing the command, i.e. last word for the command FIFO to be full), the PCI Bus State Controller is automatically set to Target Abort (within same PCI Transaction) or Disconnect Without Data for all subsequent PCI transactions that try to writes to BAR0. This automatic setting is the responsibility of the Xilinx PCI Data Interface.

The PCI Command FIFO State Controller then asserts the Command Available Interrupt to the DSP (jay assign interrupt #). The DSP services the Command Available Interrupt by reading the command data from the address range address B000 0000 to address B000 0020. It should be noted that the command FIFO is read sequentially from any data access that reads data within the aforesated address range. It is the responsibility of the DSP to understand that the data is read sequentially from any order of accesses within the data range and should thus be stored accordingly.

Upon completion of the Command Available Interrupt Service Routine the DSP executes a memory read or write to location B000 0080 within the PCI Control Register Space mapped into the DSP's CE3 (Xilinx) memory space. This resets the Command FIFO Counter back to zero. Next, the DSP executes a memory read or write to location B000 0084 (DSP Memory Space) that clears the Command Available Interrupt. Nested interrupts are not possible since the PCI Bus State Machine is not yet able to accept any Command Data at BAR0. Once the Command Available Interrupt routine has cleared the interrupt and exited, the DSP may then enable the PCI State Machine to accept a new command by reading or writing to PCI Command Enable location (TBD) within the PCI Command FIFO Control Register Space. This architecture has been selected to enable the DSRA to operate on one command at a time or to accept multiple prioritized commands in future implementations. Specifically, the decoupling of the Command Available Interrupt Service Routine from the PCI State Machine that accepts Commands at BAR0 enables the DSP's "operating system kernel" to accept additional commands at any time by software command. In single command operation, a command is accepted, the Command Available Interrupt Cleared, and the Command executed by the DSRA in PCI Master Mode prior to the enabling of the PCI State machine to accept new commands.

In a prioritized multi-command implementation, the "operating system kernel" may elect to immediately accept new commands or defer the acceptance of new commands based upon any software implemented decision criteria. In one scenario, the O/S code might only allow a pre-specified number of commands to

be queued. In another scenario, commands might only be accepted during processor idle time or when the DSP is not executing time critical (i.e. highly pipelined) compress/decompress routines. In yet another scenario, various processes are enabled based upon a pre-emptive prioritized based scheduling system.

## 12.5 Command Execution

As previously stated, the DSRA retrieves commands from the input command FIFO in 8 thirty-two bit word packets. Prior to command interpretation and execution, a command's checksum value is computed to verify the integrity of the data command and associated parameters. If the checksum fails, the host computer is notified of the command packet that failed utilizing the Command Protocol Error Handler. Once the checksum is verified the command type and associated parameters are utilized as an offset into the command "pointer" table or may other suitable command/data structure that transfers control to the appropriate command execution routine.

Commands are executed by the DSRA with the DSRA acting as a PCI Master. This is in direct contrast to command acceptance where the DSRA acts as a PCI Slave. When acting as a PCI Bus Master, the DSRA reads or writes data to the PCI Bus utilizing a separate PCI Bus Data FIFO (distinct & apart from the Command FIFO). The PCI Data FIFO is 64 (thirty-two bit) words deep and may be utilized for either data reads or data writes from the DSP to the PCI Bus, but not both simultaneously.

For data to be written from the DSRA to the Host Computer, the DSP must first write the output data to the PCI Bus Data FIFO:

The Data FIFO is commanded to PCI Bus Data Write Mode by writing to location BFF0 0100 within the Xilinx (CE3) PCI Control Register Space. Upon PCI Bus Reset the default state for the PCI Data FIFO is write mode and the PCI Data FIFO Available Interrupt is cleared. The PCI Data FIFO Available Interrupt should also be software cleared by writing to location aaaa+0x40. We are thus assuming that the first task for the DSRA is for system boot-up or application code to be downloaded from disk. For reference, PCI Data Read Mode is commanded by writing to location BFF0 0104. The PCI Bus Reset initializes the Data FIFO Pointer to the first data of the 64 data words within the FIFO. However this pointer should always be explicitly initialized by a memory write to location BFF0 0108. This ensures that the first data word written to the FIFO by the DSP performing the data write anywhere in address range B000 0000 to B000 01FF is placed at the beginning of the FIFO. Each subsequent write to any location within this address range then places one thirty-two bit data word into the next available location within the PCI Data FIFO. The FIFO accepts up to 64 thirty-two bit data words although it should be clearly understood that not all data transfers to and from the PCI Bus will consist of a full FIFO. Counting the number of thirty-two bit data words written to the PCI Data FIFO is the responsibility of the DSP Code. It is envisioned that the DSP will, in general, use 64 word DMA data transfers, thus alleviating any additional processor overhead.

When the data has been transferred from the DSP to the PCI Data FIFO, the PCI Bus Controller also needs the address of the PCI Target along with the number of data words to be transmitted. In the current DSRA implementation, the PCI Bus Address is thirty-two bits wide, although future PCI bus implementations may utilize multiword addressing and/or significantly larger (64 bit & up) address widths. The single thirty-two bit address word is written by the DSP to memory location aaaa+0x10 in the PCI Control Register Space. We also reserved locations aaaa+0x14 through aaaa+0x2F for future address bus expansion.

Finally, the PCI Bus Data Write transaction is initiated by writing the PCI Data FIFO word count to memory address  $aaaa+0x30$ . The word count value is always decimal 64 or less (0x3F). When the count register is written the value is automatically transferred to the PCI Controller for executing the PCI Bus Master writes.

When the PCI Bus has completed the transfer of all data words within the PCI Data FIFO the PCI Data FIFO Available Interrupt is set. (jay assign an interrupt number). The DSP PCI Data FIFO Available Interrupt handler will then check to see if additional data is waiting or expected to be written to the PCI Data Bus. If additional data is required the interrupt is cleared and the data transfer process repeats. If no additional data is required to be transferred then the interrupt is cleared and the routine must exit to a system state controller. For example, if the command is complete then master mode must be disabled and then slave mode (command mode) enabled – assuming a single command by command execution DSRA. The operating state system will be fully discussed in a separate section.

For data to be Read by the DSRA from the Host Computer the DSP must command the PCI Bus with the address and quantity of data to be received.

The PCI Data FIFO is commanded to PCI Bus Data Read Mode by writing to location  $aaaa+0x4$  within the Xilinx (CE3) PCI Control Register Space. Upon PCI Bus Reset the default state for the PCI Data FIFO is Write Mode and the PCI Data FIFO Full Interrupt is Cleared. The PCI Data FIFO Full Interrupt should also be cleared via software by writing to location  $aaaa+0x44$ . The PCI Bus Reset also initializes the PCI Data FIFO Pointer to the first data word of the available 64 data words within the FIFO. However this pointer should always be explicitly initialized by a memory write to location  $aaaa+0x48$  (jay assign address).

In order for data to be read from the PCI Bus by the DSRA the Xilinx PCI Bus Controller requires the address of the PCI Target along with the number of data words to be received. In the current DSRA implementation, the PCI Bus Address is thirty-two bits wide, although future PCI bus implementations may utilize multiword addressing and/or significantly larger (64 bit & up) address widths. The single thirty-two bit address word is written by the DSP to memory location  $aaaa+0x10$  in the PCI Control Register Space. We have also reserved locations  $aaaa+0x14$  through  $aaaa+0x2F$  for future address bus expansion.

Finally, the PCI Bus Data Read transaction is initiated by writing the PCI Data FIFO word count to memory address  $aaaa+0x4C$ . The word count value is always decimal 64 or less (0x3F). When the count register is written the value is automatically transferred to the PCI Controller for executing the PCI Bus Master Read.

When the PCI Bus has received all the requested data words PCI Data FIFO Full Interrupt is set. (jay assign an interrupt number). The DSP PCI Data FIFO Full Interrupt handler will then check to see if additional data is waiting or expected to be read from the PCI Data Bus. If additional data is required the interrupt is cleared and the data receipt process repeats. If no additional data is required to be transferred then the interrupt is cleared and the routine must exit to a system state controller. For example, if the command is complete then master mode must be disabled and then slave mode (command mode) enabled – assuming a single command by command execution DSRA. The operating state system will be fully discussed in a separate section.

74

It is clearly understood that there are more efficient means for handling the PCI Data transfers. The current methodology has been selected to minimize the complexity and resource utilization of the Xilinx Gate Array. It should also be understood that the utilization of asynchronous memory reads and writes to initialize system states and synchronize events at a software level aids in both hardware and system level debug at the expense of increase software overhead. Subsequent embodiments of the gate array may automate resource intensive tasks if system level performance mandates.

## READ DISK DATA

The Read Disk Data Command is executed by interpreting the

**Starting Block Address** - identify the starting block address for reading disk data. It should be noted that the starting block address parameter is a 64 bit value that by far exceeds the number of bits in most operating system's file allocation tables (for example FAT32 and NTFS). The additional bits provide for increased expansion across multiple disks, the provision to accommodate future increased capacity disks, along with the ability to provide finer granularity than current file systems.

**Number of Blocks** – identify the number of blocks to be read. The number of blocks parameter is inclusive of the first block (i.e. starting address). Thus if we are reading sixteen blocks of data 0xFF at starting block address 0x000000000000100 we would read from blocks 0x000000000000100 through 0x0000000000001FF.

**Destination Address** – identify the destination address for data read from disk. The destination address is the beginning target address for data read from disk, with data placed at increasing data addresses.

75

## 13.0 EPROM PROGRAMMING

These instructions indicate how to burn an EPROM that contains both DSP code and XILINX configuration data. If you only need one of them just leave out the other.

1. You must have a "\*.out" file from code composer and a "\*.bit" file from XILINX software.
2. Convert "\*.out" file to "\*.m0" file (Motorola S format). Open a DOS box and run the following:  
hex6x -memwidth 8 -romwidth 8 -map hex.map -m file.out  
The result will be a file called file.m0 which is an ASCII file.
3. Convert "\*.bit" file to "\*.raw" file (John Buck format). Open an DOS box and run the following:  
bitmass file.bit file.raw
4. Go to the EPROM programming machine and copy both the "\*.m0" and the "\*.raw" files to the C:\bo directory.
5. Open a DOS box and change to the C:\emp10 directory. Run the emp10 program. Running it from this directory uses an INI file that automatically selects the EPROM and the programming algorithm we are using.
6. Put a blank chip in the programmer and press 2 "Verify chip is erased".
7. Load the DSP code into the buffer.
  - A) Press <ALT>0 to configure the emp software for the data file. This sets 4 parameters as follows:

S.Buffer	00000000	0001FFFF
T.File	90000000	9001FFFF
U.File type	Motorola S	
V.Filename	C:\bo\romboot.m0	

You can then modify the filename (or any of the parameters).
  - B) Press 8 "Load file from disk".
8. Load the XILINX file into the buffer.
  - A) Press <ALT>1 to configure the emp software for the data file. This sets 4 parameters as follows:

S.Buffer	00014000	00033FFF
T.File	00000000	0001FFFF
U.File type	Binary	
V.Filename	C:\bo\romout.raw	

You can then modify the filename (or any of the parameters).
  - B) Press 8 "Load file from disk".
9. If desired you may press 7 "Buffer editor" to view or change the contents of the buffer. Note that the DSP and the XILINX files may be loaded in either order or one may be omitted.
10. Press 1 "Program with selected algorithm" to program the part.
11. Done.



**APPENDIX A  
BOARD LAYOUT**

77

**APPENDIX E  
SCHEMATICS**

NOTES, UNLESS OTHERWISE SPECIFIED:

- 1. RESISTANCE VALUES ARE IN OHMS.
- 2. CAPACITANCE VALUES ARE IN MICROFARADS.
- 3. HIGHEST REFERENCE DESIGNATOR USED: TOP (B SIDE) BOTTOM (A SIDE)
  - A. CERAMIC CAPS C157
  - B. TANTALUM CAPS CT34
  - C. DIODES D8
  - D. CONNECTORS/HEADERS J57
  - E. FILTER E1
  - F. RESISTORS R55
  - G. RESISTOR PACKS RP28
  - H. IC'S U16
  - I. CRYSTALS/OSCILLATORS TP1001 - 1036
  - J. TEST POINTS NUMBERING TP1101 - 1130
  - TP2001 - 2004

4. PARTS NOT INSTALLED ARE INDICATED WITH 'NU' THE FOLLOWING IS THE LIST OF UNINSTALLED PARTS:

RESISTORS: R34, R35, R38

DIODES: D5

IC'S U8, U11 (NOT POPULATED IN LOW COST 16 MB VERSION).

U6 (NOT POPULATED)

5. DESIGNATORS NOT USED

R36

REVISIONS

REV	DESCRIPTION	DATE	APPROVED
01	PRELIMINARY DESIGN	05.04.99	JJF
A	ENGINEERING REVIEW	05.10.99	JJF
B	MANUFACTURING REVIEW	05.18.99	JJF
C	RP's ADDED ON SHTS 2, 3, 7, 8, 9 & 13.	06.01.99	JJF
D	REMOVE SBSRAM, ADD 2ND BANK SDRAM & TEST POINTS ADD LOW COST MEMORY OPTION PAGES 24, 25, ADD INTA# IO_PIN_223 OF XILINX	06.10.99	JJF
E	ADD SHT 26, CLK GENERATORS, AECLK, JTAG OSC, XILINX ECLK & CES; DECOUPLE U12, Y1, Y2, ADD GRD POSTS, XILINX SPARE PIN TEST PTS.	07.06.99	JJF
F	PAGE 26; Y2 IS 10.00MHZ WAS 75.00MHZ, ADDED R55 68 OHM TO TESTCLK OFFPAGE AND FIX SIGNAL NAMES ON XCS40 PINS.	10.12.99	JJF
G		10.18.99	JJF

SHEET # TITLE

- 1. TITLE SHEET
- 2. CLOCK, PLL, RESET, INTERRUPTS
- 3. EXTERNAL MEMORY INTERFACE
- 4. EXTERNAL MEMORY INTERFACE SDRAM BANK 0
- 5. EXTERNAL MEMORY INTERFACE SDRAM BANK 1
- 6. EXTERNAL MEMORY INTERFACE - OTP EPROM
- 7. EXTERNAL MEMORY & CONTROL - XCS30
- 8. HOST PORT INTERFACE
- 9. ATA3 - INTERFACE
- 10. PCI - INTERFACE
- 11. ATA3 CONNECTOR
- 12. PCI CONNECTOR
- 13. PROGRAMMING PORT
- 14. POWER AND GROUND - C6211, XCS30
- 15. DECOUPLING - C6211
- 16. DECOUPLING - MEMORY & XCS30
- 17. DECOUPLING - PCIBUS
- 18. POWER REGULATORS
- 19. RESET
- 20. JTAG & EPROM HEADERS, GROUND PINS
- 21. HPI HEADER TEST CONNECTOR
- 22. DSP DATA & ADDRESS BUS TEST HEADERS
- 23. DSP & XCS30 TEST HEADERS
- 24. LOW COST MEMORY SDRAM BANK 0
- 25. LOW COST MEMORY SDRAM BANK 1
- 26. CLK GENERATORS

PATENTS PENDING.  
 COPYRIGHT © 1999.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.

REALTIME DATA COMPRESSION SYSTEMS™

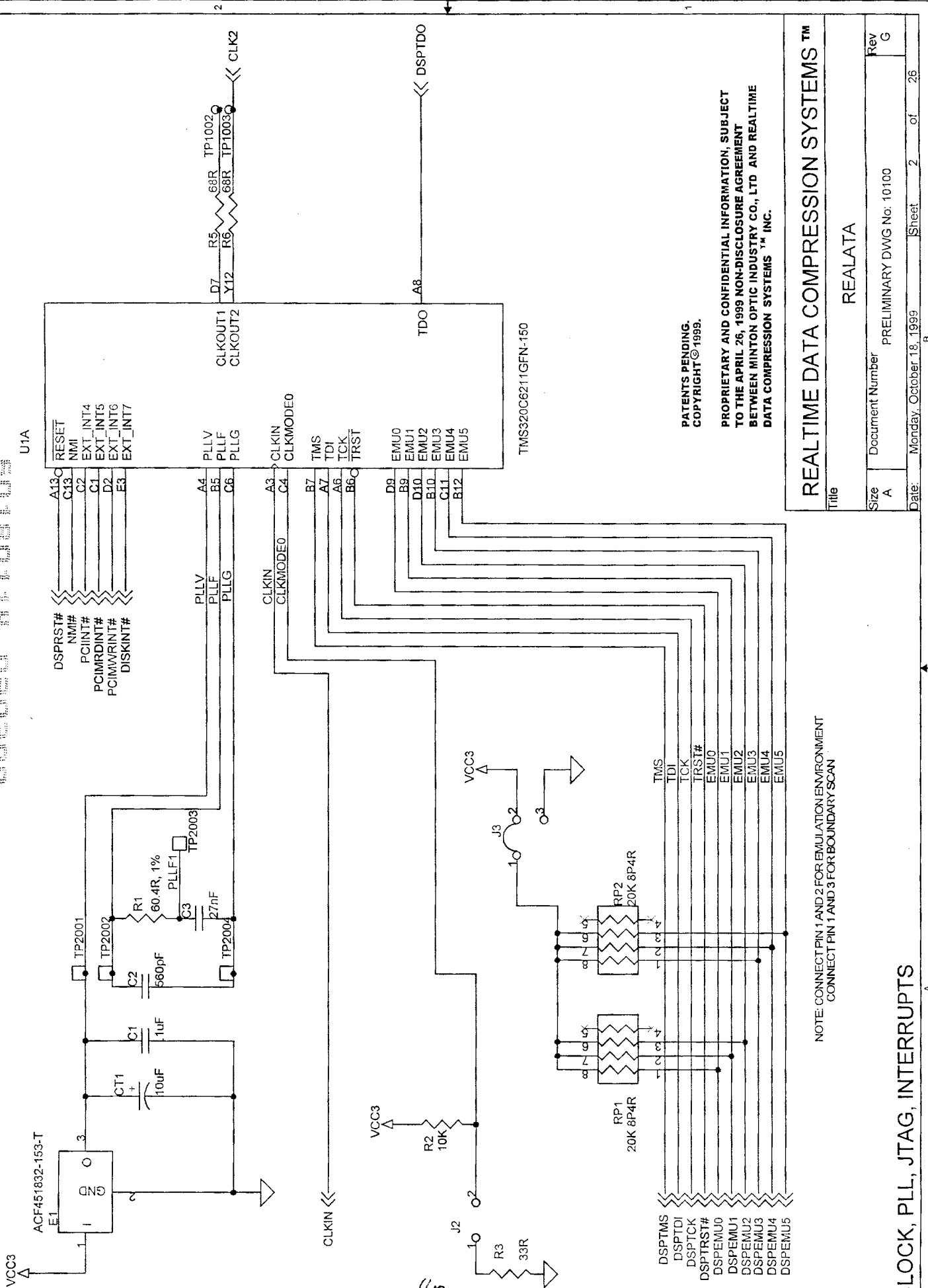
206E 63rd STREET, NEW YORK, NEW YORK 10021

REALATA

Document Number	PRELIMINARY DWG No: 10100	Rev	G
Date:	Monday, October 18, 1999	Sheet	1 of 26

REV	SHT	22	23	24	25	26				
REV	F	F	F	F	F	F	F	F	F	F
SHT	15	16	17	18	19	20	21			
REV	F	F	F	F	F	F	F	F	F	F
SHT	8	9	10	11	12	13	14			
REV	F	F	F	F	F	F	F	F	F	F
SHT	1	2	3	4	5	6	7			

# REALTIME 2010



PATENTS PENDING.  
COPYRIGHT © 1999.

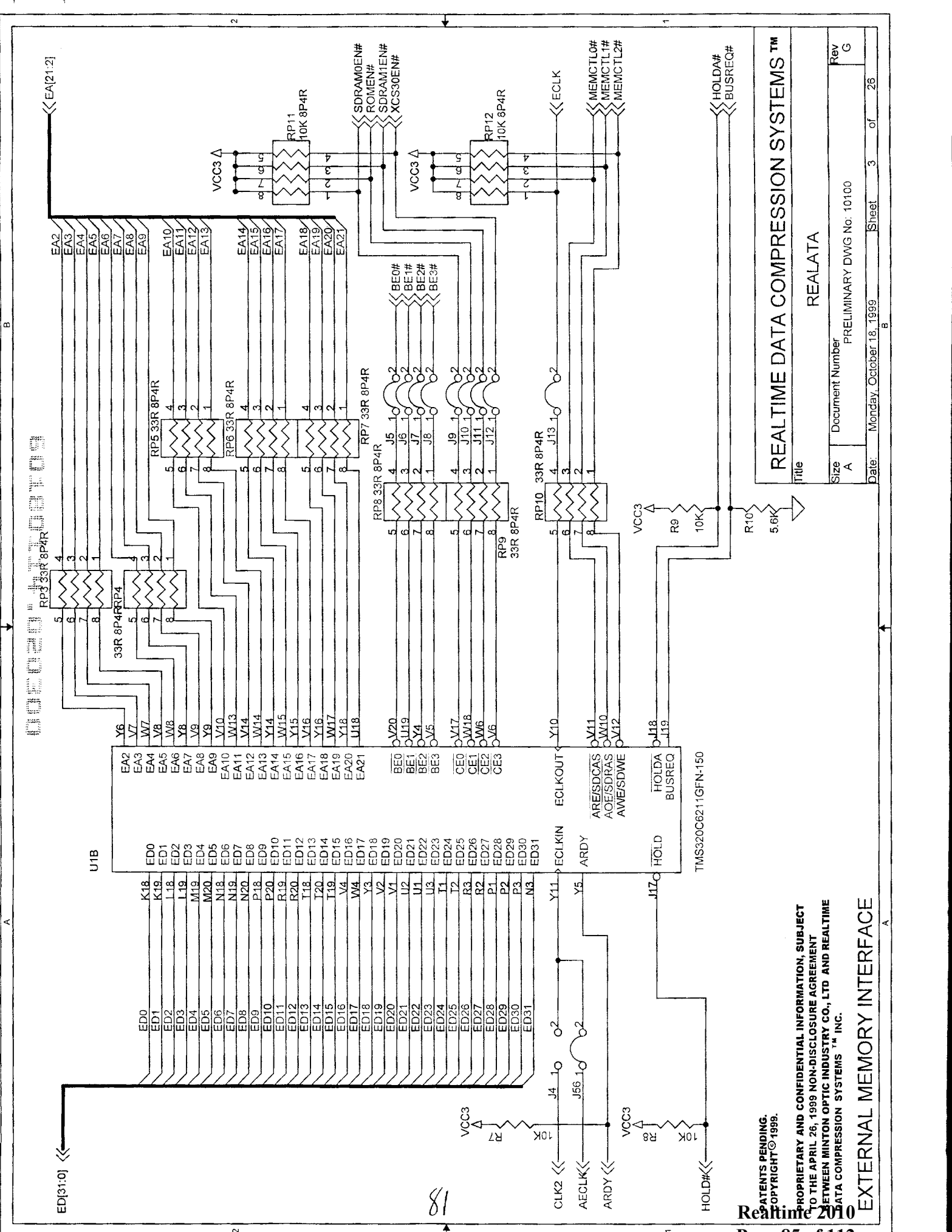
PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
DATA COMPRESSION SYSTEMS™, INC.

REALTIME DATA COMPRESSION SYSTEMS™

Title		REALATA	
Size	Document Number	PRELIMINARY DWG No. 10100	
A			
Date:	Monday, October 18, 1999	Sheet	2 of 26

NOTE: CONNECT PIN 1 AND 2 FOR EMULATION ENVIRONMENT  
CONNECT PIN 1 AND 3 FOR BOUNDARY SCAN

CLOCK, PLL, JTAG, INTERRUPTS



REALTIME DATA COMPRESSION SYSTEMS™	
Title	
Size	Document Number
A	PRELIMINARY DWG No. 10100
Rev	G
Date:	Monday, October 18, 1999
Sheet	3 of 26

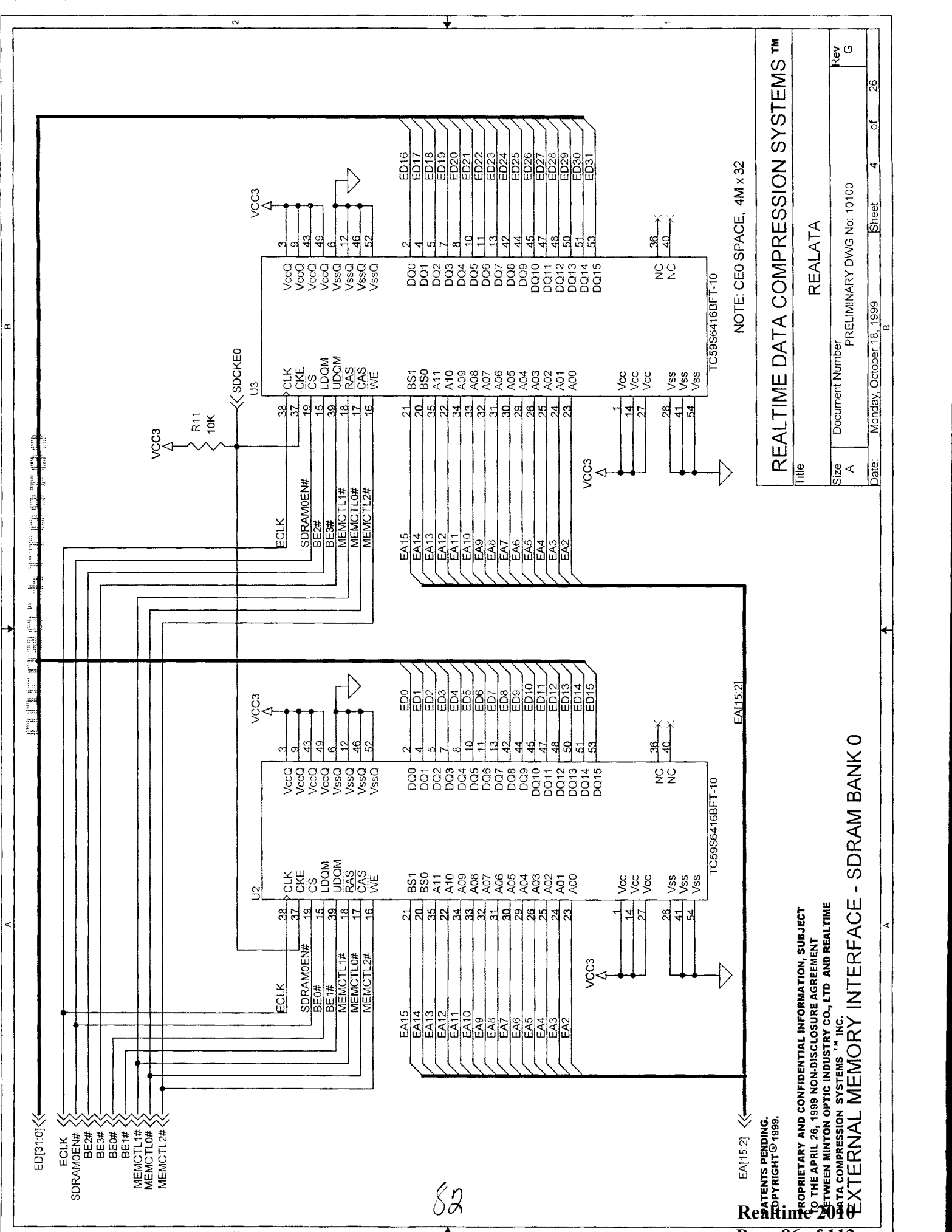
81

REALTIME DATA COMPRESSION SYSTEMS™

EXTERNAL MEMORY INTERFACE

PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME DATA COMPRESSION SYSTEMS™, INC.

Page 85 of 112

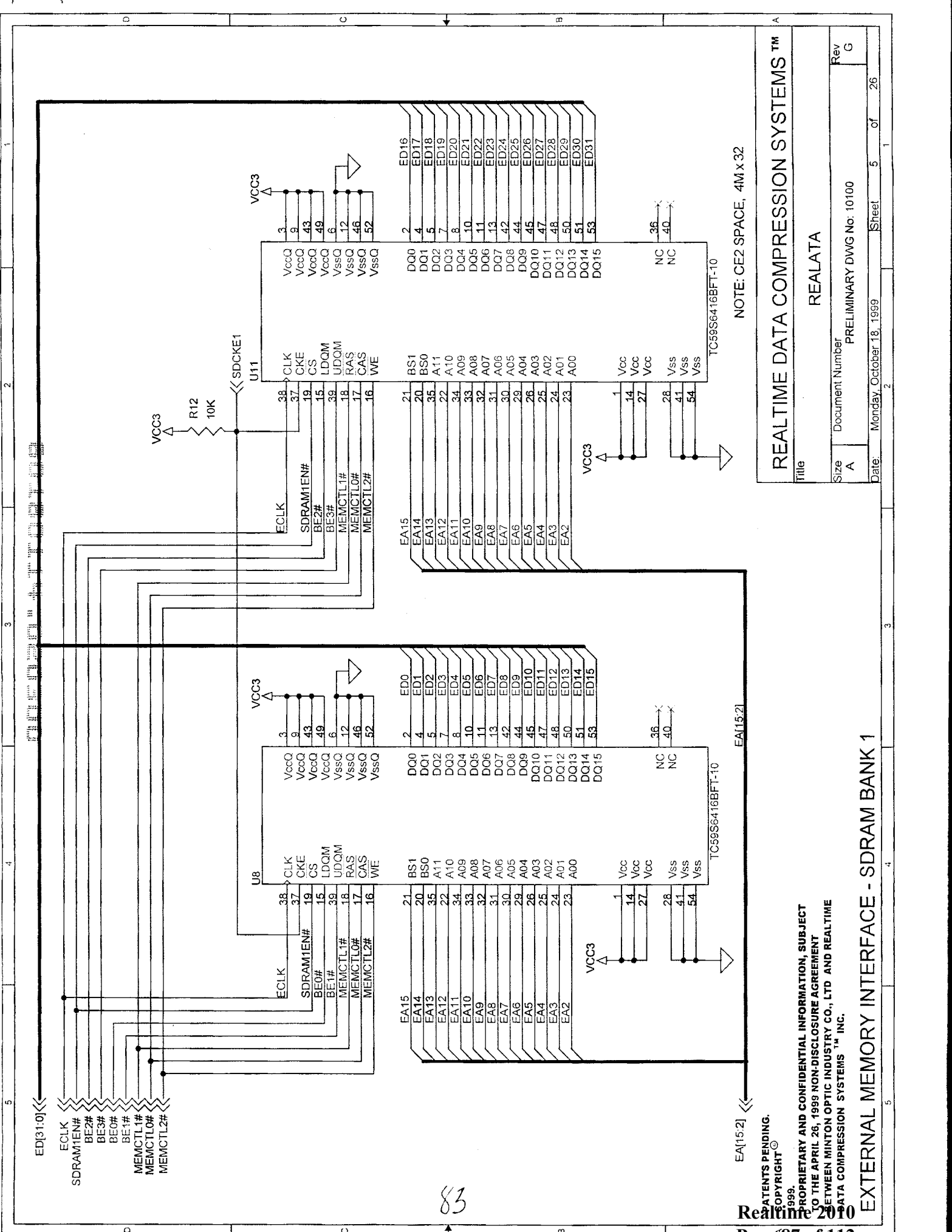


NOTE: CEO SPACE, 4M x 32

REALTIME DATA COMPRESSION SYSTEMS™	
Title REALATA	
Size A	Document Number PRELIMINARY DWG No: 10100
Date: Monday, October 18, 1999	Rev G
Sheet 4 of 26	

EXTERNAL MEMORY INTERFACE - SDRAM BANK 0

PATENTS PENDING.  
 COPYRIGHT © 1999.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.



NOTE: CE2 SPACE, 4M X 32

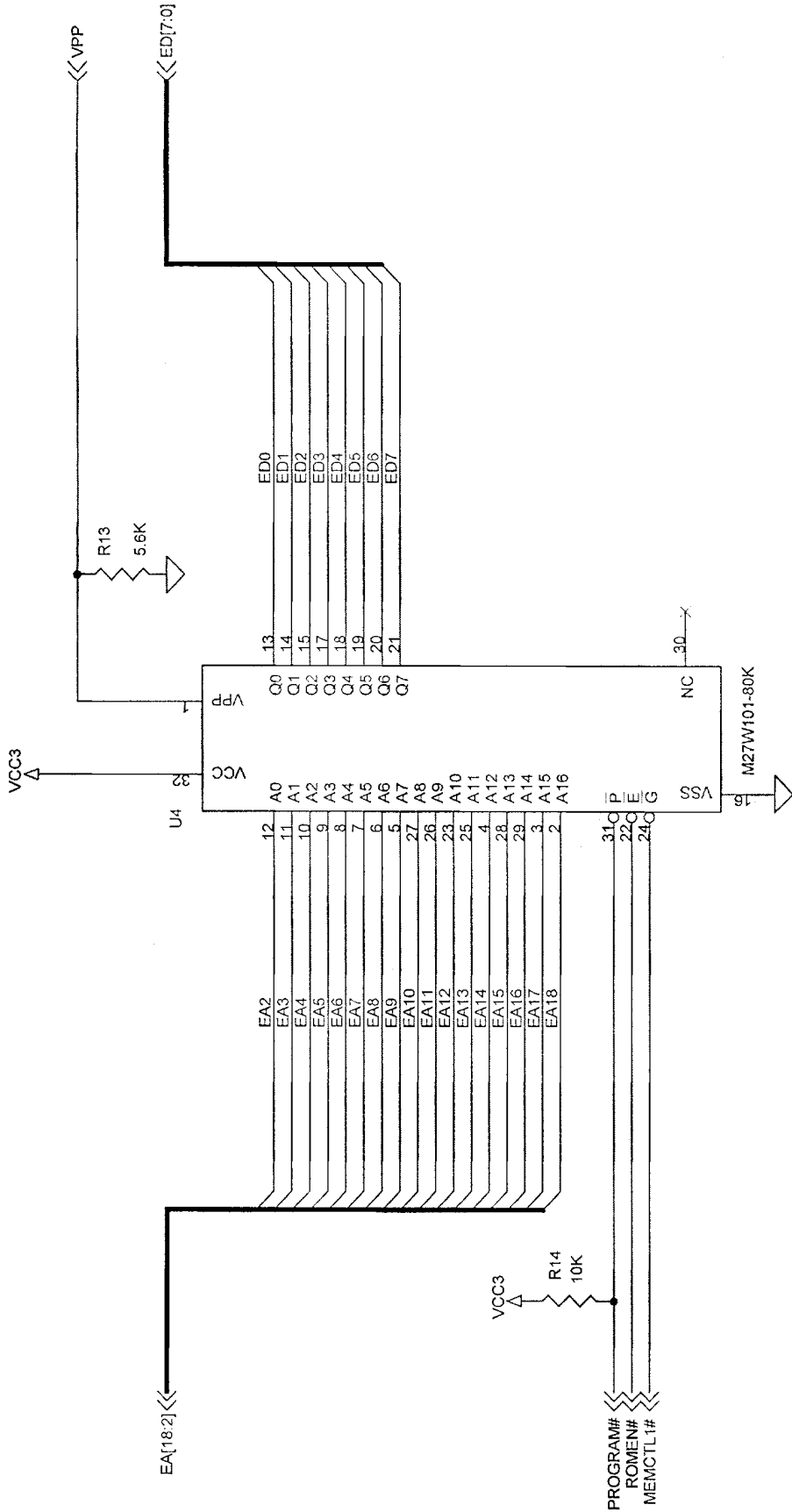
REALTIME DATA COMPRESSION SYSTEMS™	
Title: REALATA	
Size: A	Document Number: PRELIMINARY DWG No. 10100
Date: Monday, October 18, 1999	Rev: G
Sheet: 5	of 26

EXTERNAL MEMORY INTERFACE - SDRAM BANK 1

PATENTS PENDING.  
 COPYRIGHT © 1999.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.

83

REALTIME DATA COMPRESSION SYSTEMS



84

NOTE: CE1 SPACE, 128K x 8 ROM IS LENDIAN FORMAT

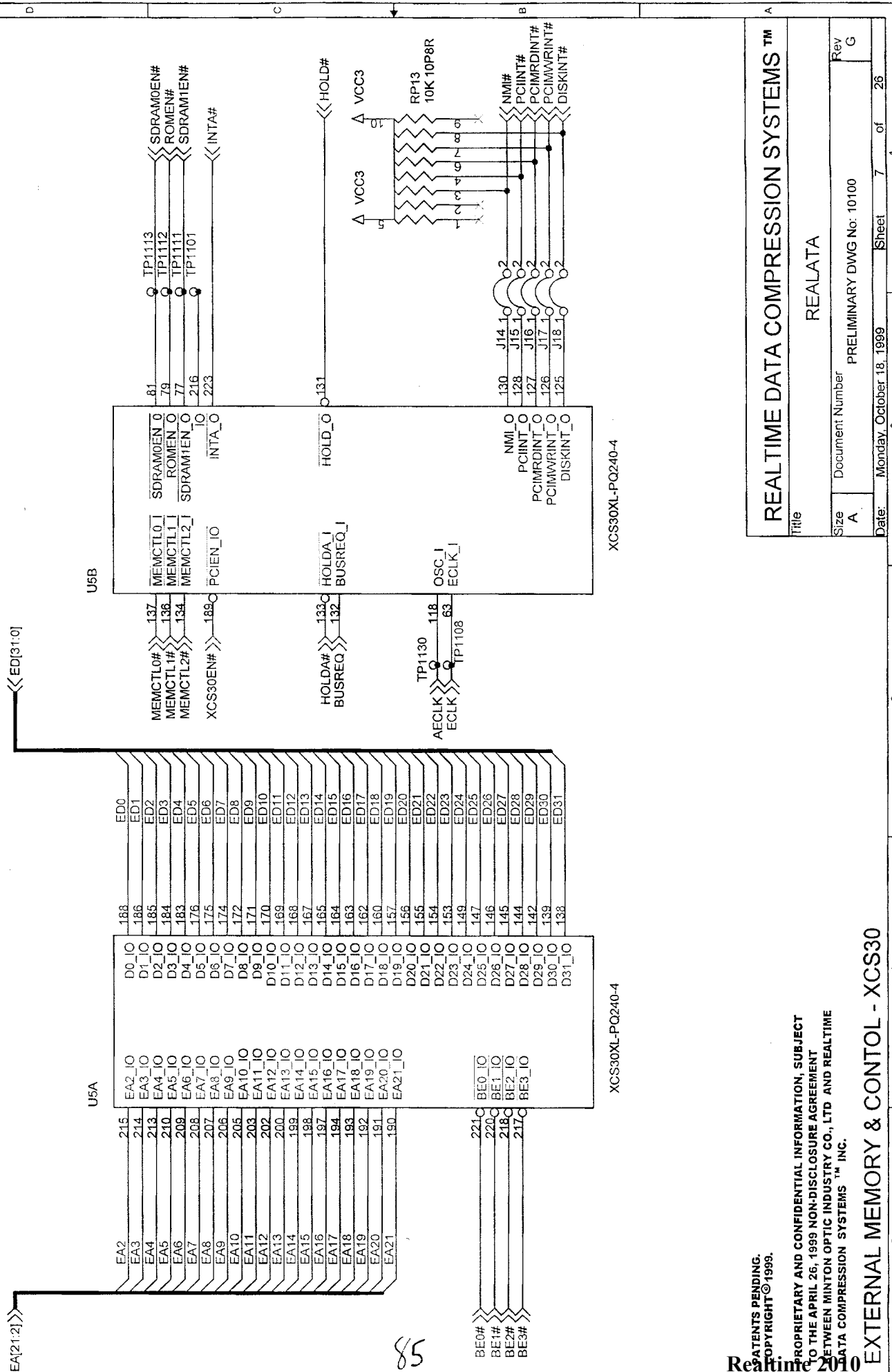
Title		REALATA
Size	Document Number	PRELIMINARY DWG No: 10100
Rev		G
Date:	Monday, October 18, 1999	Sheet 6 of 26

REALTIME DATA COMPRESSION SYSTEMS™  
 PATENTS PENDING.  
 COPYRIGHT © 1999.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.

EXTERNAL MEMORY INTERFACE - OTP EPROM

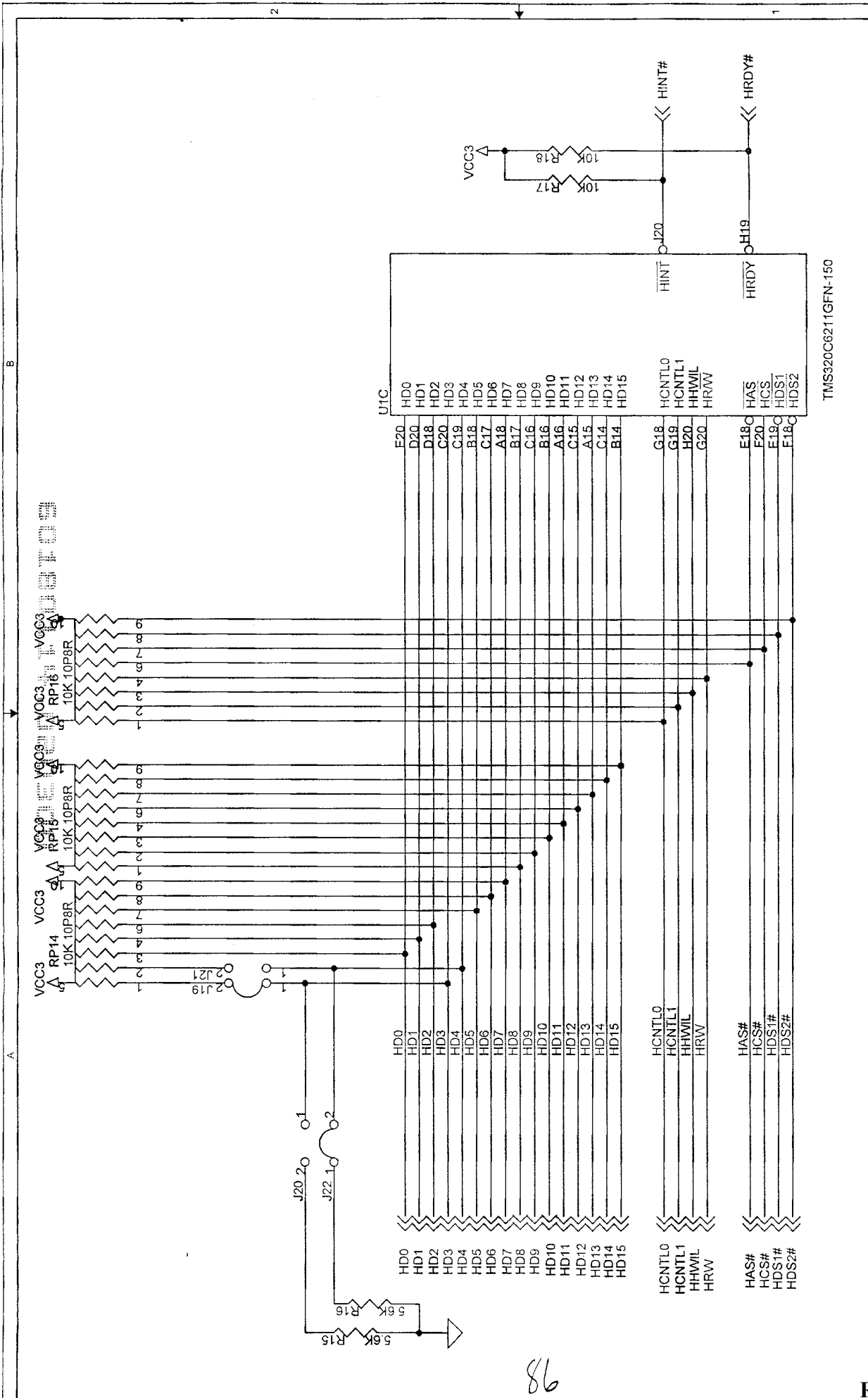


# REALTIME MEMORY & CONTROL



<b>REALTIME DATA COMPRESSION SYSTEMS™</b>	
REALATA	
Title	Rev G
Size A	Document Number PRELIMINARY DWG No: 10100
Date: Monday, October 18, 1999	Sheet 7 of 26

**EXTERNAL MEMORY & CONTROL - XCS30**  
 PATENTS PENDING.  
 COPYRIGHT © 1999.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.

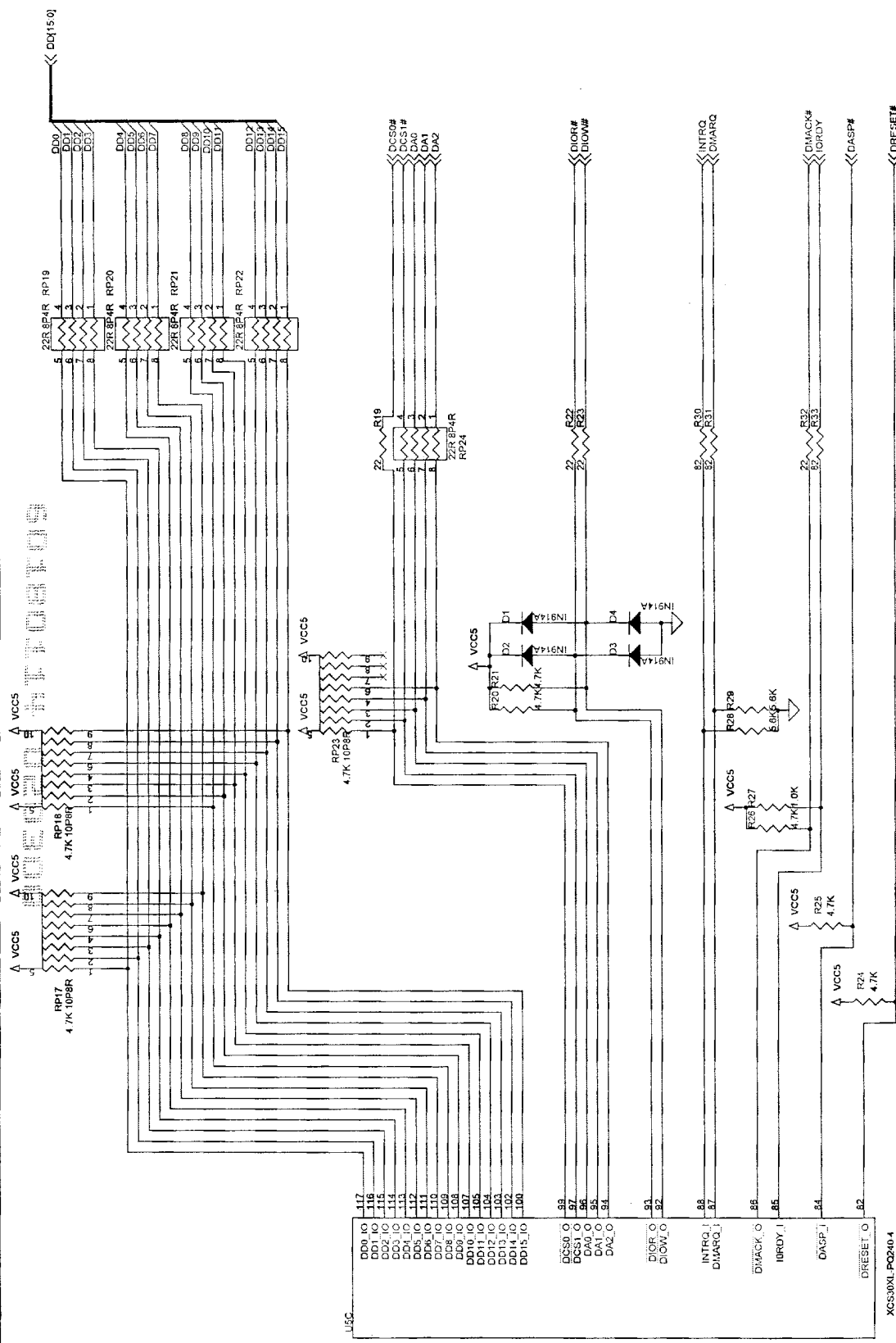


TMS320C6211GFN-150

REALTIME DATA COMPRESSION SYSTEMS™	
Title	
Size	Document Number
A	PRELIMINARY DWG No: 10100
Date:	Monday, October 18, 1999
Sheet	8 of 26
Rev	G

PATENTS PENDING.  
 COPYRIGHT © 1999.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.  
**Host Port Interface**

# REALTIME DATA COMPRESSION SYSTEMS™



REALTIME DATA COMPRESSION SYSTEMS™	
Title	
REALATA	
Size	Document Number
B	PRELIMINARY DWG No. 10100
Date	Date
Monday, October 18, 1999	Sheet 9 of 26
Rev	0

PATENTS PENDING.  
 COPYRIGHT © 1999.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1992 NON-DISCLOSURE AGREEMENT  
 BETWEEN REALATA, INC. AND REALTIME  
 DATA COMPRESSION SYSTEMS™, INC.

## ATA3 - INTERFACE

87

# CONNECTOR PINS

PCIAD[31:0] <<<

USD	USE
PCIAD0	AD-IO0
PCIAD1	AD-IO1
PCIAD2	AD-IO2
PCIAD3	AD-IO3
PCIAD4	AD-IO4
PCIAD5	AD-IO5
PCIAD6	AD-IO6
PCIAD7	AD-IO7
PCIAD8	AD-IO8
PCIAD9	AD-IO9
PCIAD10	AD-IO10
PCIAD11	AD-IO11
PCIAD12	AD-IO12
PCIAD13	AD-IO13
PCIAD14	AD-IO14
PCIAD15	AD-IO15
PCIAD16	AD-IO16
PCIAD17	AD-IO17
PCIAD18	AD-IO18
PCIAD19	AD-IO19
PCIAD20	AD-IO20
PCIAD21	AD-IO21
PCIAD22	AD-IO22
PCIAD23	AD-IO23
PCIAD24	AD-IO24
PCIAD25	AD-IO25
PCIAD26	AD-IO26
PCIAD27	AD-IO27
PCIAD28	AD-IO28
PCIAD29	AD-IO29
PCIAD30	AD-IO30
PCIAD31	AD-IO31
PCICBE0	CBE-IO0
PCICBE1	CBE-IO1
PCICBE2	CBE-IO2
PCICBE3	CBE-IO3
PAR#	PAR-IO

PCICBE[3:0] <<<

PAR# <<<

USE	USE
PCICLK <<<	PCICLK
PCIRST# <<<	RST_I
FRAME# <<<	FRAME_IO
TRDY# <<<	TRDY_IO
IRDY# <<<	IRDY_IO
STOP# <<<	STOP_IO
DEVSEL# <<<	DEVSEL_IO
IDSEL# <<<	IDSEL_IO
GNT# <<<	GNT_I
REQ_O <<<	REQ_O
PERR# <<<	PERR_IO
SERR# <<<	SERR_IO
TP1114	IO
TP1115	IO
TP1116	IO
TP1117	IO
TP1118	IO
TP1109	IO
TP1110	IO
TP1119	IO
TP1120	IO
TP1121	IO
TP1122	IO
TP1123	IO
TP1124	IO
TP1125	IO
TP1126	IO
TP1127	IO
TP1128	IO
TP1129	IO
TP1103	IO
TP1104	IO
TP1105	IO
TP1106	IO
TP1107	IO
NC	NC

XCS30XL-PQ240-4

XCS30XL-PQ240-4

PATENTS PENDING.  
COPYRIGHT © 1999.

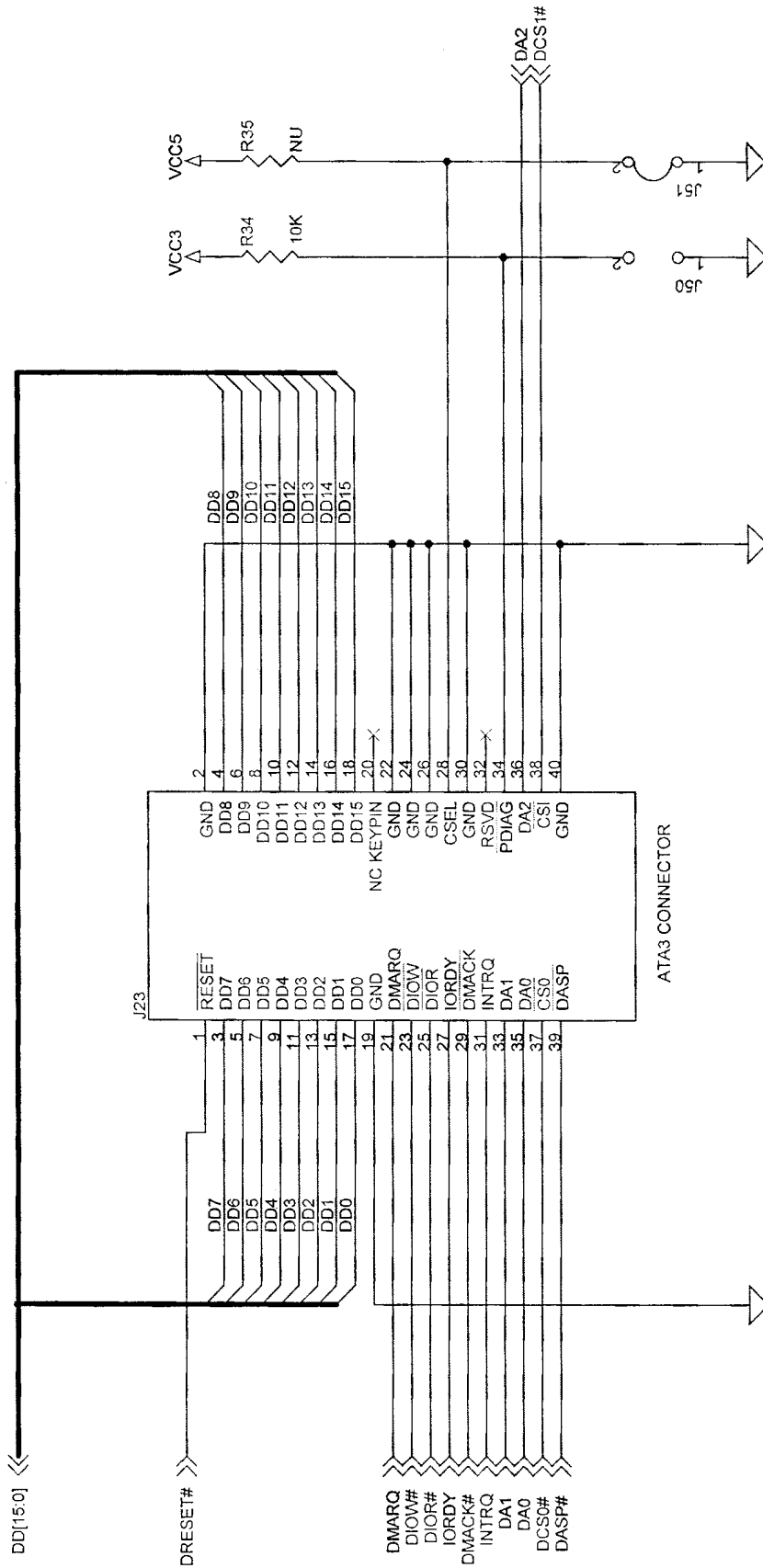
PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
DATA COMPRESSION SYSTEMS™, INC.

REALATA

Title	Document Number	PRELIMINARY DWG No: 10100
Size	A	Rev G
Date:	Monday, October 18, 1999	Sheet 10 of 26

PCI - INTERFACE

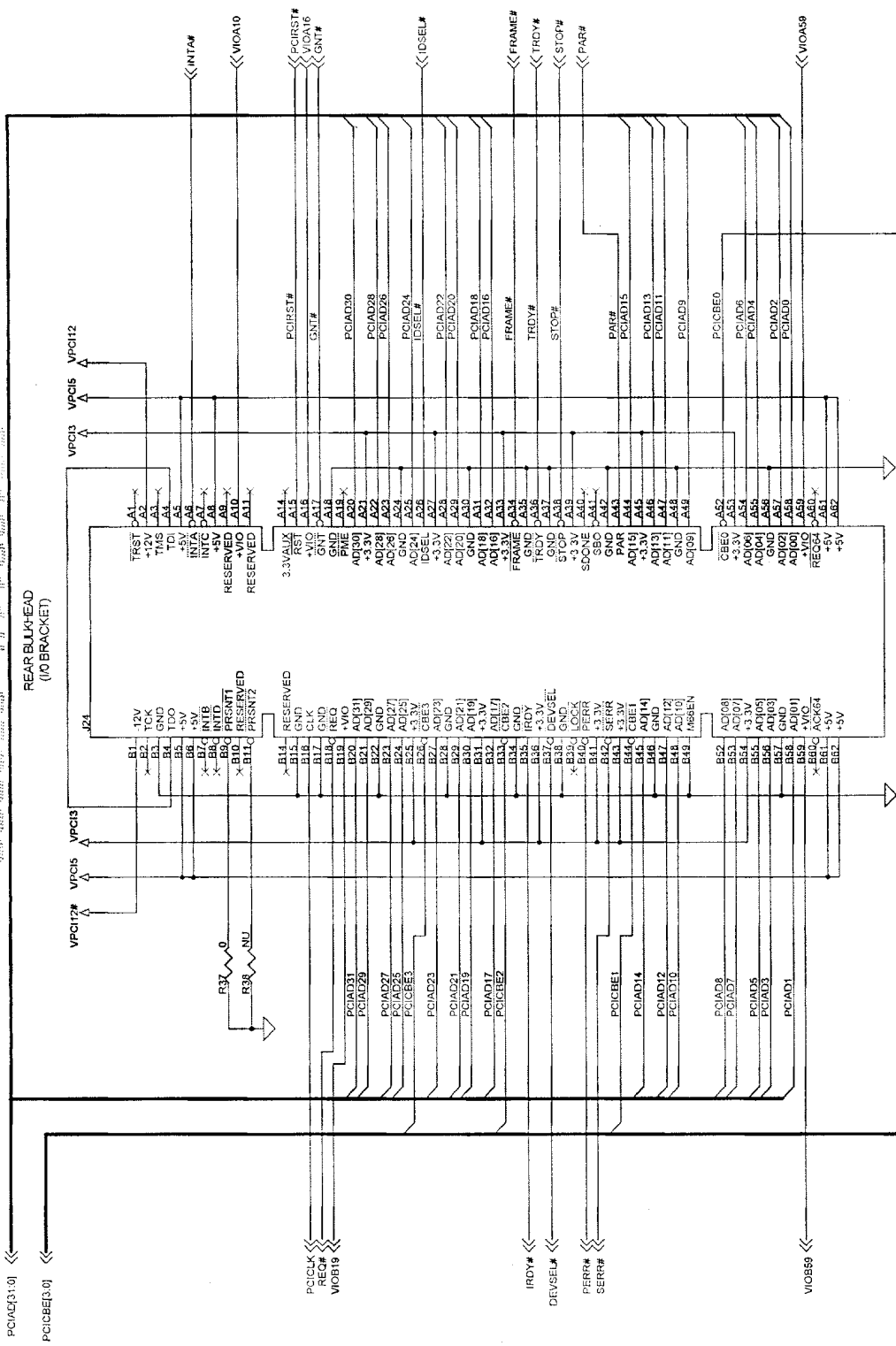
# CONNECTOR PINOUTS



<b>REALTIME DATA COMPRESSION SYSTEMS™</b>	
Title: REALATA	
Size: A	Document Number: PRELIMINARY DWG No: 10100
Date: Monday, October 18, 1999	Rev: G
Sheet: 11	of 26

**ATA3 CONNECTOR**  
 PATENTS PENDING.  
 COPYRIGHT © 1999.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.

REALTIME DATA COMPRESSION SYSTEMS



REALTIME DATA COMPRESSION SYSTEMS™	
Title	REALATA
Doc Number	PRELIMINARY DWG No: 10100
Size	B
Date	Monday, October 18, 1999
Sheet	12 of 26
Rev	G

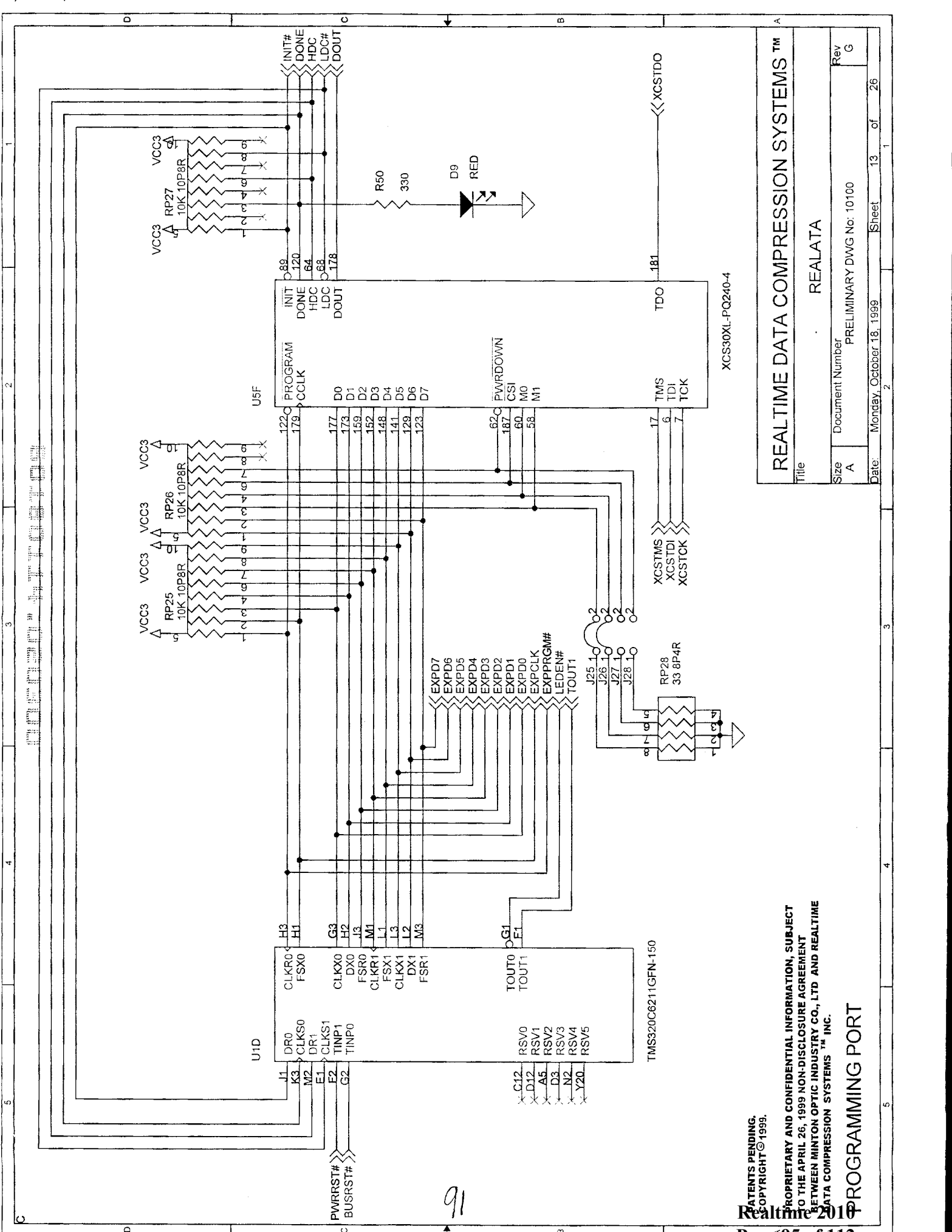
PCI EDGE CONNECTOR (TOP VIEW)

PCI CONNECTOR

PATENTS PENDING. COPYRIGHT©1999.

PROPRIETARY AND CONFIDENTIAL INFORMATION. SUBJECT TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME DATA COMPRESSION SYSTEMS™ INC.

90

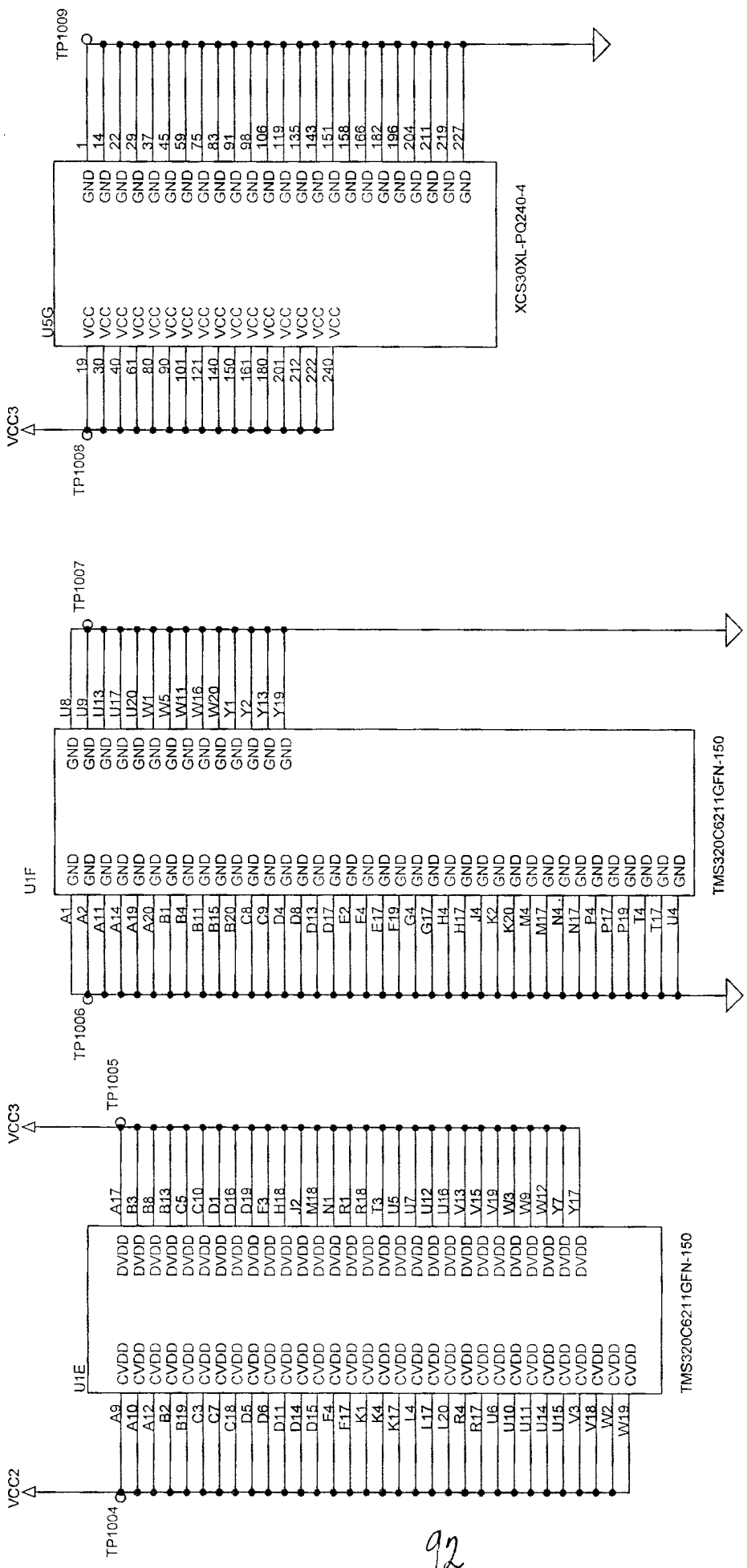


REALTIME DATA COMPRESSION SYSTEMS™	
Title	REALATA
Size	Document Number
A	PRELIMINARY DWG No. 10100
Rev	G
Date:	Monday, October 18, 1999
Sheet	13 of 26

PATENTS PENDING.  
 COPYRIGHT © 1999.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.

**PROGRAMMING PORT**

# POWER AND GROUND

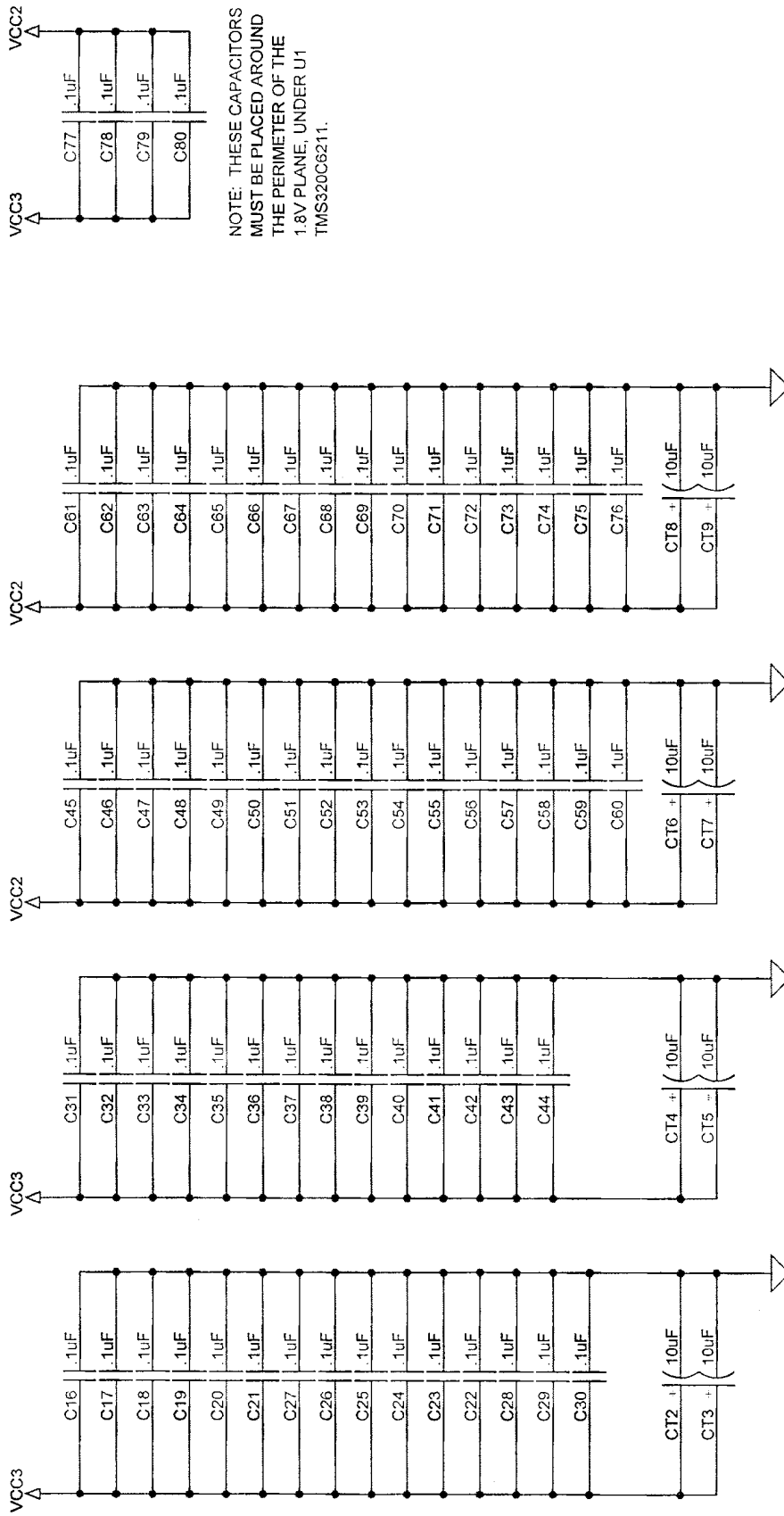


Title		REALDATA	
Size	Document Number	PRELIMINARY DWG No: 10100	
A		Rev	G
Date:	Monday, October 18, 1999	Sheet	14 of 26

**REALTIME DATA COMPRESSION SYSTEMS™**  
 PATENTS PENDING.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.  
**POWER AND GROUND - C6211, XCS30**



# REALTIME™



NOTE: THESE CAPACITORS MUST BE PLACED AROUND THE PERIMETER OF THE 1.8V PLANE, UNDER U1 TMS320C6211.

NOTE: ONE 0.1uF CAPACITOR MUST BE CONNECTED TO EACH U1 TMS320C6211 VCC2 OR VCC3 PIN, CENTRALLY LOCATE 10uF TANTALUM

PATENTS PENDING.  
COPYRIGHT © 1999.

PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME DATA COMPRESSION SYSTEMS™ INC.

REALTIME DATA COMPRESSION SYSTEMS™

REALATA

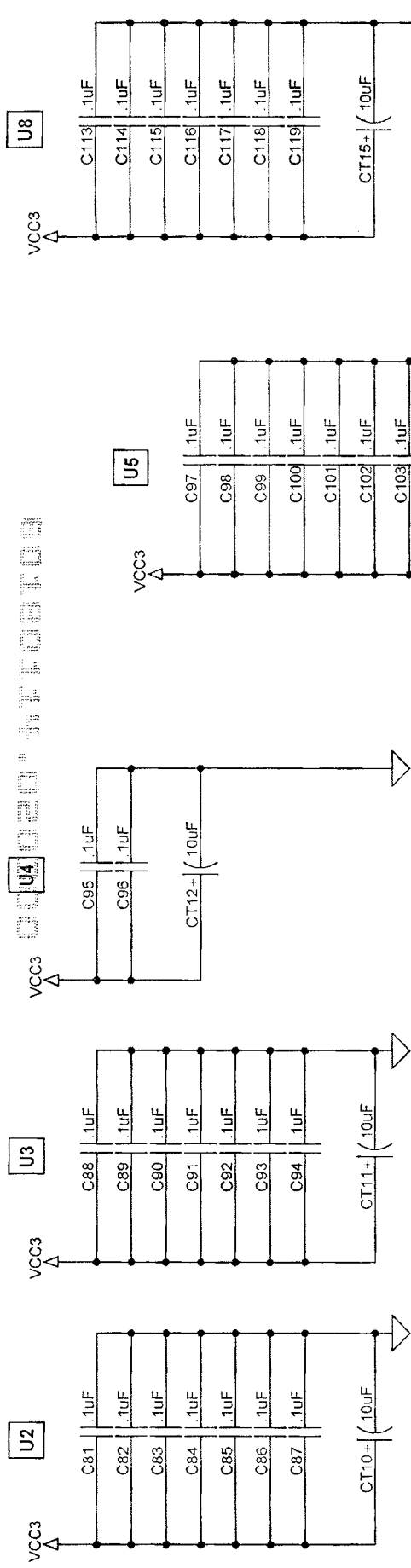
Title: Document Number: PRELIMINARY DWG No: 10100

Size: A  
Date: Monday, October 18, 1999

Sheet 15 of 26

Rev: G

# VCC3 Decoupling

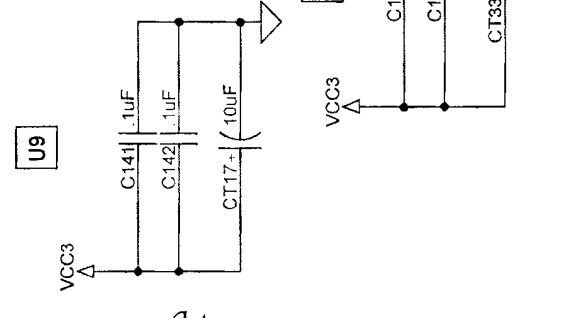


NOTE: PLACE ONE 0.1uF CAPACITOR ACROSS EACH VCC PIN, AND CENTRALLY LOCATE 10uF TANTALUM FOR U2, U3, U4, U8 & U11 RESPECTIVELY.

NOTE: ONE 0.1uF CAPACITOR MUST BE CONNECTED TO EACH U8 SBRAM VCC PIN, CENTRALLY LOCATE 10uF TANTALUMS.

NOTE: ONE 0.1uF CAPACITOR MUST BE CONNECTED TO EACH U5 XCS30 VCC PIN, CENTRALLY LOCATE 10uF TANTALUMS.

NOTE: PLACE TWO 0.1uF CAPACITORS ACROSS U9, U10, U12, Y1 & Y2. CENTRALLY LOCATE 10uF TANTALUMS ACROSS U9, U10, U12, Y1 & Y2 VCC PINS



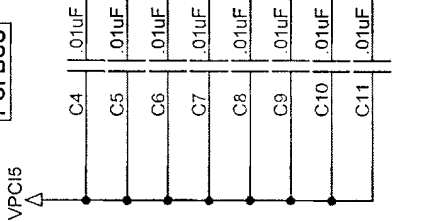
PATENTS PENDING.  
COPYRIGHT © 1999.

PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME DATA COMPRESSION SYSTEMS™ INC.

Title		REALTIME DATA COMPRESSION SYSTEMS™	
Size		REALATA	
Document Number		PRELIMINARY DWG No. 10100	
Rev	Date		Sheet
G	Monday, October 18, 1999		16 of 26

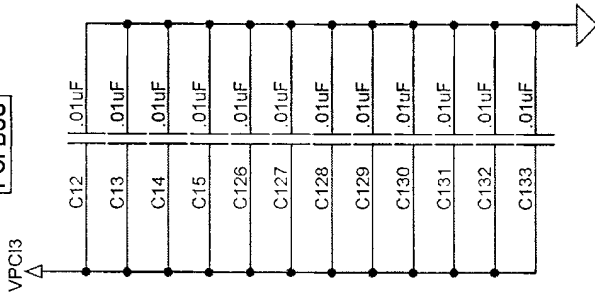
# DECOUPLING CAPACITORS

## PCI BUS



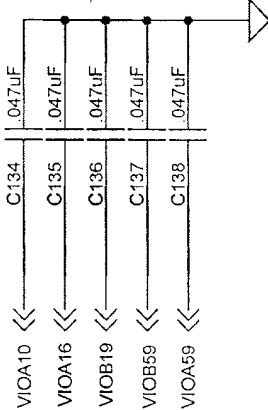
NOTE: PLACE ONE .01uF CAPACITOR WITHIN 0.25 INCHES OF EACH PCI +5V POWER PIN A5, A8, A61, A62, B5, B6, B61, B62

## PCI BUS



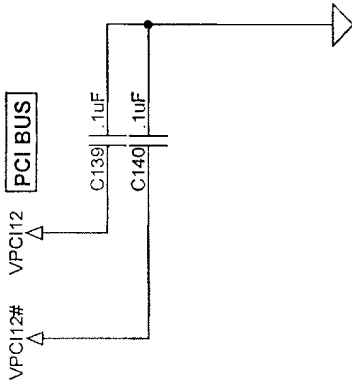
NOTE: PLACE ONE .01uF CAPACITOR WITHIN 0.25 INCHES OF EACH PCI +3.3V POWER PIN A21, A27, A33, A39, A45, A53, B25, B31, B36, B41, B43, B54

## PCI BUS



NOTE: PLACE ONE .047uF CAPACITOR WITHIN 0.25 INCHES OF EACH PCVIO POWER PIN AS LABELED. DO NOT BUS PCVIO PINS COMMON.

## PCI BUS



NOTE: PLACE CAPACITORS WITHIN 0.25 INCHES OF PCI POWER PINS.

PATENTS PENDING.  
COPYRIGHT ©1999.

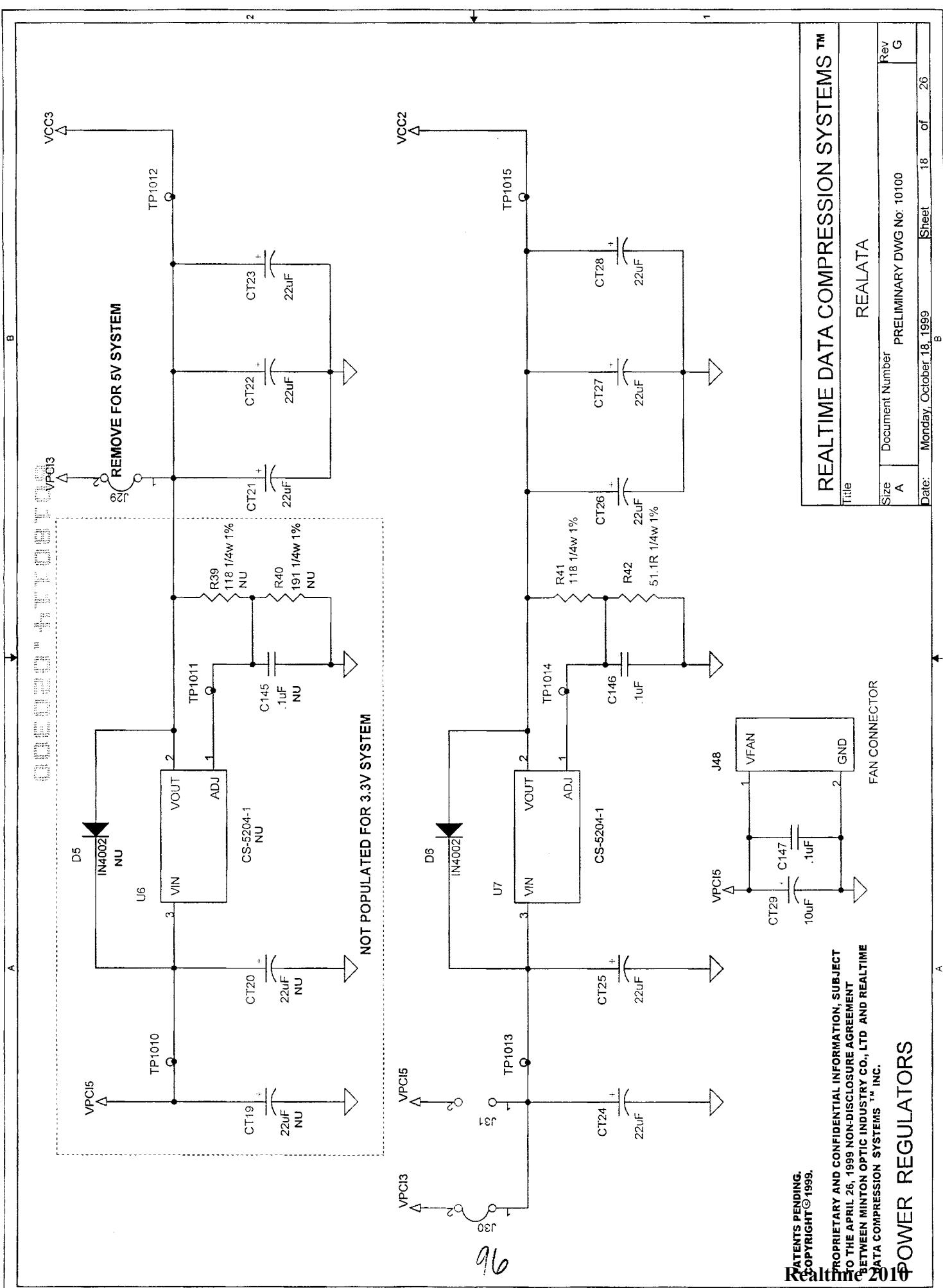
REALTIME DATA COMPRESSION SYSTEMS™  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.

## DECOUPLING - PCI BUS

REALTIME DATA COMPRESSION SYSTEMS™

Title		REALATA	
Size	Document Number	PRELIMINARY DWG No: 10100	Rev G
Date:	Monday, October 18, 1999	Sheet	17 of 26

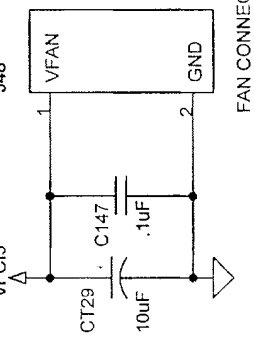
Copyright © 1999



REALTIME DATA COMPRESSION SYSTEMS™	
Title	REALATA
Size	Document Number
Rev	PRELIMINARY DWG No: 10100
G	
Date:	Monday, October 18, 1999
Sheet	18 of 26

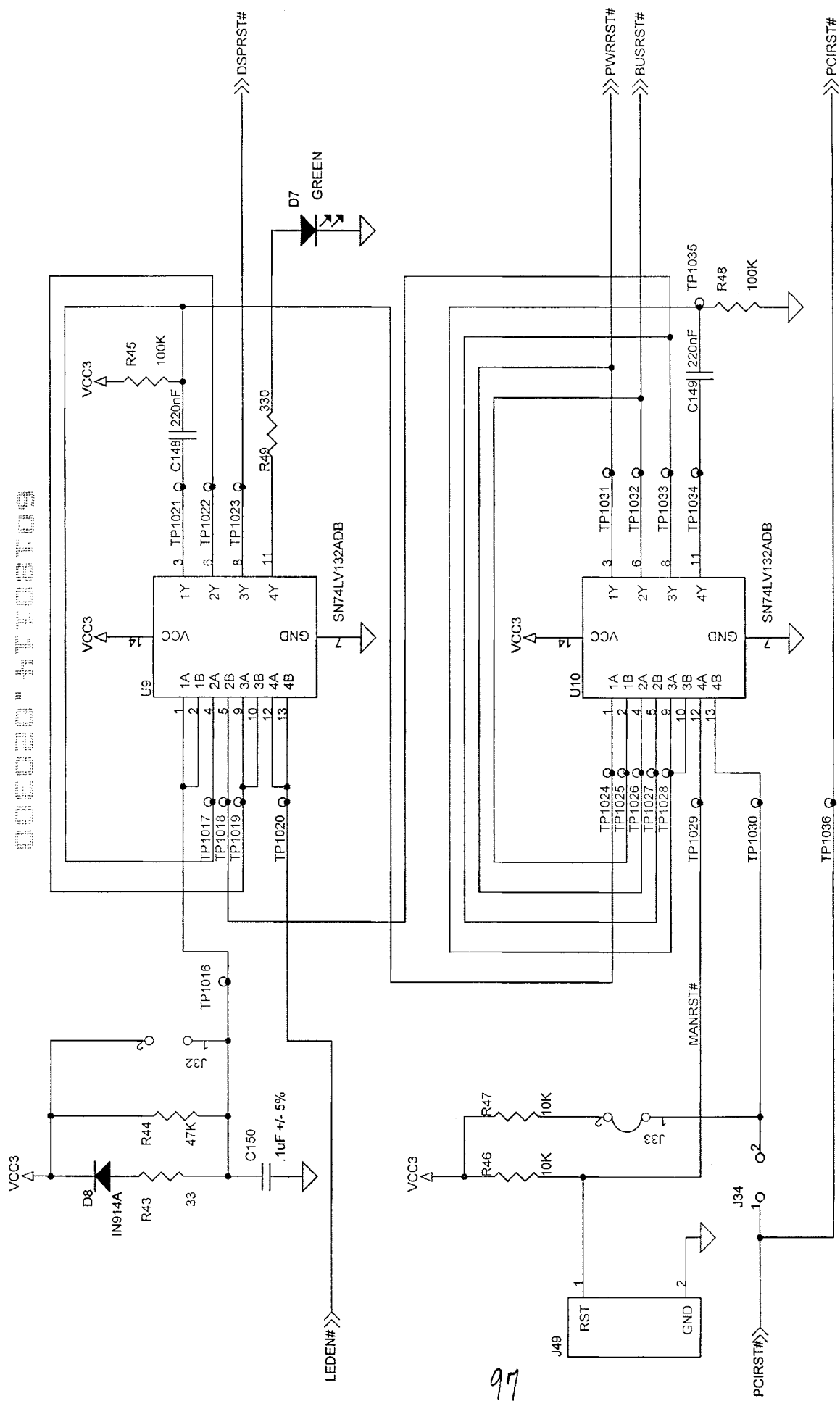
PATENTS PENDING.  
 COPYRIGHT © 1999.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.

**POWER REGULATORS**



96

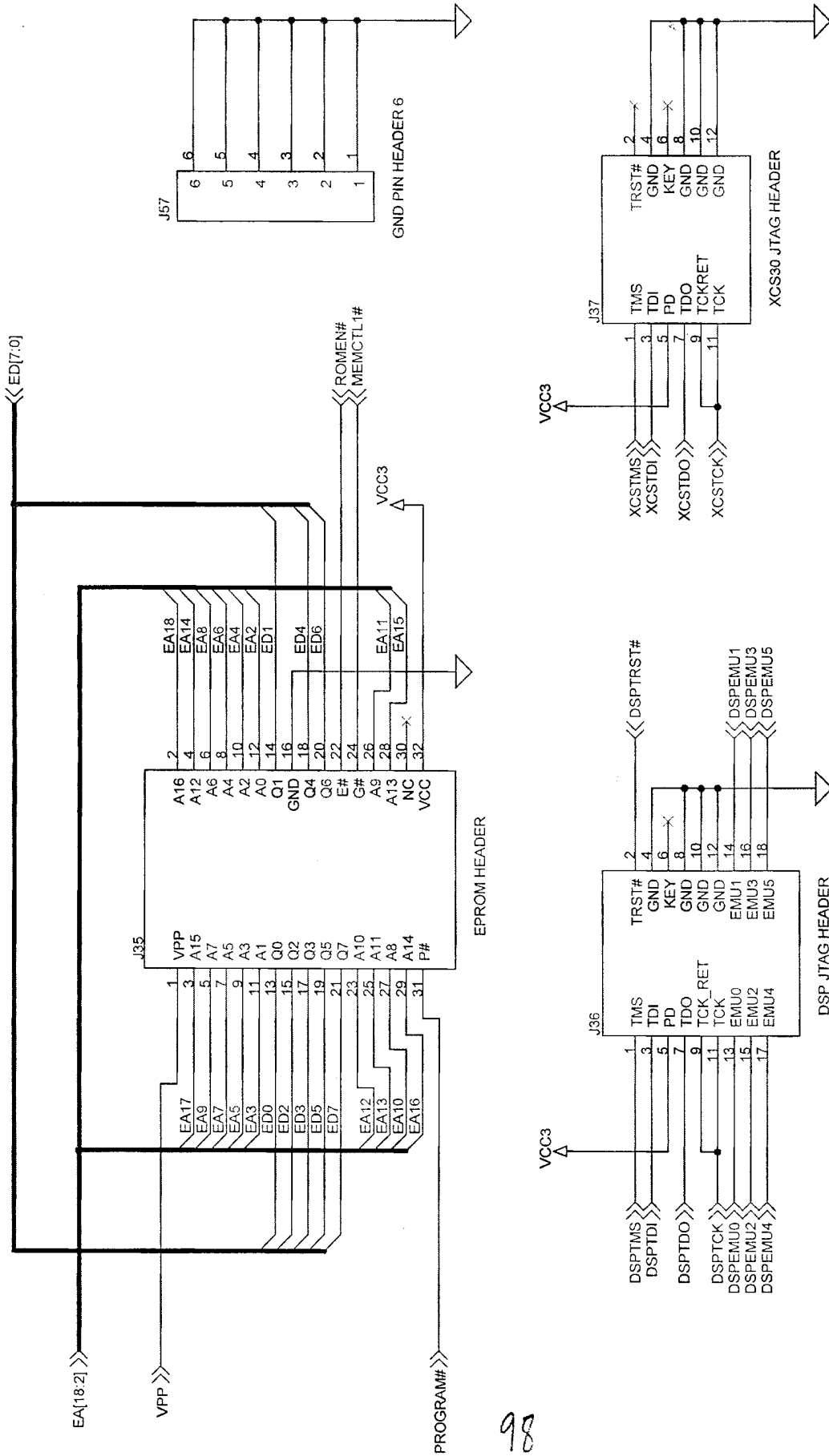
# REALTIME DATA COMPRESSION SYSTEMS



REALTIME DATA COMPRESSION SYSTEMS™	
Title	
Size	Document Number
A	REALATA
Rev	PRELIMINARY DWG No: 10100
G	
Date:	Monday, October 18, 1999
Sheet	19 of 26

**REALTIME DATA COMPRESSION SYSTEMS™**  
 PATENTS PENDING.  
 COPYRIGHT © 1999.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.  
**RESET**

# CONNECTOR HEADERS



98

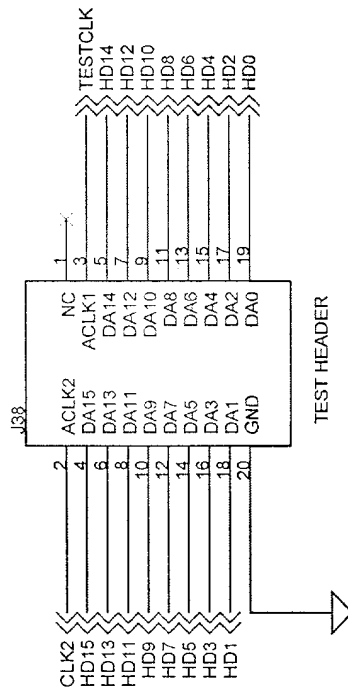
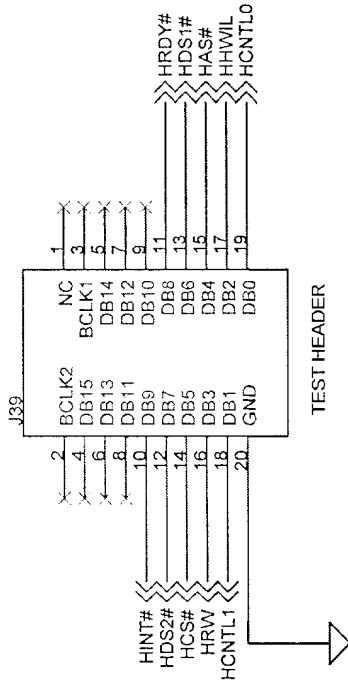
PATENTS PENDING.  
COPYRIGHT © 1999.

REALTIME DATA COMPRESSION SYSTEMS™  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™, INC.

REALTIME DATA COMPRESSION SYSTEMS™  
 JTAG & EPROM HEADERS

Title		REALTIME DATA COMPRESSION SYSTEMS™	
Size	Document Number	PRELIMINARY DWG No: 10100	Rev
A			G
Date:	Monday, October 18, 1999	Sheet	20 of 26

# CONNECTORS



99

PATENTS PENDING.  
COPYRIGHT © 1999.

REALTIME  
**REALTIME DATA COMPRESSION SYSTEMS™**  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.

**HOST PORT TEST CONNECTOR**

**REALTIME DATA COMPRESSION SYSTEMS™**

Title

REALATA

Size  
A

Document Number

PRELIMINARY DWG No: 10100

Rev  
G

Date:

Monday, October 18, 1999

Sheet

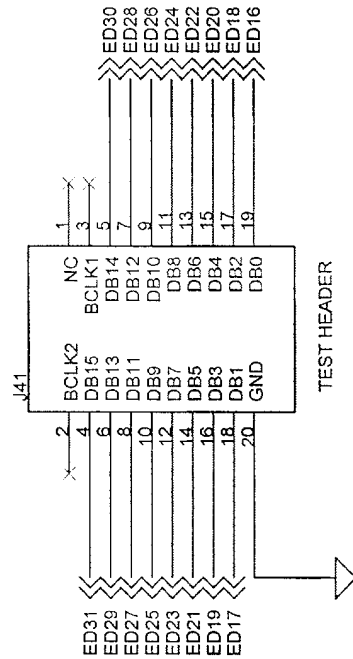
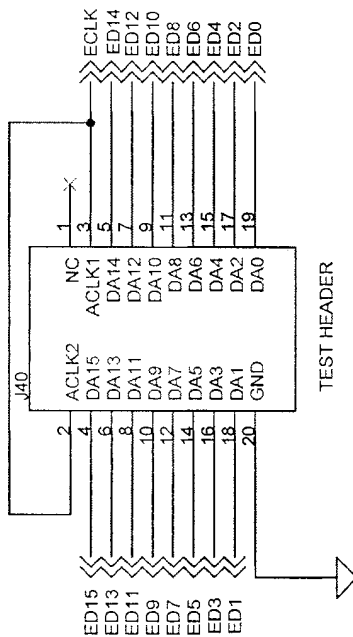
21

of

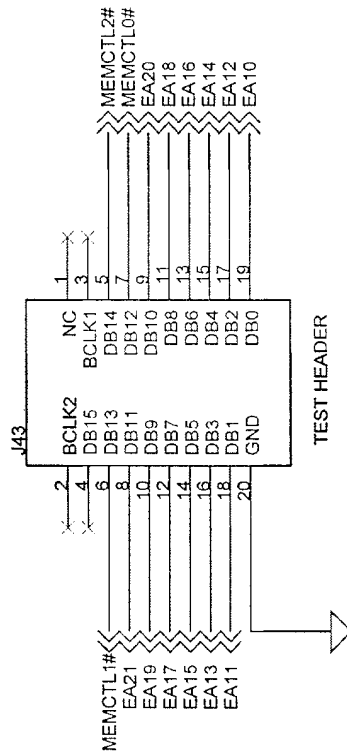
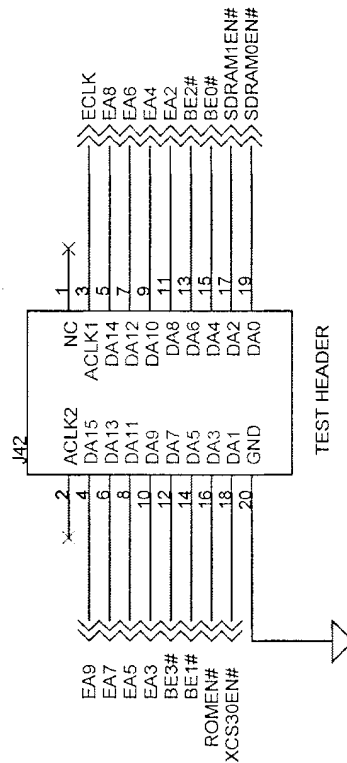
26

# CONNECTOR FOOTPRINTS

## DATA BUS TEST HEADER



## ADDRESS BUS TEST HEADER



PATENTS PENDING.  
COPYRIGHT © 1999

REALTIME DATA COMPRESSION SYSTEMS™  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.

REALTIME DATA COMPRESSION SYSTEMS™

Title

REALATA

Size

A

Document Number

PRELIMINARY DWG No: 10100

Rev

G

Date:

Monday, October 18, 1999

Sheet

22

of

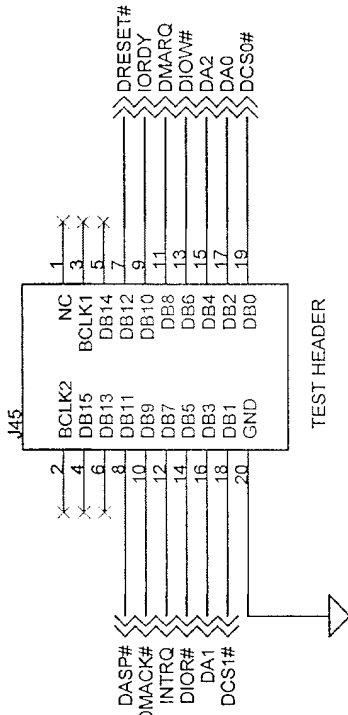
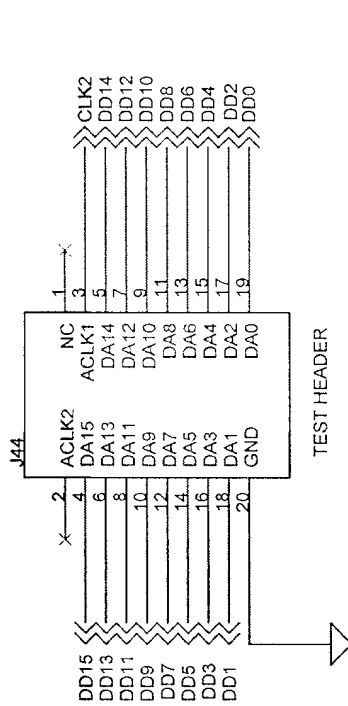
26

## DSP DATA & ADDRESS BUS TEST HEADERS

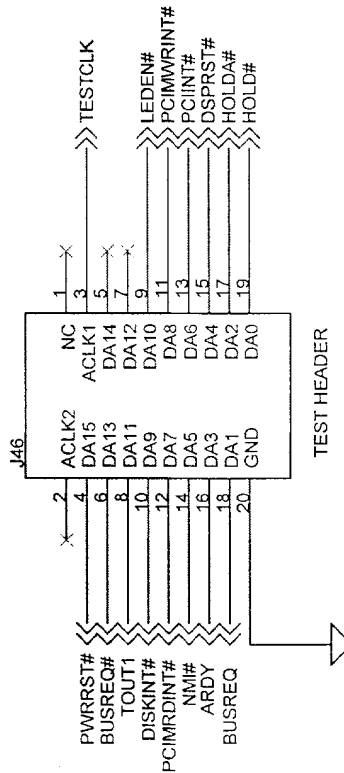


# DEFENDERS FOR

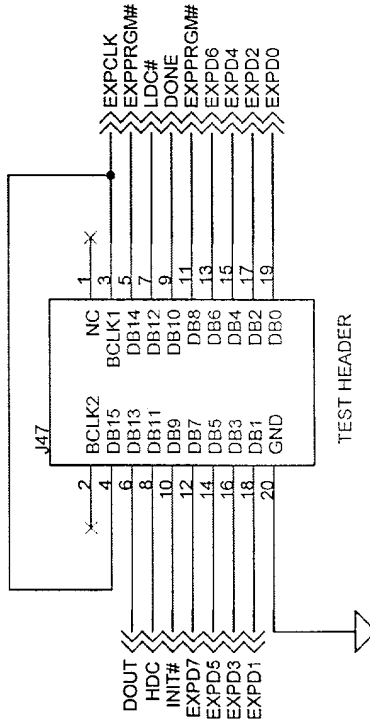
## DISK INTERFACE TEST HEADER



## DSP TEST HEADER



## XCS30 TEST HEADER



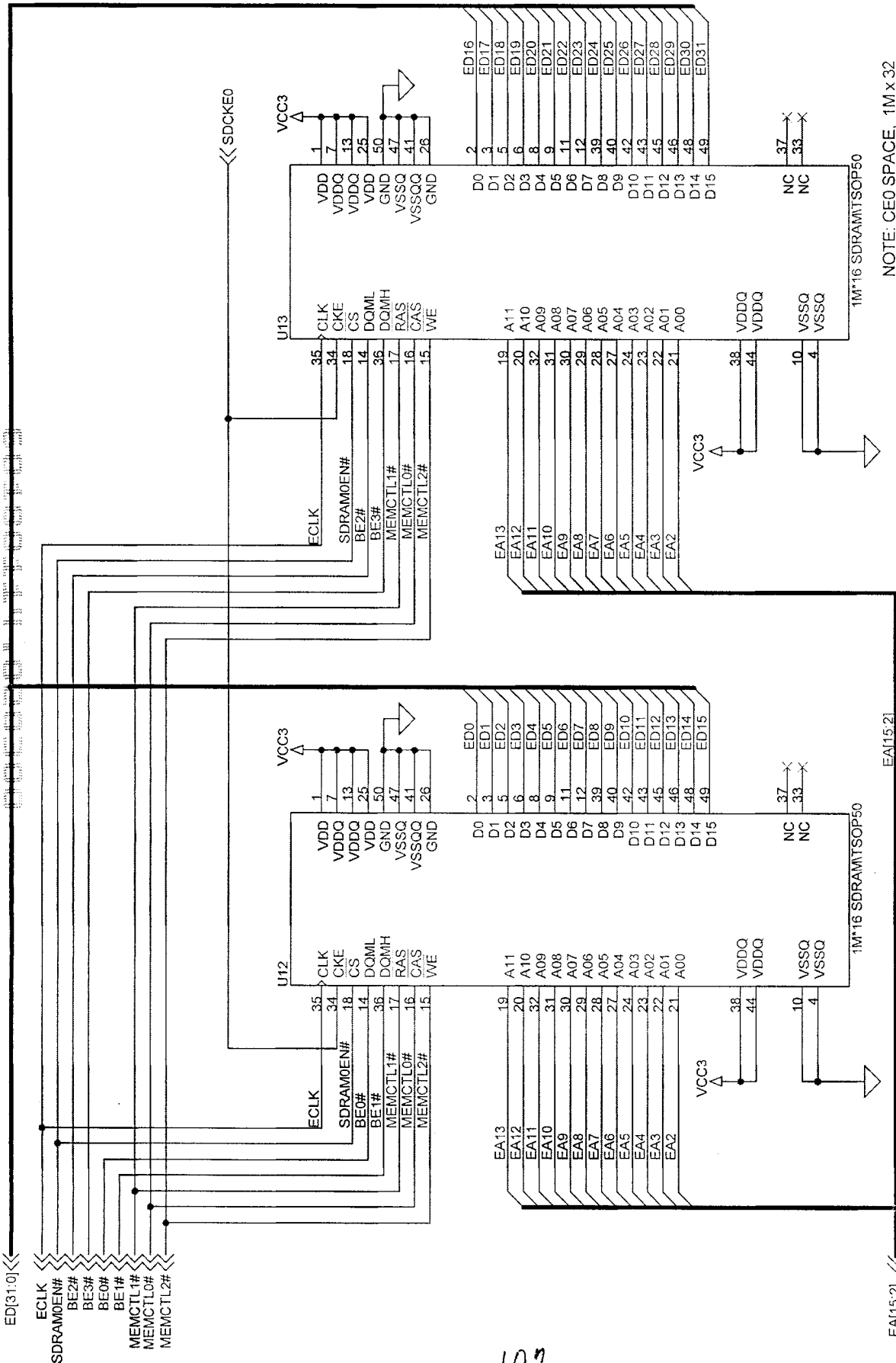
PATENTS PENDING.  
COPYRIGHT © 1999.

REALTIME  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™ INC.

## DSP & XCS30 TEST HEADERS

REALTIME DATA COMPRESSION SYSTEMS™	
Title	REALATA
Size	Document Number
Rev	PRELIMINARY DWG No: 10100
Date:	Monday, October 18, 1999
Sheet	23 of 26

101



NOTE: CEO SPACE, 1M x 32

1M\*16 SDRAMITSOP50

EA15:2	EA15:2
VDDQ	VDDQ
VDDQ	VDDQ
VSSQ	VSSQ
VSSQ	VSSQ
NC	NC
NC	NC

REALTIME DATA COMPRESSION SYSTEMS™

REALATA

Document Number: PRELIMINARY DWG No: 10100

Rev: G

Date: Monday, October 18, 1999

Sheet: 24 of 26

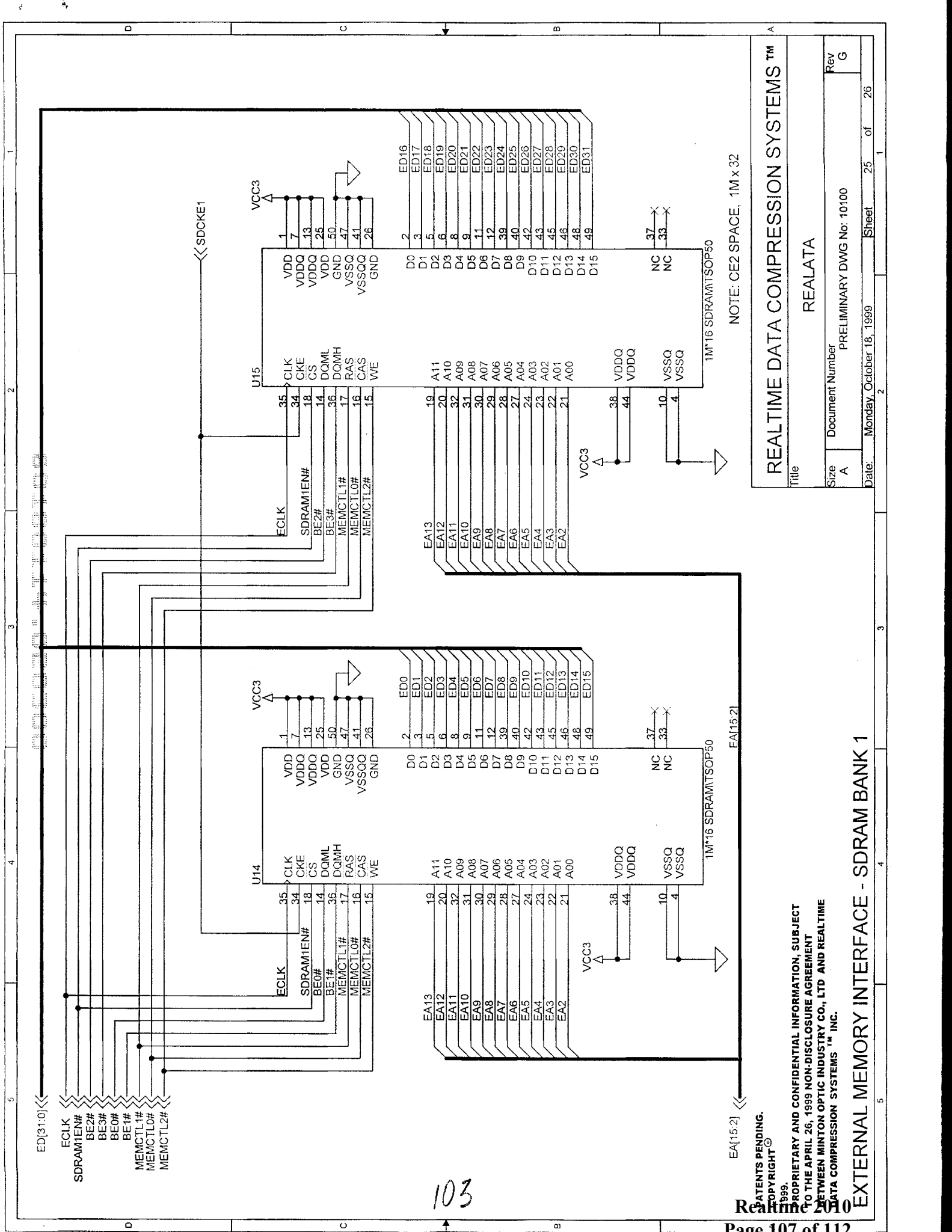
102

Page 106 of 112

REALTIME DATA COMPRESSION SYSTEMS™

PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME DATA COMPRESSION SYSTEMS™ INC.

EXTERNAL MEMORY INTERFACE - SDRAM BANK 0



NOTE: CE2 SPACE, 1M x 32

REALTIME DATA COMPRESSION SYSTEMS™

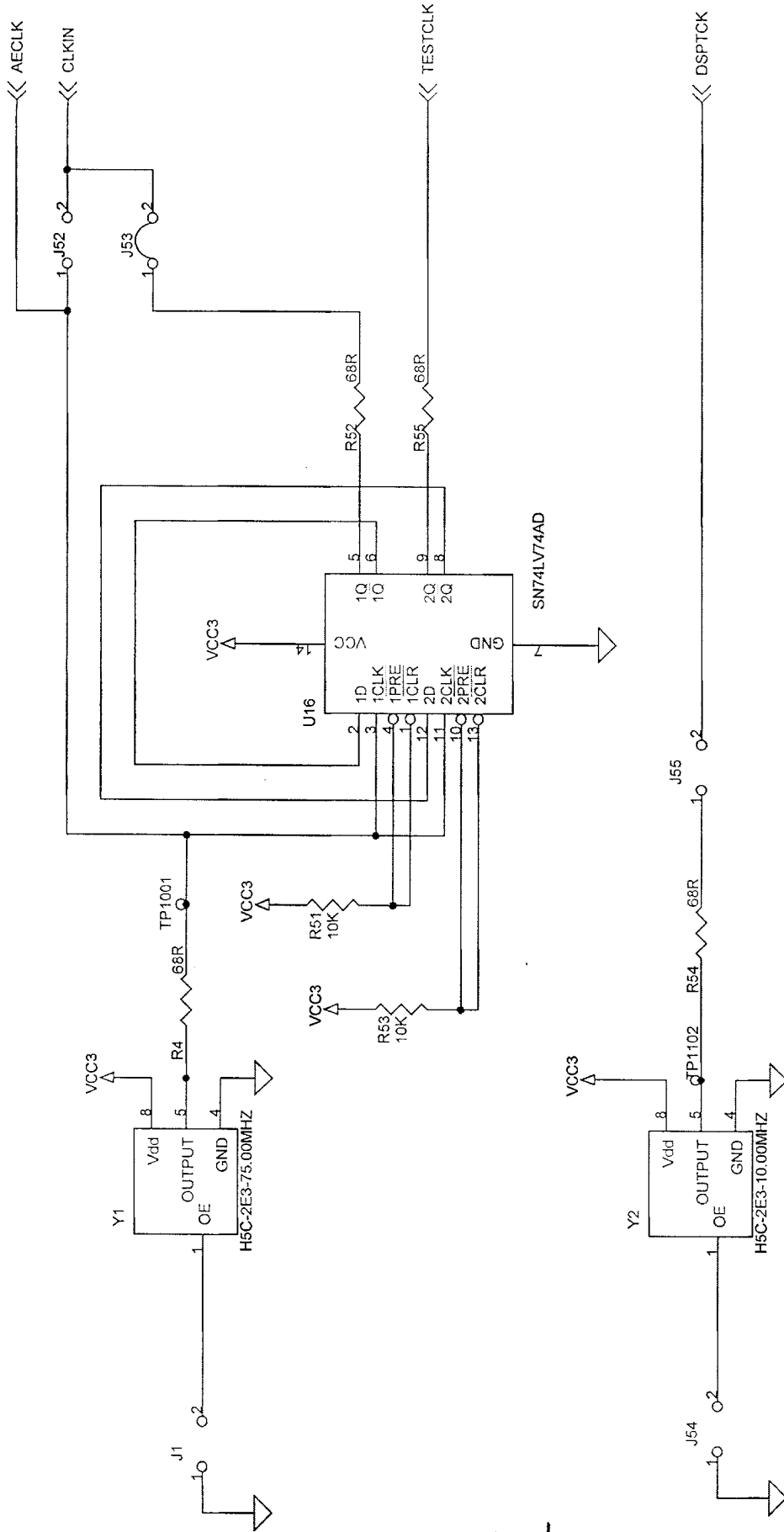
Title		REALATA	
Size	Document Number	PRELIMINARY DWG No: 10100	
A		Rev	G
Date:	Monday, October 18, 1999	Sheet	25 of 26

REALTIME DATA COMPRESSION SYSTEMS™, INC.  
 PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT  
 TO THE APRIL 26, 1999 NON-DISCLOSURE AGREEMENT  
 BETWEEN MINTON OPTIC INDUSTRY CO., LTD AND REALTIME  
 DATA COMPRESSION SYSTEMS™, INC.

EXTERNAL MEMORY INTERFACE - SDRAM BANK 1

103

# CLOCK GENERATORS



PATENTS PENDING.  
COPYRIGHT © 1999.

PROPRIETARY AND CONFIDENTIAL INFORMATION, SUBJECT TO THE  
APRIL 26, 1999 NON-DISCLOSURE AGREEMENT BETWEEN MINTON  
OPTIC INDUSTRY CO., LTD AND REALTIME DATA COMPRESSION  
SYSTEMS INC.™

NOTES: U16 & Y2 NOT USED FOR PRODUCTION UNITS;  
SUBSTITUTE: 35 MHZ Y1, REMOVE J53,  
INSERT J52 IN PRODUCTION UNITS.

REALTIME DATA COMPRESSION SYSTEMS™

Title

REALATA

Size A Document Number PRELIMINARY DWG No: 10100 Rev G

Date: Monday, October 18, 1999 Sheet 26 of 26

104



## Boot Configuration Circuit for Partially-Volatile Boot Devices

The boot configuration circuit for partially-volatile boot devices employs a look ahead differentiator to allow volatile boot devices to initialize during system configuration.

The volatile boot devices must all be initialized before the computer system bus (such as PCI interface) is fully reset. An example using the PCI bus is shown in Figure 1:

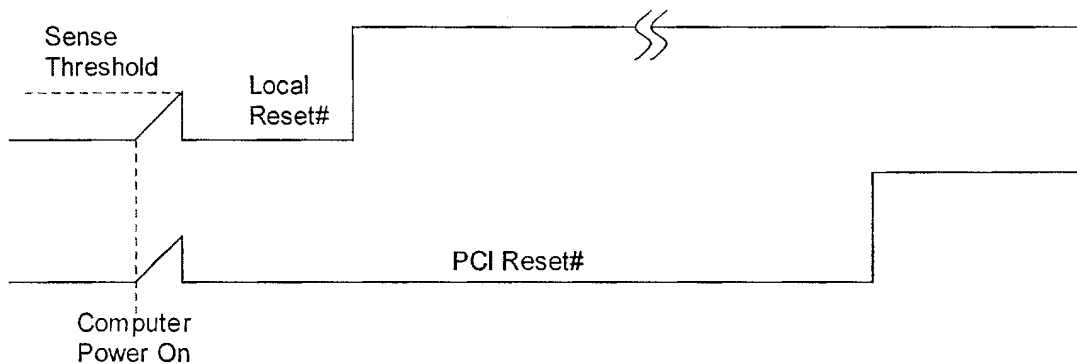


Figure 1, Local Initialization For Example Disk Controller

When power is first applied to the boot configuration circuit, it generates a power-up reset to the local system. This would typically include devices such as a DSP, memory, and I/O interfaces.

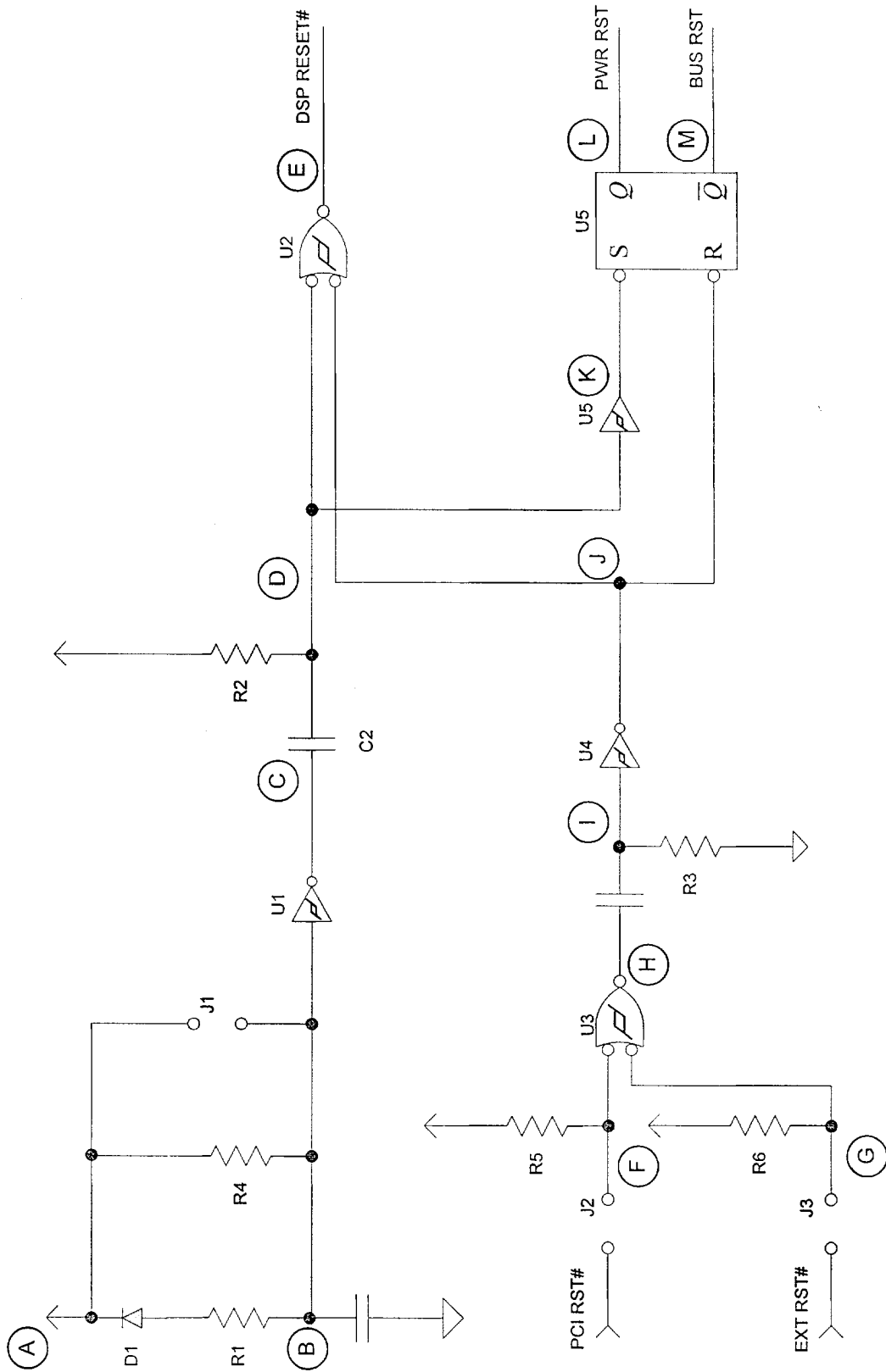
Once the local system is powered-up and reset, the controlling device (such as a DSP) will then proceed to automatically determine the system environment. The controlling device will then configure the system to work within that environment.

An example of this would be that a DSP on a disk controller would sense that the disk controller is on a PCI computer bus and has attached to it a hard disk on an IDE interface. The DSP would then load the appropriate PCI and IDE interfaces into a programmable logic device.

This can be done for all computer busses and boot device interfaces including: PCI, NuBus, ISA, Fiber Channel, SCSI, Ethernet, DSL, ADSL, IDE, DMA, Ultra DMA, and SONET.

Once the partially-volatile boot devices are completely configured for their environment, the boot device controller is reset and ready to accept commands over the computer bus.

# VOLATILE LOGIC BOOT LOADER



107

FIGURE 1

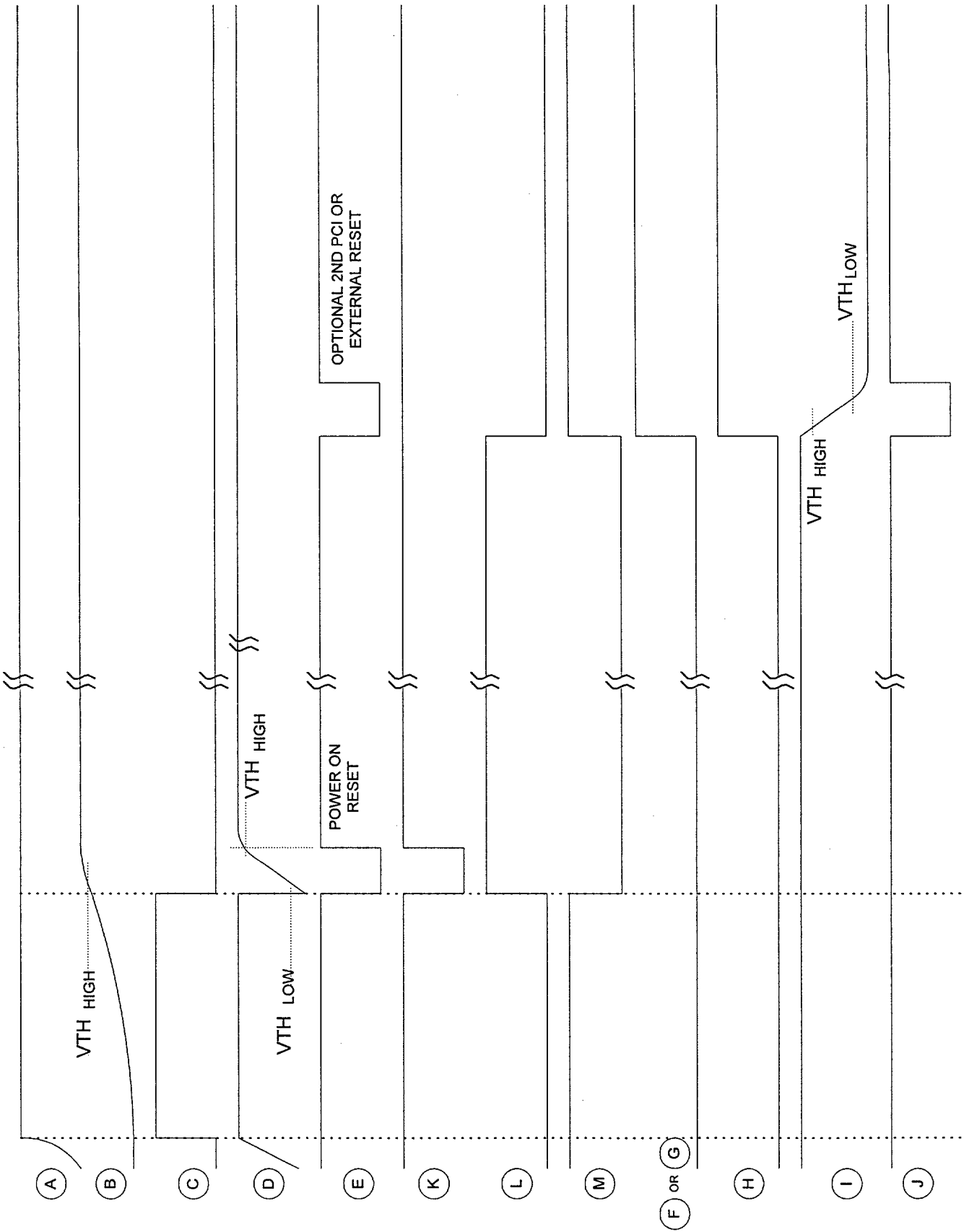


FIGURE 2