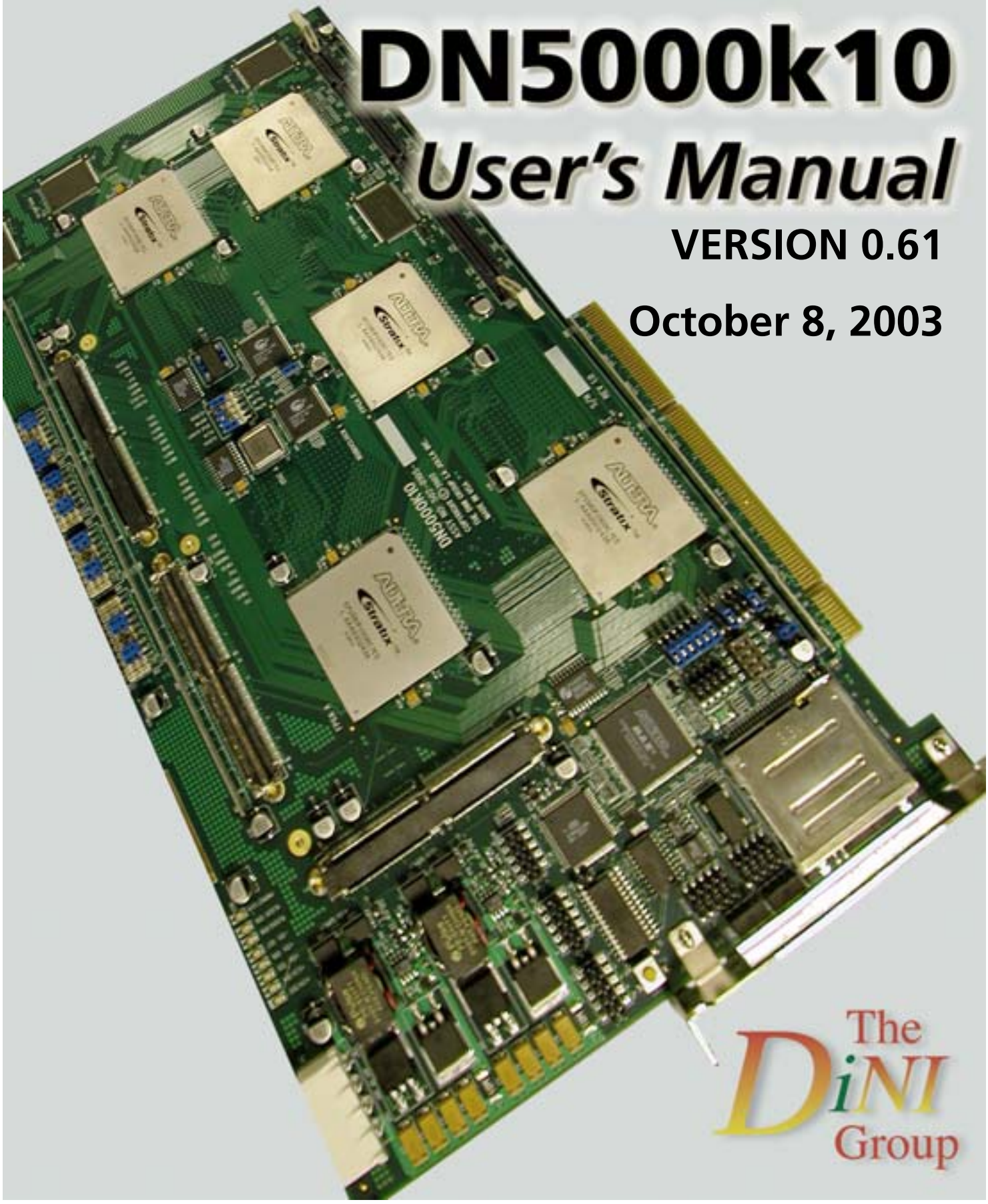
DN5000k10

User's Manual

VERSION 0.61

October 8, 2003



The DiNI Group

DN5000k10 User's Manual Version 0.61

October 8, 2003

.

The information contained within this manual and the accompanying software program are protected by copyright; all rights are reserved by the DINI Group. Therewith, the DINI group reserves a the right to make periodic modifications to this project without obligation to notify any person or entity of such revision. Copying, duplicating, selling, or otherwise distributing any part of this product without the prior written consent of an authorized representative of the DINI Group is prohibited.

DN5000K10, DN3000k10SD and DNPCIEXT-S3 are trademarks of the DINI Group.

1010 Pearl Street, Suite #6

La Jolla, CA 92037-5165

www.dinigroup.com

info@dinigroup.com

(858) 454-3419

FAX: (858) 454-1728

Copyright ©2003 The DINI Group. All Rights Reserved.

Table of Contents

Chapter 3 PCI

Chapter 4 Clocks and Clock Distribution

Chapter 5 Memories

Chapter 9 Utilities

1

Chapter A Berg Connector Datasheets

List of Figures

List of Figures (Continued)

List of Tables

Chapter 1

Getting Started

The DN5000k10 is sensitive to static electricity, so treat the PWB accordingly. The target market for this product is engineers that are familiar with FPGAs and circuit boards, so a lecture in ESD really isn't appropriate (and wouldn't be read anyway). However, we have sold some of these units to people who are not as familiar with this issue. The following web page has an excellent tutorial on the Fundamentals of ESD for those of you who are new to ESD-sensitive products:

http://www.esda.org/basics/part1.cfm.

The DINI Group Technical Support

The following means of technical support are available:

1. **The DN5000k10 User's Manual.** This is the main source of technical information. We strive to produce excellent documentation, and this manual should contain most of the answers to your questions.
2. **The DINI Group Web Page.** The web page will contain the latest manual, application notes, FAQ, articles, and any device errata and manual addenda. Please visit and bookmark: **http://www.dinigroup.com/index.php?product=5000k10**
3. **E-Mail** to **support@dinigroup.com**. You may direct questions and feedback to The DINI Group using this e-mail address.
4. **Phone Support.** We are happy to help. Call us at (858) 454-3419 during the hours of 8:00 A.M. to 5:00 P.M. Pacific Time. Some of us get in early and stay late, so you might try us outside of these hours also.
5. **Frequently Asked Questions**. In the downloads section of our web page you can find a document called **DN5000k10/S Frequently Asked Questions (FAQ)**. We will update this document occasionally with information that may not be in the User's Manual.

Relevant Information

Information about PCI can be obtained from the following sources:

The PCI Special Interest Group has a web page that has lots of good stuff. Copies of the latest PCI specification may be ordered here.

http://www.pcisig.com/
PCI Special Interest Group
2575 NE Kathryn St. #17
Hillsboro, OR 97124
FAX: (503) 693-8344

As of October 2001, the most current versions of the PCI Specifications are:

PCI Local Bus Specification, Revision 2.2
PCI Hot-Plug Specification, Revision 1.0
PCI Power Management Interface Specification, Revision 1.1
PCI-X Addendum to the PCI Local Bus Specification, Revision 1.0a

Other recommended specifications include:

PCIMG 2.0 Compact PCI Specification, Revision 2.1 (or greater)
PCI Industrial Computer Manufacturers Group (PICMG)
401 Edgewater Place, Suite 500
Wakefield, MA 01880, USA
TEL: 781-224-1100
FAX: 781-224-1239
http://www.picmg.org

The best book to get if you need an introduction to PCI is:

PCI System Architecture
Fourth Edition
MindShare, Inc.
Tom Shanley and Don Anderson

Ignore some of the ignorant statements made in the Customer Review section at **http://www.amazon.com/**. This is an excellent book for PCI and well worth the money.

The best book to get if you need an introduction to PCI-X is:

PCI-X System Architecture
MindShare, Inc.
Tom Shanely and Karen Gettman

You are going to need to know Verilog or VHDL to use the Stratix FPGA. If you need a reference, we recommend the following book for Verilog:

Verilog HDL: A Guide to Digital Design and Synthesis
Samir Palnitkar
ISBN: 0-13-451675-3

If you are one of those people that actually like VHDL, we feel sorry for you. The following books may be helpful:

Essential VHDL: RTL Synthesis Done Right
Sundar Rajan

The IQ Booster: Improve Your IQ Performance Dramatically
Edwin Breecher

Conventions

This manual uses the following conventions. An example illustrates each convention.

- The term PCI-X will be used generically unless there is a specific instance where PCI applies.
- This design guide generically refers to PCI-X protocol. When the PCI-X HalfBridge core is in PCI mode, PCI protocol will be followed.
- `Courier font` denotes the following items:
  - Signals on PCI Bus side of the PCI-X Interface

    `FRAME_IO` (PCI-X Interface signal name)

    `FRAME#` (PCI-X Bus signal name)
  - Signals within the user application

    `BACK_UP, START`
  - Command line input and output

    ```
    setenv XIL_MAP_LOC_CLOSED
    ```
  - HDL pseudocode

    ```
    assign question = to_be | !to_be;

    assign cannot = have_cake & eat_it;
    ```
  - Design file names

    `pcim_top.v, pcim_top.vhd`
- **`Courier bold`** denotes the following items:
  - Signals on the user side of the LogiCORE PCI-X Interface

    `ADDR_VLD`
  - Menu selections or button presses

    `FILE -> OPEN`
- Italic font denotes the following items:
  - Variables in statements which require user-supplied values

    ngdbuild *design_name*
  - References to other manuals

    See the *Libraries Guide* for more information.

– Emphasis in text

It is not a bug, it is a *feature*.

- Dark shading indicates items that are not supported or reserved:

SDONE_I	**in/out**	**Snoop Done signal. Not Supported.**

- Square brackets "[ ]" indicate an optional entry or a bus index:

ngdbuild [*option_name*] *design_name*

DATA[31:0]

- A vertical or horizontal ellipsis indicates repetitive material that has been omitted.

A B C... X Y Z

- The use of "`fn(SIG1. . . SIGn)`" in an HDL pseudocode fragment should be interpreted as "combinational function of signals SIG1 through SIGn.

SUM = fn(A, B, Cin);

- The prefix "`0x`" or the suffix "`h`" indicate hexadecimal notation.

A read of address `0x00110373` returned `45524943h`.

- A "#" an "_n" , an "n" or a "–" means the signal is active low

`INT#` is active low.

`fpga_inta_n` is active low.

`SRAMCS–` is active low.

`FPGA_GRSTn` is active low.

Chapter 2

DN5000k10 Features, Overview and General Description

DN5000k10 Features

The DN5000k10 features include:

- 32/64-bit, +3.3V, PCI/PCI-X-based PWB with up to five Altera Stratix™ FPGAs (FBGA1508).
  - Device availability: EP1S80 (EP1S60 to follow with slightly reduced features).
- ~3.5 million ASIC gates per PWB (with EP1S80 — LSI standard)

Device	I/O	Flip-Flops	18 x 18 Multipliers	Embedded Memory		
				M512 RAM	M4K RAM	M-RAM
EP1S60	1022	57,120	72	574	292	6
EP1S80	1203	79,040	88	767	364	9

- Fast/Easy FPGA configuration via standard SmartMedia FLASH card
  - Microprocessor controlled (ATmega128L)
  - RS232 port for configuration/operation status and control
  - Fastest possible configuration speed (via Passive Parallel method)
- 10A on-board linear regulator for +3.3V and +1.5V
  - Standalone operation via separate power connector
  - +3.3V not needed on backplane
- 6 low skew clocks distributed to all FPGA and test connectors:
  - 2 CY7B993/4 RoboclockII PLLs
  - 2 socketed oscillators
  - PCI Clock
  - 1 dividable clock via CPLD
- Direct support for Synplicity's Certify TDM interconnect multiplexing.
- Robust observation/debug with 488 connections for logic analyzer observability or for pattern generator stimulus.
- Status LEDs.

- User-designed daughter PWB for custom circuitry and interfaces.
- SignalTap and Identify (from Synplicity) fully supported via JTAG interface.

Figure 2-1 shows a block diagram of the DN5000k10.

DN5000k10 Description

The DN5000k10 is a complete logic emulation system that enables ASIC or IP designers a vehicle to prototype logic and memory designs for a fraction of the cost of existing solutions. The DN5000k10 can be hosted in a 32/64-bit PCI/PCI-X slot, or can be used as a stand-alone device. A single DN5000k10 stuffed with five EP1S80s can emulate up to 3.5 million gates of logic as measured by LSI. High I/O-count, 1508-pin, flip-chip BGA packages are employed. The F1508 package has 1203 I/Os, which allows for abundant connections to daughter connectors and external memories. A total of 488 test pins are provided on the top of the PWB via high-density connectors for logic analyzer-based debugging, or for pattern generator stimulus. Custom daughter cards such as the DN3000k10SD can be mounted to these connectors as a means of interfacing the DN5000k10 to application-specific circuits. A reference 32-bit PCI target design and test bench is provided in Verilog at no additional cost.

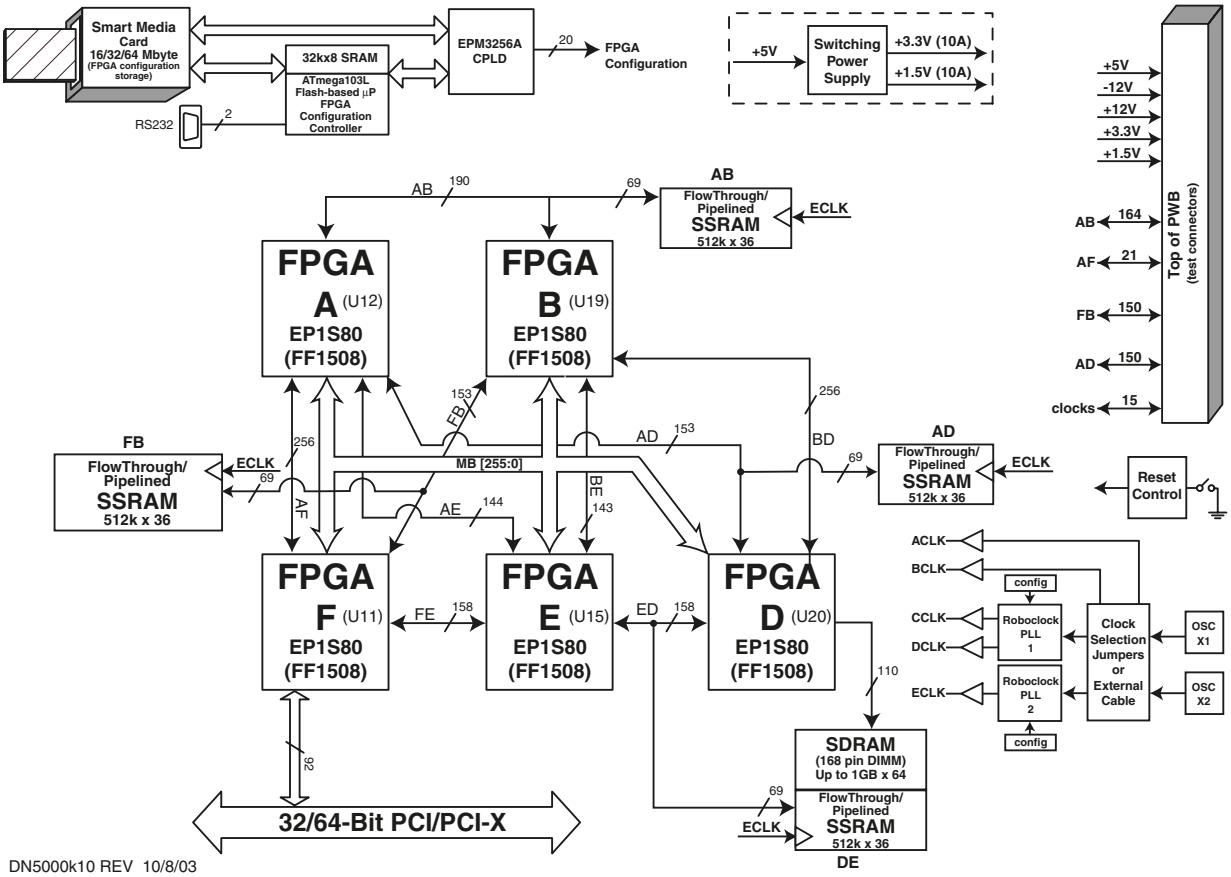


Figure 2-1 DN5000k10 Block Diagram

Easy Configuration via SmartMedia

The configuration bit files for the FPGA are copied onto a 32-megabyte SmartMedia FLASH card (provided) and an on-board microprocessor controls the FPGA configuration process. Visibility into the configuration process is enhanced with an RS232 port. Sanity checks are performed automatically on the configuration bit files, helping to avoid the time-consuming process of debugging the configuration process. FPGA configuration runs quickly at 48 MHz. Eight LEDs provide instant status and operational feedback. Four of these LEDs are connected to the CPLD and can be user-configured.

FPGA — Stratix (U11, U12, U15, U19, U20—F, A, E, B, D)

The DN5000k10 contains two to five Stratix™ FPGAs. They are called A, B, D, E, and F. The package is a flip chip fine-pitch BGA with 1508 pins (F1508). The pitch on the pins is 1 mm. This isn't important, but this pin density makes the PWB a bitch to layout. Keep that in mind if you try to make one of these at home. Most of the 1203 I/O pins are utilized on the F1508 package. The DN5000k10 can be stuffed with EP1S80 devices in any combination of locations. The EP1S60 does come in an F1508, but since it has fewer I/O pins, it should not be used. The standard speed grade we stuff is –7. We can use the –6 speed grade, but don't fall out of your chair when you get the price. Note that Altera seems to have cancelled plans for the EP1S120. Although this part appears in some Altera literature, we haven't seen any scheduled release date or other documentation for it. Don't expect to see anything larger than the EP1S80 until at least the 2004 time frame. Figure 2-2 shows the stuffing options for the DN5000k10.

What happened to C? We struggled with the layout for many months, which gave the buffoons in marketing time to rethink what they asked for. As is normally the case with marketing, after a few drinks they decided to change the requirements. The FPGA that originally was in the C position was eliminated and replaced with the four SSRAMs and a SDRAM DIMM. So much work had been done to that point that those of us in engineering didn't think it wise to re-label the FPGAs. That is why no FPGA_C exists.

The following is a very brief overview of the Stratix family. More information can be gleaned from the Stratix Datasheet (`ds_stx.pdf`). This file is on the CD-ROM supplied with the DN5000k10, but you are better off getting the latest version from the Altera Web page (**http://www.altera.com/**). Make sure to get the latest errata sheet also.

DN5000K10 STUFFING OPTION COMPARISON									
Stuffed FPGAs	PCI	Memory Access		Total Chip to Chip Connections	Non-Memory Chip to Chip Connections	Non-Header, Non-Memory Chip to Chip Connections	TOTAL Header Connections	Single-FPGA Header Connections	Single-FPGA SSRAMs
		SDRAM	SSRAMs						
ABDEF	YES	YES	4	1867	1591	1367	488	0	0
ABDE	NO	YES	4	1300	1093	908	488	171	1
ABDF	YES	YES	4	1264	1057	767	488	0	1
ABEF	YES	NO	4	1300	1162	956	488	150	2
ADEF	YES	YES	4	1125	987	879	488	314	2
BDEF	YES	YES	4	1124	986	899	488	335	2
ABD	NO	YES	4	855	717	532	488	171	2
ABE	NO	NO	4	733	664	563	488	321	3
ABF	YES	NO	3	855	717	511	488	150	1
ADE	NO	YES	3	711	573	486	338	185	1
ADF	YES	YES	4	665	596	488	488	314	3
AEF	YES	NO	4	814	814	790	488	464	4
BDE	NO	YES	4	813	744	741	467	464	3
BDF	YES	YES	4	665	596	509	488	335	3
BEF	YES	NO	3	710	641	554	338	185	2
DEF	YES	YES	3	572	503	500	324	321	2
AB	NO	NO	3	446	377	276	488	321	2
AD	NO	YES	3	409	340	253	338	185	2
AE	NO	NO	3	400	400	397	338	335	3
AF	YES	NO	3	512	512	488	488	464	3
BD	NO	YES	3	512	512	509	467	464	3
BE	NO	NO	3	399	399	396	317	314	3
BF	YES	NO	2	409	340	253	338	185	1
DE	NO	YES	2	414	345	342	153	150	1
DF	YES	YES	3	256	256	253	324	321	2
EF	YES	NO	2	414	414	411	174	171	2
A	NO	NO	2	0	0	0	338	338	2
B	NO	NO	2	0	0	0	317	317	2
D	NO	YES	2	0	0	0	153	153	2
E	NO	NO	1	0	0	0	3	3	1
F	YES	NO	1	0	0	0	174	174	1

Figure 2-2 DN5000k10 Stuffing Option Comparison

Flip-Flops and LUTs

Figure 2-3 shows what Altera calls a Logic Element, or LE. Each LE contains a flip-flop and a 4x1 look-up table (LUT). LEs are arranged in groups of 10, called Logic Array Blocks (LAB). The EP1S80 is an array of LABs with 91 rows and 101 columns, but there are 9 RAM blocks which appear in place of 13-row by 11-column sections of the grid, leaving a total of 7904 LABs and 19040 LEs. (Other blocks, such as DSP/multiplier blocks and smaller RAM, are arranged in entire columns squeezed between two LAB columns.)

Each LUT can implement any Boolean function of four inputs. An LUT can also be configured as a two-input adder/subtractor with a carry chain coming from the adjacent LE and going to the next LE. In order to reduce delays caused by long carry chains, each set of 5 LEs computes two adder results simultaneously, then uses the carry result from the previous set of 5 to select which result is correct. For more information, check www.altera.com for the Stratix datasheet.

The flip-flop in each LE includes a clock enable input, an asynchronous preset and reset, synchronous set and reset logic, and an asynchronous load function. Data input can come from the LUT in the same LE to register addition or boolean outputs, or the LUT and FF can be used independently of each other. For more information, check **www.altera.com** for the Stratix datasheet.

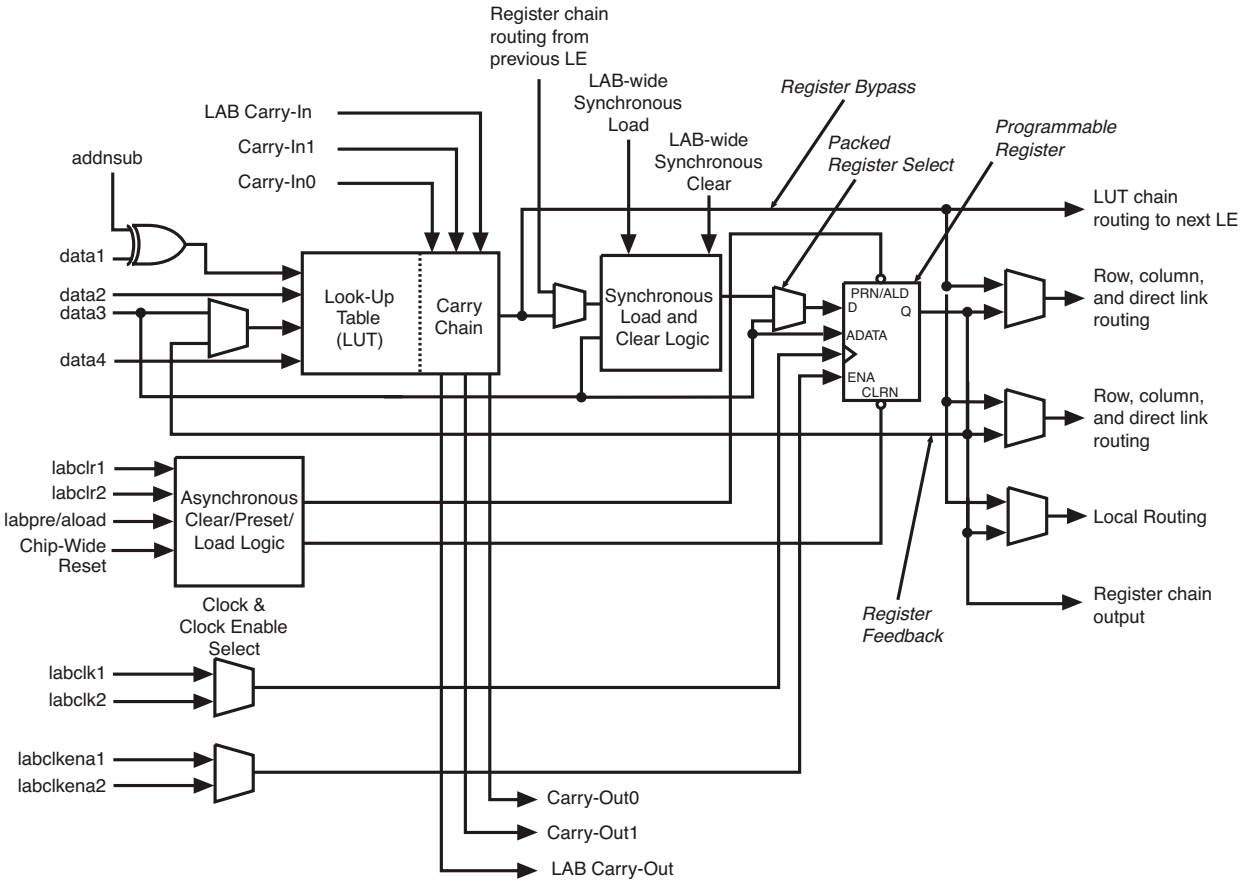


Figure 2-3 General LE Diagram

Embedded Memory

Stratix has boatloads of embedded memory. The EP1S80 contains 767 blocks of 576 bits, 364 blocks of 4.5 Kbits, and 9 blocks of 576 Kbits. The smallest memory blocks (called M512 RAM) can be configured for data widths ranging from 32 x 18 bits to 512 x 1 bit; medium-sized blocks (M4K RAM) can be configured ranging from 128 x 36 bits to 4K x 1 bit; and the largest blocks (M-RAM) can be configured anywhere from 4K x 144 bits to 64K x 9 bits. The embedded memory is dual-ported, and can be used to construct almost any type of memory - FIFOs, dual-port RAMs, single-port RAMs, etc.

The two largest blocks, M-RAM and M4K RAM, are fully dual-ported memory, with read and write functions available on two separately clocked ports. M512 RAM is a "simple dual-port" memory, meaning that one port is write-only and the other is read-only. Any of the memory blocks can be configured as simple dual-port or single-port memory. See Figure 2-4 for a diagram of the memory.

Multipliers

Stratix devices feature a large number of multipliers grouped into what Altera calls DSP blocks (see Figure 2-5). The EP1S80 contains 22 DSP blocks, each of which can provide one 36x36 bit multiplier, four 18x18 bit multipliers, or eight 9x9 bit multipliers. Each block also contains adder/subtractor/accumulator registers which can be configured to provide many common DSP functions, such as FIR or IIR filters, FFT, or DCT, without the use of LAB resources. The Stratix datasheet (available at

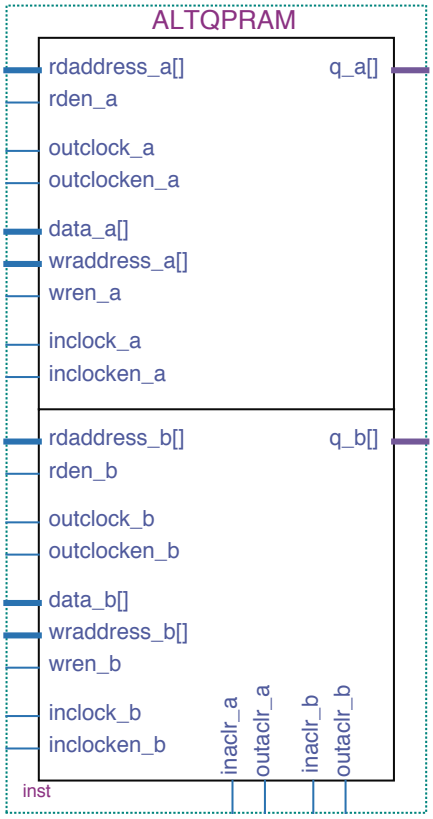


Figure 2-4 Dual-Port Data Flows

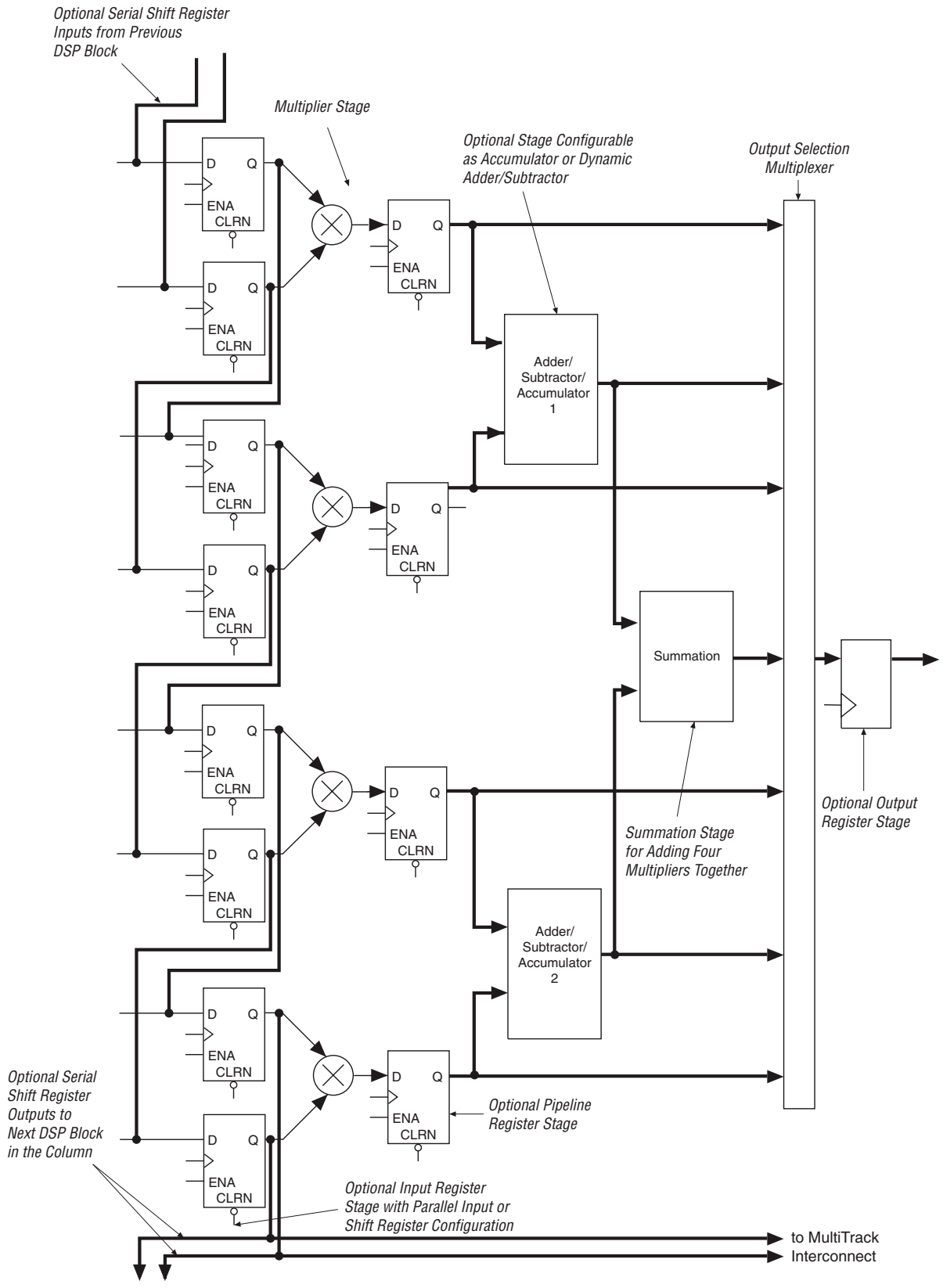


Figure 2-5 DSP Block Diagram

www.altera.com) has more detailed information on how the multipliers and adders are configured for some common functions.

Figure 2-5 shows a DSP block configured for four 18x18 bit multipliers. A DSP block can be configured as two parallel systems of 9x9 bit multipliers, each of which is also described by Figure 2-5. The adder blocks can be used to add or subtract two or four multipliers, such as in complex multiplication, or to add a new result each clock cycle to an accumulated sum. They are also used to configure the DSP block as a 36x36 bit multiplier, with or without an accumulator.

All registers in Figure 2-5 are optional, as shown by Figure 2-6, which is a detailed view of a single 18x18 bit or 9x9 bit multiplier. Any or all of the registers may be used to pipeline the multiplier logic and improve the clock speed, or the alternate path may be used to bypass the register. Figure 2-6 also shows more detail about the optional shift register path, which makes FIR or IIR filters easy to implement.

Most synthesis tools will accept Verilog or VHDL descriptions of multpliers and infer a DSP block with the appropriate configuration. For those that don't, Altera provides a megafunction generator to help with direct instantiation of the hardware resources. See "Synthesis and Emulation Issues" on page 2-26 for more detail.

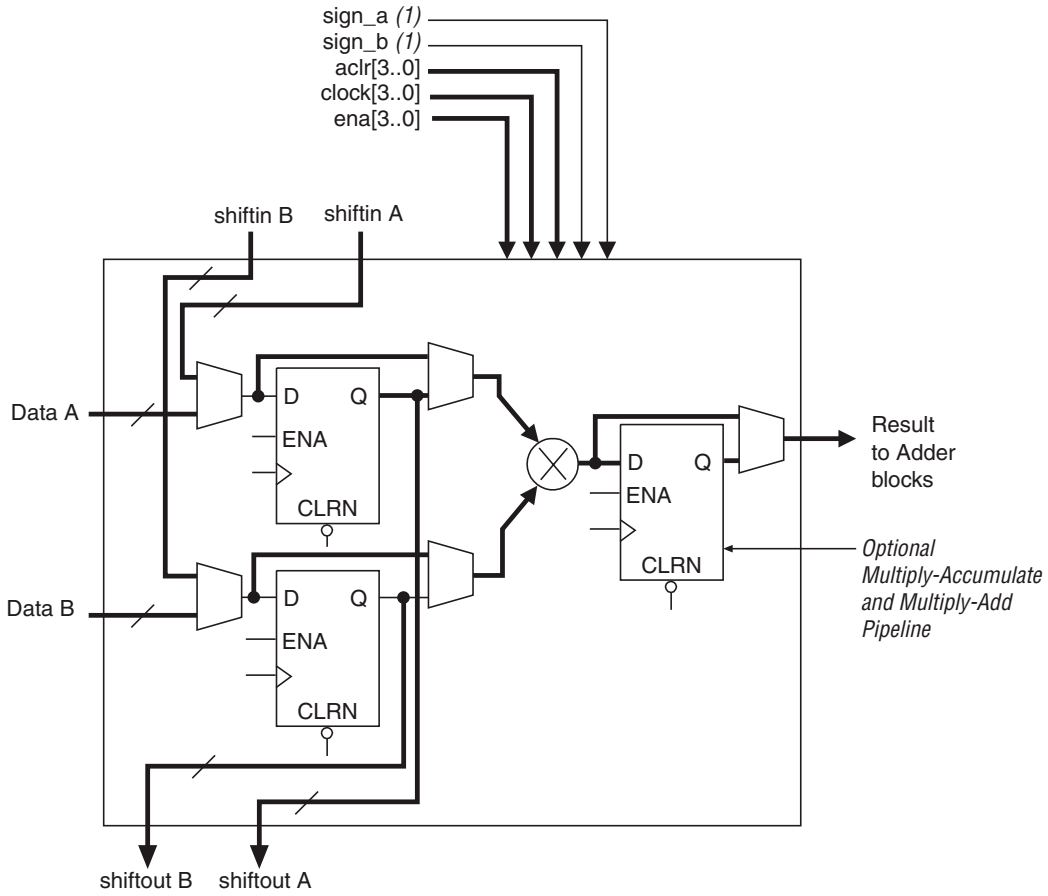


Figure 2-6 Multiplier Sub-Component Block Diagram

I/O Issues

Terminator technology is supported on all pins. The resistors used for RDN and RUP should be 250 ohms for series termination or impedance matching I/O standards. Parallel termination requires 1000 ohm resistors for RDN and RUP. Terminator technology is a very nice feature and we recommend you use it on all I/O signals. The default IO_STANDARD attribute for the CSF file is **LVTLL**.

All VCCO pins are connected to +3.3 V. The VREF pins are connected to +1.5 V, so the DN5000k10 does not support I/O standards that require other values of VREF. So the I/O standards supported are:

LVTTL — Low-Voltage TTL
The low-voltage TTL, or LVTTL, standard is a general-purpose EIA/JESDSA standard for 3.3 V applications that use the LVTTL input buffer and a Push-Pull output buffer. The standard requires a 3.3 V input and output source voltage (V_{CCO}) but does not require the use of a reference voltage (V_{REF}) or a termination voltage (V_{TT}).

LVCMOS33 — 3.3 Volt Low-Voltage CMOS
This standard is an extension of the LVCMOS standard (JESD8. –5). It is used in general-purpose 3.3 V applications. The standard requires a 3.3 V input/output source voltage (V_{CCO}) but does not require the use of a reference voltage (V_{REF}) or a termination voltage (V_{TT}).

PCI-X — Peripheral Component Interface
The PCI standard specifies support for 33 MHz, 66 MHz and 133 MHz PCI bus applications. It uses a LVTTL input buffer and a Push-Pull output buffer. This standard does not require the use of a reference voltage (V_{REF}) or a board termination voltage (V_{TT}); however, it does require 3.3 V input output source voltage (V_{CCO}).

SSTL-3 class I and II
SSTL-3 uses a series termination resistor on output signals and a parallel termination resistor on input signals. Stratix devices use a VREF of +1.5V to enable the appropriate resistors internally. Because SSTL-3 requires parallel termination, it is only available on banks 3, 4, 7 and 8, and on clock output signals.

CTT
CTT uses a parallel termination resistor on input signals, with no termination resistors on output signals. Stratix devices use a VREF of +1.5V to enable the appropriate resistors internally. Because CTT requires parallel termination, it is only available on I/O banks 3, 4, 7 and 8, and on clock output signals.

Differential Termination Standards
Stratix devices provide several differential I/O standards, available only in I/O banks 1, 2, 5 and 6, and on differential clock output signals. However, the DN5000k10 is not routed for differential signaling, so these features are not likely to be very useful.

Bitstream Encryptions

Stratix devices have no special bitstream encryption function. The Dini Group may be able to assist with scrambling bitfiles to protect IP on the SmartMedia card, which would then be descrambled in the programming CPLD. Users should be aware, however, that the bitstream would be

unprotected between the CPLD and FPGA, so they could still be examined and reverse-engineered. Our DN3000k10 products use Xilinx FPGAs which can be used to decrypt the bitstream inside the FPGA, providing complete design protection. If you are interested in this feature, please be aware that there are some issues with the Xilinx encryption feature, described in the DN3000k10 FAQ on our website.

µP and FPGA Configuration

The DN5000k10 has an ATmega128L microprocessor (µP) that is used to control the configuration process (**U8**). The amount of internal SRAM (4 Kbytes) was not large enough to hold the FAT needed for SmartMedia, so an external 32 k x 8 SRAM was added. The address latching function is done via an LVT373 (**U3**).

The microprocessor has the following responsibilities:

- Reading the SmartMedia card
- Configuring the Stratix FPGA
- Executing DN5000k10 self tests.

Other than FPGA configuration, the µP has no responsibilities. Less than 25% of the 128 Kbytes of FLASH is used for FPGA configuration and utilities, so you are welcome to use the rest of the resources of the µP for your own purposes. Instructions for customizing the µP are contained in the file `Custom_ATmega128L.pdf`. This file is on the CD-ROM, or it can be downloaded from the DINI Group web page.

REMEMBER: You can use the microprocessor for your own purposes!

We ship a programming cable for the ATmega128L with the DN5000k10. Updates to the code will be posted on our web site. If you wish to do your own development you will need the compiler, which we do not ship with the product. The compiler is available from IAR (**http://www.iar.com/**). The part number is EWA90PCUBLV150.

Note that if you are willing to program the FPGA with the JTAG or serial cable, the CLPD and the µP have **no** function. In this case you can use all of the resources of the µP for your own purposes.

The µP: Some Details

The ATmega128L is gross overkill for the FPGA configuration function. The datasheet and user's manual are on the CD-ROM that was shipped with the DN5000k10. The file names are `ATmega128_UM.pdf` and `ATmega128_DS.pdf`. But if you intend to use the µP for your own purposes, you should check the Atmel web page to get a copy of the latest user's manual, datasheet, and erratas. The Atmel web page is **http://www.eu.atmel.com/atmel/**. The ATmega128L is under the section called "Flash Microcontroller, AVR 8-Bit RISC." Most of the features are unused. A variety of test headers allow for possible use of these features. Each header and the various possible functions are described in the

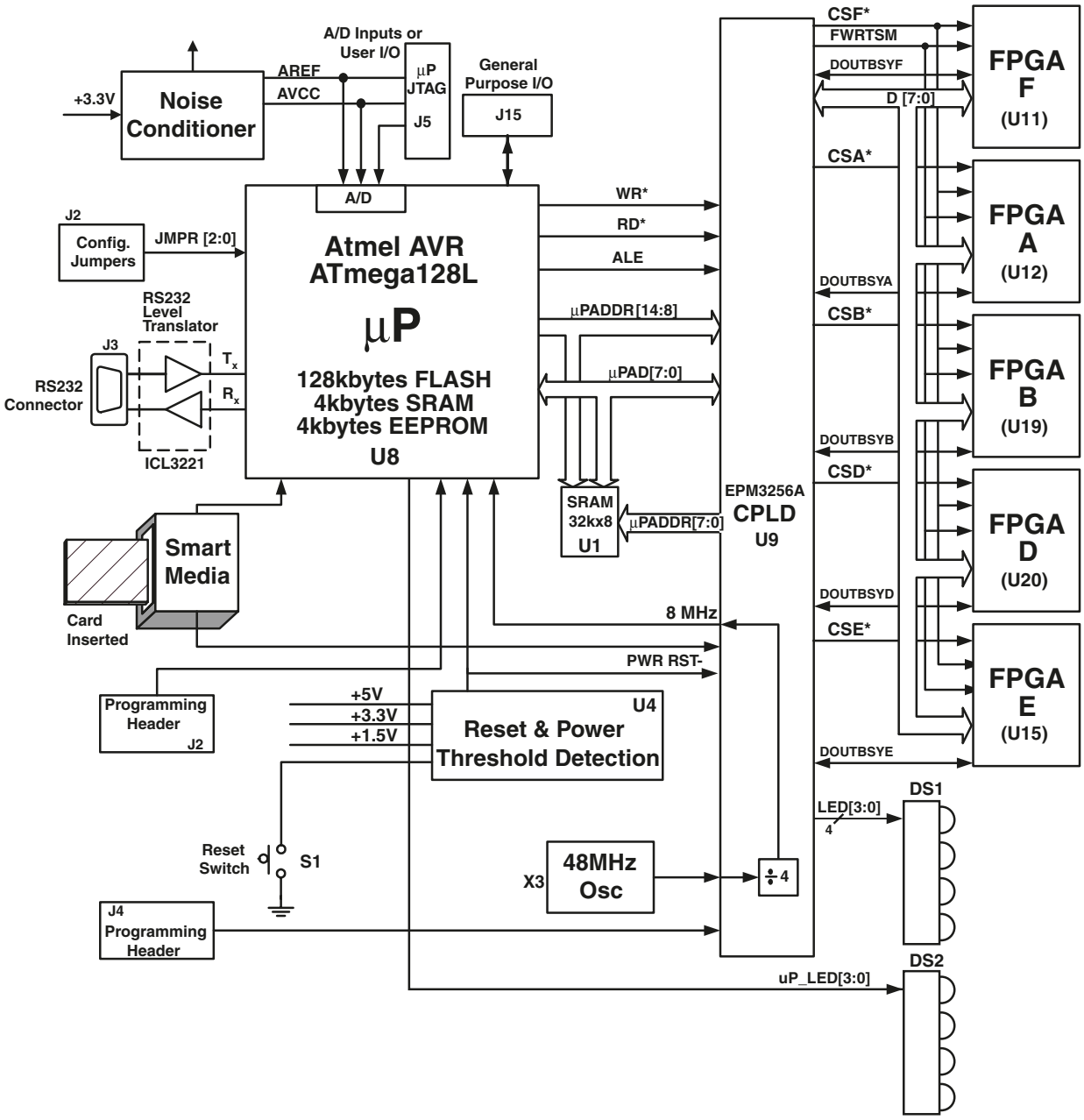


Figure 2-7 DN5000k10Block Diagram of ATmega128L and DN5000k10 Interfaces

sections that follow. Figure 2-7 is a block diagram of the ATmega128L and its various interfaces on the DN5000k10.

J6: Unused µP Connections

J6 contains connections to the ATmega128L that were not used elsewhere. These ten connections can be used for external TTL connections to the µP, externally generated interrupts, or any other function that the ATmega128L supports on these pins. Remember that the ATmega128L is not +5 V tolerant, so if you attach external TTL signals to these pins, the voltage level of these signals must not exceed +3.3 V.

The **J6** schematic is shown in Figure 2-8.

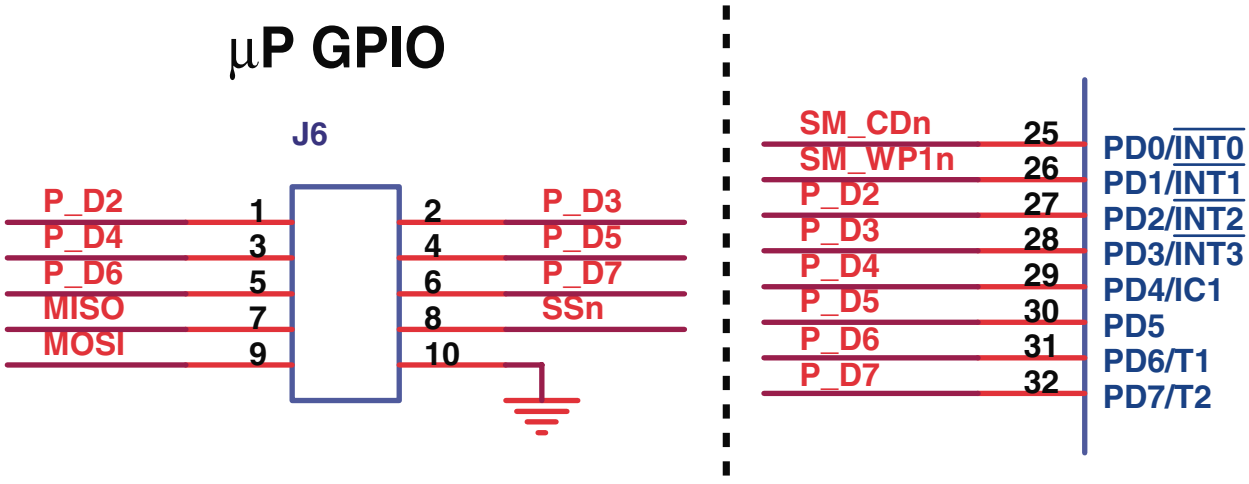


Figure 2-8 J6: Unused µP Connections

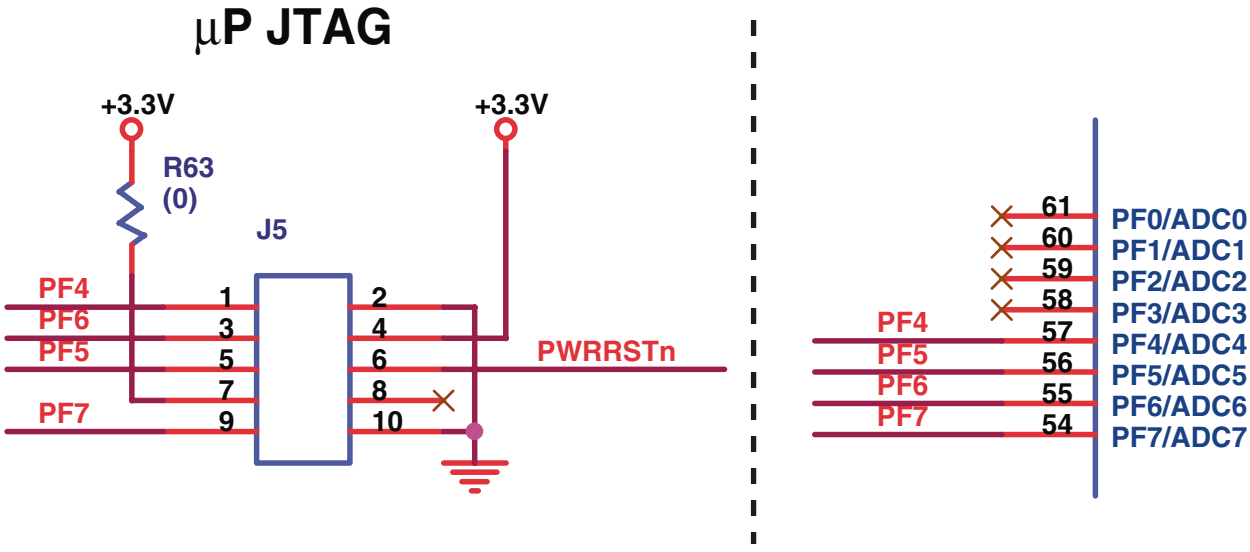


Figure 2-9 J5 JTAG Interface

ATmega128L JTAG Interface

The ATmega128L processor has a JTAG interface that can be used for on-chip debugging, real-time emulation, and programming of FLASH, EEPROM, fuses, and Lock Bits. In order to take advantage of the JTAG interface, you must have the Atmel AVR JTAG ICE kit (part number ATAVRJTAGICE) and AVR studio software that Atmel provides free at **www.atmel.com**. The JTAG interface for the ATmega128L can be accessed through header **J5** of the DN5000k10 (see Figure 2-9).

Programming the ATmega128L (U8)

A cable used to reprogram the ATmega128L is shipped with the DN5000k10. You will need to reprogram the ATmega128L if we update the code or you intend to use the processor for your own application. **J2** is used for this purpose.

Figure 2-10 illustrates **J2**.

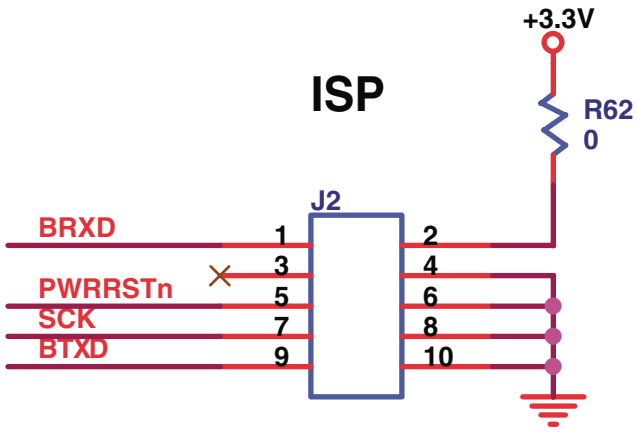


Figure 2-10 J2 Schematic

Detailed Instructions

1. Download the latest update for the processor and CPLD at **www.dinigroup.com** (file `uP_CPLD.zip`).
2. You will first need to reprogram the CPLD. Please see "CPLD—EPM3256A" on page 2-15 for instructions (use the file `DNk10S_CPLD.jed` that can be found in the downloaded zip file.
3. Next, you will program the processor (ATmetga128L). Connect the AVR cable that was shipped with the DN5000k10 to header **J2** with the red/purple wire on the cable connected to pin 1 and connect the other end to the serial port of your PC.
4. In order to program the processor, you will need to install AVR Studio that is included on the Atmel CD that was shipped with the DN5000k10. This software can also be downloaded at **www.atmel.com**.
5. From the Windows **START** menu, choose **PROGRAMS–>Atmel AVR Studio x.xx** (where x.xx is the version number).
6. Once AVR Studio is open, select **TOOLS–>STK500/AVRISP/JTAG ICE** and a new window should appear with the title **STK500**. At the bottom of the **STK500** window, if you see:

   ```
   Detecting...FAILED!
   ```

   that means either there is no power on the DN5000k10, there is another program open that is using the serial port, or the serial cable connecting the AVR tool is not connected properly. If this happens, you should close down the window titled **STK500**, correct the situation, and then select **TOOLS–>STK500/AVRISP/JTAG ICE** again. You

will not be able to continue unless you see something very similar to the following at the bottom of the **STK500** window:

```
Detecting…AVRISP found on COM1:
Getting revisions…HW: 0x01, SW Major: 0x01, SW
Minor: 0x07…OK
```

7. On the **PROGRAM** tab, select the **ATmega128** under the **DEVICE** drop down menu, and in the **FLASH** section where it says **INPUT HEX FILE**, browse and select the file `DN5000k10_128.a90` that can be found in the downloaded zip file (`uP_CPLD.zip`) from the Dini Group website. To program the device all you need to do is hit the **PROGRAM** button in the **FLASH** section. When the programming is complete (it takes about 45 seconds) you should see a message at the bottom of the window that looks something like this:

```
Detecting…AVRISP found on COM1:
Getting revisions…HW: 0x01, SW Major 0x01, SW
Minor: 0x07…OK
Reading FLASH input file…OK
Setting device parameters, serial programming
mode…OK
Entering programming mode…OK
Erasing device…OK
Programming FLASH using block mode…100% OK
Leaving programming mode…OK
```

8. After programming the processor, close all AVR Studio windows and setup the serial port according to the section titled "Setting up the Serial Port (J3 — RS232 Port)" on page 2-19. Please note that in this situation, connecting the serial port is mandatory and the FPGA cannot be configured via the SmartMedia card until you have completed all the instructions in this section.

9. Reset the DN5000k10 by pressing S1. After about 5 seconds, you should see the following in the HyperTerminal window:

```
**********NEED FPGA STUFFING INFORMATION*********

Enter number of FPGAs on Board (1–6):
```

Using the keyboard, enter the number of FPGAs on the board (should be between 1 and 5 for the DN5000k10). After you have entered this, you should see the following query:

```
Please select the FPGA on the board (F, A, E, B, or D):
```

Enter one of the FPGA locations on your board that contains an FPGA, and you should see the following menu:

```
1) Virtex II 1000 (FG456)
2) Virtex II 6000 (FF1152)
3) Virtex II 4000 (FF1152)
4) Virtex II 3000 (FG676)
5) Virtex II 8000 (FF1152)
```

```
6) Altera Apex II (2A40)
7) Altera Apex II (2A70)
8) Altera Stratix (EP1S80F1508C7)

  Please enter selection (1-6): for FPGA D:
```

Enter option 8 for Stratix FPGAs. Then, repeat these two steps for each FPGA location that contains an FPGA. If you enter the wrong number of FPGAs, or the wrong types or locations, you will need to reprogram the processor and follow these steps again. If you enter the same location twice, you'll probably need to start again also.

10. The processor and the CPLD are now ready to configure the FPGA(s). Please see the section titled "Starting Fast Passive Parallel Configuration" on page 2-22 for further instructions.

CPLD—EPM3256A

Some non-volatile logic is needed to handle the counters and state machines associated with the high-speed interface to the SmartMedia card. We used an EPM3256A CPLD from Altera for this function. The datasheet is on the CD-ROM and is titled `epm3256a.pdf`. Approximately 90% of the resources of this device are utilized, so 10% are available for your own purposes. The Verilog source for the CPLD is provided on the CD–ROM. The file name is `CPLD.V`.

The CPLD performs the following functions:

- Interface to ATmega128L µP and SRAM
  - Clock Output to µP: `BUP_CLK`
  - Data/Lower Address: `UPAD[7:0]`
  - Upper Address:`UPPADDR[15:8]`
  - Control Signals: `UP_ALE, UP_RDn, UP_WRn`
  - SRAM Select: `SRAM_CSn`
- Data Retrieval from SmartMedia Card
  - Data Bus: `SM_D[7:0]`
  - Control: `SM_CLE, SM_ALE, SM_WEn, SM_WPn, SM_CEn, SM_REn, SM_RDYBUSYn`
- Configuration and Clock Status Reporting:
  - `CPLD_LED[3:0], ROBO_LOCK1, ROBO_LOCK2`
- Control of FPGA Parallel Configuration (names with the letter "X" indicate one signal for each FPGA, for example: `FPGA_CSnA, FPGA_CSnB, FPGA_CSnD`, etc.)
  - Clock: `FPGA_DCLK`
  - Chip Select: `FPGA_CSnX, FPGA_CEnX`
  - Control: `FPGA_nCONFX, FPGA_CDONEX, FPGA_IODONEX, FPGA_RDYnBUSYX`
  - Data Bus: {`FPGA_D[7:1], FPGA_D0X`}
  - Mode Selector Switches: F`PGA_MSEL[2:0], DIP1_0`
- Pass-Through of Serial/JTAG Cable Signals
  - Cable: `DCLK/TCK, CONF_DONE/TDO, nCONFIG/TMS, nSTATUS, DATA0/TDI`

  - FPGA Chain: CPLD_TMS, CPLD_TDO, CPLD_TDI, CPLD_TCK, CPLD_TRST

- Support for Clocking Schemes:
  - CPLD Clock Input: CLK[48]
  - Inputs from Clock Buffers: ACLK[8], BCLK[8]
  - Output to Clock Grid: BCPLD_CLKOUT
- Interface to Reset Schemes:
  - FPGA_GRSTn, PWR_RSTn

We may periodically update the CPLD. The CPLD can be reprogrammed using the Altera JTAG cable supplied with the DN5000k10. The connections are on header **J4**. The relevant signals and the connections to **J4** are listed in Table 2-1. Figure 2-10 shows the location of **J**4.

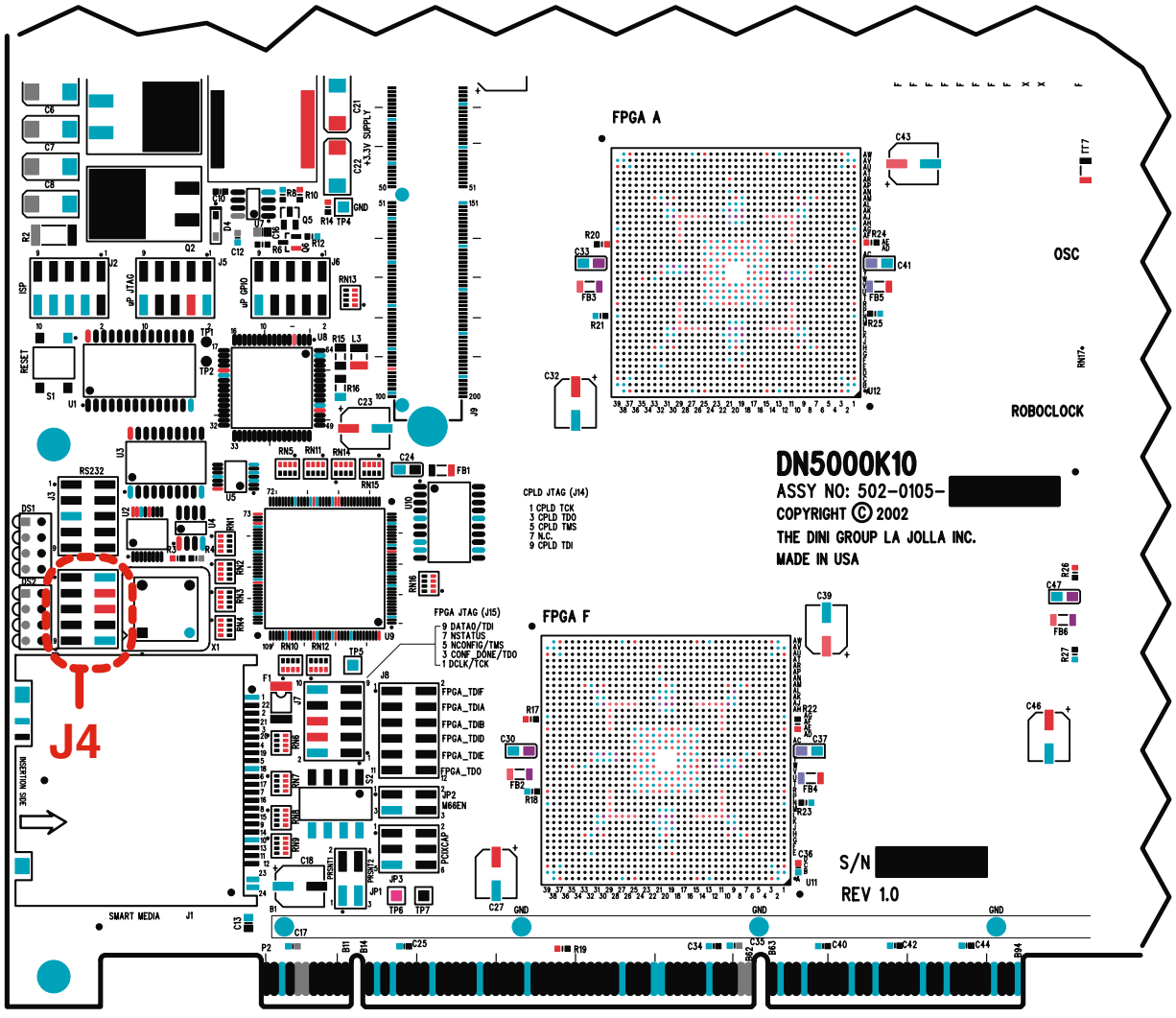


Figure 2-11 Location of J4 on the DN5000k10

Table 2-1 Signals and Connections to J4

JTAG Cable	J4 Signal Name	J4 Pin
VCC	+3.3 V	4, 6
GND	GND	2, 10
TCK	JTAG_CPLD_TCK	1
TDO	JTAG_CPLD_TDO	3
TDI	JTAG_CPLD_TDI	9
TMS	JTAG_CPLD_TMS	5

Some Miscellaneous Notes on the CPLD

X1 is a 48 MHz oscillator. This part is soldered down to the PWB and is not intended to be user-configurable. The 48 MHz is divided down to 8 MHz in the CPLD to provide the clock for the ATmega128L µP. The processor clock signal is labeled CPUCLK (and BCPUCLK) on the schematic.

The 48 MHz is used directly for the state machines in the CPLD for controlling the interface to the SmartMedia card. The frequency of 48 MHz is interesting because it is the closest frequency to 50 MHz that can be divided by an integer to get 8 MHz. The frequency 50 MHz is the fastest that the Altera Stratix parts can be configured with SelectMap without wait states. So FPGA configuration using Fast Passive Parallel occurs at very nearly the fastest theoretical speed.

Serial and JTAG configuration of the Stratix FPGA are back off positions only—that is why those signals are connected to the CPLD. Fast Passive Parallel is the quickest configuration method, but we wanted to provide the user as many options as possible.

If you want to use 100% of the CPLD and µP for your own purposes, you can configure the FPGA using the JTAG cable.

The 48 MHz clock can be divided down in the CPLD and used to drive the PWB clock network. See Chapter 4 for a more detailed description of this option.

Notes on Header J7

Fast Passive Parallel using the SmartMedia card is the best way to configure the FPGA. Two other options exists if, for some reason, the SmartMedia card method is not applicable.

1. Serial Programming Using the Cable. Header **J7** has the 5 serial connections that are used to configure the FPGA using the serial method. Table 2-2 has the pinouts. Note that this is a back-off position to SmartMedia and JTAG and should only be used in dire circumstances. Note also that the switches on **J2** will need to change to reflect “slave-serial” configuration.

2. JTAG Programming.
   The JTAG connection can be used to configure the FPGA and can also be used to connect the SignalTap Logic Analyzer (See Application Note 175 at **www.altera.com/literature/lit-qts.html**) or other solutions such as the Bridges2silicon system, which was recently acquired by Synplicity (see **www.bridges2silicon.com**). The JTAG method of configuration should be used if the SmartMedia method isn't working. Remember that programming a Stratix part through JTAG uses a `.sof` file, not a `.rbf` file. Table 2-2 has the pinouts.

Table 2-2 FPGA Serial/JTAG Configuration Header

Name on Schematic	Name on Cable		Header Pin (J7)
	Serial Mode	JTAG Mode	
DCLK/TCK	DCLK	TCK	1
CONF_DONE/TDO	CONF_DONE	TDO	3
DATA0/TDI	DATA0	TDI	9
nCONFIG/TMS	nCONFIG	TMS	5
nSTATUS	nSTATUS	(none)	7
GND	GND	GND	2, 10
VCC	VCC	VCC	4, 6

Fast Passive Parallel Configuration Instructions

The FPGA on the DN5000k10 can be configured in Fast Passive Parallel mode using a Smart Media card. Fast Passive Parallel configuration is the easiest and quickest way to configure the FPGA. The DN5000k10 is shipped with two 32 MB Smart Media cards. One of these Smart Media cards contains reference design bit files produced for Fast Passive Parallel configuration, and files `main.txt` and `iotst.txt` that sets options for the configuration process (for description of options, see "Creating Main Configuration File main.txt" on page 2-20). This Smart Media card has been labeled with a sticker marked "reference design." The other Smart Media card is empty and is for use with your own designs. To configure the FPGA with the reference design, please skip to "Starting Fast Passive Parallel Configuration" on page 2-22.

Creating RBF Files for Fast Passive Parallel

To create an RBF file with QuartusII software:

Go to Assignments menu and drag down to Settings. Click on Device under Compiler Settings on the left, then click the Device & Pin Options button on the right. Go to the Configuration tab, select Configuration Scheme = Fast Passive Parallel, and disable the option to Use Configuration Device. Go to the Programming Files tab, turn on Raw Binary File (`.rbf`), and turn off all other options. (Note: the `.sof` file for JTAG programming will also be created.)

The easy way to assign pins is to create your project, then open the .csf file created in a text editor. If your pinlist is formatted correctly, you can copy it and paste it into the `.csf` file in the section labeled "CHIP (design_name)". Sample files on the software CD provided, found in the folder labeled Verilog, show how to format the information and provide the correct pinlist for the signal names used on the board.

Setting up the Serial Port (J3 — RS232 Port)

J3 is for an RS232 connection to a terminal. An ICL3221 (**U2**) provides voltage translation to RS232 levels. A cable that converts the 10-pin header to a DB9 is shipped with the DN5000k10. This cable comes packaged with a bracket attached. Remove the bracket to eliminate the possibility of it falling on the DN5000k10, which could short signals and damage the board. After you have removed the bracket, plug the cable into **J3**. **J3** is not keyed—so make sure you get the orientation correct. Pin 1 is identified with the number 1 and a dot. Figure 2-12 is a cutout from the assembly drawing, and shows the location of **J3** and Pin 1.

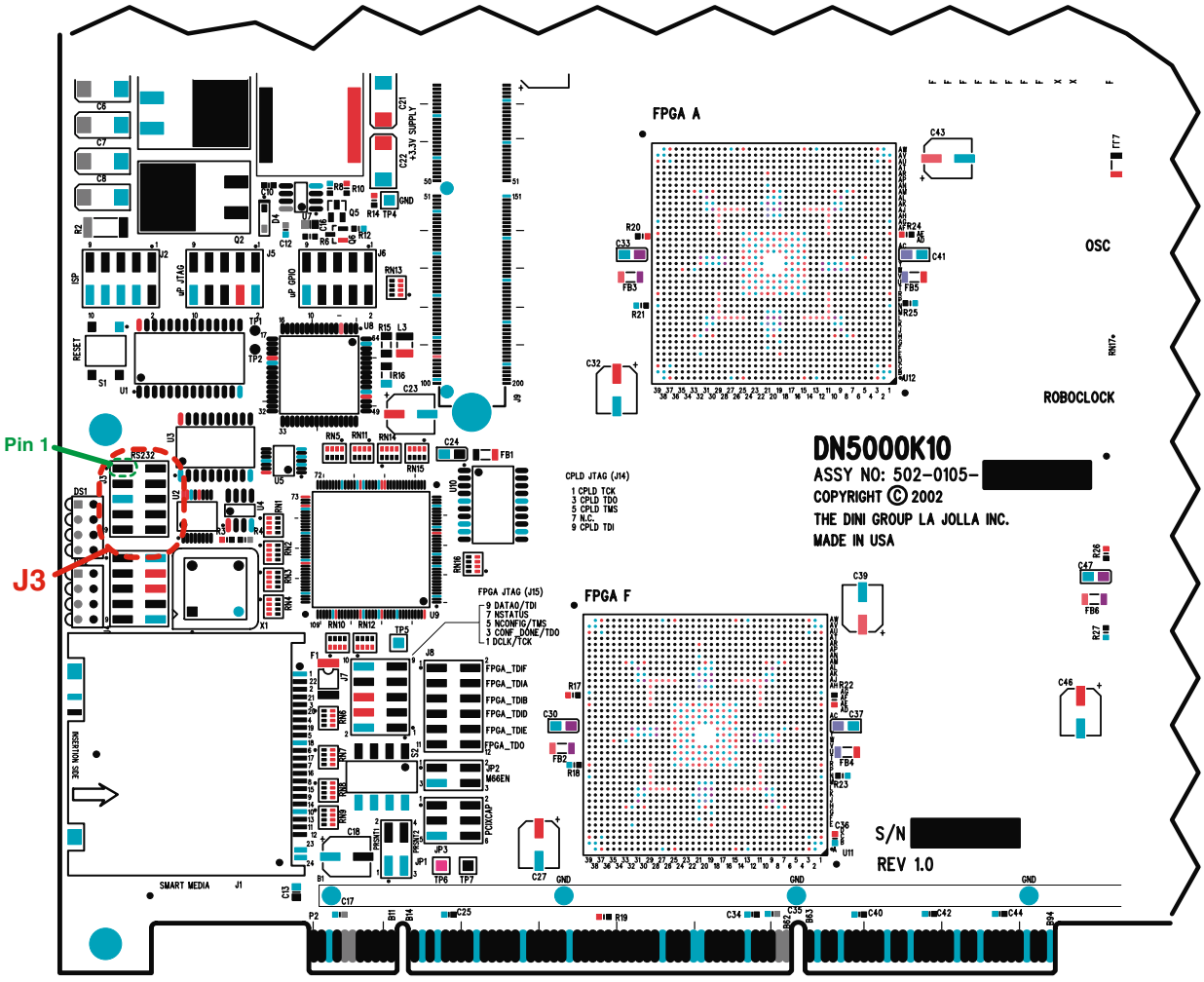


Figure 2-12 J3 Serial Port Locations

A female-to-female RS232 cable is provided with the DN5000k10. This cable will attach directly to the RS232 port of a PC. We get our cables from Jameco (**http://www.jameco.com**).The part number is 132345. Male-to-female extension cables are part number 25700.

The RS232 port is configured with the following parameters:

Bits per second:9600
Data bits:8
Parity:None
Stop Bits:1
FLow control:None
Terminal Emulation:VT100

We use the Windows-based program HyperTerminal (`Hypertrm.exe`). The configuration file `DN5000k10.ht` is supplied on the CD-ROM or can be downloaded from our web page.

Users have the option of connecting the serial port if they wish to see any messages during the configuration process.

NOTE: It is NOT mandatory to have the serial port connection in order to configure the FPGA in SelectMAP mode. However, if an error occurs during the configuration, then without a serial port connection the user will not be able to see any error messages. In addition, without a serial port connection, a user cannot select any Main Menu options after the configuration process is complete.

Creating Main Configuration File main.txt

To control which bit file on the Smart Media card is used to configure the FPGA in SelectMAP mode a file named `main.txt` must be created and copied to the root directory of the Smart Media card. The configuration process cannot be performed without this file. Below is a description of the options that can be set in the file, a description of the format this file needs to follow, and an example of a `main.txt` file.

Options:

Verbose Level — During the configuration process, there are three different verbose levels that can be selected for the serial port messages:

- Level 0:
  - Fatal error messages
  - Sanity Check errors (e.g., RBF file was created for the wrong part, RBF file was created with wrong version of Altera tools, or Quartus options are set incorrectly)
  - Initializing message will appear before configuration
  - A single message will appear once the FPGA is configured
- Level 1:
  - All messages that Level 0 displays
  - Displays configuration type (should be Fast Passive Parallel
  - Displays current FPGA being configured if the configuration type is set to Fast Passive Parallel
  - Displays a message at the completion of configuration for each FPGA configured.

- Level 2:
  - All messages that Level 1 displays
  - Options that are found in `main.txt`
  - RBF file names for each FPGA as entered in `main.txt`
  - Maker ID, Device ID, and size of Smart Media card
  - All files found on Smart Media card
  - If sanity check is chosen, the RBF file attributes will be displayed (part, package, date, and time of the RBF file)
  - During configuration, a "." will be printed out after each block (16 KB) has successfully been transferred from the Smart Media to the current FPGA.

Sanity Check — The Sanity Check if enabled, verifies that the RBF file was created for the right part, the right version of Altera was used, and the Quartus options were set correctly. If any of the settings found in the RBF file are not compatible with the FPGA, a message will appear from the serial port, and the user will be asked whether or not they want to continue with the RBF file. Please see the section "Creating RBF Files for Fast Passive Parallel" on page 2-18 for details on which Quartus options need to be changed from the default settings.

Format:

The format of the `main.txt` file is as follows:

- The first nonempty/uncommented line in `main.txt` should be:

  ```
  Verbose level: X
  ```

  where "X" can be 0, 1 or 2. If this line is missing or X is an invalid level, then the default verbose level will be 2.

- The second nonempty/uncommented line in `main.txt` tells whether or not to perform a sanity check on the bit files before configuring an FPGA:

  ```
  Sanity check: y
  ```

  where "y" stands for yes, "n" for no. If the line is missing or the character after the ":" is not "y" or "n" then the sanity check will be enabled.

- For each FPGA that the user wants to configure, there should be exactly one entry in the `main.txt` file with the following format:

  ```
  FPGA F: example.rbf
  ```

  In the above format, the "F" following FPGA is to signal that this entry is for FPGA F, and FPGA F would then be configured with the bit file `example.rbf`. The DN5000k10 has one to five FPGAs, which are FPGA A, B, D, E and F. The example has only one FPGA, which is FPGA F. There can be any number of spaces between the ":" and the configuration file name, but they need to be on the same line.

- Comments are allowed with the following rules:
  1. All comments must start at the beginning of the line.
  2. All comments must begin with //
  3. If a comment spans multiple lines, then each line must start with //

  Commented lines will be ignored during configuration, and are only for the user's purpose.
- The file `main.txt` is **NOT** case sensitive.

IMPORTANT: All configuration file names have a maximum length of eight (8) characters, with an additional three (3) for the extension. Do not name your configuration files with long file names. In addition, all file names should be located in the root directory of the Smart Media card—no subdirectories or folders are allowed. Since the `main.txt` file controls which file is used to configure the FPGA, the Smart Media card can contain other files.

Example of `main.txt`:

```
//start of file "main.txt"
Verbose level: 2
Sanity check: y

FPGA F: fpgaF.rbf
//the line above configures FPGA F a file "fpgaF.rbf"

//end of main.txt
```

Given the above example file:

- Verbose level is set to 2
- A sanity check on the bit files will be performed
- FPGA F will be configured with file `fpgaF.rbf`.

Starting Fast Passive Parallel Configuration

If using the reference design SmartMedia card that came with the DN5000k10 then no files need to be copied to the card. Otherwise, copy your RBF file and `main.txt` to the root directory of the SmartMedia card using the FlashPath floppy adapter. Make sure the switches on **S2** are set for Fast Passive Parallel as shown in Table 2-3.

Table 2-3 J2 Configuration Jumper Settings

Switch 0 MSEL[2]	Switch 1 MSEL[1]	Switch 2 MSEL[0]	Configuration Mode
off	off	off	Fast Passive Parallel
off	on	off	Passive Serial

Set up the serial port connection as described above in "Setting up the Serial Port (J3 — RS232 Port)" on page 2-19. Next, place the SmartMedia card in the SmartMedia socket on the DN5000k10 and turn on the power (NOTE: the card can only go in one way). The SmartMedia card is hot-

swappable and can be taken out or put into the socket even when the power is on. Once the power has been turned on, the configuration process will begin as long as there is a valid SmartMedia card inserted properly in the socket. If there is not a valid SmartMedia card in the socket, then UP_LED[3:0] will flash (see Figure 8-2 on page 8-3 for LED descriptions) and the Main Menu will appear from the serial port. A SmartMedia card is determined to be invalid if either the format of the card does not follow the SSFDC specifications, or if it does not contain a file named `main.txt` in the root directory. If the configuration was successful, a message stating so will appear and the **Main Menu** will come up. Otherwise, an error message will appear.

The LEDs on **DS1** and **DS2** give feedback during and after the configuration process; see "LEDs" on page 8-3 for further details.

After the FPGA has been configured, the following **Main Menu** will appear on the serial port:

```
1. Configure FPGA(s) using main.txt
2. Interactive FPGA configuration menu
3. Check Configuration status
4. Select file to use in place of main.txt
5. List files on SmartMedia
6. Select FPGA to program via JTAG
```

Description of Main Menu Options.

1. **Configure FPGAS Using "`main.txt`" as the Configuration File**— By selecting this option, the FPGA will configure in Fast Passive Parallel mode. You can also press the reset button (**S1**) to reconfigure the FPGA in Fast Passive Parallel mode.
2. **Interactive FPGA configuration menu** — This option takes you to a menu titled "**Interactive Configuration Menu**" and allows the FPGA to be configured through a set of menu options instead of using the `main.txt` file. The menu options are described below.

   Description of **Interactive Configuration Menu** options:

   1. **Select a bit file to configure FPGA(s)** — This menu option allows the user to select a file from a list of files found on the SmartMedia card to use to configure the FPGA.
   2. **Set verbose level (current level = 2)** — This menu option allows the user to change the verbose level from the current setting. Please note: if the user goes back to the main menu and configures the FPGA(s) using `main.txt`, the verbose level will be set to whatever setting is specified in `main.txt`.
   3. **Disable/Enable sanity check for bit files** — This menu option either allows the user to disable or enable the sanity check, depending on what the current setting is. Please note: if the user goes back to the main menu and configures the FPGA(s)

using main.txt, the sanity check will be set to whatever setting is specified in main.txt.

M) **Main menu** — This menu option takes the user back to the Main Menu described above.

3. **Check Configuration status** — This option checks the status of the **DONE** pin and prints out whether or not the FPGA(s) have been configured along with the file name that was used for configuration.
4. **Select file to use in place of** main.txt — By default, the processor uses the file main.txt to get the names of the files to be used for configuration as well as options for the configuration process. However, a user can put several files that follow the format for main.txt on the SmartMedia card that contain different options for the configuration process. By selecting the main menu option 4, the user can select a .txt file from a list of files that should be used in place of main.txt. After selecting a new file to use in place of main.txt, the user should select Main Menu option 1 to configure the FPGA(s) according to this new file. If the power is turned off or the reset button (**S1**) is pressed, the configuration file is changed back to the default, main.txt.
5. **List files on SmartMedia** — This option prints out a list of all the files found on the SmartMedia card.
6. **Select FPGA to Program with JTAG** — This option must be set to enable an FPGA before it can be programmed through JTAG.

SmartMedia

The configuration file for the FPGA is copied to a SmartMedia card using the SmartDisk FlashPath Floppy Disk Adapter. The approximate file size for each possible Stratix FPGA is shown below in Table 2-4. Note that several files can be put on a 32-megabyte card. We supply two 32-megabyte SmartMedia cards with the DN5000k10. SmartMedia is a standard, so you can get more SmartMedia cards if you want. The DN5000k10 requires a +3.3 V card. Card sizes of 16, 32, 64, and 128 megabytes have been tested on the DN5000k10. We have not seen 256 MB or larger cards for sale yet, but when we do there will probably be an update to the CPLD and processor on our website to support them.

Table 2-4 Stratix FPGA Approximate File Sizes

Stratix FPGA	Number of Configuration Bytes
SOF for EP1S80	2,954,672
RBF for EP1S80	2,992,071

We get our SmartMedia cards from **http://www.computers4sure.com/**. A Delkin Devices 16-megabyte card (part number DDSMFLS2-16) sells for about $15. A 32-megabyte card (part number DDSMFLS2-32) will set you

back about $20 (see Figure 2-13). New SmartMedia cards do not require formatting before use.

NOTE: SmartMedia cards do not need to be formatted before they are used. The Windows format command **does not work**—it is necessary to use the FlashPath utility to format a SmartMedia card.

Figure 2-13 Delkin 32 MB 3.3 V Smart Media Card

Do not press down on the top of the SmartMedia Connector **J1** if a Smart-Media card is not installed. The metal case shorts to the +3.3 V power supply and the case gets hot enough to burn your finger. We suggest that you leave a SmartMedia card in the connector to prevent this from occurring. A polyswitch fuse (**F1**) has been added so that the PWB and the SmartMedia connector are protected if you do accidently press on the top of the connector.

NOTE: Do NOT press on the SmartMedia Connector J28 if a card is not installed!

WARNING:

Do NOT format a SmartMedia card using the default Windows format program. All Smart Media cards come preformatted from the factory, and files can be deleted from the card when they are no longer needed. If for some reason you absolutely need to format a SmartMedia card, you must use the format program that is included in the FlashPath (Smart-Media floppy adapter) software.

Synthesis and Emulation Issues

The QuartusII™ software from Altera is able to synthesize directly from Verilog or VHDL code. However, third-party synthesis tools provide an advantage: to create a memory block or multiplier, all you need to do is describe them functionally, and the tool will infer the appropriate DSP and RAM megafunctions for Quartus to place and route. On the other hand, if you are using Quartus to synthesize, and you try to infer an M-RAM block using a functional description, Quartus will attempt to route 200,000 LEs as a memory array. So, if you don't have any other synthesis tool, you will need to become familiar with Quartus megafunctions.

We have tried the following tools for synthesis:

Synplicity Synplify (http://www.synplicity.com/)

Synopsys FPGA Express (http://www.synopsys.com/)

Synopsys FPGA Compiler II

Exemplar LeonardoSpectrum (http://www.exemplar.com/products/leonardospectrum.html)

Of the four listed here, we find that Synplicity offers the best performance, followed by Exemplar. The Synopsys products are not the easiest products to use, and probably should be avoided until Synopsys decides that they want to be in this market. It is generally not worth your time to preserve your Synopsys ASIC compiler directives and scripts by using the FPGA synthesis products from Synopsys. The time you save using Snopsys products is offset by other hassles.

Synthesis Notes

1. The FPGAs used on your DN5000k10 are EP1S80s in an F1508 package (EP1S60s available on request). Unless you paid for a faster speed grade, the –6 is what you will be getting.
2. Assuming you have a synthesis tool other than QuartusII, memories are best implemented by describing them behaviorally in your RTL. All four synthesis products are sophisticated enough to map your behavioral descriptions into the memory blocks. It is **NOT** necessary to instantiate memories manually, unless you are synthesizing with Quartus. Make sure, however, to check the report files to make sure that your memories were implemented in memory blocks (if this is possible). If input and output registers in your RTL don’t match the behavior of the embedded memory blocks, the synthesis program may not recognize what you intended, and give you arrays of LEs instead.
3. Much to our surprise, the synthesis programs recognized RTL multiplier code and used the embedded multipliers without any trouble. So, like the memories, RTL description of your multipliers is all that is necessary unless you are synthesizing with Quartus. Make sure to check the report files—multipliers that are implemented using logic blocks (as opposed to the embedded memory blocks) take huge amounts of FPGA resources.

4. Clocks are the biggest problem when converting ASIC code to FPGA code. FPGAs only have a limited number of clock arrays. This is far too complicated to describe here, so get the *Stratix Data Sheet* and read about the clocks.

Chapter 3

PCI

Overview

The DN5000k10 can be hosted in a 32-bit, or 64-bit PCI slot. PCI-X is also supported. Stand-alone operation is described in "Stand-Alone Operation" on page 6-3. An EP1580-7, with care, should be able to support a 64-bit, 66 MHz PCI or PCI-X controller. We have not tested the PWB at PCI-X speeds of 100 MHz and 133 MHz. We suspect, but won't guarantee, that the DN5000k10 can support these high frequencies, provided the speed grade of the FPGA is adequate. Figure 3-1 shows the FPGA pin connections for the PCI signals. This data is provided on the CD-ROM in a CSF file titled `pins_F.csf`, for your convenience.

The PCI/PCI-X edge connector is shown in Figure 3-2.

Stratix parts cannot tolerate +5 V TTL signaling, so the DN5000k10 must be plugged into a +3.3 V PCI slot. PCI-X, by definition, is +3.3 V signaling. The PWB is keyed so that it is not possible to mistakenly plug the board into a +5 V PCI slot. Do NOT grind out the key in the PCI host slot, and Do NOT modify the DN5000k10 to get it to fit into the slot. If you need a +3.3 V PCI slot, the DNPCIEXT-S3 Extender card can do this function. The link is **http://www.dinigroup.com/products/pciextender.html**. This extender also has the capability to slow the clock frequency of the PCI bus by a factor of two—a function that is very useful when prototyping ASICs.

NOTE: +5 V Signaling on Stratix parts causes them to smoke! This is quite BAD! Do NOT Modify the DN5000k10 board to fit into your pci slot.

PCI Mechanical Specifications

The DN5000k10 is **not** a standard sized PCI card—it is too tall and slightly too long. This is sometimes an issue in servers that have a bracket installed over the top of the PCI cards. If you need to close the case on a DN5000k10, some tower configurations may work. Figure 3-3 shows the exact dimensions of the DN5000k10.

Some Notes on the DN5000k10 and PCI/PCI-X

+3.3 V power is not needed on the host PCI connector. +3.3 V power is derived from +5 V using an on-board 10 A switching regulator. Power distribution for the DN5000k10 is described in "Power Supplies and Power Distribution" on page 6-1.

`LOCK#` has a pull-up. This is technically a violation of the PCI specification, but we have seen systems (from SUN!) that have the `LOCK#` pin floating. Remember that the function of this pin was deleted in the 2.2 version of

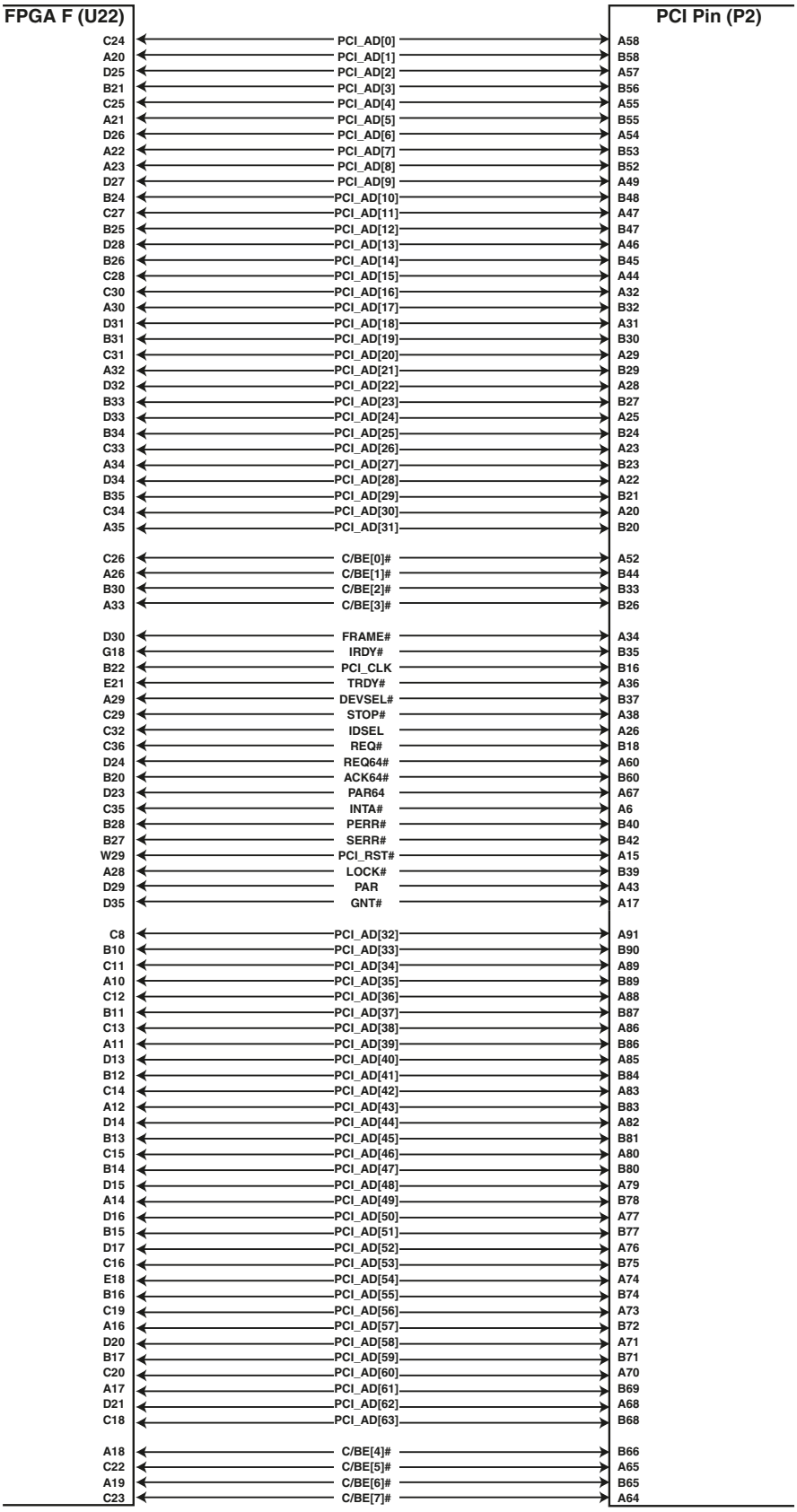


Figure 3-1 FPGA Pin Connections for PCI Signals

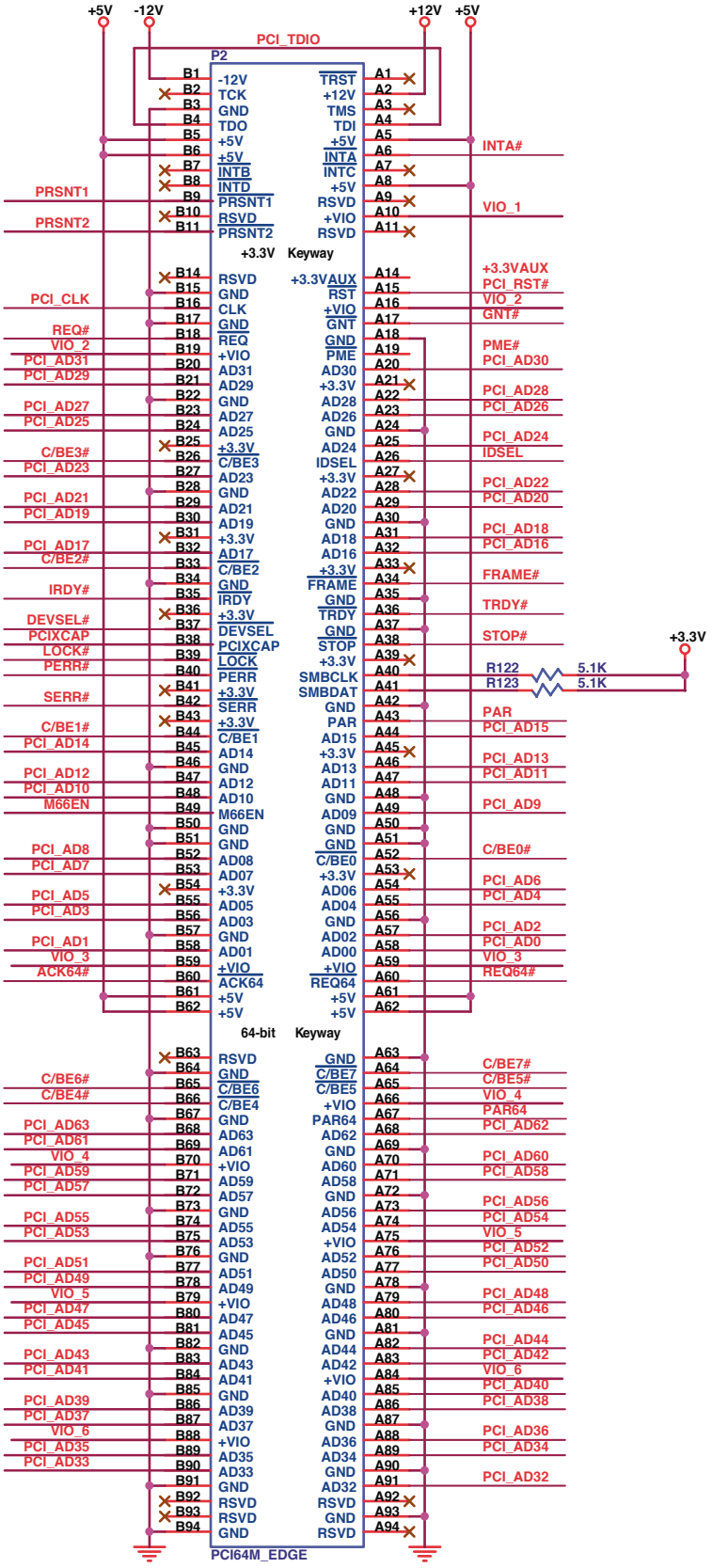


Figure 3-2 PCI/PCI-X Edge Connector

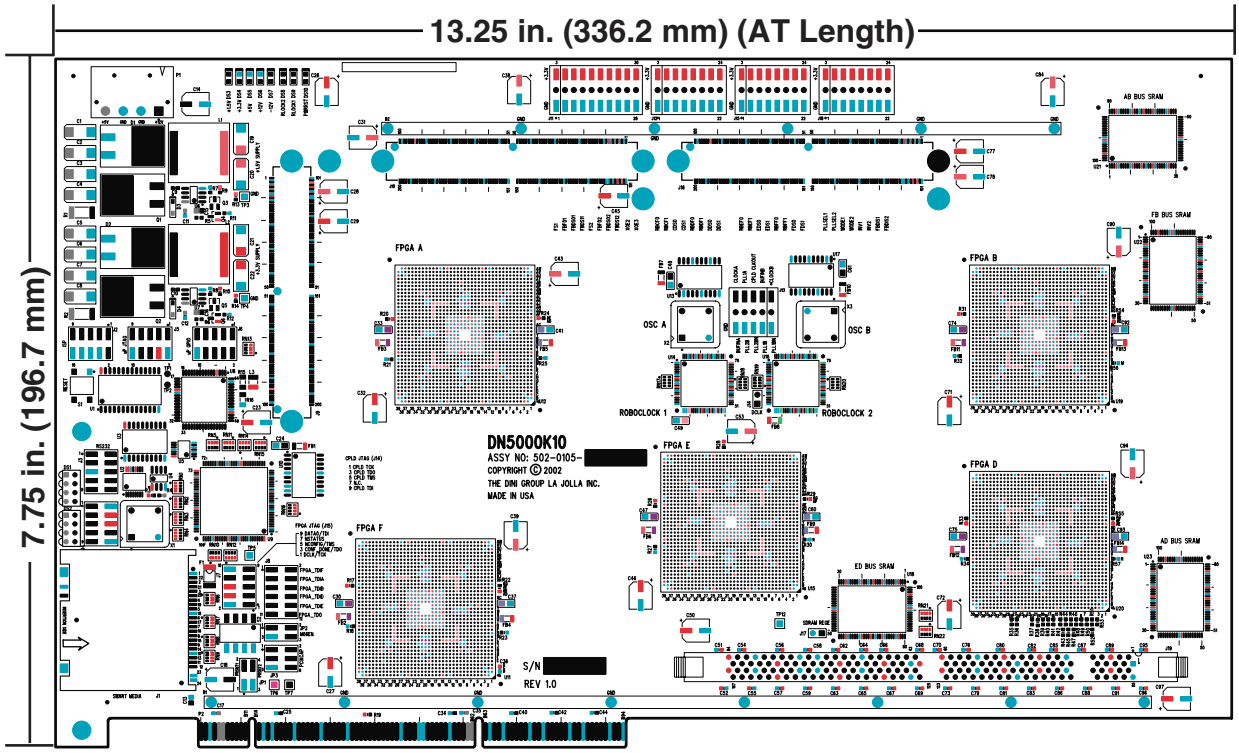


Figure 3-3 DN5000k10 Dimensions

the PCI Specification. The pull-up is 1M, which should not adversely impact PCI functionality in any way.

The PCI JTAG signals TDI, TDO, TCK, TMS, TRST#, are not used. TDI and TDO are connected together per the PCI Specification to maintain JTAG chain integrity on the motherboard. The signals TMS, TCK, and TRST# are left unconnected.

The Stratix FPGAs are not +5 V tolerant, so you must plug the DN5000k10 into a +3.3 V PCI slot. Do NOT modify the connector to get the board to fit. If you need +5 V to +3.3 V PCI voltage translation, get one of our extenders. The link is:

http://www.dinigroup.com/products/pciextender.html.

The FPGA is volatile, meaning it loses its brains when power is off. The SmartMedia method takes about 1 second to configure an EP1580 after power is stable. It takes about 5 seconds to configure five EP1580s. It is likely that FPGA F will finish the configuration process before RST# is deasserted. If your system has an unusually fast RST#, it is possible that the FPGA will not be configured when RST# deasserts. A RST# that deasserts before the FPGA has finished cannot properly configure the PCI/PCI-X mode latch.

The signal 3.3Vaux is not connected.

The signals INTB#, INTC#, and INTD# are not connected.

JP1: Present Signals for PCI/PCI-X

The present signals indicate to the system board whether an add-in card is physically present in the slot and, if one is present, the total power requirements of the add-in card.

The **JP1** PCI-X Present Header is shown in Figure 3-4.

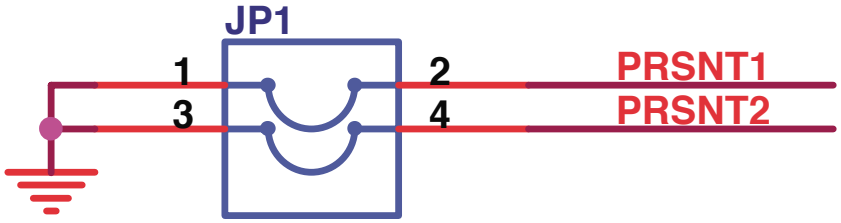


Figure 3-4 JP1 PCI-X Present Header

Table 3-1 shows the Present Signal Definitions for PCI/PCI-X.

Table 3-1 Present Signal Definitions

PRSNT1#	PRSNT2#	Expansion Configuration
Open	Open	No expansion board present
Ground	Open	Expansion board present, 25W maximum
Open	Ground	Expansion board present, 15W maximum
Ground	Ground	Expansion board present, 7.5W maximum.

We have never seen the present signals used anywhere, but we have heard of systems that will not PNP (Plug-and-Play) configure a PCI board if both the present pins are left open. We recommend installing a jumper in location 1-2 (for PRSNT1-), or 3–4(for PRSNT2-), or both.

JP2: M66EN—66MHz Enable

The `66MHZ_ENABLE` pin (**M66EN**) indicates to the host whether the device can operate at 66 MHz or 33 MHz. Section 7.5.1 in the *PCI Specification 2.2* provides the gory details. For 33 MHz only FPGA designs, install a jumper between pins 3 and 4. For 66 MHz capable designs, install a jumper between pins 1 and 2 instead. Table 3-2 shows the jumper descriptions for M66EN.

Table 3-2 M66EN Jumper Descriptions

	Jumper	Description
M66EN	Pins 3-4	33 MHz
	Pins 1-2	66 MHz

TP7: PME–, Power Management Enable

This board does not have built-in support for **PME–** (power management enable). Connecting **PME–** to an FPGA that is not powered is a bad idea—the system powers up as the board is installed. **PME–** is connected to **TP7**. This test pin allows the user to connect external circuitry to **PME–** if this functionality is desired.

JP3: PCI/PCI-X Capability

Figure 3-5 shows the PCI-X Capabilities Header. Add in PCI-X boards tell the system what speed they are capable of running by the correct setting of this header.

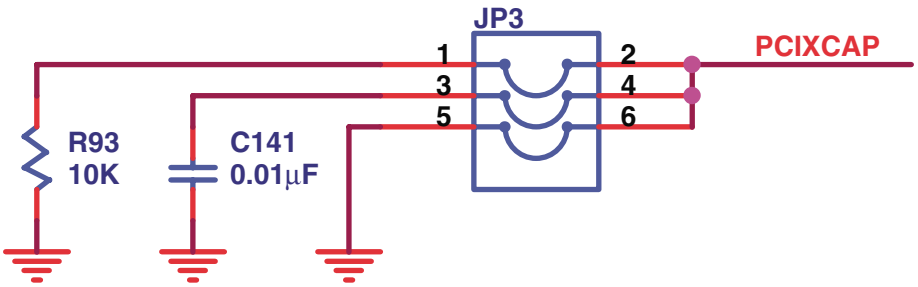


Figure 3-5 PCI-X Capability Header

Add-in cards indicate at which frequency they support PCI-X, using a pin called **PCIXCAP**. If the card's maximum frequency is 133 MHz, this pin is left unconnected (except for a decoupling capacitor **C141**). If the card's maximum frequency is 66 MHz, it connects **PCIXCAP** to ground through a resistor **R93** (and decoupling capacitor **C141**). Conventional PCI cards connect this pin to ground.

JP3—PCIXCAP

For PCI only (not PCI-X capable), jumper between pins 5 and 6.

For PCI-X 133 MHz capable, jumper between pins 3 and 4.

For PCI-X 66 MHz capable, jumper between pins 1 and 2 and pins 3 and 4.

The **PCIXCAP** jumpers are detailed in Table 3-3.

Table 3-3 PCIXCAP Jumpers

PCIXCAP	Jumper(s) Installed
PCI Only	5–6
PCI-X 133 MHz	3–4
PCI-X 66 MHz	1-2, 3–4

The M66EN and PCIXCAP Encodings are shown in Table 3-4.

Table 3-4 M66EN and PCIXCAP Encoding

M66EN	PCIXCAP	Conventional Device Frequency Capability	PCI-X Device Frequency Capability
Ground	Ground	33 MHz	Not Capable
Not Connected	Ground	66 MHz	Not Capable
Ground	Pull-down	33 MHz	PCI-X 66 MHz
Not Connected	Pull-down	66 MHz	PCI-X 66 MHz
Ground	Not Connected	33 MHz	PCI-X 133 MHz
Not Connected	Not Connected	66 MHz	PCI-X 133 MHz

Chapter 4

Clocks and Clock Distribution

Functional Overview

The DN5000k10 ASIC emulation board has a flexible and configurable clock scheme. Figure 4-1 is a block diagram showing the clocking resources and connections.

The clocking structures for the DN5000k10 include the following features:

- 2 user-selectable socketed oscillators (**X2**, **X3**)
- 1 48 MHz oscillator (**X1**)
- 2 CY7B993 (or CY7B994) RoboclockII™ Multi-Phase PLL Clock Buffers
- 2 FCT3807 Low-Skew Clock Buffers

The Clock Grid, **J13:** a 5X3 0.1 in. header distributes clock signals to two FCT3807 clock buffers and two RoboclockII™ PLL clock buffers (CY7B993 or CY7B994). The clock outputs from the buffers are dispersed throughout the board.

Two 3.3 V half-can oscillator sockets (**X2** and **X3**) and the signal CLKOUT from the CPLD provide on-board input clock solutions. The DN5000k10 is shipped with both a 14.318 MHz (**X2**) and a 33 MHz (**X3**) oscillator. Neither **X2** nor **X3** are used by the configuration circuitry, so the user is free to stuff any standard 3.3 V half-can oscillator in the **X2** and **X3** positions (more detail later in "Customizing the Oscillators" on page 4-14). The Clock Grid can also accept a 5X2 ribbon cable. This cable can provide input clocks to both of the RoboclockII's and one of the 3807 buffers.

The FCT3807 clock buffer provides a high speed 1-to-10 buffer with low skew (0.35 ns) allowing clocks A (ACLK[9:0]) and B (BCLK[9:0]) to be distributed point-to-point. The two RoboclockII PLL clock buffers (**U14** and **U16**) offer functional control of clock frequency and skew, among other things. They are configured via header arrays (**J11, J12, J15** and **J18)**. The DN5000k10 comes from the factory stuffed with CY7B994V, which can operate at frequencies from 24 MHz to 200 MHz. They can also be stuffed with CY7B993V which operate from 12 MHz to 100 MHz. (Note: Output frequency can be as low as 1 MHz, depending on the operating frequency—see below for details.) Each chip has 16 output clocks along with 2 feedback output clocks. Two sets of eight output clocks are jumper selectable for each chip. The feedback clocks are controlled separately.

The PLL clock buffers can accept either 3.3 V LVTTL or LV Differential (LVPECL) reference inputs. The devices can operate at up to 12x the input frequency while the output clocks can be divided up to 12x the operating

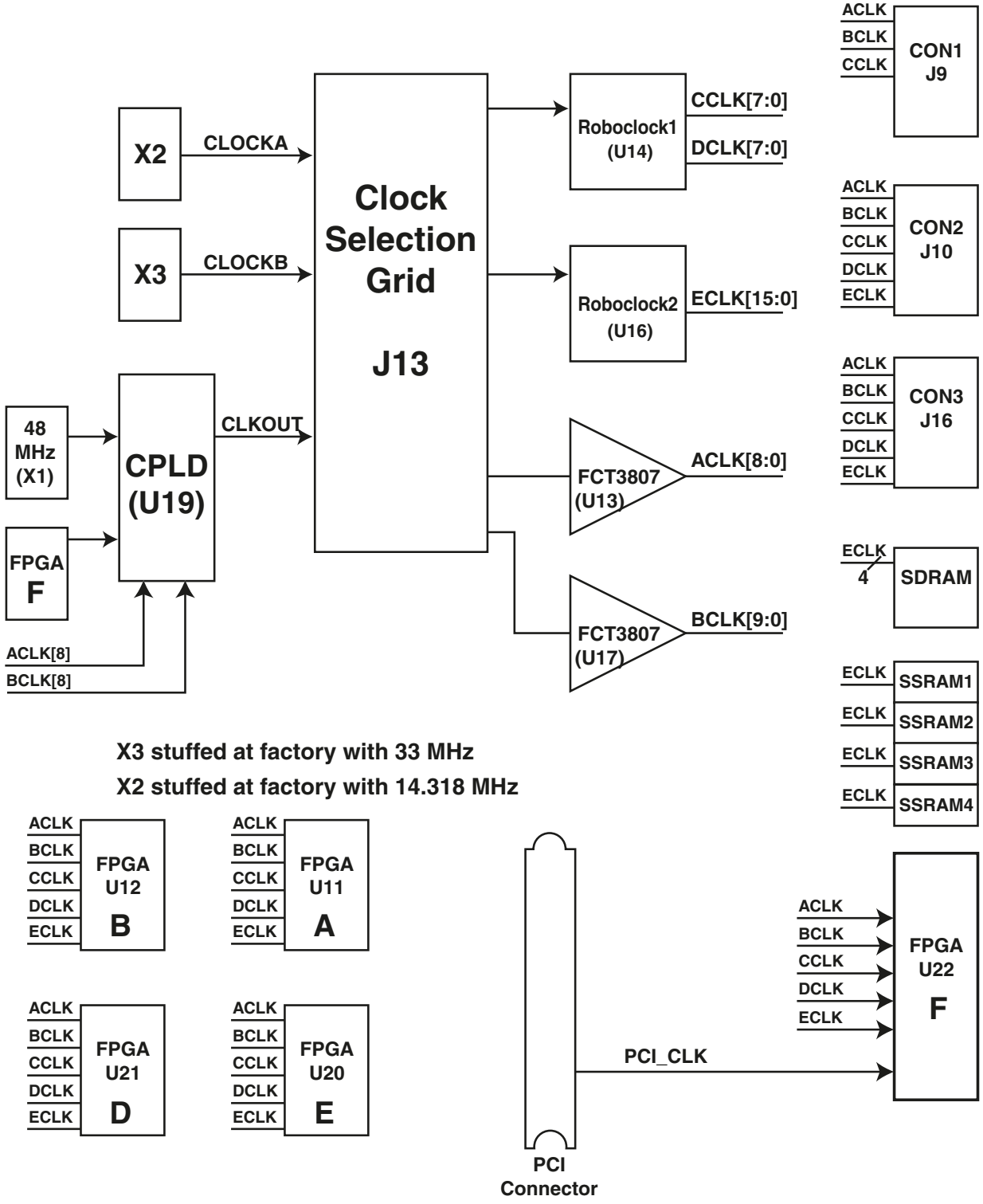


Figure 4-1 Clock Distribution Block Diagram

frequency. Phase adjustments can be made in 625 ps or 1300 ps steps up to ±10.4 ns. All adjustments are jumper selectable.

Clock Grid

Orientation and Description

The clock grid, **J13**, gives the user the ability to customize the clock scheme on the DN5000k10. A brief description of each pin is given in Table 4-1. The physical orientation of the pins is diagrammed in Figure 4-2.

.

Table 4-1 Clock Grid Signal Descriptions

Signal	Description
CLKOUT	**Clock signal from CPLD. Typically 12 MHz.**
PLL1A	**Input to RoboclockII #1**
CLOCKA	**Clock signal of oscillator #1 (X1)**
BUFINB	**Clock input to 3807 #2**
CLOCKB	**Clock signal of oscillator #2 (X2)**
PLL2B_PRE	**Secondary clock input to RoboclockII II #2. Differential pair with PLL2BN_PRE.**
PLL2BN_PRE	**Secondary clock input to RoboclockII II #2. Differential pair with PLL2B_PRE.**
BUFINA	**Clock input to 3807 #1**
PLL1BN_PRE	**Secondary clock input to RoboclockII #1. Differential pair with PLL1B_PRE.**
PLL1B_PRE	**Secondary clock input to RoboclockII #1. Differential pair with PLL1BN_PRE.**
GND	**Ground signals to provide signal integrity for ribbon cables.**

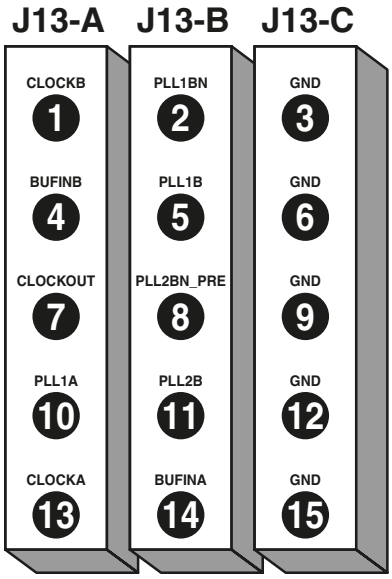


Figure 4-2 Clock Grid

Jumper Control for the Most Common Applications

Three main configurations are the most common.

First, the grid may be jumpered as follows:

Configuration #1: CLKOUT <–>PLL1A, CLOCKA <–>BUFINA and CLOCKB <–> BUFINB

Both 3807s receive their inputs from the oscillators. RoboclockII #1 receives a clock input from the CPLD. Also, RoboclockII #2 can use DCLK[7] from RoboclockII #1 as an input. This is explained in "Roboclock PLL Clock Buffers" on page 4-7. The common clock configurations are diagrammed in Figure 4-3.

Second, the input clock distribution can be configured as:

Configuration #2: CLKOUT <–>PLL2BN_PRE, CLOCKA <–>PLL1A and CLOCKB <–> BUFINB

In this configuration, a 3807 #2 receives an oscillator input. RoboclockII #1 receives an oscillator input while RoboclockII #2 receives the CPLD output clock signal. 3807 #1 is unused.

Finally, the grid may be configured as:

Configuration #3: CLKOUT <–>BUFINB, CLOCKA <–>BUFINA, and CLOCKB <–> PLL1BN_PRE

The 3807 #1 receives an oscillator input, and the 3807 #2 receives a CPLD input. Meanwhile, RoboclockII #1 receives the other oscillator input. The user can wire-wrap a clock to the unused driver(s) as needed. This enables full use of the timing devices on the DN5000k10.

Also, the destination of the output clocks might dictate some other configuration. This manual and other documentation should provide more than enough information to satisfy the user's needs (See Figure 4-4).

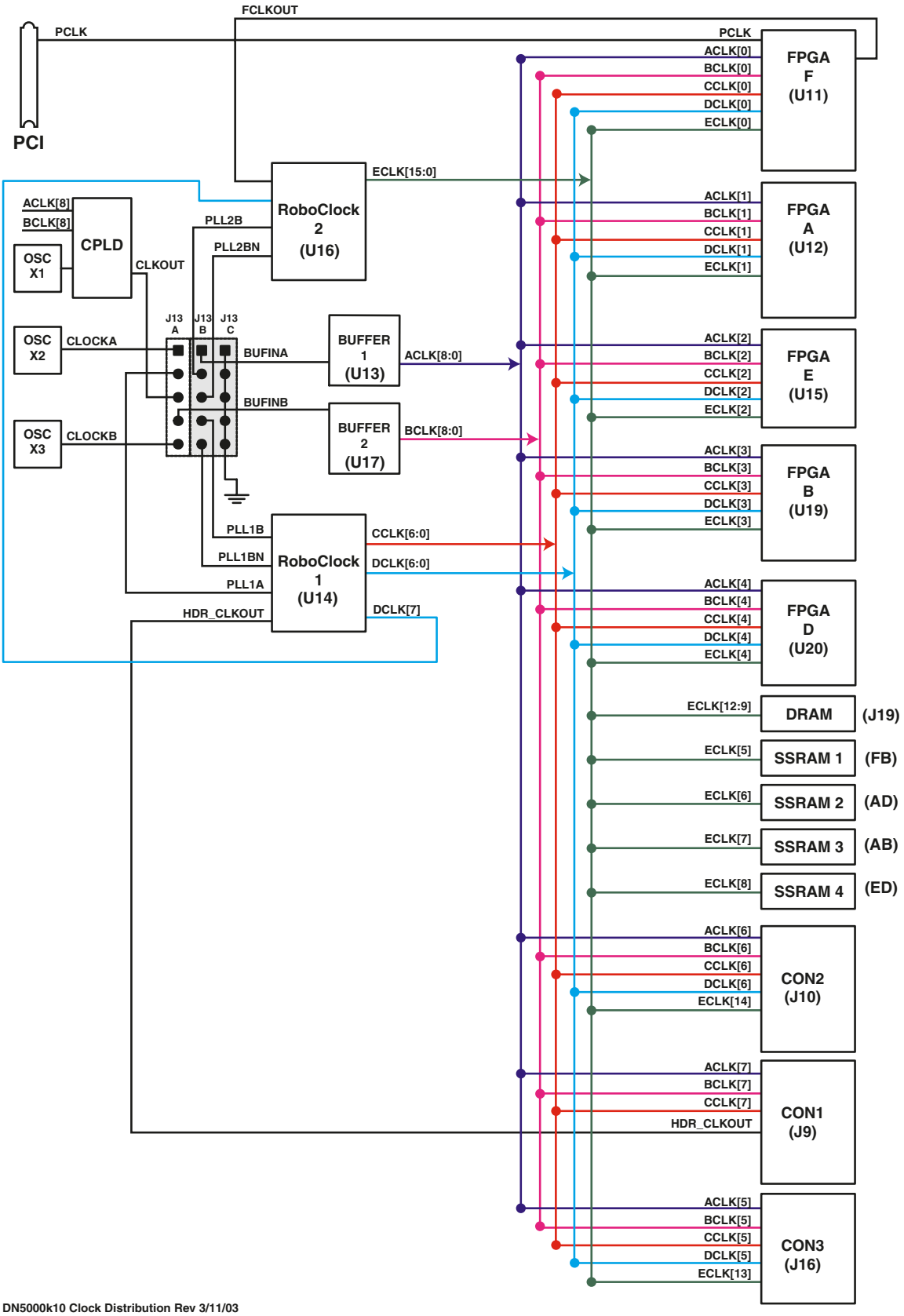


Figure 4-3 Common Clock Configurations

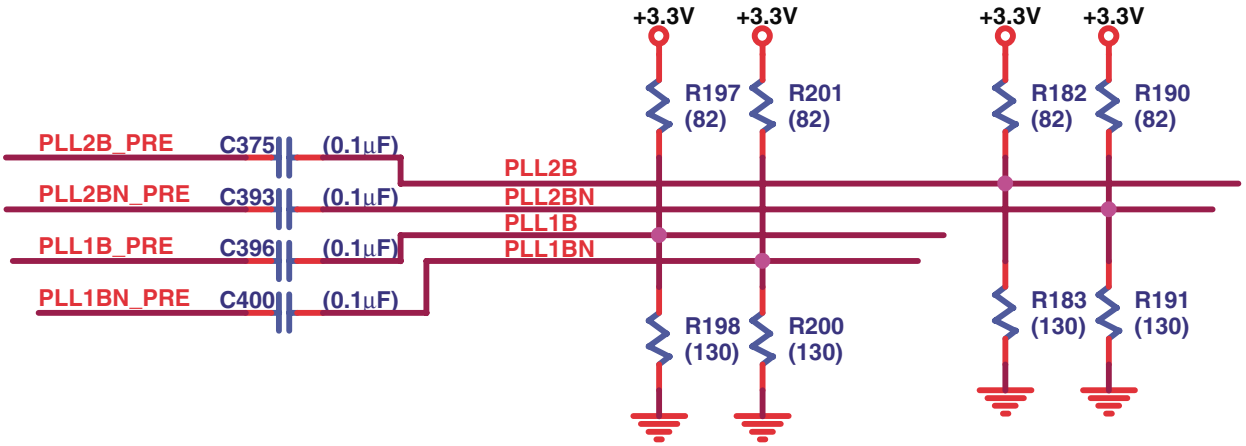


Figure 4-4 PECL Clock Input and Termination

NOTE: C375, C393, C396 and C400 are stuffed with 0-ohm resistors!

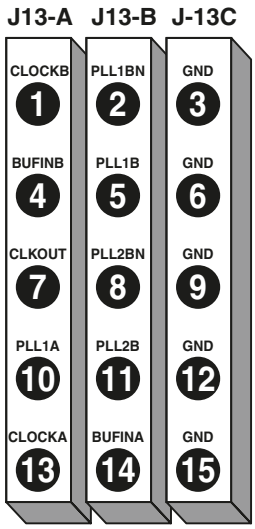
Note that the schematic shows capacitors in positions **C375**, **C393**, **C396** and **C400**. The DN5000k10 has 0-ohm resistors in these capacitor positions. The termination resistors **R182-R183, R190–R191, R197–198** and **R200–R201** are not stuffed.

Ribbon Cable: Providing an Off-Board Clock to the DN5000k10

The DN5000k10 gives the user a simple means to bring off-board clocks onto the board. The user can attach 10-pin ribbon cable to rows B and C of the Clock Grid. **J13-B** consists of an input to 3807 #1 and differential pair inputs to both RoboclockII's. **J13-C** consists of ground pins for signal integrity. These signals are described in Table 4-1 on page 4-3. `BUFINA` is a standard 3.3 V TTL input.

Both differential pairs provide some flexibility. The user can provide a single 3.3 V TTL input. It can be attached to either input. However, the other input must be left open. The user can provide a differential clock input to the pair. The differential clock inputs must obey the electrical specifications listed in Table 4-8 on page 4-13.

While attaching a ribbon cable, the user can jumper oscillator signal CLOCKB to BUFINB (3807 #2) on **J24**. This results in full use of all of the timing devices on the DN5000k10 (See Figure 4-5).



WITHOUT 10-PIN RIBBON CABLE				
Option	**CLKOUT (J13.7) goes to:**	**CLOCKA (J13.13) goes to:**	**CLOCKB (J13.1) goes to:**	**Notes**
1	PLL1A (J13.10)	BUFINA (J13.14)	BUFINB (J13.4)	Roboclock 2 may be driven by Roboclock 1
2	BUFINB (J13.4)	BUFINA (J13.14)	PLL1BN (J13.2)	Roboclock 2 may be driven by Roboclock 1
3	PLL2BN (J13.8)	BUFINA (J13.14)	PLL1BN (J13.2)	Buffer B is undriven
		PLL1A (J13.10)	BUFINB (J13.4)	
4	BUFINB (J13.4)	PLL2B (J13.11) (requires wire wrap)	PLL1BN (J13.2)	Buffer A is undriven

BUFINA → Buffer A → ACLK
BUFFINB → Buffer B → BCLK
PLL1A → Roboclock1 → CCLK, DCLK (PLLSEL1 low)
PLL1B → Roboclock1 → CCLK, DCLK (PLLSEL1 high)
PLL2B → Roboclock2 → ECLK (PLLSEL2 high)
Roboclock1 → Roboclock2 with PLLSEL2 low, J27 installed, and FCKLOUT unused.

WITH 10-PIN RIBBON CABLE	
Option	**(Connected to J21 and J22)**
3 external clocks	PLL1B, PLL2B and BUFINA are driven from cable. BUFINB can be jumpered to CLKOUT or CLOCKB, or left undriven.
2 external clocks	PLL2B and BUFINA are driven from cable, with PLL1A jumpered to CLKOUT or CLOCKA. Same options as above for BUFINB.
PECL clocks	The board can be set up for PECL inputs in PLL1B and PLL1BN, and in PLL2B and PLL2BN. PECL-ready boards cannot function without the cable except as in options 3 and 4 above.

Figure 4-5 External Ribbon Cable Connections

Roboclock PLL Clock Buffers

Figure 4-6 is a functional diagram of Roboclock 1 and Roboclock 2.

Jumper Descriptions

Headers **J11, J12, J15** and **J18** are used to control the PLLs. Each header consists of GND pins in row A, various PLL inputs in row B, and +3.3 V pins in row C. The layout of the headers is shown in Figure 4-7.

The Header Classifications are shown in Table 4-2.

Table 4-2 Header Classification

Controls	Header
Group I: General Control	**J18**

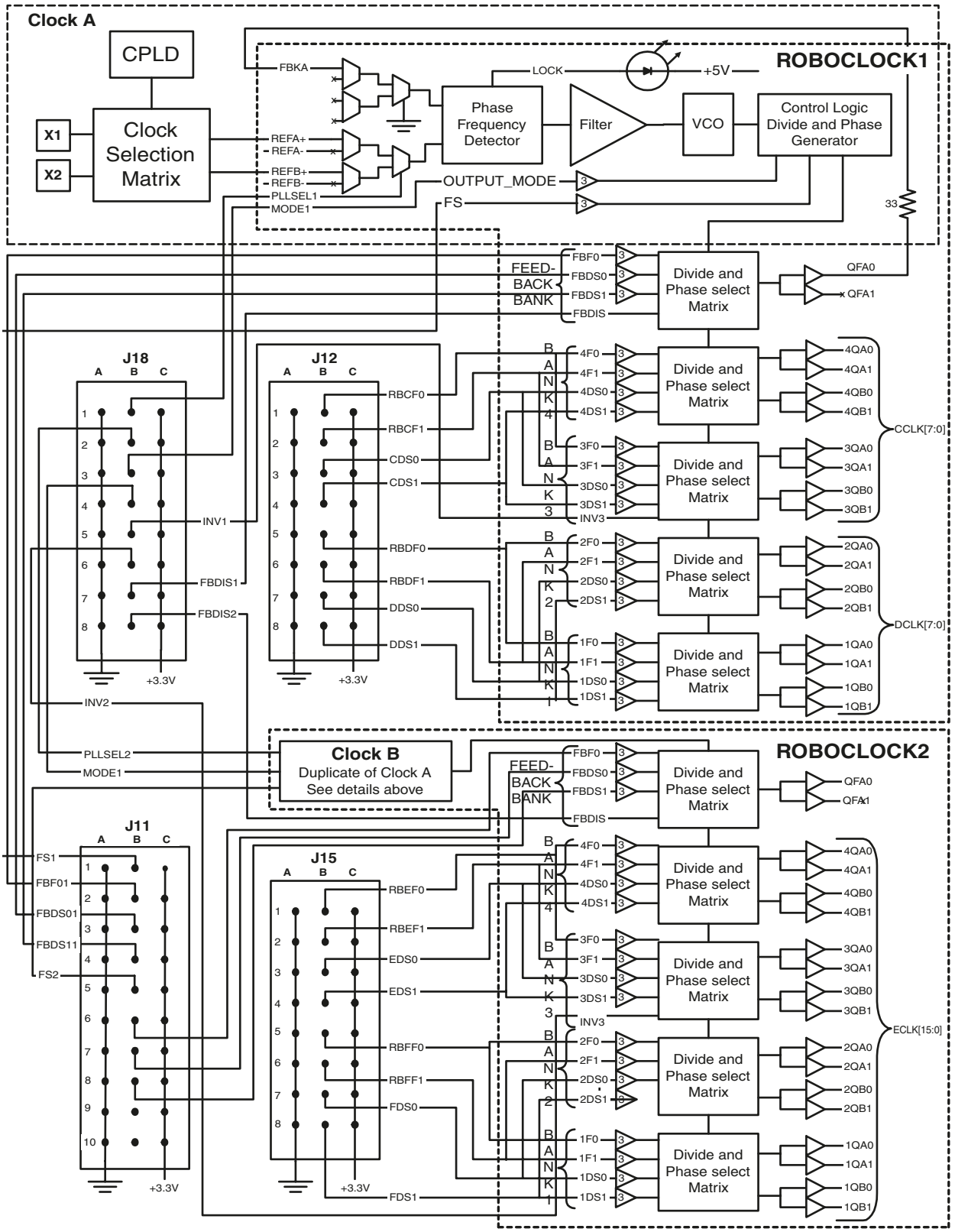


Figure 4-6 Functional Diagram of Roboclock 1 and Roboclock 2

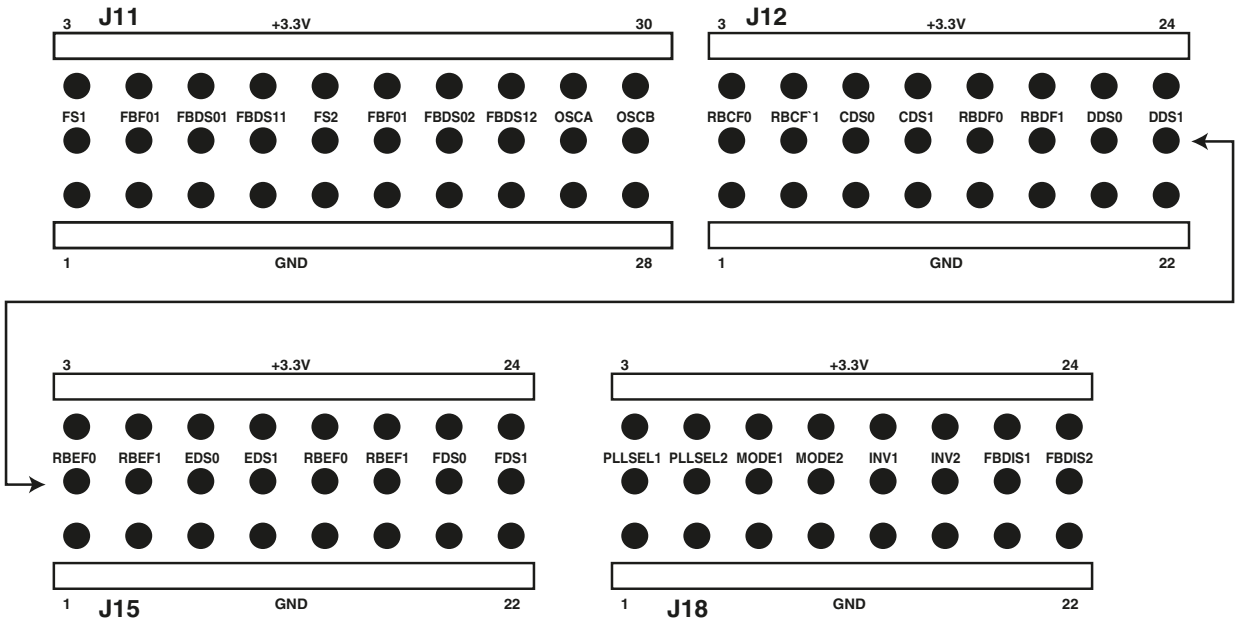


Figure 4-7 Header Layout

Table 4-2 Header Classification

Controls	Header
Group 2: **PLL1 Divider Control**	**J12**
Group 3: **PLL2 Divider Control**	**J15**
Group 4: **Feedback and f_{NOM} Control**	**J11**

The input pins are either LVTTL or 3-level input pins. The LVTTL pins need to be jumpered HIGH or LOW, which is achieved by connecting the input pin to the neighboring +3.3 V or GND pin, using a jumper. The 3-level input pins can be in a HIGH, MID, or LOW state. The HIGH and LOW states are achieved in the same way as the LVTTL pins. The MID state is reached by leaving the input pin unjumpered. The RoboclockII's have internal circuitry to bring the pin to 1.5 V when left open. The Jumper Definitions are shown in Table 4-3.

Table 4-3 Jumper Definitions

Name	Type	Default	Description
PLLSEL2[1]	LVTTL	LOW	Input Clock Select: If LOW, U16:DCLK[7] or FCLKOUT (U14:PLL1A or HDR_CLKOUT) is selected as the input clock. If HIGH, the U16:PLL2B_N (U14:PLL1B_N) pair is selected as the input clock.
MODE[2:1]	3-Level	HIGH	Output Mode: If HIGH, clock outputs disable to high-Z state. If LOW, clock outputs disable to "HOLD-OFF' mode. If MID, clock outputs disable to factory test mode.
INV2[1]	3-Level	MID	Invert Mode: When HIGH, clocks CCLK[3:0] (ECLK[3:0]) are inverted. When MID, these clock outputs are non-inverting. When LOW, the pairs CCLK[1:0] and CCLK[3:2] (ECLK[1:0] and ECLK[3:2]) will be complementary.
FBDIS[2:1]	LVTTL	LOW	Feedback Disable: When HIGH, feedback is disabled. When LOW, feedback is enabled.
RB[C-F]F[1:0]	3-Level	MID	Output Phase Function: Each pair controls the phase function of the respective group of outputs. See "Clock Skew" on page 4-12 for more information.
[C-F]DS[1:0]	3-Level	LOW	Output Divider Function: Each pair controls the divider function of the respective group of outputs. See "Clock Division" on page 4-11 for more information.
FS[2:1]	3-Level	LOW	Frequency Select: The input specifies the operating range of the nominal frequency (f_{NOM}). See "General Control" on page 4-11 for more information.
FBF0[2:1]	3-Level	LOW	Feedback Output Phase Function: The input controls the phase function of the feedback outputs. See "Feedback and Clock Multiplication" on page 4-11 for more information.
FBDS[1:0][2:1]	3-Level	MID	Feedback Output Divider Function: Each pair controls the divider function of the feedback outputs. See "Feedback and Clock Multiplication" on page 4-11 for more information.

General Control

FS[2:1] is a 3-Level input which determines the allowable range for the operating frequency f_{NOM} of the device. Depending on the chip grade, the PLL can operate between 12–100 MHz or 24–200 MHz. The actual f_{NOM} frequency can be determined by setting all jumpers to their defaults. Thus, f_{NOM} will be seen on all of the "divide-by-one" clock outputs. The user can set FS accordingly. The Frequency Range Settings are shown in Table 4-4.

Table 4-4 Frequency Range Settings

FS[2:1]	CY7B993V		CY7B994V	
	f_{NOM} (MHz)		f_{NOM} (MHz)	
	MIN	MAX	MIN	MAX
LOW	12	26	24	52
MID	24	52	48	100
HIGH	48	100	96	200

Feedback and Clock Multiplication

First of all, FBDIS[2:1] must be set LOW, enabling feedback. The feedback output is looped back to the feedback input. When a divided output is applied to the feedback input, the VCO (voltage controlled oscillator) of the PLL aligns the feedback input with the original input clock. Thus, with a 10 MHz input clock and the feedback outputs set to divide by 2, f_{NOM} must be 20 MHz. Consequently, 10 MHz is seen on the feedback output clocks and can be aligned with the input clocks. The feedback clock divider function actually serves as a clock multiplication mechanism for the operating frequency f_{NOM}. The divider function and the clock skew function are set in the same manner for the feedback and the normal clock outputs. See "Clock Division" on page 4-11 and "Clock Skew" on page 4-12, respectively.

Clock Division

The three pairs of DS inputs per chip are used to control the two groups of clock outputs and the feedback outputs of each PLL. The user can simply follow the Divider Function Table to acquire the desired output frequency. There are two things to remember. First, FS[2:1] must be set properly according to fNOM. Second, the FBDS feedback inputs act as operating clock frequency multipliers. The Output Divider Settings are shown in Table 4-5.

Table 4-5 Output Divider Settings

Input Signals		Output Divider Function	
[C-fF]DS1 and FBDS1[2:1]	[C-F]DS0 and FBDS0[2:1]	Output Signals	Feedback Output Signals
LOW	LOW	/1	/1
LOW	MID	/2	/2
LOW	HIGH	/3	/3
MID	LOW	/4	/4
MID	MID	/5	/5
MID	HIGH	/6	/6
HIGH	LOW	/8	/8
HIGH	MID	/10	/10
HIGH	HIGH	/12	/12

Clock Skew

Clock skew is controlled by the "F" inputs. The clock skew may be any integer value from 0 to ±8 times the RoboclockII time unit t_U. The time unit value is derived from the operating frequency f_{NOM} and the `FS[2:1]` setting. The following equation yields the time unit t_U.

$$t_U = \frac{1}{f_{NOM} \bullet N}$$

The possible values for N are given in Table 4-6. The available skew for each RoboclockII derived clock is given in Table 4-7. Based on the following information, the user will be able to adjust the skew for any of the RoboclockII outputs.

Table 4-6 Time Unit N-factor

FS	CY7B993V		CY7B994V	
	N	f_{NOM} (MHz) at which t_U = 1 ns	N	f_{NOM} (MHz) at which t_U = 1 ns
LOW	64	15.625	32	31.25
MID	32	31.25	16	62.5
HIGH	16	62.5	8	125

Table 4-7 Clock Skew Settings

Input Signals		Output Skew Function				
RB[C-F]F1	RB[C-F]F0 and FBF0[2:1]	DCLK[3:0] or ECLK[11:8]	DCLK[7:4] or ECLK[15:12]	CCLK[3:0] or ECLK[3:0]	CCLK[7:4] or ECLK[7:4]	Feedback Output Signals
LOW	LOW	$-4t_U$	$-4t_U$	$-8t_U$	$-8t_U$	$-4t_U$
LOW	MID	$-3t_U$	$-3t_U$	$-7t_U$	$-7t_U$	N/A
LOW	HIGH	$-2t_U$	$-2t_U$	$-6t_U$	$-6t_U$	N/A
MID	LOW	$-1t_U$	$-1t_U$	COL1*	COL1*	N/A
MID	MID	$0t_U$	$0t_U$	$0t_U$	$0t_U$	$0t_U$
MID	HIGH	$+1t_U$	$+1t_U$	COL2**	COL2**	N/A
HIGH	LOW	$+2t_U$	$+2t_U$	$+6t_U$	$+6t_U$	N/A
HIGH	MID	$+3t_U$	$+3t_U$	$+7t_U$	$+7t_U$	N/A
HIGH	HIGH	$+4t_U$	$+4t_U$	$+8t_U$	$+8t_U$	$+4t_U$

***The clock skew is equivalent to the skew on** DCLK[3:0] **or** ECLK[11:8]
****The clock skew is equivalent to the skew on** DCLK[7:4] **or** ECLK[15:12]

Differential Clocks

In addition to LVTTL clock signals, the RoboclockII clock buffers can handle LV Differential (LVPECL) clocks. The user can cable in an acceptable differential signal to PLL1B and PLL1BN, or PLL2B and PLL2BN through the clock grid **J13**. The signals must obey the specifications given in Table 4-8. Onboard circuitry is available to center the signals about the proper voltage, if needed.

Table 4-8 LVPECL Input Specifications

Description	Min	Max
Differential Voltage	0.4	3.3
Highest HIGH Voltage	1.0	3.3
Lowest LOW Voltage	GND	2.9
Common Mode range (crossing voltage)	0.8	3.3

The clock input of the RoboclockII can accept a superset of PECL. PECL involves a 1 V swing about $V_{CC}/2$. The RoboclockII clock input can accept a swing of up to 3.3 V about $V_{CC}/2$, which gives the user another dimension of flexibility.

The CY7B993V/4V can output LVTTL complementary (differential) signals, too. Setting INV1 (INV2) LOW will result in clocks CCLK[1:0] and CCLK[3:2] (ECLK[1:0] and ECLK[3:2]) becoming complementary pairs. A network of series and parallel resistors could be used to reduce the nominal swing of the clock signals.

Useful Notes and Hints

The CYB993V consistently outputs ~32.5 MHz signals in cases of improper settings or unacceptable clock inputs. This was observed when:

- The CY7B993V part was operating at a nominal frequency f_{NOM} of 36.4 MHz with FS set LOW.
- Identical clocks were sent to PLL2B and PLL2BN.

For the CY7B994V part, the operating frequency can reach up to 200 MHz. However, the maximum output frequency is 185 MHz. This means when 185 MHz ≤ f_{NOM} ≤ 200 MHz, the output divider must be set to at least 2. Otherwise, the RoboclockII's will output garbage.

Customizing the Oscillators

The user can customize the frequency of the clock networks by stuffing oscillators in **X2** and **X3**. The DN5000k10 is shipped with a 14.318 MHz oscillator in location **X2** and a 100 MHz oscillator in **X2**. The RoboclockII's are **not** +5 V tolerant, so +3.3 V oscillators are necessary.

NOTE: If you stuff your own oscillators, +3.3 V CMOS outputs are necessary since the RoboclockII's are not +5 V signalling tolerant!

We get our oscillators from Digi-Key (**http://www.digikey.com/**). Of note is an Epson line of oscillators called the **SG-8002 Programmable Oscillators**. Any frequency between 1.00 MHz–106.25 MHz can be procured in the normal Digi-Key shipping time of 24 hours. A half-can, +3.3 V CMOS version is needed with a tolerance of 50 ppm. The part number for an acceptable oscillator from this family would be:

SG-8002DC-PCB-ND
- package SG-531
- output enable
- 3.3 V CMOS
- 50 ppm

If the order is placed via the web page, the requested frequency to two decimal places is placed in the Web Order Notes. The datasheet is on the CD-ROM for this oscillator. The file name is SG8002DC.pdf.

Any polarity of output enable for each oscillator (on pin 1) is acceptable. Make sure that you have the proper jumper settings at positions 9 and10 of **J14**, **J15** and **J16**. See Figure 4-8 and Figure 4-9 for a description.

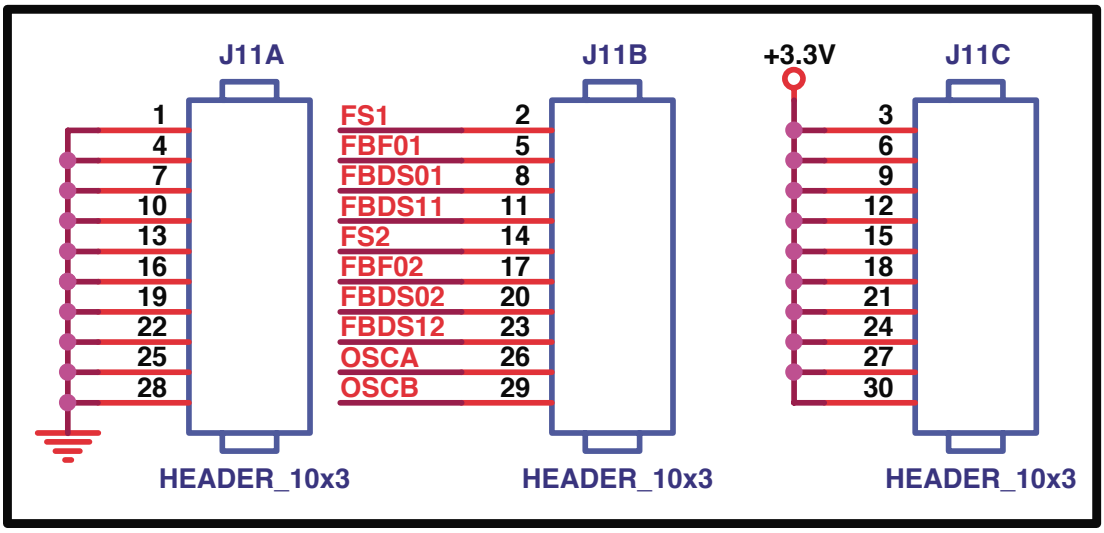


Figure 4-8 Clock OE Pin Jumper Settings

Table 4-9 Clock OE Pin Jumper Settings

Clock OE	Jumper Settings
Active High OE for X1	Jumper J11.26 to J11.27
Active Low OE for X1	Jumper J11.26 to J11.25
Active High OE for X2	Jumper J11.29 to J11.30
Active Low OE for X2	Jumper J11.29 to J11.28

DN5000k10 PCI_CLK Operation

The DN5000k10 ASIC emulation board has the ability to run all FPGAs, all SRAMs, and all the SDRAM off of PCI_CLK. PCI_CLK is a single destination clock which is routed to FPGA F (**U11**) from the PCI connector. The user can input PCI_CLK to one of the Stratix Enhanced PLLs in FPGA F. The resulting PLL output can be sent to the Roboclocks via the signal FCLKOUT. All of the aforementioned devices receive one of the ECLK clock outputs from Roboclock 2. Consequentially, the DN5000k10 can be clocked exclusively by PCI_CLK

PCI_CLK Details

PCI_CLK is connected to one of FPGA F's global clock inputs from the PCI connector. Then, a PLL must be instantiated. The PLL will require a minimum of three connections. For further information on Stratix™ PLL operation, see the Altera website at **http://www.altera.com**. The Stratix™ Datasheet (ds_stx.pdf), which can be found on the DN5000k10 CD-ROM, also provide useful information on PLLs. PCI_CLK must be connected to the "INCLK[0]" input of the DCM. The DCM output, "EXTCLK[0]" will be connected to the output signal FCLKOUT. One more connection must be made to the PLL. This will come up later in this section.

The FCLKOUT signal is connected to one of the four input pins on Roboclock 2. FCLKOUT's complementary input is DCLK[7](R). FCLKOUT and DCLK[7](R) are both single ended TTL inputs. When either of them is being used, the other one must be left open. For complete PCI_CLK operation, jumper **J14** must be unstuffed leaving DCLK[7](R) open.

If set to the default configuration, Roboclock 2 will drive a one-to-one PCI_CLK derived clock on its outputs. See "Roboclock PLL Clock Buffers" on page 4-7 for more information. Roboclock 2 has 16 clock outputs (ECLK[15:0]). Each FPGA (ECLK[4:0]), each SSRAM (ECLK[9:6]), and the SDRAM (ECLK[13:10]) receive ECLK signals.

To complete this setup, a feedback signal must be connected to the PLL in FPGA F. Roboclock 2 sends ECLK[15] to the feedback input of FPGA F. ECLK[15] needs to be connected to the "fbin" input signal of the PLL. Using ECLK[15] as feedback allows the PLL to properly synchronize the DN5000k10 PCI_CLK network which completes the setup (see Figure 4-9 for a diagram of the PCI_CLK PLL circuit).

The DN5000k10 can be run off any single-ended TTL clock signal which is sent to the Roboclocks. The ECLK distribution provides the DN5000k10 this flexibility. PCI_CLK has special implications for DN300k10 PCI operation.

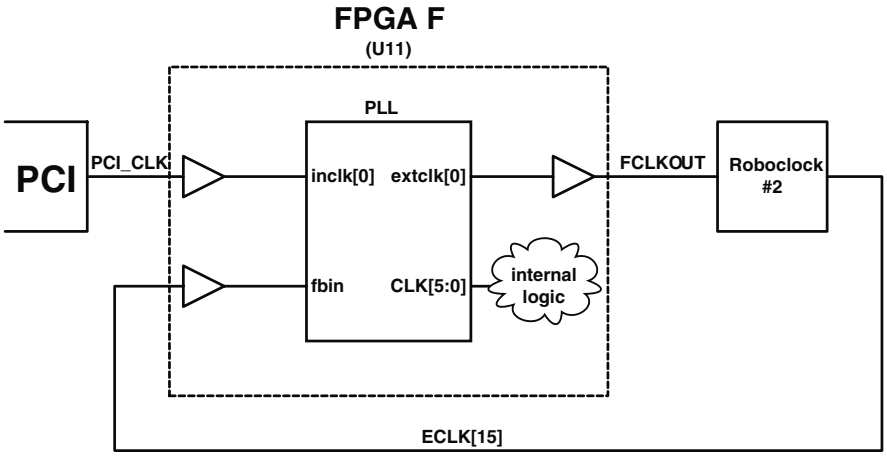


Figure 4-9 PCI_CLK PLL Circuit

BCLKOUT and FCLKOUT

FCLKOUT is assigned to a dedicated clock output pin in the Stratix architecture, and can be used to drive Roboclock 2 (see the section "DN5000k10 PCI_CLK Operation" on page 4-16 for details). The DN5000K10 differs from previous Dini Group emulator boards because the Stratix architecture assigns each PLL to specific clock pins. In the case of FCLKOUT, the only possible source is from PLL5, which has only one possible input, PCI_CLK. So, the sole purpose of FCLKOUT is to provide the means to run the whole board with PCI_CLK.

Header Clocks

Each header receives a clock signal from each of the five clock groups, except that J9 does not receive DCLK or ECLK. The last available DCLK signal was needed to drive Roboclock 2, and the last ECLK signal was needed to drive the feedback loop for the PCI_CLK function. Instead, **J9** has a signal, HDR_CLKOUT, which connects to an input of Roboclock 1, providing another way to drive CCLK and DCLK from an off-board source. This can be used for systems where additional hardware on a daughtercard needs to communicate with the FPGAs on the DN5000K10. Note that HDR_CLKOUT and PLL1A are connected to a differential pair of inputs on Roboclock 1, so if HDR_CLKOUT is being used, pin 10 on **J13** must not be connected to anything so that there is no interfering signal.

DCLK[7](R)

The signal DCLK[7](R) is routed from one of the Roboclock 1 outputs to one of the Roboclock 2 inputs. Jumper **J14** lies along this connection route. **J14** must be installed in order to utilize DCLK[7](R). DCLK[7](R)and FCLKOUT are complementary input on Roboclock 2, and are both single-ended TTL inputs. When either of them is being used, the other one must be left open. Thus, FCLKOUT must be undriven on FPGA F for DCLK[7](R) to operate.

DCLK[7](R) provides two useful results. First, any clock signal or some derivation sent to Roboclock 1 can be driven onto RoboclockII #2 for full distribution.

Second, running a clock through Roboclock 1 to Roboclock 2 gives the user more divide and multiply options for the clock frequencies. Here is an

example. If you have a 40MHz input clock, the user cannot output a 30MHz clock with a single RoboclockII's multiply and divide options. However, the user can input a 40MHz to RoboclockII #1 and divide it by 4. By installing the **J27** jumper, a 10MHz clock will be driven onto RoboclockII #2. Setting RoboclockII #2's feedback outputs to divide by 3, the operating frequency will become 30MHz. Thus, a 30MHz could be driven onto the RoboclockII #2 output signals.

NOTE: The signal **FCLKOUT** must be left open in order to utilize **DCLK[7](R)**

Chapter 5

Memories

The DN5000k10 has five external memories: four 36-bit SSRAMs, and one 72-bit SDRAM DIMM. The four SSRAMS are referred to as SSRAM FB (**U22)**, SSRAM AD (**U23**), SSRAM AB (**U21**), and SSRAM ED (**U18**).

SSRAMs

The SSRAMs can be stuffed with ZBT, non-ZBT, pipeline, or flowthrough parts. We believe we have anticipated the additional address lines for the 1 M x 36 and 2 M x 36 parts when they are available. The DN5000k10 is stuffed at the factory with 512 K x 36-bit Synchronous Pipeline Burst SRAM. Samsung K7A163600M-QC1400 are probably the parts you will have stuffed into your DN5000k10. The datasheet is on the CD-ROM in the file DS_K7A1636(18)00M.pdf. The SSRAMs are tested at 100 MHz.

SSRAM Notes

All SSRAMs use ECLK for their clock. SSRAMs FB, AB, and AD share most of their I/Os with the 200 pin connectors. The exceptions that are not on the connectors are pins **ZZ**, **OE**, and **CE**. So, if an SSRAM is not needed, one of the FPGAs connected to it may drive **ZZ** high, putting the SSRAM to sleep so that it does not interfere with the 200-pin connector. Alternatively, the 200-pin connector is available to probe SSRAM signals for debugging purposes. However, if someone wanted to design a system where hardware on a daughtercard can access an SSRAM directly, some rework would be needed to connect it to the **OE** and **CE** pins. The connections between the FPGAs and the SSRAMS is shown in Figure 5-1.

The signal connections for SSRAM FB are shown in Figure 5-2.

The signal connections for SSRAM AD are shown in Figure 5-3.

The signal connections for SSRAM AB are shown in Figure 5-4.

The signal connections for SSRAM ED are shown in Figure 5-5.

Flowthrough SSRAMs are functionally the closest to ASIC-style memories. Pipeline SSRAMs can be clocked at faster frequencies. ZBT SSRAMs are typically one generation behind in density. The subtle differences between the styles of memories are described in the next section.

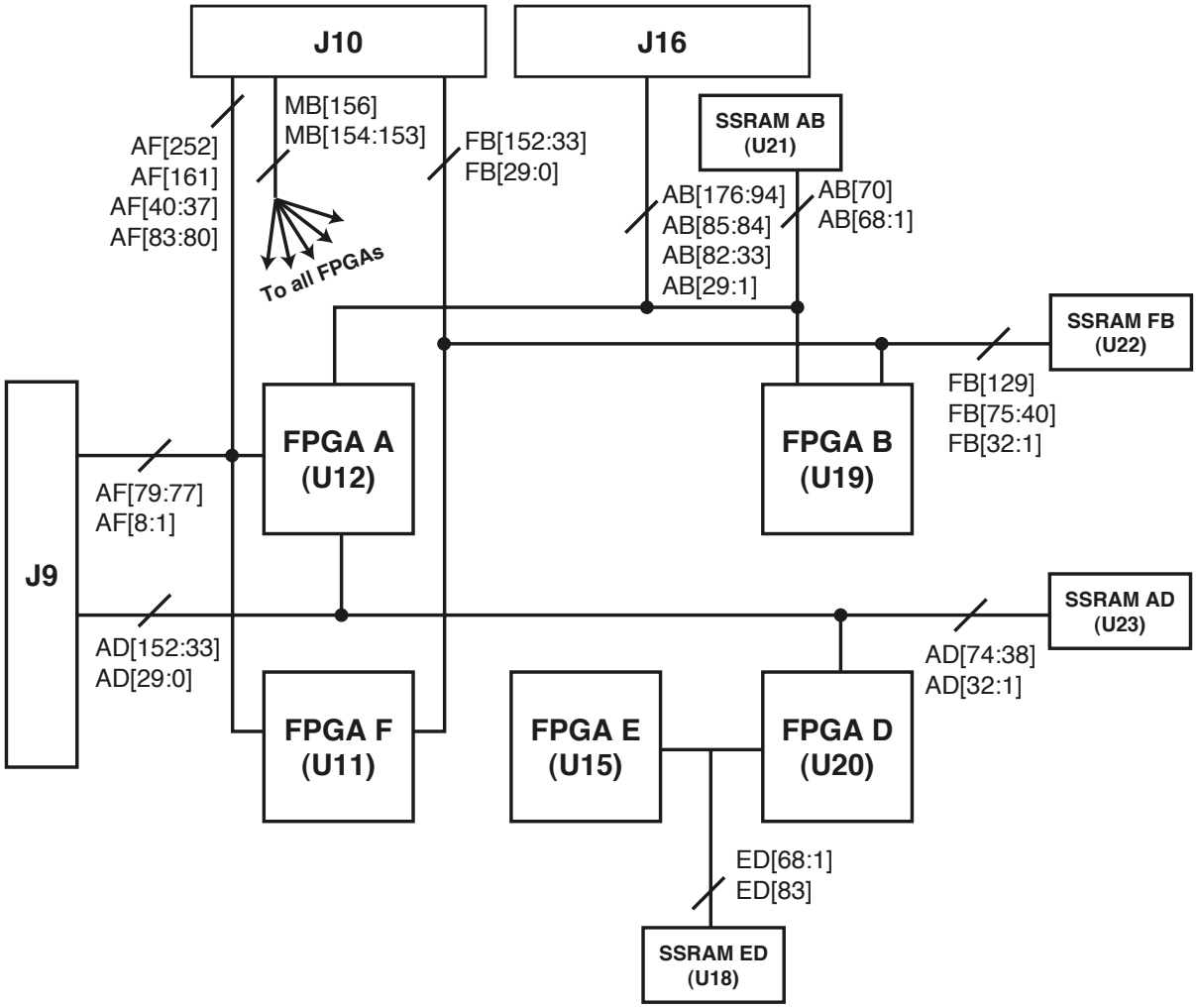


Figure 5-1 FPGA Interconnect Block Diagram

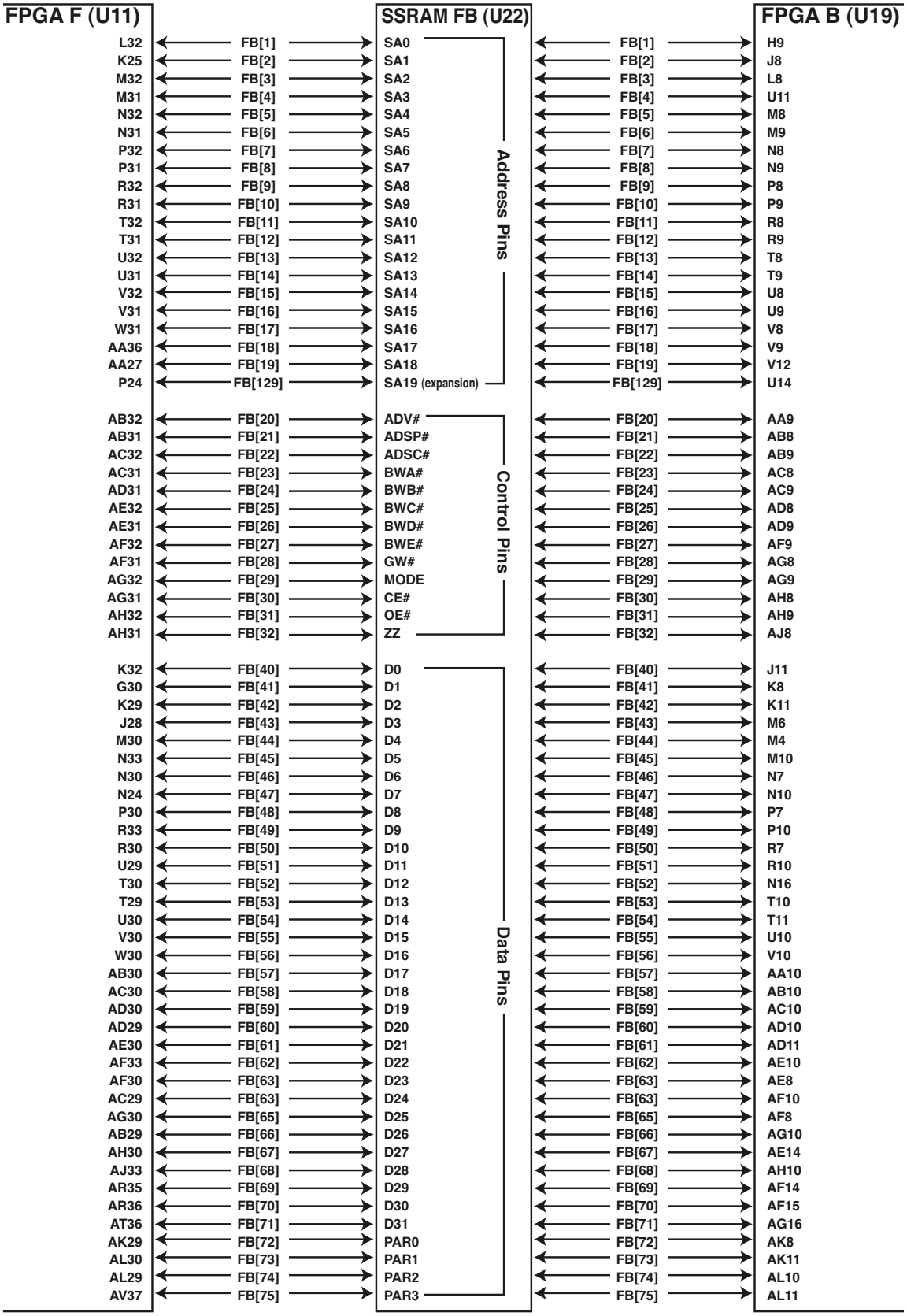


Figure 5-2 SSRAM FB (U22) Bus Signals

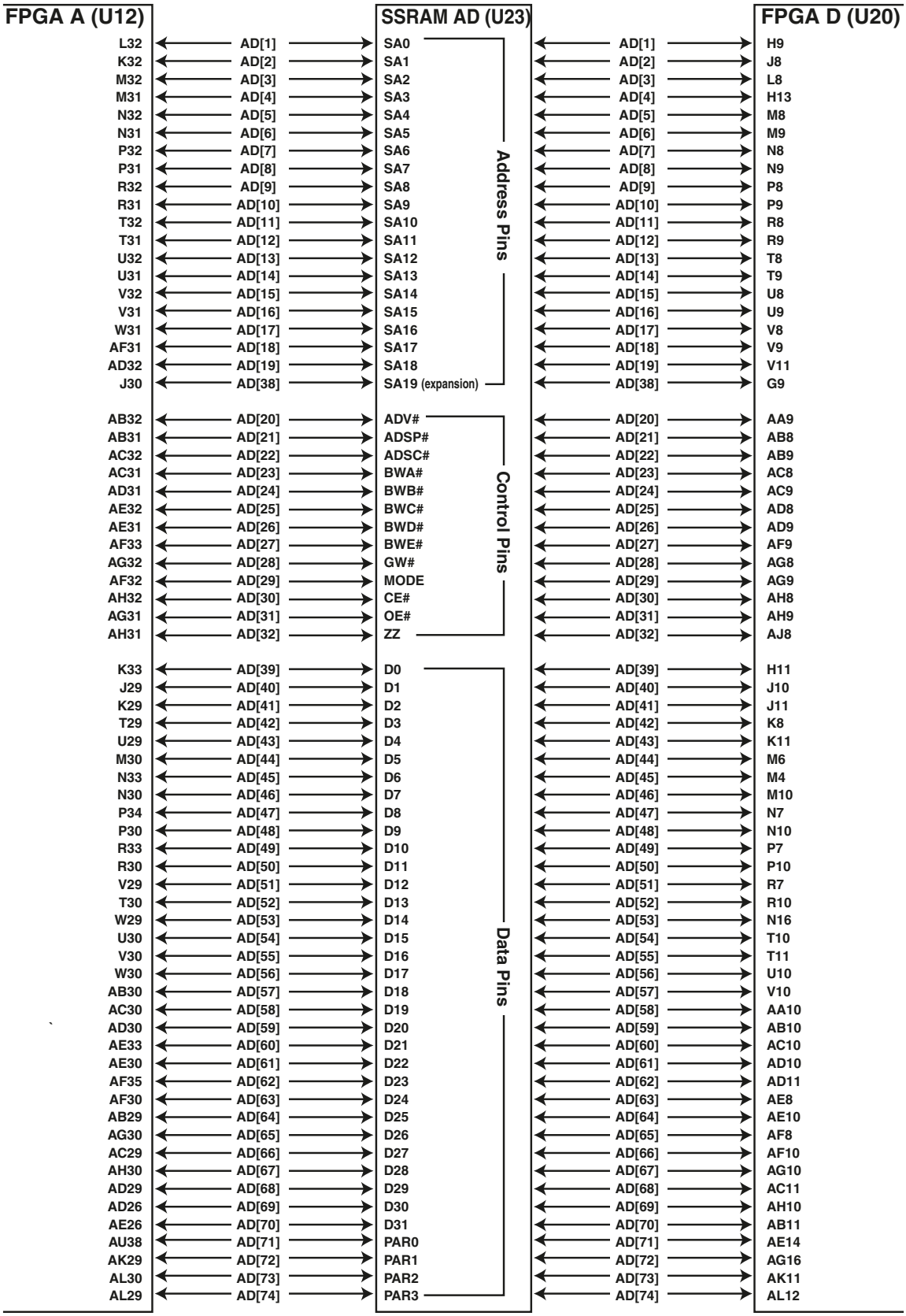


Figure 5-3 SSRAM AD (U23) Bus Signals

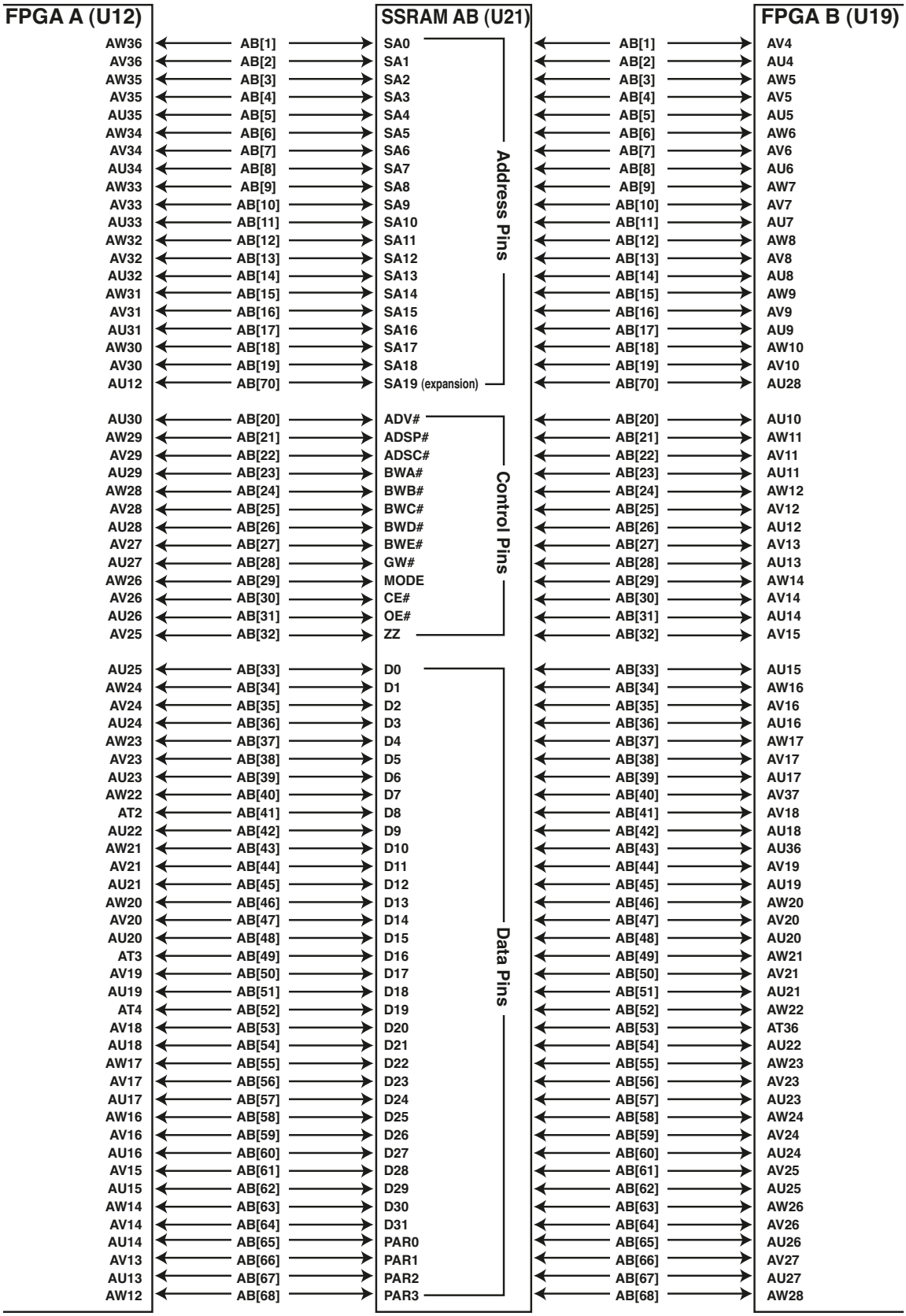


Figure 5-4 SSRAM AB (U21) Bus Signals

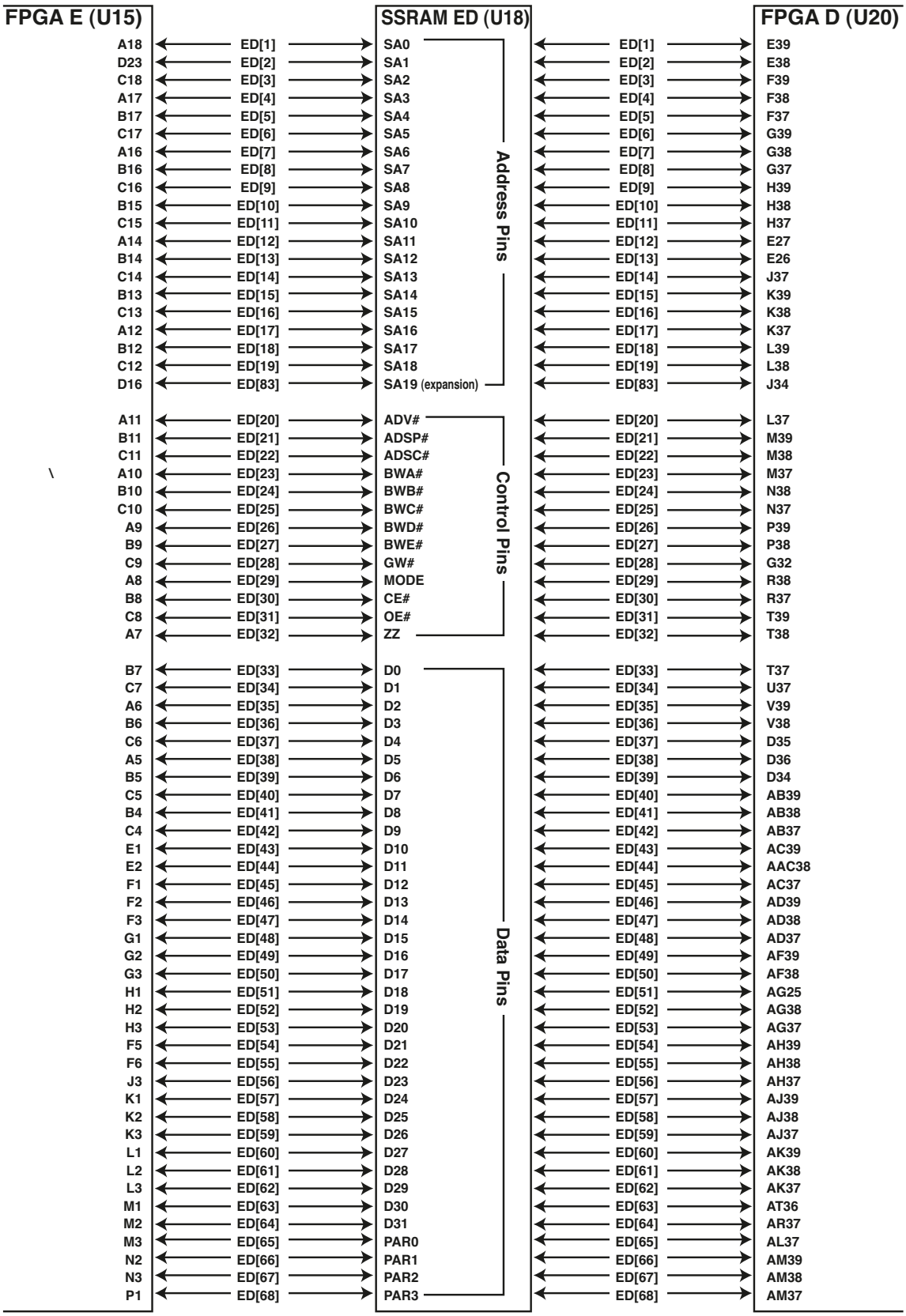


Figure 5-5 SSRAM ED (U18) Bus Signals

Pin 14 of each SSRAM may be pulled high, pulled low, or left unconnected. Table 5-1describes which 0-ohm resistors must be used for each type of SSRAM to function correctly.

Table 5-1 Requirements for Non-Standard SSRAMs

	FB	AD	AB	ED
ZBT Pipeline	**Install R290 R291 R296**	**Install R293 R294 R298**	**Install R286 R287 R288**	**Install R219 R220 R221**
ZBT Flowthrough	**Install R289 R291 R295**	**Install R292 R294 R297**	**Install R284 R285 R288**	**Install R219 R222 R223**
Syncburst Flowthrough or Pipeline	**No Extra Resistors**			

Pipeline, Flowthrough, ZBT

Syncburst FT (Flowthrough) (Figure 5-6) is the most straightforward type of SSRAM available for the DN5000k10. Write data may be accepted on the same clock cycle as the activation signal and address, and read data is returned one clock cycle after it is requested. Syncburst is designed to allow two controllers to access the same SSRAM, using two activation signals, ADSC# and ADSP#; an activation with ADSP# requires data and byte enables one clock cycle after the address and activation. Syncburst PL (Pipelined) (Figure 5-7) is identical except for registered outputs, which delay read data an additional clock cycle but may be necessary for high-speed designs.

Zero-Bus-Turnaround (ZBT) SSRAMs are designed to eliminate wait states between reads and writes by synchronizing data. Thus, ZBT FT SSRAMs

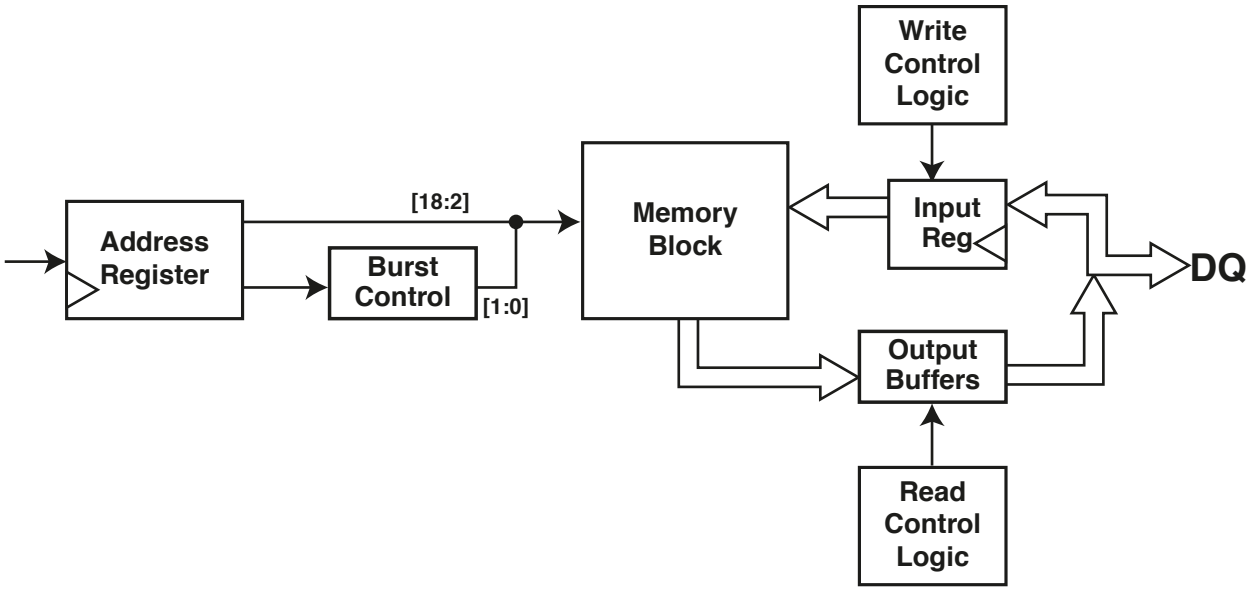


Figure 5-6 Syncburst FT

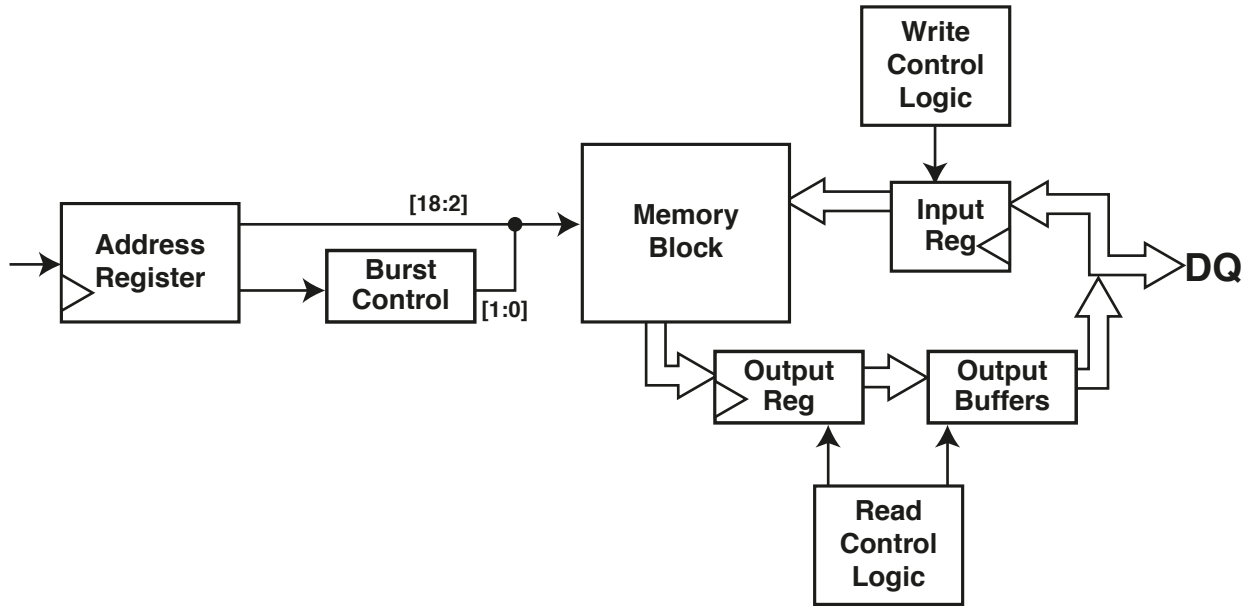


Figure 5-7 Syncburst PL

(Figure 5-8) accept and return data one clock cycle after the address phase, and ZBT PL SSRAMs (Figure 5-9) accept and return data two clock cycles after the address phase. This allows the user to begin a write burst immediately after the last word of a read burst, because read data will be returned before the first write data is required. The timing is illustrated in Figure 5-10 and Table 5-2.

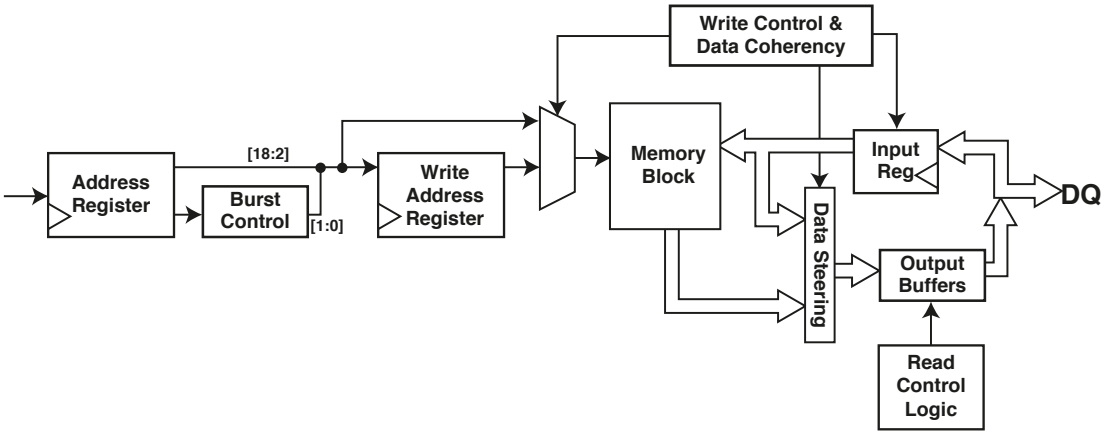


Figure 5-8 Syncburst ZBT FT

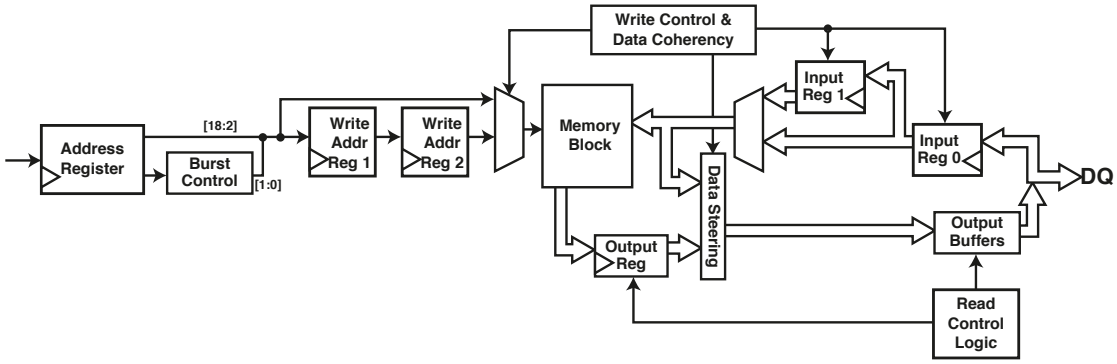


Figure 5-9 Syncburst ZBT PL

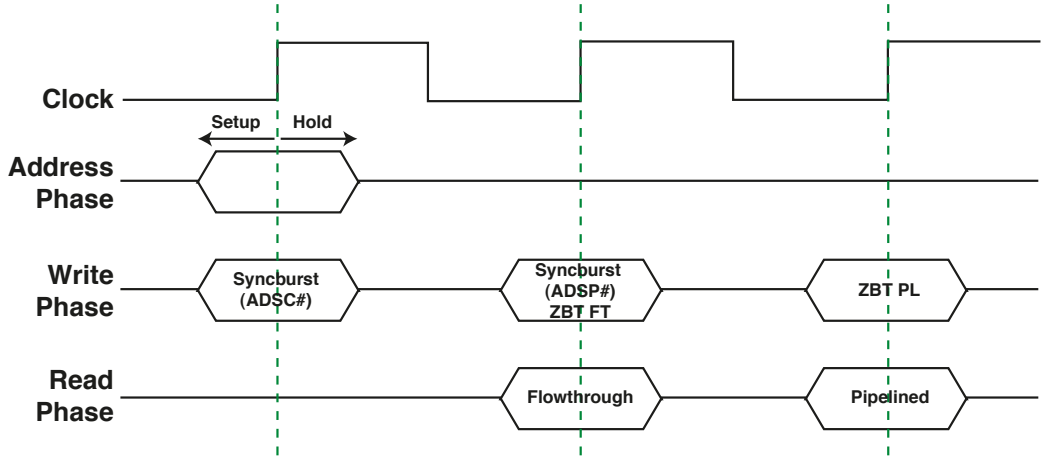


Figure 5-10 Syncburst and ZBT SSRAM Timing

Table 5-2 Syncburst and ZBT SSRAM Timing

	Syncburst	ZBT
Address Phase	CE#, CE2#, CE ADSC#, or ADSP# **address** **or** ADV#[1]	CE#, CE2#, CE R/W#, LD#[2], BWx# **address** **or** ADV[2], BWx#[3]
Write Phase	BWE#, BWx#, **or** GW#[4] **data**	**data**
Read Phase	**Valid Data**	**Valid Data**

[1]**To continue a burst.**
[2]ADV/LD# **is low to load a new address, high to continue bust.**
[3]**For write access only.**
[4]**Writes to all four bytes.**

SDRAM

The DN5000k10 has a socket for a +3.3 V 168-pin SDRAM DIMM. Either registered or unbuffered modules fit in the socket (**U7**). The same PC100/PC133 SDRAM modules that you put into your PC are used here. Your DN5000k10 will be stuffed and tested with a 1 Gbyte PC133 SDRAM DIMM, unless otherwise requested.

All DIMM pins are connected to the FPGA and the pins are shown in Figure 5-11 and Figure 5-12. We aren't quite sure what the largest size SDRAM DIMM is that will work in the DN5000k10, but here is the math as best we understand it:

14 Address lines A[13:0] (multiplexed between RAS* and CAS*, address 10 not used for CAS*)	27
2 bank address BA[1:0]	2
4 chip selects (S[3:0]*) used in pairs	1

So, we think that there are 29 address bits (27 + 2) and 2 possible chips selects, which add one more address bit. This totals 30 address bits — 1 G of 72-bit long words, which is 8 Gbytes. Please tell us if this math is wrong.

SDRAM modules require 4 clocks — CK[3:0]. These clocks are driven by the RoboclockII 2 and the signal names are ECLK.

The CD-ROM has a datasheet of an acceptable 1 Gbyte SDRAM module from Micron. The file name is SDF36C64_127x72G_B.pdf.

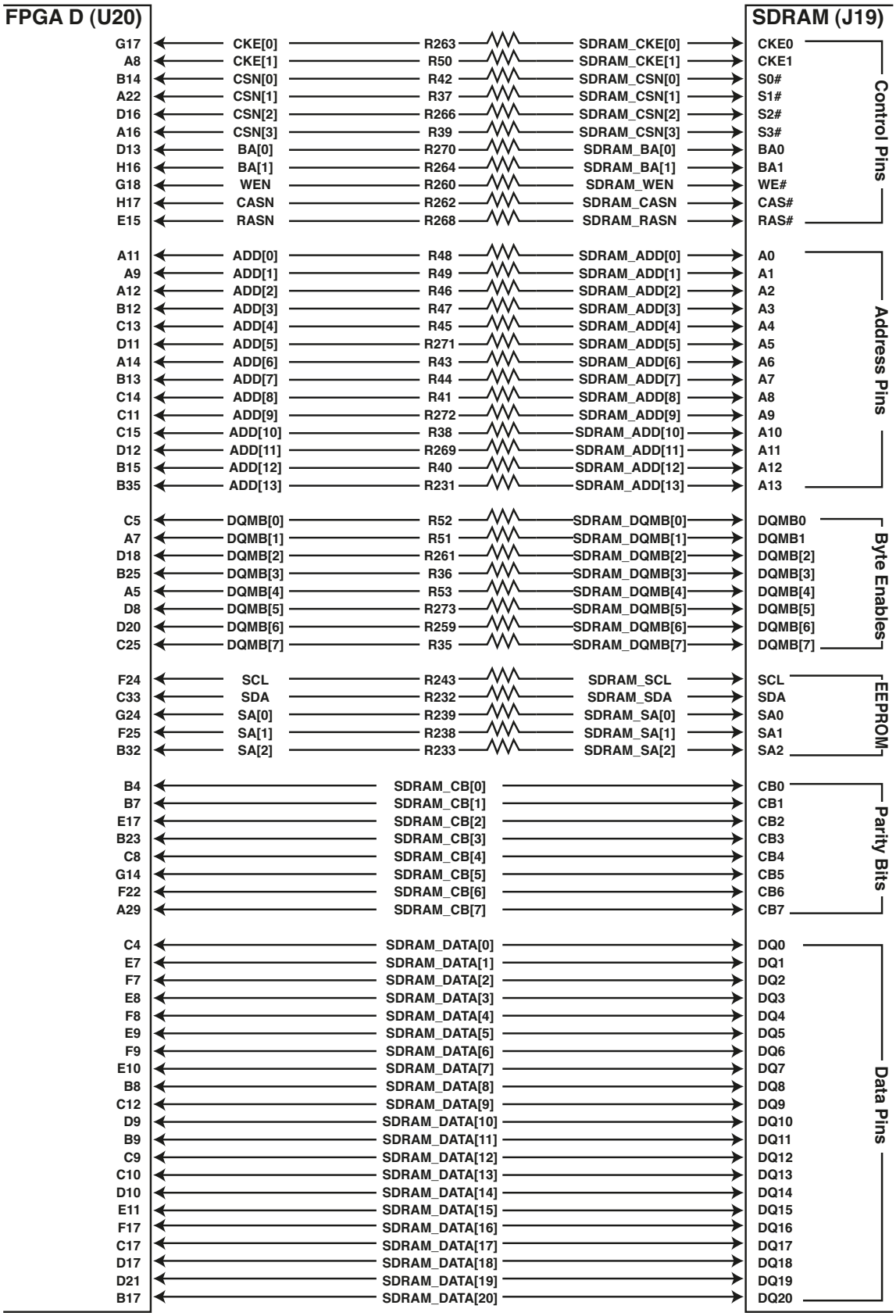


Figure 5-11 SDRAM (J19) Bus Signals (Page 1 of 2)

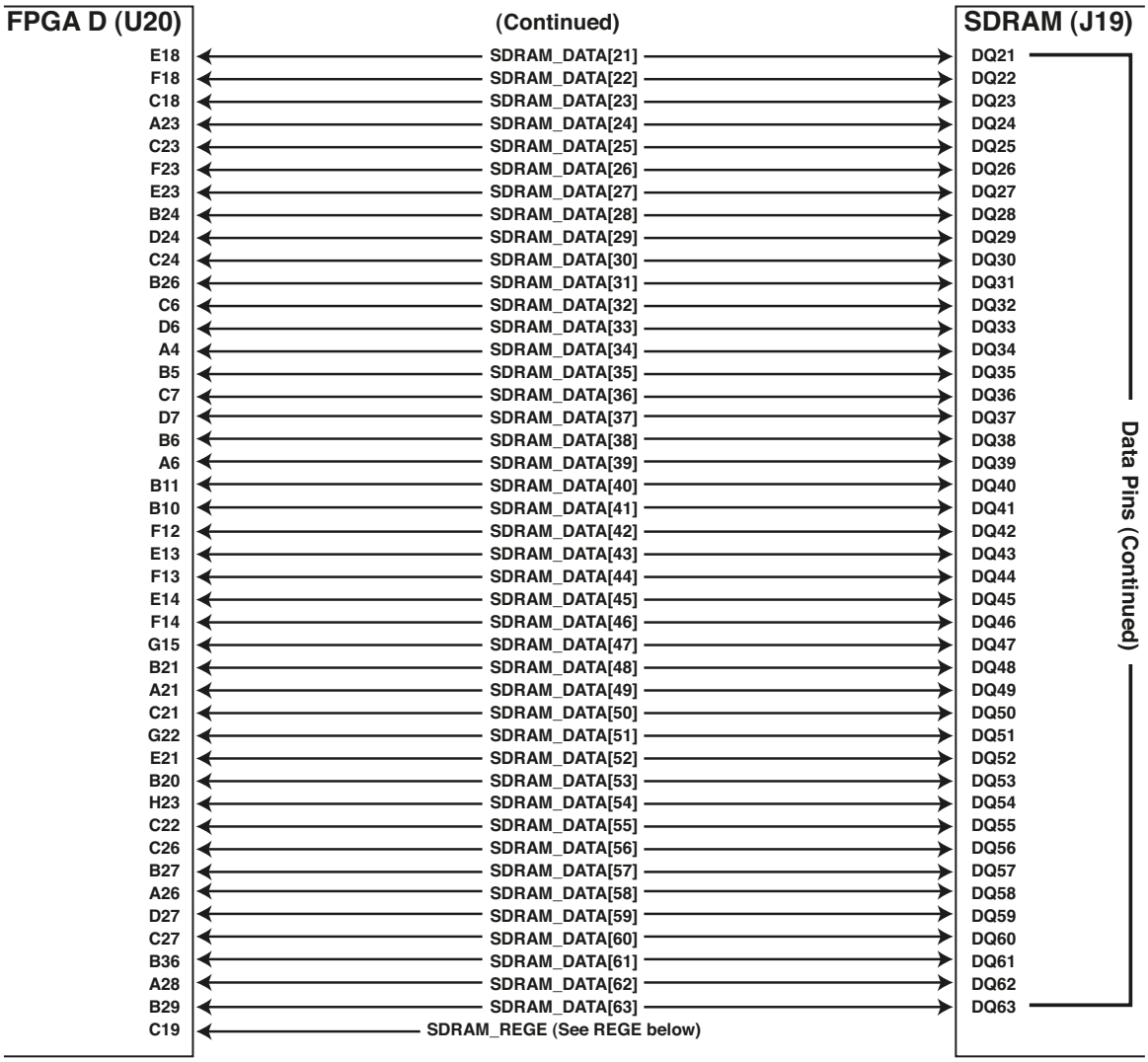


Figure 5-12 SDRAM (J19) Bus Signals (Page 2 of 2)

SDRAM On-Board Options

R189 and R193 are connected to the **WP** (Write Protect) input of the SDRAM EEPROM. Stuffing a 0-ohm resistor in R193 will keep the **WP** signal high, whereas stuffing it in R189 drives the signal low. The default configuration is RS193 stuffed. **NEVER** stuff both resistors at the same time.

The EEPROM holds data describing the size, configuration, and timing characteristics of the SDRAM. The data is write-protected when the **WP** signal is high. There should be little or no reason to want to overwrite the EEPROM data. Some SDRAM manufacturers simply connect the **WP** pin of the EEPROM chip to the power supply of the SDRAM, in which case the **WP** resistors have no effect whatsoever.

Header **J17** is connected to the **REGE** (Register Enable) input of the SDRAM and to ground. A pull-up resistor keeps the **REGE** signal high when the header is unconnected; adding a jumper between the two pins drives the signal low. The default configuration is no jumper. **REGE** is also connected

to FPGA D, intended as an input so that the design can check the status of **REGE**. Do **NOT** drive this signal high when **J17** is jumpered.

On some SDRAMs, the **REGE** input may be used to select Registered or Non-Registered behavior. If **REGE** is high, the control signals will go through registers before being sent to the individual DRAMs, delaying access by one clock cycle but improving fanout; if it is low, the signals will be passed directly to the DRAMs.

Chapter 6

Power Supplies and Power Distribution

The DN5000k10 can be hosted in a +3.3 V PCI slot, or it can be used stand-alone. Figure 6-1 shows the various supplies used on the DN5000k10 and the connections of these supplies on the circuit board. The supply, +5 V, from the PCI connector (or **P1**) supplies the basic power to the DN5000k10. The +3.3 V power from the PCI connector is ***not*** used, nor is it connected to any circuitry on the DN5000k10.

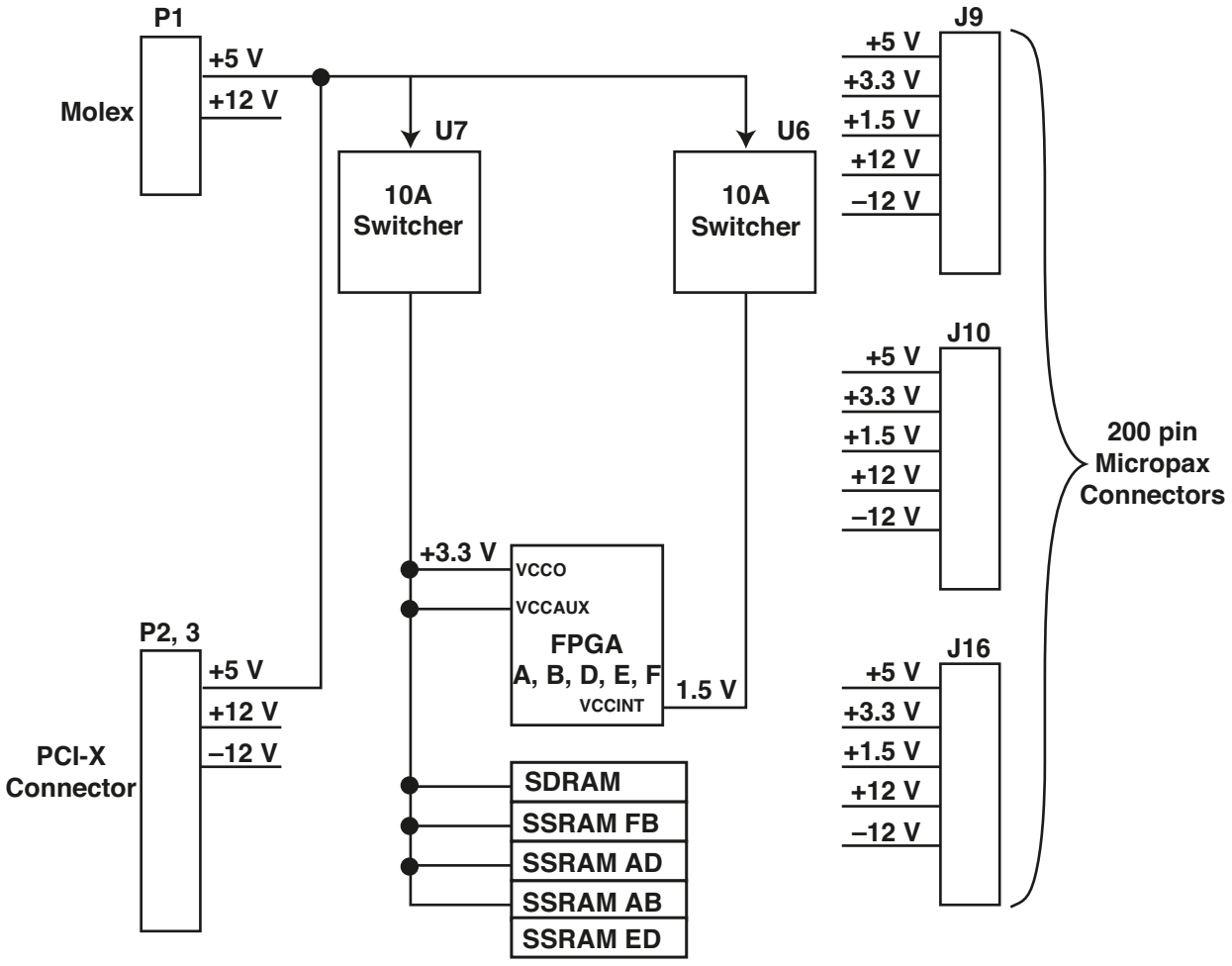


Figure 6-1 DN5000k10 Power Distribution

The DN5000k10, when plugged into a PCI slot, has the following different power rails:

- +5 V
- +3.3 V

- +1.5 V
- –12 V
- +12 V

The power rails +3.3 V and +1.5 V are created using a switching regulator with +5 V as the input. +3.3 V from the PCI fingers is ***not*** used. **U7** is for +3.3 V and **U6** is for +1.5 V. Heat is not an issue with this style of switching regulator. Each regulator should be able to supply the minimum 10 A of current without strain. The most demanding application of the DN5000k10 should fit within the 10 A budget on these two power rails.

+3.3 V Power

The specification for the +3.3 V power is shown in Table 6-1. The +3.3 V supply is used by the following components on the DN5000k10:

- Stratix FPGA I/O (**U11, U12, U15, U19, U20**)
- Roboclocks **U14, U16**
- 2 Clock buffers **U13**, **U17**
- CPLD (**U9**)
- Microprocessor (**U8**)
- Microprocessor SRAM (**U1**)
- 4 SSRAMs (**U18, U21, U22, U23**)
- SDRAM DIMM (**J19**)
- 3 Oscillators **X1**, **X2**, **X3**

We do run +3.3 V a little hot. At worst case for all components, the +3.3 V power supply should never fall below +3.30 V.

Table 6-1 Specification for +3.3 V Power

	Minimum	Typical	Maximum
Voltage	**+3.35 V**	**+3.39 V**	**+3.44 V**
Current	**N/A**	**N/A**	**12 A**

+1.5 V Power

The specification for the +1.5 V power is shown in Table 6-2. The +1.5 V supply is used by the following component on the DN5000k10:

- Stratix FPGA V_{cc}INT (**U11, U12, U15, U19, U20**)

We also run +1.5 V a little hot. At worst case for all components, the +1.5 V power supply should never fall below +1.50 V.

Table 6-2 Specification for +1.5 V Power

	Minimum	Typical	Maximum
Voltage	**+1.55 V**	**+1.56 V**	**+1.58 V**
Current	**N/A**	**N/A**	**12 A**

If you use the DN5000k10 in a lab environment, the Stratix FPGA will never see worst case power and temperature—so you can use typical, commercial timing.

NOTE: In a lab environment, the FPGA never sees the worst-case temperature and power. You can use typical, commercial timing!

Stand-Alone Operation

The DN5000k10 can be used stand-alone, meaning it doesn't have to be plugged into a PCI slot. Connector **P1** is used to provide power to the DN5000k10 in this configuration. **P1** is a Molex drive power connector and will connect to any standard ATX power supply (see Figure 6-2). The power supply that we used is shown in Figure 6-3, but any ATX or AT style power supply will work. We use a 250-watt ATX supply. Since the DN5000k10 does not draw enough current to meet the minimums required by the supply, we plug an old disk drive into another one of the Molex connectors. The current drawn by the disk drive sinks enough current to make the switchers in the power supply happy.

The **P1** connector is rated to 13 A, far more current than the DN5000k10 can use.

The DN5000k10, when used stand-alone, has the following different power rails:

- +5 V
- +3.3 V
- +1.5 V
- +12 V

NOTE: If you use the DN5000k10 stand-alone with an ATX power supply, the DN5000k10 may not draw enough current to meet the minimum current required by the switchers in the supply. Connecting a disk drive to another connector will solve this problem!

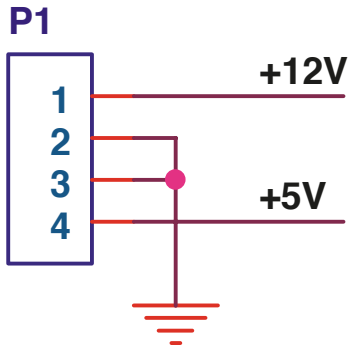


Figure 6-2 Molex Connector P1—Auxiliary Power

Figure 6-3 Example ATX Power Supply

By specification, a PCI board may consume a maximum of 25 watts from the fingers of the PCI connector. This power limit is below that the DN5000k10 is capable of consuming, even if daughter cards and/or large SDRAM banks are installed. The **P1** connector can be used to augment the power obtained from the PCI fingers. **P1** can be used provided that the +5 V and +12 V power rails on the connector are supplied by the same power source as the PCI fingers.

NOTE: P1 and PCI may provide power at the same time, but ONLY if the same power source used to supply P1 is also supplying power to PCI.

Chapter 7

Daughter Connections to DN3000k10SD—Observation Daughter Card for 200-pin Connectors

The traditional approach to experiment with new devices involving wiring together some ICs on a breadboard is fast becoming impractical and ineffective. Instead, designers using new high-density devices need custom PC boards representing a substantial investment of time and money.

Prototype boards from manufacturers can meet this demand for experimentation while eliminating the expense and time involved with custom PC boards. Additionally, such prototype boards facilitate the understanding and advantages of new device features.

Purpose

The DN3000k10SD daughter card allows external connection to the signals present on the DN5000k10 series ASIC prototyping boards. The DN5000k10 allows logic emulation with Stratix™ devices prior to committing to using them for specific applications. It allows designers to try Stratix features such as BlockRAM, DLLs, and SelectI/O™ resource, with an off-the-shelf resource.

Features

The DN3000k10SD Daughter Card has the following features:

- Buffered I/O, Passive and Active Bus Drivers
- Unbuffered I/O
- Differential LVDS pairs (Note: Not available on DN5000k10 ASIC prototyping board)
- Headers for Test Points

The daughter card contains headers that may be useful with certain types of oscilloscope probes, or when wiring pins to prototype areas.

Figure 7-1 is a block diagram of the DN3000k10SD Daughter Card.

The DN3000k10SD Daughter Card is pictured in Figure 7-2.

Figure 7-3 shows the assembly drawing of the DN3000k10SD Daughter Card.

The DN3000k10SD Daughter Card provides 16 differential pairs, 48 buffered (passive/active) I/O, and 66 unbuffered I/O signals. The

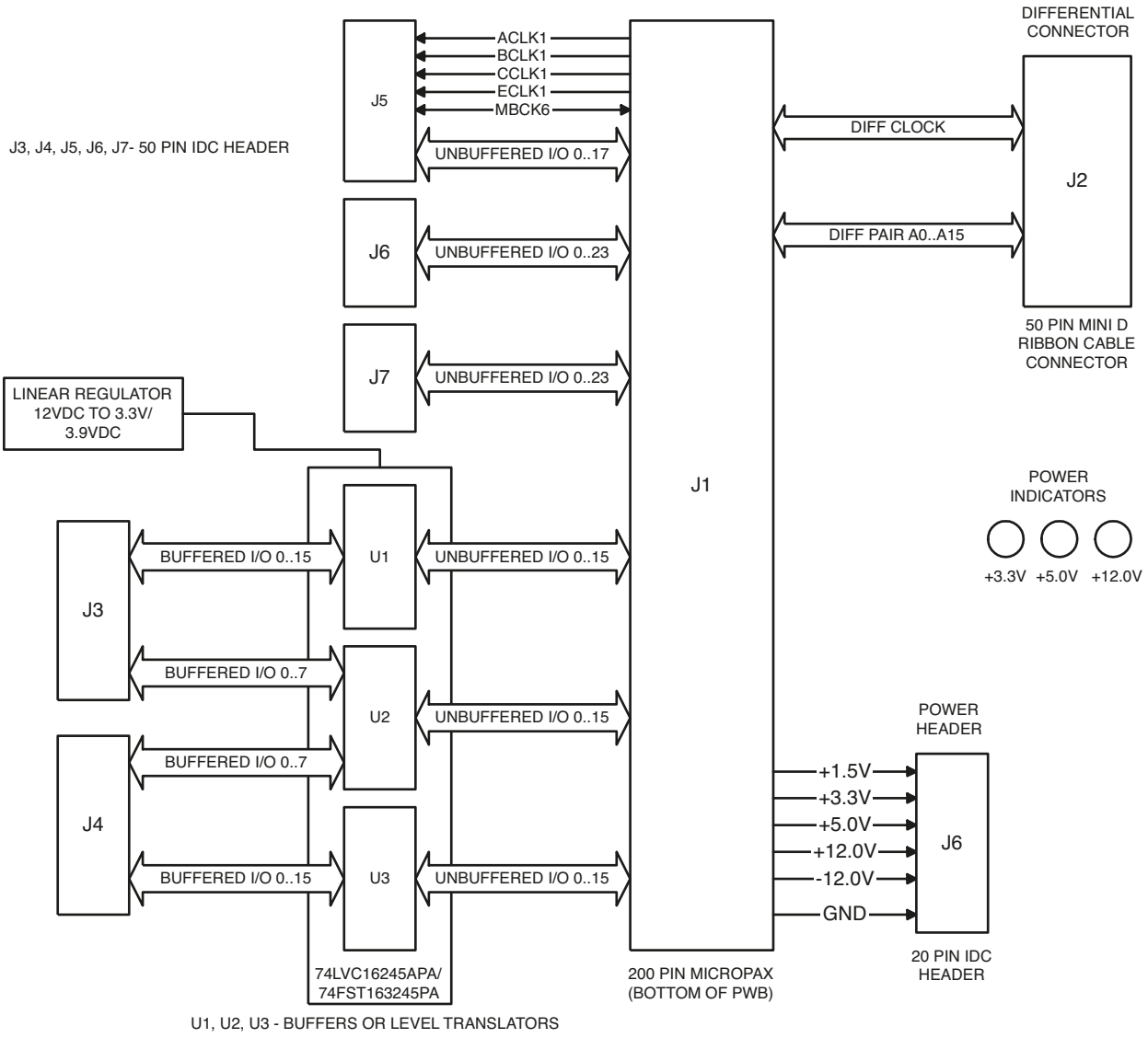
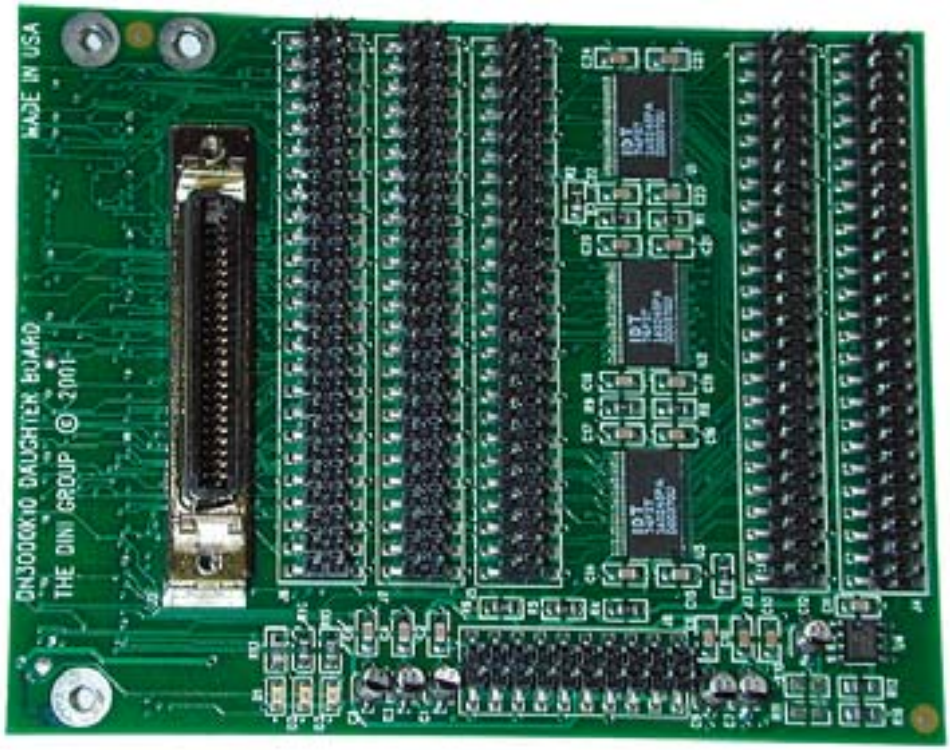
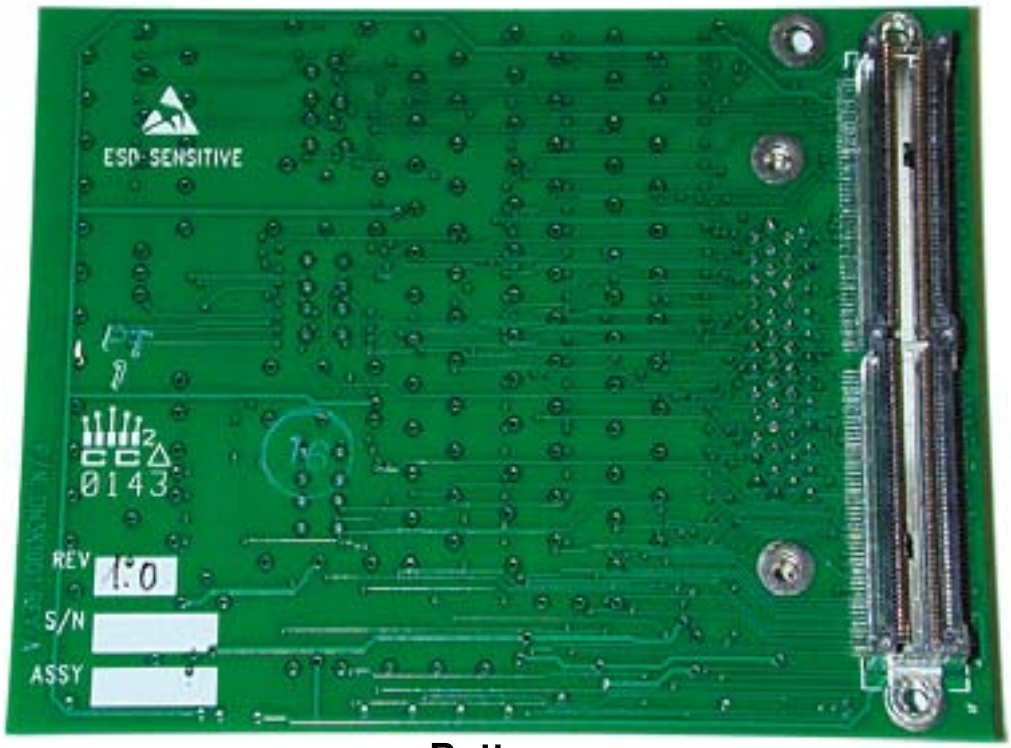


Figure 7-1 DN3000k10SD Daughter Card Block Diagram



Top



Bottom

Figure 7-2 DN3000k10SD Daughter Card

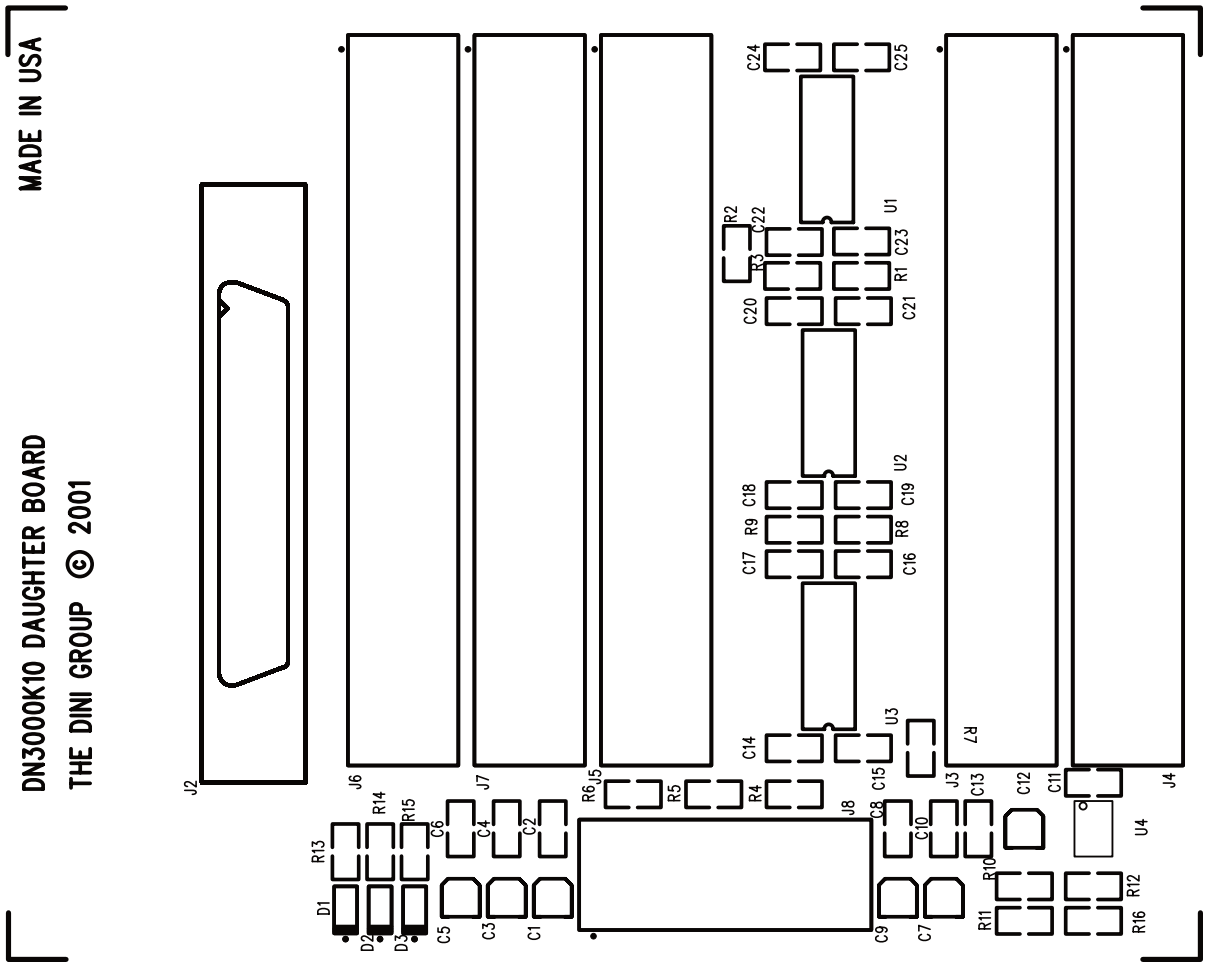


Figure 7-3 DN3000k10SD Daughter Card Assembly Drawing

IDT74FST163245 chips are used as bus switches in the passive mode, and the IDT74LVC16245A chips are used as bus transceivers in the active mode. The DN3000k10SD has separate enable/direction signals for each driver.

NOTE: Availability of these I/O signals depends on the location of the daughter card with respect to the development board.

Daughter Card LEDs

The LEDs act as visual indicators, representing the active power sources.

- **D1** — LED indicating +3.3 V present
- **D2** — LED indicating +5.0 V present
- **D3** — LED indicating +12 V present

Under normal operating conditions, all LEDs should be on.

Power Supply

A linear power supply (**U4**) is present to provide level shift/translation functions when the board is populated with bus switches.

Options

Resistors R10 and R11 can be used to select different voltage sources, +5 V or +3.3 V, respectively. When used, **U4** must be removed in order to prevent contention.

NOTE: Never populate **R10/R11** simultaneously: this will result in a shorted power supply.

Power Rating

- +5 V power supply is rated for 1 A.
- +3.3 V power supply is rated for 1 A.
- +1.5 V power supply is rated for 1 A.
- +12 V power supply is rated for 0.5 A.
- –12 V power supply is rated for 0.5 A.

Connector J8

Table 7-1 shows the connections of **J8**.

Table 7-1 Connector J8 Pins External Power

Pin	Function	Pin	Function
1	GND	11	GND
2	+5 V	12	+1.5 V
3	GND	13	GND
4	+5 V	14	+12 V
5	GND	15	GND
6	+3.3 V	16	+12 V
7	GND	17	GND
8	+3.3 V	18	–12 V
9	GND	19	GND
10	+1.5 V	20	–12 V

LVDS

Low-voltage differential signaling (LVDS) is a signaling method used for high-speed transmission of binary data over copper. It is well recognized that the benefits of balanced data transmission begin to outweigh the costs over single-ended techniques when the signal transmission times approach 10 ns. This represents signaling rates of about 30 Mbps or clock rates of 60 MHz (in single-edge clocking systems) and above. LVDS is defined in the TIA/EIA-644 standards.

NOTE: Not available on the DN5000k10 ASIC prototyping board.

Connector J2

This is a Mini D Ribbon (MDR) connector (50 pin) manufactured by 3M, used specifically for high speed LVDS signaling. The connector mates with a standard off-the-shelf 3M-cable assembly:

P/N 14150-EZBB-XXX-0LC

where XXX is:
050 = 0.5 m
150 = 1.5 m
300 = 3.0 m
500 = 5.0 m

Please contact 3M for further details: **http://www1.3m.com/**.

Unbuffered I/O

The DN3000k10SD Daughter Card provides 66 unbuffered I/O signals, including 5 single ended clock signals. The function of these signals is position dependent.

NOTE: SIgnals `P4NX7` and `P4NX6` are also used for direction select and output enable on **U2** and **U3** respectively.

Connectors J3, J4

J3, **J4**— Buffered Interface header

IDC headers (50 pin) providing 48 buffered I/O signals.

See Table 7-2 on page 7-8.

Connector J5, J6, J7

J5, J6, J7 — Unbuffered Interface Header

IDC headers (50 pin) providing 66 buffered I/O signals.

See Table 7-2 on page 7-8.

Buffered I/O

The DN3000k10SD Daughter Card provides 48 buffered I/O signals. The function of these signals is position dependent. **U1**, **U2**, and **U3** allow for different populating options, and devices can be active or passive.

Active

The LCV162245A is used for asynchronous communication between data buses. It allows data transmission from the A to the B or from the B to the A bus, depending on the logic level at the direction-control (**DIR**) input. The output-enable (**OE#**) input can be used to disable the device so that the busses are effectively isolated.

Passive

The FST163245 bus switches are used to connect or isolate two ports without providing any current sink or source capabilities. Thus, they generate little or no noise of their own while providing a low resistance path for an external driver. The output-enable (**OE#**) input can be used to disable the device so that the busses are effectively isolated.

Test Interface

The DN3000k10SD Daughter Card provides a 200-pin connector to interface to one of three test connectors on the DN5000k10 ASIC prototyping board: **J9**, **J10** and **J16**.

Connector J1

J1—Test Interface Connector

Micropax connector (200 pin) used as a standard interface to all the DINI Group development boards. This connector has a specified current rating of 0.5 amps per contact. See Table 7-2 on page 7-8.

Daughter Card I/O Connections

Table 7-2 shows the DN3000k10SD Daughter Card I/O Interconnects to connectors **J25, J26** and **J28**.

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN5000k10 I/O Connector							
			J9, J10, or J16 Pin No.	Header J9		Header J10		Header J16	
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin
1	+12 V		1	+12 V		+12 V		+12 V	
2	GND		2	GND		GND		GND	
3	ACLK[1]	J5.1	3	ACLK[7]		ACLK[6]		ACLK[5]	
4	+5 V		4	+5 V		+5 V		+5 V	
5	BCLK[1]	J5.3	5	BCLK[7]		BCLK[6]		BCLK[5]	
6	+5 V		6	+5 V		+5 V		+5 V	
7	CCLK[1]	J5.5	7	CCLK[7]		CCLK[6]		CCLK[5]	
8	GND		8	GND		GND		GND	
9	+3.3 V		9	+3.3 V		+3.3 V		+3.3 V	
10	P2N[3]	J3.1	10	HDR_ CLKOUT		ECLK[14]		ECLK[13]	
11	**GND**		**11**	**GND**		**GND**		**GND**	
12	P2N[2]	J3.3	12	AD[149]	U20.AH15	FB[135]	U11.AN27	AB[172]	U19.AT33
13	P2N[1]	J2.8	13	AD[148]	U20.AH14	FB[141]	U11.AH22	AB[171]	U19.AP32
14	P2N[0]	J2.9	14	AD[151]	U20.AM15	FB[140]	U11.AF21	AB[170]	U19.AR32
15	P2NX[7]	J3.5	15	AD[150]	U20.AL15	FB[139]	U11.AC26	AB[169]	U19.AT32
16	P2NX[6]	J3.7	16	AD[152]	U20.AK14	FB[138]	U11.AB26	AB[168]	U19.AP31
17	P2NXP5[	J3.9	17	AD[0]	U20.AL14	FB[137]	U11.AR34	AB[167]	U19.AR31
18	P2NX[4]	J3.11	18	AD[115]	U20.AM12	FB[136]	U11.AP34	AB[166]	U19.AT31
19	P2NX[1]	J2.10	19	AD[114]	U20.AL13	FB[36]	U11.AV36	AB[165]	U19.AP30
20	P2NX[0]	J2.11	20	AD[112]	U20.AK13	FB[35]	U11.AU38	AB[164]	U19.AR30
21	P3NX[9]	J2.40	21	AD[111]	U20.AK12	FB[34]	U11.AK32	AB[163]	U19.AT30
22	**GND**		**22**	**GND**		**GND**		**GND**	
23	P3NX[8]	J2.41	23	AD[110]	U20.AA13	AF[252]	U11.AR19	AB[162]	U19.AP29
24	P3NX[5]	J3.13	24	AD[109]	U20.AA12	FB[33]	U11.AJ32	AB[161]	U19.AR29
25	P3NX[4]	J3.15	25	AD[108]	U20.AH13	AF[161]	U11.AT19	AB[160]	U19.AT29
26	P3N[89]	J3.17	26	AD[107]	U20.AH12	AF[40]	U11.AP19	AB[159]	U19.AP28

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN5000k10 I/O Connector						
			J9, J10, or J16 Pin No.	Header J9		Header J10		Header J16	
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin
27	P3N[88]	J3.19	27	AD[106]	U20.AG13	FB[28]	U11.AF31	AB[158]	U19.AR28
28	P3N[87]	J3.21	28	AD[105]	U20.AG12	FB[29]	U11.AG32	AB[157]	U19.AT28
29	P3N[86]	J3.23	29	AD[104]	U20.AF13	FB[27]	U11.AF32	AB[156]	U19.AP27
30	P3N[83]	J3.25	30	AD[103]	U20.AF12	FB[25]	U11.A32	AB[155]	U19.AR27
31	P3N[82]	J3.27	31	AD[102]	U20.AE13	FB[26]	U11.AE31	AB[154]	U19.AT27
32	P3N[77]	J3.29	32	AD[101]	U20.AE12	FB[24]	U11.AD31	AB[153]	U19.AP26
33	**GND**		**33**	**GND**		**GND**		**GND**	
34	P3N[76]	J3.31	34	AD[100]	U20.AD13	AF[252]	U11.AR19	AB[162]	U19.AP29
35	P3N[75]	J3.33	35	AD[99]	U20.AD12	FB[33]	U11.AJ32	AB[161]	U19.AR29
36	P3N[74]	J3.35	36	AD[93]	U20.U13	AF[161]	U11.AT19	AB[160]	U19.AT29
37	P3N[69]	J2.42	37	AD[98]	U20.AC13	AF[40]	U11.AP19	AB[159]	U19.AP28
38	P3N[68]	J2.43	38	AD[92]	U20.U12	FB[28]	U11.AF31	AB[158]	U19.AR28
39	P3N[67]	J3.37	39	AD[97]	U20.AC12	FB[29]	U11.AG32	AB[157]	U19.AT28
40	P3N[66]	J3.39	40	AD[91]	U20.T13	FB[27]	U11.AF32	AB[156]	U19.AP27
41	P3N[63]	J3.41	41	AD[96]	U20.AB13	FB[25]	U11.AE32	AB[155]	U19.AR27
42	P3N[62]	J3.43	42	AD[90]	U20.T12	FB[26]	U11.AE31	AB[154]	U19.AT27
43	P3N[57]	J3.45	43	AD[95]	U20.W13	FB[24]	U11.AD31	AB[153]	U19.AP26
44	**GND**		**44**	**GND**		**GND**		**GND**	
45	P3N[56]	J3.47	45	AD[94]	U20.V13	FB[13]	U11.U32	AB[142]	U19.AT23
46	P3N[55]		46	AD[89]	U20.R13	FB[12]	U11.T31	AB[141]	U19.AP22
47	P3N[54]		47	AD[87]	U20.P13	FB[11]	U11.T32	AB[140]	U19.AR22
48	P3N[49]	J4.1	48	AD[88]	U20.R12	FB[10]	U11.R31	AB[139]	U19.AN22
49	P3N[48]	J4.3	49	AD[86]	U20.P12	FB[9]	U11.R32	AB[138]	U19.AT21
50	P3N[47]	J2.19	50	AD[85]	U20.N13	FB[8]	U11.P31	AB[137	U19.AN18
51	P3N[46]	J2.20	51	AD[84]	U20.N12	FB[7]	U11.P32	AB[136]	U12.AT19
52	P3N[43]	J4.5	52	AD[83]	U20.M13	FB[6]	U11.N31	AB[135]	U19.AT20
53	P3N[42]	J4.7	53	AD[82]	U20.M12	FB[5]	U11.N32	AB[134]	U19.AP19
54	P3N39	J4.9	54	AD[81]	U20.K16	FB[4]	U11.M31	AB[133]	U12.AT20
55	**GND**		**55**	**GND**		**GND**		**GND**	
56	P3N[38]	J4.11	56	AD[79]	U20.K13	FB[3]	U11.M32	AB[132]	U12.AT21

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN5000k10 I/O Connector						
			J9, J10, or J16 Pin No.	Header J9		Header J10		Header J16	
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin
57	P3N[35]	J4.13	57	AD[80]	U20.K15	FB[2]	U11.K25	AB[131]	U19.AP18
58	P3N[34]	J4.15	58	AD[77]	U20.J13	FB[1]	U11.L32	AB[130]	U19.AR18
59	P3N[29]	J4.17	59	AD[78]	U20.K12	FB[129]	U11.P24	AB[129]	U19.AT18
60	P3N[28]	J4.19	60	AD[76]	U20.J12	FB[130]	U11.U26	AB[128]	U19.AM18
61	P3N[27]	J4.21	61	AD[55]	U20.T11	FB[127]	U11.J27	AB[127]	U19.AR17
62	P3N[26]	J4.23	62	AD[75]	U20.H12	FB[128]	U11.T26	AB[126]	U19.AT17
63	P3N[23]	J2.21	63	AD[54]	U20.T10	FB[125]	U11.J29	AB[125]	U19.AP16
64	P3N[22]	J2.22	64	AD[116]	U20.K14	FB[126]	U11.R26	AB[124]	U19.AR16
65	P3N[19]	J4.25	65	AD[53]	U20.N16	FB[60]	U11.AD29	AB[123]	U19.AT16
66	**GND**		**66**	**GND**		**GND**		**GND**	
67	P3N[18]	J4.27	67	AD[52]	U20.R10	FB[59]	U11.AD30	AB[122]	U19.AP15
68	P3N[15]	J4.29	68	AD[56]	U20.U10	FB[58]	U11.AC30	AB[121]	U19.AR15
69	P3N[14]	J4.31	69	AD[51]	U20.R7	FB[57]	U11.AB30	AB[120]	U19.AT15
70	P3N[9]	J2.23	70	AD[117]	U20.G7	FB[56]	U11.W30	AB[119]	U19.AP14
71	P3N[8]	J2.24	71	AD[50]	U20.P10	FB[55]	U11.V30	AB[118]	U19.AR14
72	P3N[7]	J4.33	72	AD[118]	U20.G6	FB[54]	U11.U30	AB[117]	U19.AT14
73	P3N[6]	J4.35	73	AD[49]	U20.P7	FB[53]	U11.T29	AB[116]	U19.AP13
74	P3N[3]	J4.37	74	AD[48]	U20.N10	FB[52]	U11.T30	AB[115]	U19.AR13
75	P3N[2]	J4.39	75	AD[9]	U20.P8	FB[51]	U11.U29	AB[114]	U19.AT13
76	P4N27	J4.41	76	AD[47]	U20.N7	FB[50]	U11.R30	AB[113]	U19.AP12
77	**GND**		**77**	**GND**		**GND**		**GND**	
78	P4N[26]	J4.43	78	AD[46]	U20.M10	FB[49]	U11.R33	AB[112]	U19.AR12
79	P4N[21]	J4.45	79	AD[6]	U20.M9	FB[48]	U11.P30	AB[111]	U19.AT12
80	P4N[20]	J4.47	80	AD[4]	U20.H13	FB[47]	U11.N24	AB[110]	U19.AP11
81	P4N[19]		81	AD[5]	U20.M8	FB[46]	U11.N30	AB[109]	U19.AR11
82	P4N[18]		82	AD[3]	U20.L8	FB[45]	U11.N33	AB[108]	U19.AT11
83	P4N[13]		83	AD[2]	U20.J8	FB[44]	U11.M30	AB[107]	U19.AP10
84	P4N[12]		84	AD[1]	U20.H9	FB[43]	U11.J28	AB[106]	U19.AR10
85	P4N[11]		85	AD[39]	U20.H11	FB[42]	U11.K29	AB[105]	U19.AT10
86	P4N[10]		86	AD[38]	U20.W14	FB[41]	U11.G30	AB[104]	U19.AP9

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN5000k10 I/O Connector						
			J9, J10, or J16 Pin No.	Header J9		Header J10		Header J16	
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin
87	P4N[7]		87	AD[133]	U20.W14	FB[40]	U11.K32	AB[103]	U19.AR9
88	**GND**		**88**	**GND**		**GND**		**GND**	
89	P4N[6]		89	AD[134]	U20.G5	FB[39]	U11.K33	AB[102]	U19.AT9
90	P4N[3]		90	AD[131]	U20.V14	FB[38]	U11.G31	AB[101]	U19.AP8
91	P4N[2]		91	AD[132]	U20.F5	FB[37]	U11.J32	AB[100]	U19.AR8
92	P4NX[11]		92	AD[129]	U20.U14	FB[123]	U11.F34	AB[99]	U19.AT8
93	+1.5 V		93	+1.5 V		+1.5 V		+1.5 V	
94	P4NX[10]		94	AD[130]	U20.E3	FB[124]	U11.F35	AB[98]	U19.AP7
95	P4NX[7]	J7.45	95	AD[125]	U20.R14	FB[121]	U11.M25	AB[97]	U19.AR7
96	P4NX[6]	J7.47	96	AD[126]	U20.D2	FB[122]	U11.N26	AB[96]	U19.AT7
97	P4NX[5]		97	AD[127]	U20.T14	FB[119]	U11.H27	AB[95]	U19.AT6
98	P4NX[4]		98	AD[128]	U20.C2	FB[120]	U11.M26	AB[94]	U19.AT5
99	**GND**		**99**	**GND**		**GND**		**GND**	
100	–12 V		100	–12 V		–12 V		-12 V	
101	GND		101	GND		GND		GND	
102	MBCK[1]	J2.27	102	MB[156]	U11.AV8	AD[84]	U19.AV33		
103	+1.5 V		103	+1.5 V		+1.5 V		+1.5 V	
104	MBCK[0]	J2.28	104	MB[154]	U11.AW6	AB[174]	U19.AP33		
105	+3.3 V		105	+3.3 V		+3.3 V		+3.3 V	
106	MBCK[6]	J5.9	106	DCLK[6]		DCLK[5]			
107	GND		107	GND		GND		GND	
108	ECLK[1]	J5.7	108	MB[153]	U11.AW5	AB[173]	U19.AR33		
109	**GND**		**109**	**GND**		**GND**		**GND**	
110	**GND**		**110**	**GND**		**GND**		**GND**	
111	P2N[5]	J5.15	111	AD[57]	U20.V10	FB[149]	U11.AH25	AB[85]	U19.AU33
112	P2N[4]	J5.17	112	AD[58]	U20.AA10	FB[148]	U11.AH26	AB[80]	U19.AW32
113	P2NX[11]	J2.2	113	AD[59]	U20.AB10	FB[147]	U11.AG25	AB[81]	U19.AV32
114	P2NX[10]	J2.1	114	AD[60]	U20.AC10	FB[146]	U11.AG26	AB[82]	U19.AU32
115	P2NX[9]	J5.19	115	AD[61]	U20.AD10	FB[145]	U11.AF25	AB[77]	U19.AW31
116	P2NX[8]	J5.21	116	AD[62]	U20.AD11	FB[144]	U11.AF26	AB[78]	U19.AV31

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN5000k10 I/O Connector						
			J9, J10, or J16 Pin No.	Header J9		Header J10		Header J16	
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin
117	P2NX[3]	J5.23	117	AD[63]	U20.AE8	FB[143]	U11.AE26	AB[79]	U19.AU31
118	GND		118	GND		GND		GND	
119	P2NX[2]	J5.25	119	AD[64]	U20.AE10	FB[142]	U11.AD26	AB[74]	U19.AW30
120	P3NX[11]	J2.29	120	AD[65]	U20.AF8	FB[75]	U11.AV37	AB[75]	U19.AV30
121	P3NX[10]	J2.30	121	AD[66]	U20.AF10	FB[74]	U11.AL29	AB[76]	U19.AU30
122	P3NX[7]	J2.31	122	AD[67]	U20.AG10	FB[73]	U11.AL30	AB[71]	U19.AW29
123	P3NX[6]	J2.32	123	AD[68]	U20.AC11	FB[72]	U11.AK29	AB[72]	U19.AV29
124	P3NX[3]	J5.27	124	AD[69]	U20.AH10	FB[71]	U11.AT36	AB[73]	U19.AU29
125	P3NX[2]	J5.29	125	AD[70]	U20.AB11	FB[70]	U11.AR36	AB[68]	U19.AW28
126	P3NX[1]	J5.31	126	AD[71]	U20.AE14	FB[69]	U11.AR35	AB[69]	U19.AV28
127	P3NX[0]	J5.33	127	AD[72]	U20.AG16	FB[68]	U11.AJ33	AB[70]	U19.AU28
128	P3N[85]	J5.35	128	AD[73]	U20.AK11	FB[67]	U11.AH30	AB[66]	U19.AV27
129	GND		129	GND		GND		GND	
130	P3N[84]	J5.37	130	AD[74]	U20.AL12	FB[66]	U11.AB29	AB[67]	U19.AU27
131	P3N[81]	J5.39	131	AD[113]	U20.AM13	FB[65]	U11.AG30	AB[63]	U19.AW26
132	P3N[80]	J5.41	132	AD[144]	U20.AF14	FB[64]	U11.AC29	AB[64]	U19.AV26
133	P3N[79]	J2.3	133	AD[145]	U20.AF15	FB[63]	U11.AF30	AB[65]	U19.AU26
134	P3N[78]	J2.4	134	AD[146]	U20.AG14	FB[62]	U11.AF33	AB[61]	U19.AV25
135	P3N[73]	J2.6	135	AD[147]	U20.AG15	FB[61]	U11.AE30	AB[62]	U19.AU25
136	P3N[72]	J2.7	136	AD[143]	U20.AR5	FB[151]	U11.AM25	AB[58]	U19.AW24
137	P3N[71]	J2.33	137	AD[142]	U20.AR4	FB[150]	U11.AL25	AB[59]	U19.AV24
138	P3N[70]	J2.34	138	AD[141]	U20.AR6	FB[0]	U11.AL26	AB[60]	U19.AU24
139	P3N[65]	J5.43	139	AD[140]	U20.AD14	FB[152]	U11.AK26	AB[55]	U19.AW23
140	GND		140	GND		GND		GND	
141	P3N[64]	J5.45	141	AD[139]	U20.AP6	FB[115]	U11.AM27	AB[56]	U19.AV23
142	P3N[61]	J5.47	142	AD[138]	U20.AC14	FB[114]	U11.AM28	AB[57]	U19.AU23
143	P3N[60]	J5.49	143	AD[137]	U20.AN6	FB[113]	U11.AL27	AB[52]	U19.AW22
144	P3N[59]	J6.1	144	AD[136]	U20.AB14	FB[112]	U11.AL28	AB[53]	U12.AV18
145	P3N[58]	J6.3	145	AD[135]	U20.AA14	FB[111]	U11.AK27	AB[54]	U19.AU22
146	P3N[53]	J6.5	146	AD[37]	U20.AM9	FB[110]	U11.AK28	AB[49]	U19.AU22

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN5000k10 I/O Connector							
			J9, J10, or J16 Pin No.	Header J9		Header J10		Header J16	
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin
147	P3N[52]	J6.7	147	AD[36]	U20.AK7	FB[109]	U11.AK25	AB[50]	U19.AV21
148	P3N[51]	J2.17	148	AD[35]	U20.AL8	FB[108]	U11.AJ24	AB[51]	U19.AU21
149	P3N[50]	J2.18	149	AF[77]	U11.F36	FB[107]	U11.AH27	AB[46]	U19.AW20
150	P3N[45]	J6.9	150	AD[34]	U20.AK8	FB[106]	U11.AH28	AB[47]	U19.AV20
151	**GND**		**151**	**GND**		**GND**		**GND**	
152	P3N[44]	J6.11	152	AD[33]	U20.AA11	FB[105]	U11.AG27	AB[48]	U19.AU20
153	P3N[41]	J6.13	153	AF[78]	U11.G36	FB[104]	U11.AG28	AB[43]	U12.AW21
154	P3N[40]	J6.15	154	AF[79]	U11.H36	FB[103]	U11.AF27	AB[44]	U19.AV19
155	P3N[37]	J6.17	155	AD[28]	U20.AG8	FB[102]	U11.AF28	AB[45]	U19.AU19
156	P3N[36]	J6.19	156	AD[29]	U20.AG9	FB[101]	U11.AE27	AB[40]	U12.AW22
157	P3N[33]	J6.21	157	AD[27]	U20.AF9	FB[100]	U11.AE28	AB[41]	U19.AV18
158	P3N[32]	J6.23	158	AD[26]	U20.AD9	FB[99]	U11.AD27	AB[42]	U12.AU22
159	P3N[31[	J2.44	159	AD[25]	U20.AD8	FB[98]	U11.AD28	AB[37]	U19.AW17
160	P3N[30]	J2.45	160	AD[24]	U20.AC9	FB[97]	U11.AC27	AB[38]	U19.AV17
161	P3N[25]	J6.25	161	AD[23]	U20.AC3	FB[134]	U11.W26	AB[39]	U19.AU17
162	**GND**		**162**	**GND**		**GND**		**GND**	
163	P3N[24]	J6.27	163	AD[22]	U20.AB9	FB[133]	U11.V29	AB[34]	U19.AW16
164	P3N[21]	J6.29	164	AD[21]	U20.AB8	FB[132]	U11.V26	AB[35]	U19.AV16
165	P3N[20]	J6.31	165	AD[20]	U20.AA9	FB[131]	U11.P26	AB[36]	U19.AU16
166	P3N[17]	J6.33	166	AD[19]	U20.V11	FB[96]	U11.AC28	AB[33]	U19.AU15
167	P3N[16]	J6.35	167	AD[18]	U20.V9	FB[95]	U11.AB27	AB[29]	U19.AW14
168	P3N[13]	J6.37	168	AD[17]	U20.V8	FB[94]	U11.AB28	AB[176]	U19.AT35
169	P3N[12]	J639	169	AD[16]	U20.U9	FB[93]	U11.W27	AB[175]	U19.AT34
170	P3N[11]	J2.47	170	AD[15]	U20.U8	FB[92]	U11.V27	AB[27]	U19.AV13
171	P3N[10]	J2.48	171	AD[14]	U20.T9	FB[91]	U11.V28	AB[28]	U19.AU13
172	P3N[5]	J6.41	172	AD[13]	U20.T8	FB[90]	U11.U27	AB[24]	U19.AW12
173	**GND**		**173**	**GND**		**GND**		**GND**	
174	P3N[4]	J6.43	174	AD[12]	U20.R9	FB[89]	U11.U28	AB[25]	U19.AV12
175	P3N[1]	J6.45	175	AD[11]	U20.R8	FB[88]	U11.T27	AB[26]	U19.AU12
176	P3N[0]	J6.47	176	AD[10]	U20.P9	FB[87]	U11.T28	AB[21]	U19.AW11

Table 7-2 DN3000k10SD Daughter Card I/O Interconnects

Daughter Card Connections			DN5000k10 I/O Connector						
			J9, J10, or J16 Pin No.	Header J9		Header J10		Header J16	
J1	Signal	Conn		Signal	FPGA Pin	Signal	FPGA Pin	Signal	FPGA Pin
177	P4N[25]	J7.1	177	AD[8]	U20.N9	FB[86]	U11.R27	AB[22]	U19.AV11
178	P4N[24]	J7.3	178	AD[7]	U20.N8	FB[85]	U11.R28	AB[23]	U19.AU11
179	P4N23	J7.5	179	AD[45]	U20.M4	FB[84]	U11.P27	AB[18]	U19.AW10
180	P4N[22]	J7.7	180	AD[44]	U20.M6	FB[83]	U11.P28	AB[19]	U19.AV10
181	P4N[17]	J7.9	181	AD[43]	U20.K11	FB[82]	U11.N27	AB[20]	U19.AU10
182	P4N[16]	J7.11	182	AD[42]	U20.K8	FB[81]	U11.N28	AB[15]	U19.AW9
183	P4N[15]	J7.13	183	AD[41]	U20.J11	FB[80]	U11.M27	AB[16]	U19.AV9
184	**GND**		**184**	**GND**		**GND**		**GND**	
185	P4N[14]	J7.15	185	AD[40]	U20.J10	FB[79]	U11.M28	AB[17]	U19.AU9
186	P4N[9]	J7.17	186	AD[123]	U20.P14	FB[78]	U11.K27	AB[12]	U19.AW8
187	P4N[8]	J7.19	187	AD[124]	U20.P15	FB[77]	U11.L34	AB[13]	U19.AV8
188	P4N[5]	J7.21	188	AD[121]	U20.N14	FB[76]	U11.K28	AB[14]	U19.AU8
189	P4N[4]	J7.23	189	AD[122]	U20.N15	FB[116]	U11.J26	AB[9]	U19.AW7
190	P4N[1]	J7.25	190	AD[119]	U20.M14	FB[117]	U11.K26	AB[10]	U19.AV7
191	P4N[0]	J7.27	191	AD[120]	U20.M15	FB[118]	U11.H34	AB[11]	U19.AU7
192	P4NX[13]	J7.29	192	AF[1]	U11.E39	AF[38]	U11.AN30	AB[6]	U19.AW6
193	P4NX[12]	J7.31	193	AF[2]	U11.E38	AF[39]	U11.AN24	AB[7]	U19.AV6
194	P4NX[9]	J7.33	194	AF[3]	U11.F39	AF[37]	U11.AM31	AB[8]	U19.AU6
195	**GND**		**195**	**GND**		**GND**		**GND**	
196	P4NX[8]	J7.35	196	AF[4]	U11.F38	AF[83]	U11.J34	AB[3]	U19.AW5
197	P4NX[3]	J7.37	197	AF[5]	U11.F37	AF[80]	U11.H35	AB[4]	U19.AV5
198	P4NX[2]	J7.39	198	AF[6]	U11.G39	AF[82]	U11.J35	AB[5]	U19.AU5
199	P4NX[1]	J7.41	199	AF[7]	U11.G38	AB[1]	U19.AV4		
200	P4NX[0]	J7.43	200	AF[8]	U11.G37	AF[81]	U11.J36	AB[2]	U19.AU4

Chapter 8

Reset Schemes, LEDs, Bus Bars and 200 Pin Connectors

Reset Schemes

A LTC1326 chip from Linear Technology controls reset functionality for the DN5000k10. Figure 8-1 shows the distribution of the reset signal PWRRST–. In addition to controlling the reset, the power supplies rails +5 V, +3.3 V, and +1.5 V are threshold detected by the LTC1326. Undervoltage conditions will case the assertion of the reset signal.

The LTC1326 has a push-button. Momentarily depressing this button causes a 200 ms reset pulse on the signal PWRRST–. If the push-button is depressed for 2 seconds and held, PWRRST– is asserted continuously. LED5, when lit, means that reset is asserted, so if you press and hold **S1**, you should see LED5 illuminate after a few seconds. If LED5 illuminates for any reason during normal operation, this indicates that PWRRST– is active and that something is wrong. Note that if you press **S1** and release it quickly, you probably won't see LED5 since the 200 ms reset pulse is not strong long enough for the eye to observe.

Depressing the push-button S1 causes the following sequence of events:

1. Reset of the CPLD and µP
2. FPGA configuration is cleared
3. If the switches on **S2** are set for Fast Passive Parallel and there is a valid SmartMedia card inserted into the socket, then the FPGA will be configured. A SmartMedia card is valid if it complies with the SSFDC specification and contains a file named main.txt in the root directory. If the card is invalid or there is no card present, then the FPGA will not be configured.
4. The Main Menu will appear.

The identical sequence of events occurs at power-up.

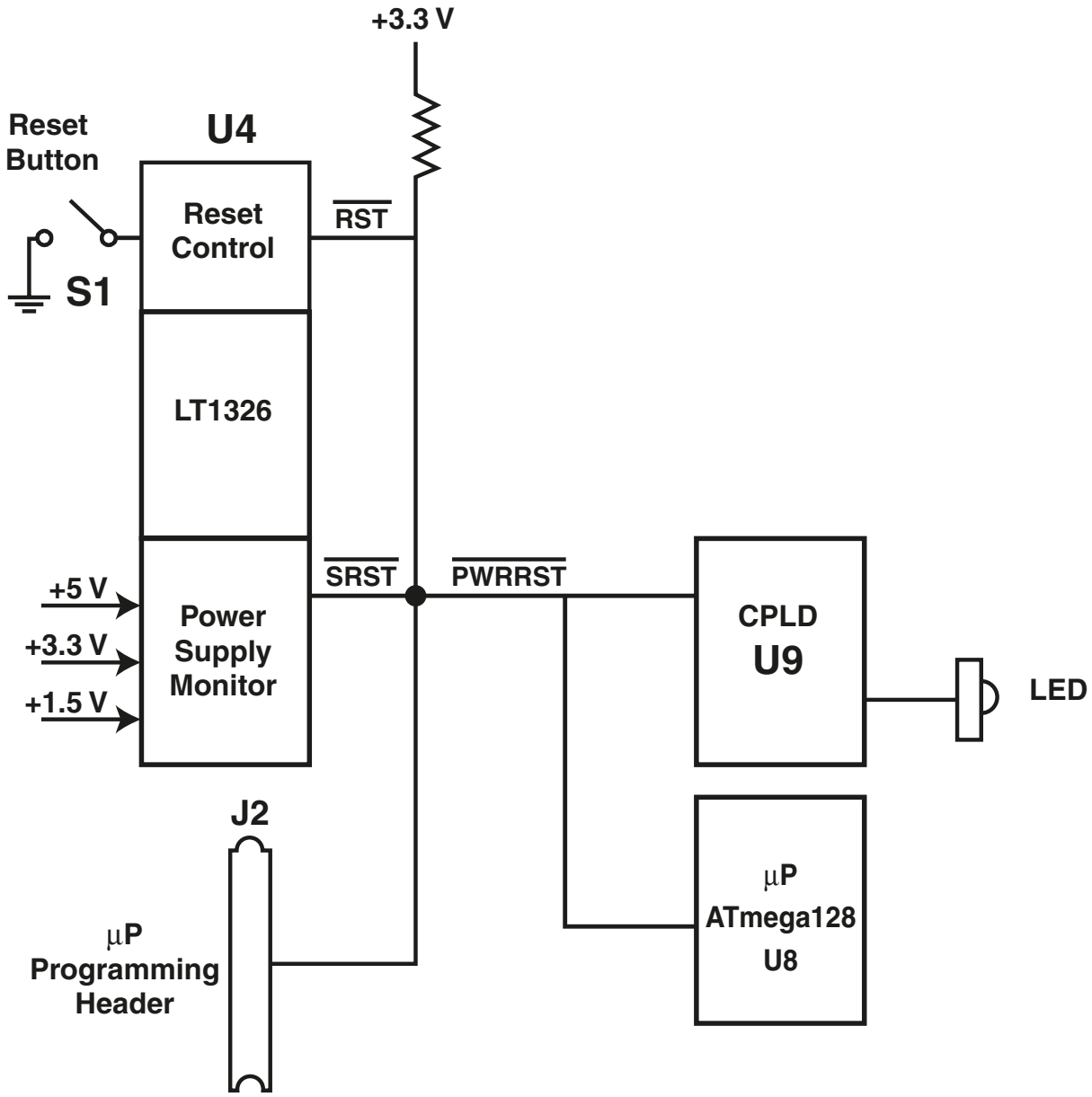


Figure 8-1 Reset Functionality

LEDs

The DN5000k10 has eight LEDs that are used to visually communicate the status of circuitry (Figure 8-2).

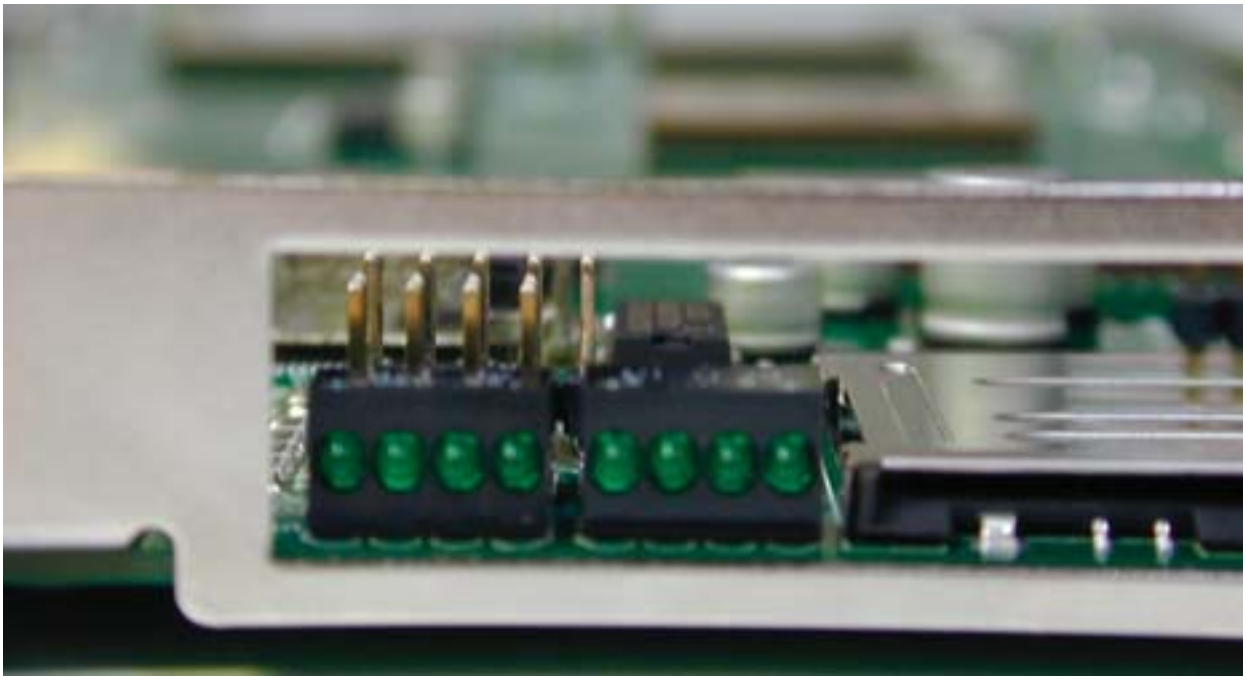


Figure 8-2 DN5000k10 LEDs

From left to right, the LEDs are labeled: CPLD_LED0, CPLD_LED1, CPLD_LED2, CPLD_LED3, UP_LED0, UP_LED1, UP_LED2, UP_LED3 (see Figure 8-3).

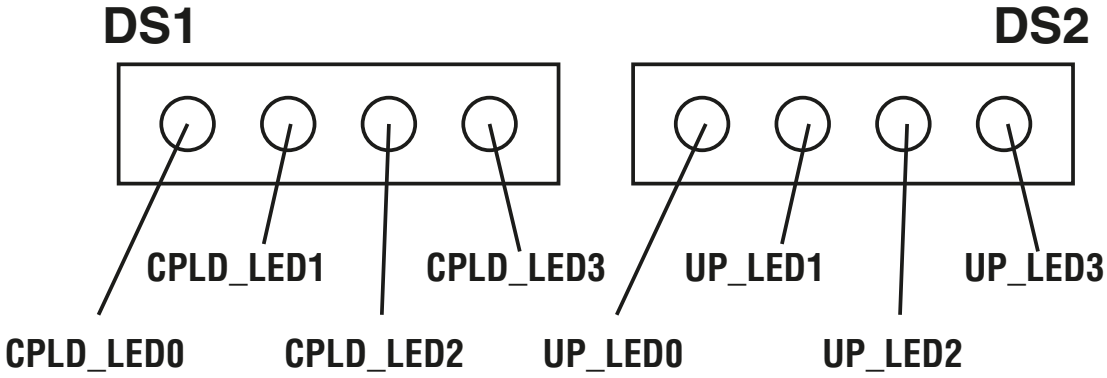


Figure 8-3 DN5000k10 LED Diagram

The LEDs have the following functions:

UP_LED3 Lights when the configuration process from the SmartMedia was successful.

UP_LED[2:0] These three LEDs have multiple meanings:

- When all 3 LEDs are blinking, then the µP has been reprogrammed and is waiting for the user to enter FPGA stuffing

information via serial port or there is not a valid SmartMedia card present and the FPGA has been configured.

- During configuration, the combination of LEDs lit tells the user which FPGA is currently being configured (UP_LED2, UP_LED1, UP_LED0):
  - off, off, on = FPGA F
  - off, on, off = FPGA A
  - off, on, on = FPGA E
  - on, off, off = FPGA B
  - on, off, on = FPGA D

CPLD_LED3 lights when any of the FPGAs are NOT configured

CPLD_LED2 lights when reset is asserted (PWRRST–)

CPLD_LED1 lights when the PLL in RoboclockII 1 is LOCKED

CPLD_LED0 lights when the PLL in RoboclockII 2 is LOCKED

You are free to reprogram the CPLD and/or microprocessor to use any or all of the LEDs for your own purposes.

Bus Bars

The two bus bars, **B1** and **B2,** are installed to prevent flexing of the PWB and serve no other purpose. They are connected quite solidly into the ground plane of the DN5000k10 at every hole, and you can use the metal bars to ground-test equipment such as oscilloscopes and pattern generators. Be careful not to short any power rails or signals to these metal bars—they can carry a lot of current. The PCI bracket, **BRK1**, is also connected to the ground plane at each of the screw mounts.

The 200 Pin Connectors: J9, J10, J16

The DN5000k10 contains three 200-pin connectors, **J9**, **J10** and **J16**. Daughter cards of any sort may be plugged into these connectors. The relative pin location of the powers, grounds, and signals is identical for each of the three connectors, with the exception of a clock output on **J9**. A hole that can be used to attach a standoff is located at the same relative position from each connector (seeFigure 8-4). This hole is grounded on the DN5000k10, so connect this mounting hole to digital ground on your daughter card.

The mechanical position of the 200-pin connectors on the DN5000k10 is shown in Figure 8-4. The 200-pin connector used on the DN5000k10 is a Berg Electronics 91294-003 in the Micropax™ family. This link will take you to the Berg website: **http://www.berg.com/**. This Berg connector was chosen because of its high pin density, performance, and availability. The part number for the mating connector is 91403-003. We stock the mating connector at our offices in La Jolla, CA, so if you are designing a daughter card and are having trouble getting this part, call us. We would be happy to send you a few at our cost. Appendix A contains a mechanical datasheet for both the Berg 91403-003 and 91294-003 connectors.

This style of connector has four mounting holes—two screw holes at each end and two alignment holes between pins 50–51 and after pin 100 (see Figure 8-4). These mounting holes are part of the metal shell of the

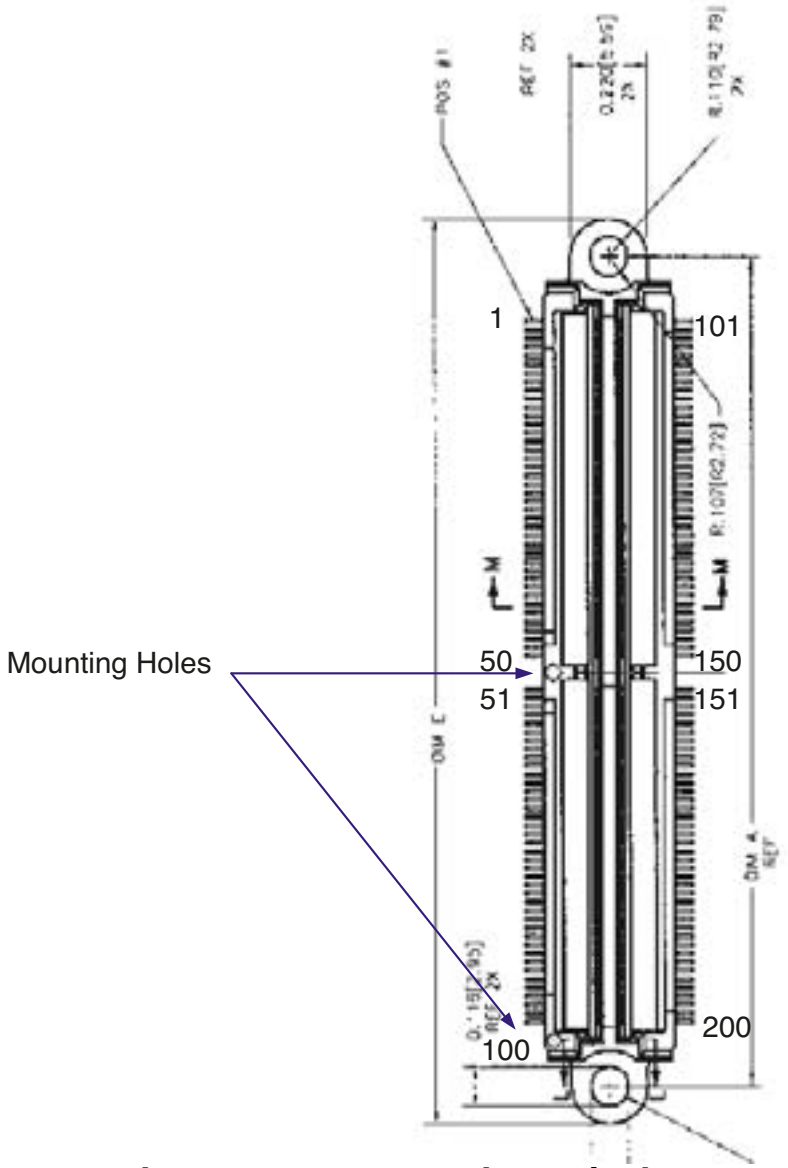


Figure 8-4 91294-003 Pin Numbering

connector and make an important connection to the mating connector. All four of these mounting holes are connected to digital ground on the DN5000k10—therefore, the shell of the connector is grounded.

We used the pin numbering shown in Figure 8-4 for the 200-pin, 91294-003 connectors.

The Signals

Each of the three 200-pin connectors has the following:

- 162 signals connected to the FPGA
  - All 162 are connected to the FPGA
  - A subset of the 162 are also connected to the memories
- 7 clocks
- The following power rails:
  - +12V (1 pin)

- -12V (1 pin)
- +5V (2 pins)
- +3.3V (2 pins)
- +1.5V (2 pins)
- GND (23 pins + case)

Regarding the amount of current that the power pins can carry, the following text is lifted directly from the specification for the Micropax™ family of connectors:

> *6.1 Current Rating. Current rating shall be evaluated in still air at 25°C ambient temperature. Under the following conditions, the temperature rise shall be no greater than 30°C:*
> *All contacts powered at 0.5 amp*
> *One contact powered at 3.0 amps*

Most of the signals are TTL (or some low current variation such as LVDS), so you can reasonably expect to get up to 3 amps per power pin through this connector. Remember that the +3.3V and +1.5V power supplies are limited to 5 amps total—the memories, the FPGA and the clock circuitry on the DN5000k10 consume +3.3V. The FPGA only consumes +1.5V.

If you use the DN5000k10 stand-alone (meaning that it is *not* plugged into a PCI slot), the auxiliary power connector has +5V and +12V, but does not have –12V. So unless you provide –12V to the DN5000k10 via another connection, –12V will not be available for use by a daughter card.

NOTE: –12V is not required by the DN5000k10. The DN5000k10 will operate normally without a –12V power supply.

Some of the interconnect with the 200-pin connectors is shared with the memories. See Figure 5-1 on page 5-2 to see which signals are shared. The 200-pin connectors are shown in Figure 8-5.

Notes:

- The signal labeled HDR_CLKOUT on **J9** is connected to a Roboclock input, and may be used to supply a clock to the FPGAs. The same pin on **J10** and **J16** is connected to a Roboclock output, so if a daughtercard is built to supply a clock output to HDR_CLKOUT it should not be put on **J10** or **J16**. All other clock, data, and power pin locations are the same on each header, so a daughtercard designed for **J10** can be put on **J16** instead with no problems. Cards designed for **J9** may be built with a jumper between their clock source and the HDR_CLKOUT pin, in which case they can be put on **J10** or **J16** if the jumper is disconnected.

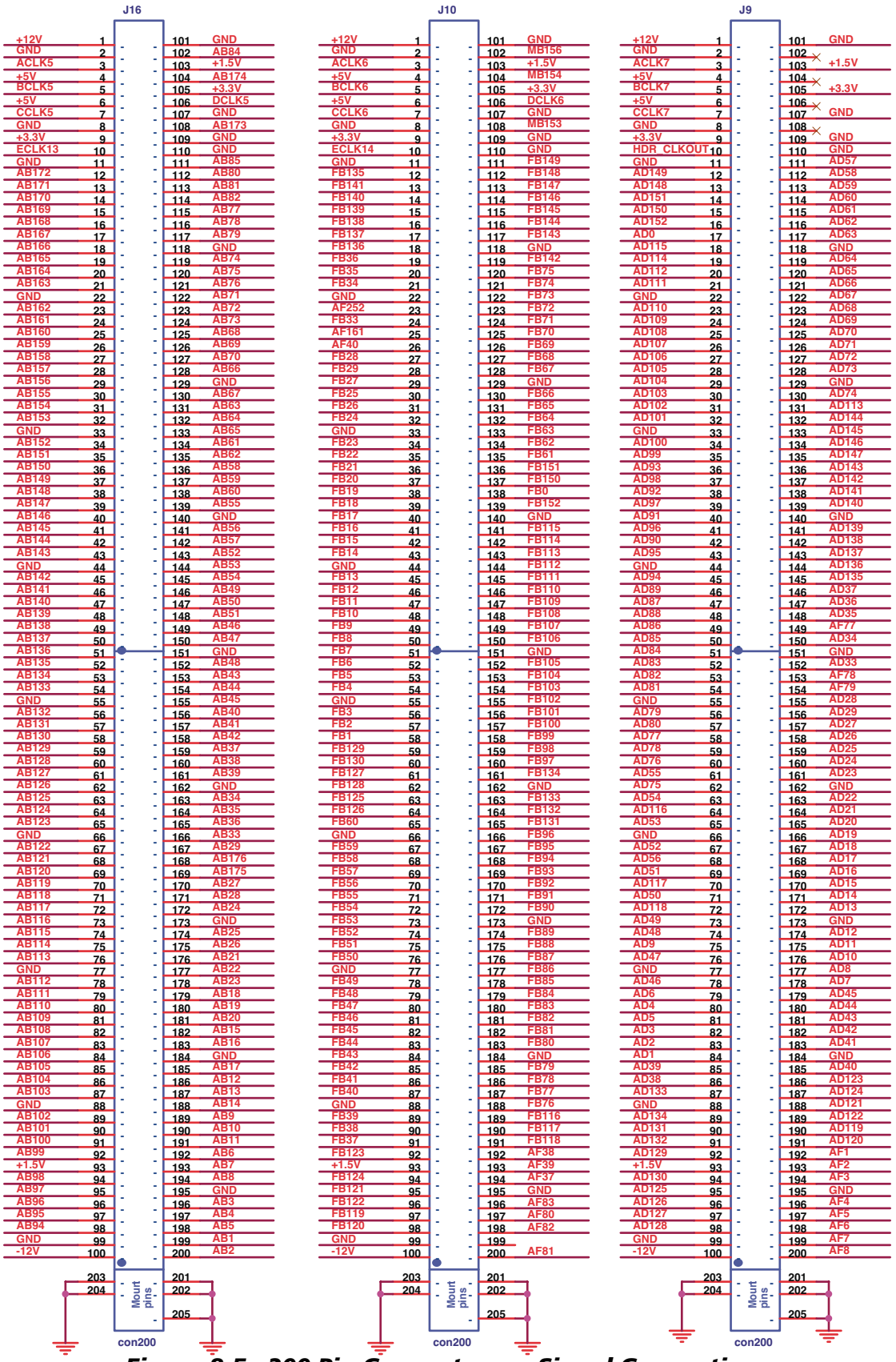


Figure 8-5 200 Pin Connectors — Signal Connections

Chapter 9

Utilities

PCI Debug—General Pontificating

Debugging of PCI-based hardware can be troublesome, so it is best to do so with a tiered approach. The following sequence of events needs to occur for a PCI-based peripheral to start working:

1. The hardware must boot itself at power-up — in the case of the DN5000k10:
   a. The µP must boot.
   b. Recognize the SmartMedia card.
   c. Configure the FPGA. (Hopefully, all this occurs before RST# on the PCI bus is deasserted.)
2. The PCI BIOS executes the PNP routines and configures the BARs on all PCI peripherals.
3. The operating system driver initializes the card.
4. The application initiates communication with the driver and the application executes.

The steps are dependant. Each of the steps must start and execute flawlessly before the next step occurs. When you get a PCI card for the first time, it is necessary to debug each step before attempting to go to the next. We provide utilities to help with each step.

Steps 1 and 2 are best done without an operating system in place. Windows NT-based systems take minutes to reboot after a crash (the-BLUE-screen-of-death) and an NT driver won't work unless the hardware is debugged. Since crashing is a regular occurrence in a PCI hardware debug environment, we find it easiest to do our debug and manufacturing test in the old DOS environment. Virtually all PCI peripherals get configured with addresses beyond the IM boundary. On a PC, C programs cannot access memory locations beyond IM unless special programs called DOS extenders are used. Several freeware DOS extenders are available. We use a free DOS-extender called DJGPP. More information can be found at **http://www.delorie.com/DJGPP**.

PC-Based—AETEST.EXE

A utility program called AETEST is provided with the DN5000k10. AETEST can be run under DOS, Windows 98/ME, Windows NT/2000, or LINUX. When used under DOS, you must boot your PC with a DOS disk. We ship one with the DN5000k10 in case you don't know how to make one on your own. All features work in the native mode of AETEST, which is DOS.

All source code for AETEST is provided, so you are welcome to customize the program to your own applications.

AETEST is not a stable program. We add and subtract features when we need to for debug and verification purposes, so don't be concerned if the screens that you see aren't exactly replicated here.

In a nutshell, AETEST lets you do the following:

- Determine if PCI recognizes the DN5000k10
- Read/write/loop any memory location
- Read/write/loop configuration space
- Display all configured PCI devices
- Display memory setting from any locations
- Fill memory with various patterns
- Run various tests on the DN5000k10
  - SSRAM Test
  - Multiplier Test
  - SDRAM Test
  - Interconnect Test
  - Daughter Card Test

AETEST Utility Installation Instructions

Installation Instructions for DOS

1. The files `aetestdj.exe` and `cwsdpmi.exe` (the DOS extender) need to be in the same directory.
2. Run `aetestdj.exe`.

Installation Instructions for Windows NT

1. Install the device driver: `install.exe` and `qldriver.sys` must be in the same directory.
2. Type "`install`"
3. After the driver is installed, start the driver by selecting **Control Panel–>Devices–>**find **"QLDriver"** –>click **"Start"**
4. Run `aetestnt.exe`.

Installation Instructions for Windows 2000

1. Install the device driver: `qldriver2000.inf` and the driver file (`qldriver.sys`) should be in the same directory.
2. Open **Control Panel**, click on "**Add/Remove Hardware**" and then go to "**Next–>**."
3. Choose **Add/Troubleshoot a device** (the default option) and click on "**Next–>**."
4. Wait until it finishes new hardware device searching; choose "**Add a new device**," and click on "**Next–>**."
5. Choose "**No, I want to select the hardware from a list**," and press "**Next–>**."
6. Choose "**Other Devices**" from **Hardware Types** list and press "Next–>"
7. Click on "**Have Disk…**"

8. In "**Copy Manufacturer's Files From:**" window, find the directory where `qldriver.sys` is located, then press "**OK**."
9. You should see "`dn2000k10 driver`" under **Models**; click on "**Next–>**."
10. Press "**Next–>**" and then "**Finish**."
11. Run `aetestnt.exe`.

Installation Instructions for LINUX

This has been tested on Red Hat Linux 7.2 (kernel version 2.4.x).

Note that all the text files, including the scripts, are DOS text format (with an extra carriage return character after every new line), so you need to convert them.

1. You must be root to start the driver and the program. "`dndev_load`" and "`dndev_unload`" are scripts that load and unload the driver; "`dndev.o`" is the driver file.
2. Load the driver; type "`sh dndev_load`"
3. Unload the driver; type "`sh dndev_unload`"
4. After driver is loaded, run the utility `aetest_linux`.
5. Note: You might need to run chmod on `aetest_linux` to make it executable: type "`chmod u+x aetest_linux`."

Installation Instructions for Solaris

The utility and driver are tested on Solaris 7.0/Sparc, with the 32-bit kernel.

Note that all the text files, including the scripts, are DOS text format (with an extra carriage return character after every new line), so you need to convert them.

1. To install the driver, go to the driver directory, make sure the driver file "`dndev`" is in the sparc sub-directory, and run "`sh dndev_uninstall.sh`"
2. To uninstall the driver, run "`sh dndev_uninstall.sh`"
3. To run the test utility, run "`aetest_solaris`" as root after the driver is loaded.

The driver is compiled with the gcc compiler.

`aetest_solaris` is compiled with "gmake." You can download it from the GNU website. The "make" from the Solaris installation does not work with our makefile format.

You may need to make `aetest_solaris` executable; run "`chmod u+x aetest_solaris`."

Installation Instructions for Windows 98/ME

There are two ways to run AETEST: You can run the DOS version "aetestdj.exe" directly, or you can run AETEST with a device driver.

To run AETEST with a device driver follow the steps below.

1. Choose a default PCI driver for the device. When Windows first starts with the device plugged in, it should ask for a device driver. Select **"Specify the location of the driver."**
2. Select **"Display a list of the drivers in a specific location…"**
3. Select **"Other devices."**
4. Under **"Manufacturers"** tab, select **"unknown device**."
5. Under "Models" select **"unsupported device**."
6. The driver file (pcifg.vxd) and aetest98.exe must be in the same directory. Run aetest98.exe.

NOTE: To re-compile the driver file pcicfg.vxd, you need the VtoolsD compiler from **www.numega.com**.

AETEST Options: Description and Definitions

Startup

When AETEST is first started, it tries to find a device that it recognizes. We have arbitrarily defined the DN5000k10 with a DEVICE_ID of 0x1250 or 0x1251 and a VENDOR_ID of 0x17DF. We have arbitrarily defined the DN5000k10S with a DEVICE_ID of 0x1240 and a VENDOR_ID of 0xABCD or 0x17DF. The DN2000k10 series has a DEVICE_ID of 0x1234, 0x1235, 0x1236 or 0x1237 and VENDOR_ID of 0xABCD. You should see the following screen if AETEST recognizes a DN5000k10 (Figure 9-1):

```
searching for "DN2000K10 Asic Emulator (1000E)" VENDOR_ID==abcd, DEVICE_ID==1236

searching for "DN2000K10 Asic Emulator (1600E)" VENDOR_ID==abcd, DEVICE_ID==1237

searching for "DN3000K10S Asic Emulator (6000)" VENDOR_ID==abcd, DEVICE_ID==1240

found device ---- vabcd, d1240 name="DN3000K10S Asic Emulator (6000)"
Configuration space:
00: 1240abcd      04: 0000001f
08: ff000047      0c: 00000000
10: fd800000      14: e0000000
18: 00000000      1c: 00000000
20: 00000000      24: 00000000
28: 00000000      2c: 90ab5678
30: 00000000      34: 00000000
38: 00000000      3c: 00000000
BAR0: base: 0xfd800000, size: 0x00800000
BAR1: base: 0xe0000000, size: 0x10000000
BAR2: base: 0x00000000, size: 0x00000000
BAR3: base: 0x00000000, size: 0x00000000
BAR4: base: 0x00000000, size: 0x00000000
BAR5: base: 0x00000000, size: 0x00000000
press any key
```

Figure 9-1 DN5000k10AETEST Startup Screen, DN5000k10 Recognized

Most of this initial display is debug information. The program is looking for a Vendor and Device ID that it recognizes, and finds `vendor=abcd` and `device=1240`, which is a DN5000k10 stuffed with a 2V60000. The lines after `Configuration space:` show what is in the configuration space and how the BARs are configured.

If AETEST does **not** see a PCI peripheral it recognizes, you will see the following (Figure 9-2):

```
searching for "Quad Sharc" VENDOR_ID==5045, DEVICE_ID==1
searching for "DN2000K10 Asic Emulator" VENDOR_ID==abcd, DEVICE_ID==1234
searching for "DN2000K10 Asic Emulator (2000E)" VENDOR_ID==abcd, DEVICE_ID==1235
searching for "DN2000K10 Asic Emulator (1000E)" VENDOR_ID==abcd, DEVICE_ID==1236
searching for "DN2000K10 Asic Emulator (1600E)" VENDOR_ID==abcd, DEVICE_ID==1237
searching for "DN3000K10S Asic Emulator (6000)" VENDOR_ID==abcd, DEVICE_ID==1240
searching for "ql5064 interface test" VENDOR_ID==1234, DEVICE_ID==5678
searching for "ql5064 64MB dram+LFSR" VENDOR_ID==1234, DEVICE_ID==5679
searching for "ql5064 PowerPC bridge" VENDOR_ID==11e3, DEVICE_ID==6
searching for "ql5064 PowerPC bridge (old VID/DID)" VENDOR_ID==1010, DEVICE_ID==5064
searching for "ql5064 AntiFuse test #1" VENDOR_ID==71f3, DEVICE_ID==2454
searching for "ql5064 AntiFuse test #2" VENDOR_ID==507, DEVICE_ID==2367
searching for "ql5064 AntiFuse test #3" VENDOR_ID==bc92, DEVICE_ID==2e6c
searching for "ql5064 AntiFuse test #4" VENDOR_ID==e125, DEVICE_ID==c38c
searching for "ql5064 AntiFuse test #5" VENDOR_ID==e62c, DEVICE_ID==ca76
searching for "ql5064 AntiFuse test #6" VENDOR_ID==448b, DEVICE_ID==e6a
searching for "ql5064 Emulated with 8051" VENDOR_ID==1243, DEVICE_ID==4321
searching for "Greg's PCI Device" VENDOR_ID==5143, DEVICE_ID==2
searching for "Cohu's PCI Device (sensor board)" VENDOR_ID==dead, DEVICE_ID==beef
searching for "LYNX 9610" VENDOR_ID==10b5, DEVICE_ID==9610

Didn't find known device in the following list:
   vendor_id=5045, device_id=1
   vendor_id=abcd, device_id=1234
   vendor_id=abcd, device_id=1235
   vendor_id=abcd, device_id=1236
   vendor_id=abcd, device_id=1237
   vendor_id=abcd, device_id=1240
   vendor_id=1234, device_id=5678
   vendor_id=1234, device_id=5679
   vendor_id=11e3, device_id=6
   vendor_id=1010, device_id=5064
   vendor_id=71f3, device_id=2454
   vendor_id=507, device_id=2367
   vendor_id=bc92, device_id=2e6c
   vendor_id=e125, device_id=c38c
   vendor_id=e62c, device_id=ca76
   vendor_id=448b, device_id=e6a
   vendor_id=1243, device_id=4321
   vendor_id=5143, device_id=2
   vendor_id=dead, device_id=beef
   vendor_id=10b5, device_id=9610

Hit a key to continue
```

Figure 9-2 AETEST Startup Screen, No PCI Peripheral Recognized

AETEST will still run, but many DINI product-specific options will not be available.

AETEST Main Screen

The AETEST Main Screen is shown in Figure 9-3.

```
              -=-= ASIC Emulator PCI Controller Driver =-=- v8

                      0) Read FPGA revision
                      1) PCI Menu
                      2) Memory Menu
                      3) Flash Menu
                      4) Clock Menu
                      5) Dedicated Multiplier Test

                      Q) Quit

                       -=-= PCI BASE ADDRESS =-=-
            0 : fd800000    1 : e0000000    2 : 00000000
            3 : 00000000    4 : 00000000    5 : 00000000



     Please select option:
```

Figure 9-3 AETEST Main Screen

Options

Read FPGA Revision. Display the revision ID of the FPGA. We will update the revision ID of the FPGA every time we change the reference design.

PCI Menu. Display the PCI utilities menu.

Memory Menu. Display the Memory Menu.

Flash Menu. Display the Flash Utilities Menu (DN2000k10 series only!).

Clock Menu. Display the Clock Utilities Menu.

Dedicated Multiplier Test. Execute the multiplier test.

Q. Quit and return to the DOS prompt

The selections are sometimes case sensitive, so be aware of the status of the CAPS LOCK on your keyboard. The base addresses for each of the configured BARs is displayed on all screens. You will need these addresses if you want to manually read and write to address locations within the PCI reference design. In this example (Figure 9-3, above), BAR0 is configured to `0xFD800000` and BAR1 is configured to `0xE0000000`. BAR[5:2] are not configured so they show up as `0x0`.

PCI Menu

The AETEST PCI menu is shown in Figure 9-4.

```
          -=-= ASIC Emulator PCI Controller Driver =-=- v8
      PCI Device/Function Num: 0x7F, 0x00

S) Set PCI Device Number
F) Set PCI Function Number
D) Display all Configured PCI Devices
1) Display Vendor and Device ID for PCI device-function: 7f-0
2) Loop on PCI device-fun: 7f-0 and Display Vendor and Device ID
3) Loop on PCI device-fun: 7f-0 and Don't Display Vendor and Device ID
4) Loop on all PCI device numbers and Display Device/Vendor ID's
5) Display all PCI information for PCI device-function: 7f-0
6) Write config dword
7) Read config dword

C) Configure BAR's from File
V) Save BAR Configuration to File
M) Main Menu
Q) Quit

                -=-= PCI BASE ADDRESS =-=-
        0 : fd800000    1 : e0000000    2 : 00000000
        3 : 00000000    4 : 00000000    5 : 00000000

Please select option:
```

Figure 9-4 AETEST PCI Menu

Set PCI Device Number. Sets a PCI device number of your choice as the "active" device (hex input). This option lists the available Device Numbers to help you match up your device IDDevice ID and Vendor ID with the device number.

Set PCI Function Number. Sets a PCI function number of your choice as the "active" function of a multi-function device (hex input). This option lists the Device ID and Vendor ID of each function within the "active" device number to help you to choose the desired function.

Display all Configured PCI Devices. Displays the PCI Device Numbers and corresponding Device ID and Vendor ID of all devices seen on the bus. This does not display device numbers with a Device ID and Vendor ID of all ones (`OxFFFF`).

Display Vendor and Device ID for PCI device-function.

Displays the Vendor ID and Device ID of the active device and function number. In the example above, this would display the Vendor ID and Device ID of the PCI device at device number `0x7F`, function number `0x00`.

Loop on PCI device-fun: 7f-0 and Display Vendor and Device ID. Reads and displays the Vendor ID and Device ID of the "active" device number and function number. Repeats this action until the user hits a key to stop it.

Loop on PCI device-fun: 7f-0 and Don't Display Vendor and Device ID. Same as previous menu option, except doesn't display results. This menu option is useful when using an oscilloscope to debug configuration reads.

Loop on all PCI device numbers and Display Device/Vendor ID's. Loops on each device number, reading the Vendor ID and Device ID for each. It moves onto the next device number when you press any key. That is, it continually reads the Vendor ID and Device ID from device number 0 until you hit a key, at which point it continually reads the Vendor ID and Device ID from device number 1. It moves all the way through device number 0 to device number `0x7F` (in case there are any bridges on your PCI bus).

Display all PCI information for PCI device-function: 7f-0.

Reads and displays all of the configuration space for the "active" device and function number. Use options "S" and "F" to change between the "active" device number and function number, and then use this option to view the entire configuration space.

Write config(uration) DWORD. Allows write to configuration space. The following text will appear to remind you what is in configuration space for a PCI device:

```
PCI_CS_VENDOR_ID                0x00
PCI_CS_DEVICE_ID                0x02
PCI_CS_COMMAND                  0x04
PCI_CS_STATUS                   0x06
PCI_CS_REVISION_ID              0x08
PCI_CS_CLASS_CODE               0x09
PCI_CS_CACHE_LINE_SIZE          0x0c
PCI_CS_MASTER_LATENCY           0x0d
PCI_CS_HEADER_TYPE              0x0e
PCI_CS_BIST                     0x0f
PCI_CS_BASE_ADDRESS_0           0x10
PCI_CS_BASE_ADDRESS_1           0x14
PCI_CS_BASE_ADDRESS_2           0x18
PCI_CS_BASE_ADDRESS_3           0x1c
PCI_CS_BASE_ADDRESS_4           0x20
PCI_CS_BASE_ADDRESS_5           0x24
PCI_CS_EXPANSION_ROM            0x30
PCI_CS_INTERRUPT_LINE           0x3c
PCI_CS_INTERRUPT_PIN            0x3d
PCI_CS_MIN_GNT                  0x3e
PCI_CS_MAX_LAT                  0x3f

Input config offset (hex 0x00-0xff):
word to write (in hex):

Loop indefinitely? (y or n)?
```

If looping was selected, any keypress will stop the loop.

Read config(uration) DWORD. Allows read from configuration space. Has options for single read, loop read with display, and loop read without display.

Configure BARs from File. Reloads the PCI configuration of the "active" device from a file. It writes 0x001F to the command register, and writes the 6 bars with the values from the file. This is useful for hot-swapping devices (power switch still required on extender), or reinitializing a device when its configuration has been altered.

WARNING: Because the PCI BIOS is not assigning the BARs for this device, you may induce a memory conflict by using this option. This option is for advanced users only!

Save Bar Configuration to File. Writes PCI Device ID, Vendor ID and the BARs into a file (from the "active" device). This option is for advanced users only!

Memory Menu

The memory menu (Figure 9-5) allows you to perform a variety of tests of PCI memory along with some DN5000k10 specific tasks.

```
 -=-= ASIC Emulator PCI Controller Driver =-=- v8

1) Write To Memory Test        2) Read Memory Test
3) Write/Read Test             4) Memory Fill
8) Memory Display
9) Write Memory Byte
a) Read Memory Byte
b) Write/Read Memory Byte
c) memory test on SSRAM 1
d) memory test on SSRAM 2
e) memory test on SSRAM 3
f) memory test on SDRAM
g) full memory test (including blockram)
n) memory test on FPGA block memory
p) bar memory range test
u) SRAM memory test
M) Main Menu                   Q) Quit

          -=-= PCI BASE ADDRESS =-=-
0 : fd800000    1 : e0000000    2 : 00000000
3 : 00000000    4 : 00000000    5 : 00000000
```

Figure 9-5 AETEST Memory Menu

Write to Memory Test. Write a selected number of long words to a specific PCI memory location (Figure 9-6).

```
                          --------
Memory location (hex)     :fb000000
Numbers of long words to write (in decimal) ?  2
long word to write (in hex) :aaaaaaaa
long word to write (in hex) :55555555

Loop indefinitely? (y or n)?

Hit a key to continue....
```

Figure 9-6 AETEST Write to Memory Test

You will be prompted for the memory location (in hex). The physical address is needed. All 4 gigabytes of PCI memory can be accessed. A minimum of 1 to a maximum of 1024 long words can be written, in sequential order, to the same address. A looping option is available if you want to use an oscilloscope. If you are in a scope loop, any keypress will terminate the loop and return you to the main menu.

Read Memory Test. Read a single long word from a specific PCI memory location (Figure 9-7).

```
              --------Input address :fb000000

fb000000
        1.Display result
        2.Display result and loop indefinitely
        3.Don't display result and loop indefinitely
Please select:
```

Figure 9-7 AETEST Read Memory Test

You will be prompted for the memory location (in hex). The physical address is needed. All 4 gigabytes of PCI memory can be read. Three options are available:

1. Read once and display.
2. Read indefinitely and display.
3. Read indefinitely and don’t display.

Write/Read Test. Write a long word to a specific PCI memory location and immediately read what was written. Repeat for a selected number of long words (Figure 9-8).

```
Input address :fb000000

Numbers of long words to write (in decimal) ?  2

long word to write (in hex) :000
long word to write (in hex) :aaa

        1.Display result
        2.Display result and loop indefinitely
        3.Don't display result and loop indefinitely
Please select:
```

Figure 9-8 AETEST Write/Read Test

You will be prompted for the memory location (in hex). The physical address is needed. All 4 gigabytes of PCI memory can be read. The program will prompt for the number of long words you with to write (1 to 1024). Three options are available:

1. Read once and display.
2. Read indefinitely and display.
3. Read indefinitely and don't display.

Option 3 is a very useful scope loop.

Memory Fill. Fill memory with a selected pattern (Figure 9-9).

```
Input starting address (hex and 32 bit aligned): fb000000

Input number of bytes (divisible by 4): 1000
1 -- Fill with 0
2 -- address=data
3 -- 0x55555555, 0xaaaaaaaa
4 -- 0xffffffff
5 -- data=~address
```

Figure 9-9 AETEST Memory Fill

You will be prompted for the memory location (in hex). The physical address is needed. All 4 gigabytes of PCI memory can be written. The program will prompt for the number of bytes (in hex) you wish to fill (`4` to `0xfffffffc`). The following fill options are available:

1. **fill with 0** — fill all the locations with `0x00000000` (clear the memory)
2. **address=data** — fill each long word with its address
3. alternating `0x55555555`, `0xAAAAAAAA`
4. `0xffffffff` — set all of memory
5. **data=~address** — fill each long word with the address (each bit inverted).

Memory Display. Display 160 long words of memory. You are prompted for the starting address (in hex):

```
Input starting address (hex and 32 bit aligned):
```

The following screen is displayed (Figure 9-10):

```
          0        4        8        c       10       14       18       1c
000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000020 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000040 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000080 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000a0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000c0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0000e0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000100 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000120 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000140 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000160 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000180 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001a0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001c0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
0001e0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000200 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000220 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000240 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000260 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
(f)orward  (b)ack  (j)ump  goto(0)  (q)uit
```

Figure 9-10 AETEST Memory Display

(f) forward — pages the screen forward in memory.
(b) back — pages the screen backwards in memory.
(j) jump — jump to a specific location (in hex).
(0) goto — jump back to the original address location specified at the beginning.
(d) delay and display loop — display, wait for a second, and display again. Loop until a key is struck.

Write Memory Byte. Write a specific number of bytes to a single memory address (Figure 9-11).

```
Input address :fb000000

Numbers of long words to write (in decimal) ?  2

byte to write (in hex) :ff
byte to write (in hex) :aa

        1.Display result
        2.Display result and loop indefinitely
        3.Don't display result and loop indefinitely
Please select:
```

Figure 9-11 AETEST Write Memory Byte

You will be prompted for the memory location (in hex). The physical address is needed. All 4 gigabytes of PCI memory can be accessed. A minimum of 1 to a maximum of 1024 bytes will be written, in sequential order, to the same address. A looping option is available if you want to use

an oscilloscope. If you are in a scope loop, any keypress will terminate the loop and return you to the main menu.

Read Memory Byte. Read a single byte from a specific PCI memory location (Figure 9-12).

```
Input address :fb000000

fb000000
        1.Display result
        2.Display result and loop indefinitely
        3.Don't display result and loop indefinitely
Please select:
```

Figure 9-12 AETEST Read Memory Byte

You will be prompted for the memory location (in hex). The physical address is needed. All 4 gigabytes of PCI memory can be read. Three options are available:

1. Read once and display.
2. Read indefinitely and display.
3. Read indefinitely and don't display.

Write/Read Memory Byte. Write and read a single DWORD from a specific PCI memory location. After entering a memory address (hex, 32 bits), you specify how many DWORDS you want written and read back, and the data. Then, you choose from the 3 options as above. The menu option does not perform any data checking. (Figure 9-13)

```
Numbers of long words to write (in decimal) ?  2

byte to write (in hex) :88888888
byte to write (in hex) :99999999

        1.Display result
        2.Display result and loop indefinitely
        3.Don't display result and loop indefinitely
Please select:
```

Figure 9-13 AETEST Write/Read Memory Byte

Memory test on SSRAM1. Tests one of the SSRAM chips on the DN5000k10.

Memory test on SSRAM2. Tests one of the SSRAM chips on the DN5000k10.

Memory test on SSRAM3. Tests one of the SSRAM chips on the DN5000k10.

Memory test on SDRAM. Tests the SDRAM chip on the DN5000k10.

Full Memory Test (Including BlockRAM). Tests all of the memories. This includes the SSRAM chips, the SDRAM, and the BlockRAM internal to the FPGA.

Memory test on FPGA block memory. Tests the BlockRAM inside the FPGA. On the DN2000k10, the BlockRAM is only in FPGA F.

BAR memory range test. Generic memory test that prompts the user for BAR number, starting address offset, DWORD count, and number of iterations. The user is also prompted if the program should stop if error occurs, or if the program should display any errors that occur. This allows for maximum flexibility when debugging a design with an oscilloscope, or debugging any memories or memory locations on your PCI bus. The memory test is very complete, performing a write then a read to every location, a read from every location, and then a read/write/read test to every location. All other memory test options listed in the memory menu are based on this generic memory test function.

Appendix A

Berg Connector Datasheets

Figure A-1 and Figure A-2 contain the schematics for the Berg 91403-003 Connector.

Figure A-3 through Figure A-5 contain the schematics for the Berg 91294-003 connector.

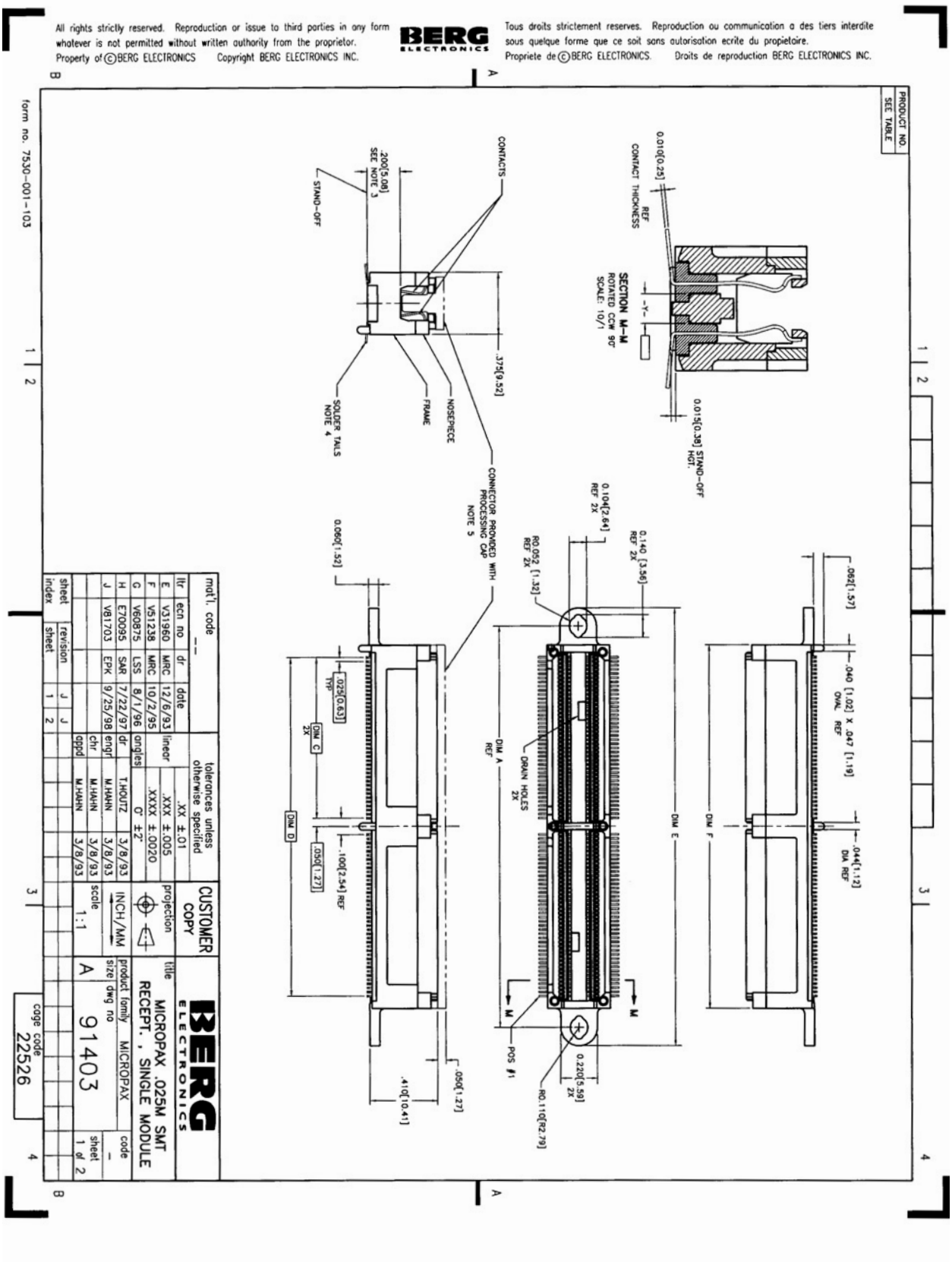


Figure A-1 Berg 91403-003 Datasheet Page 1 of 2

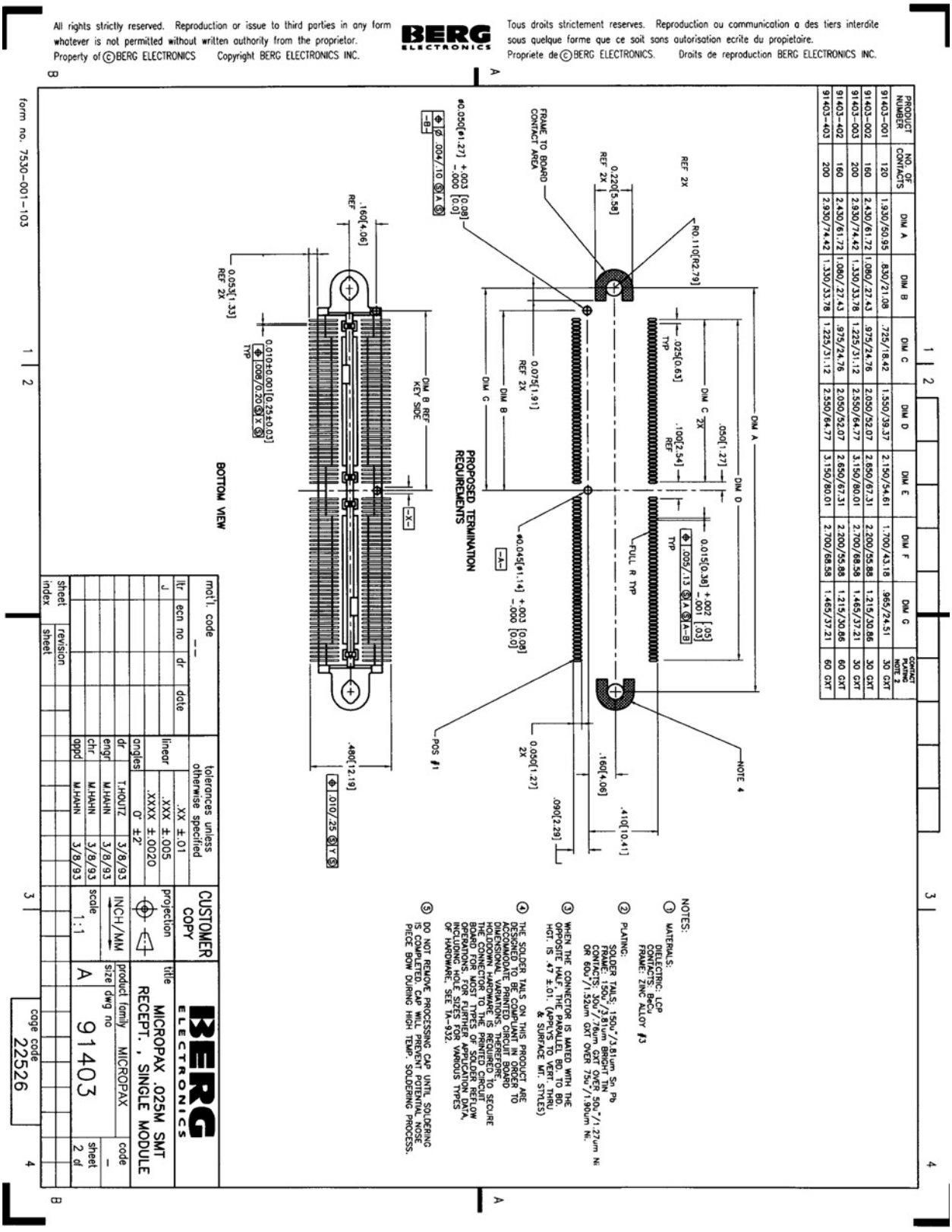


Figure A-2 Berg 91403-003 Datasheet Page 2 of 2

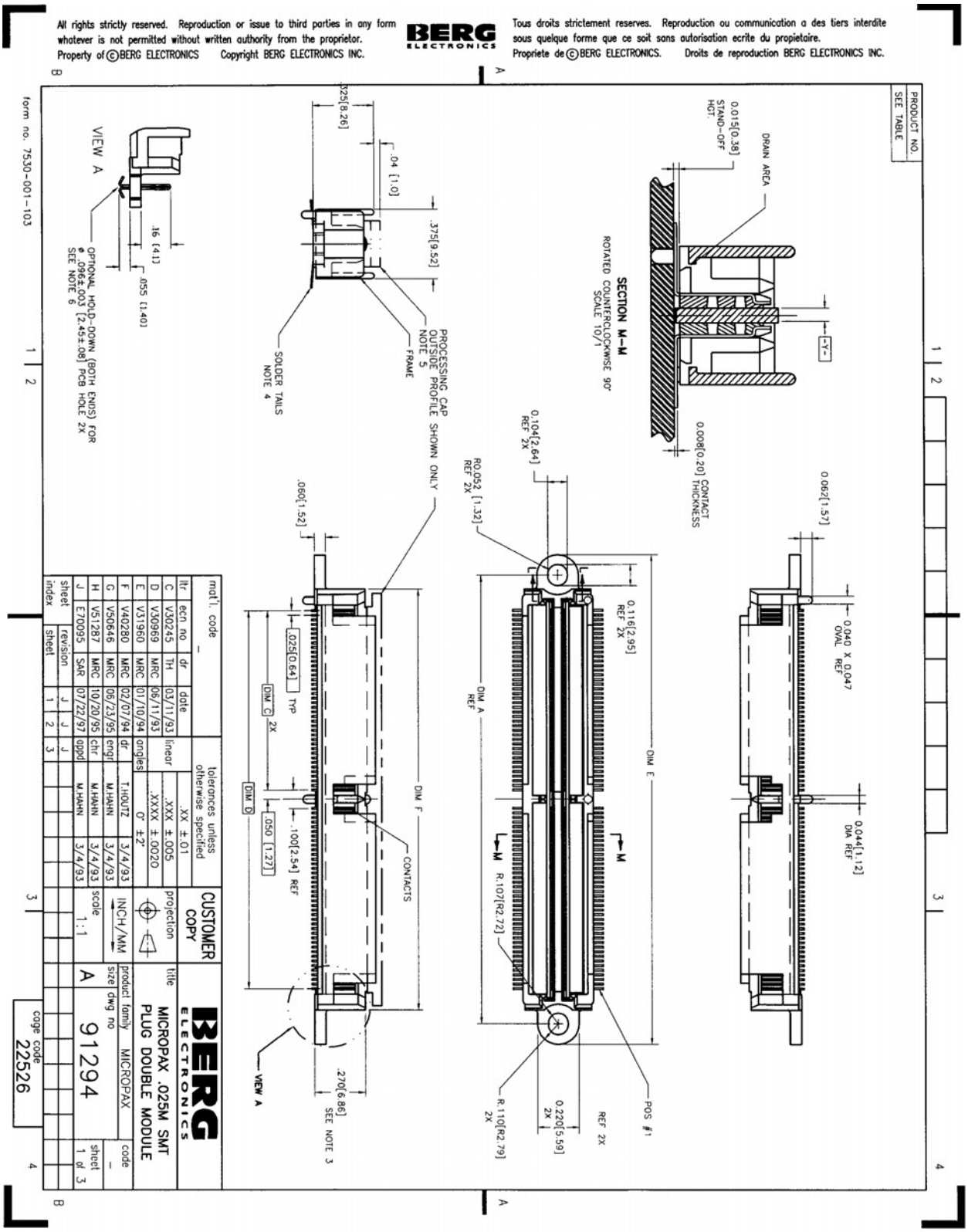


Figure A-3 Berg 91294-003 Datasheet Page 1 of 3

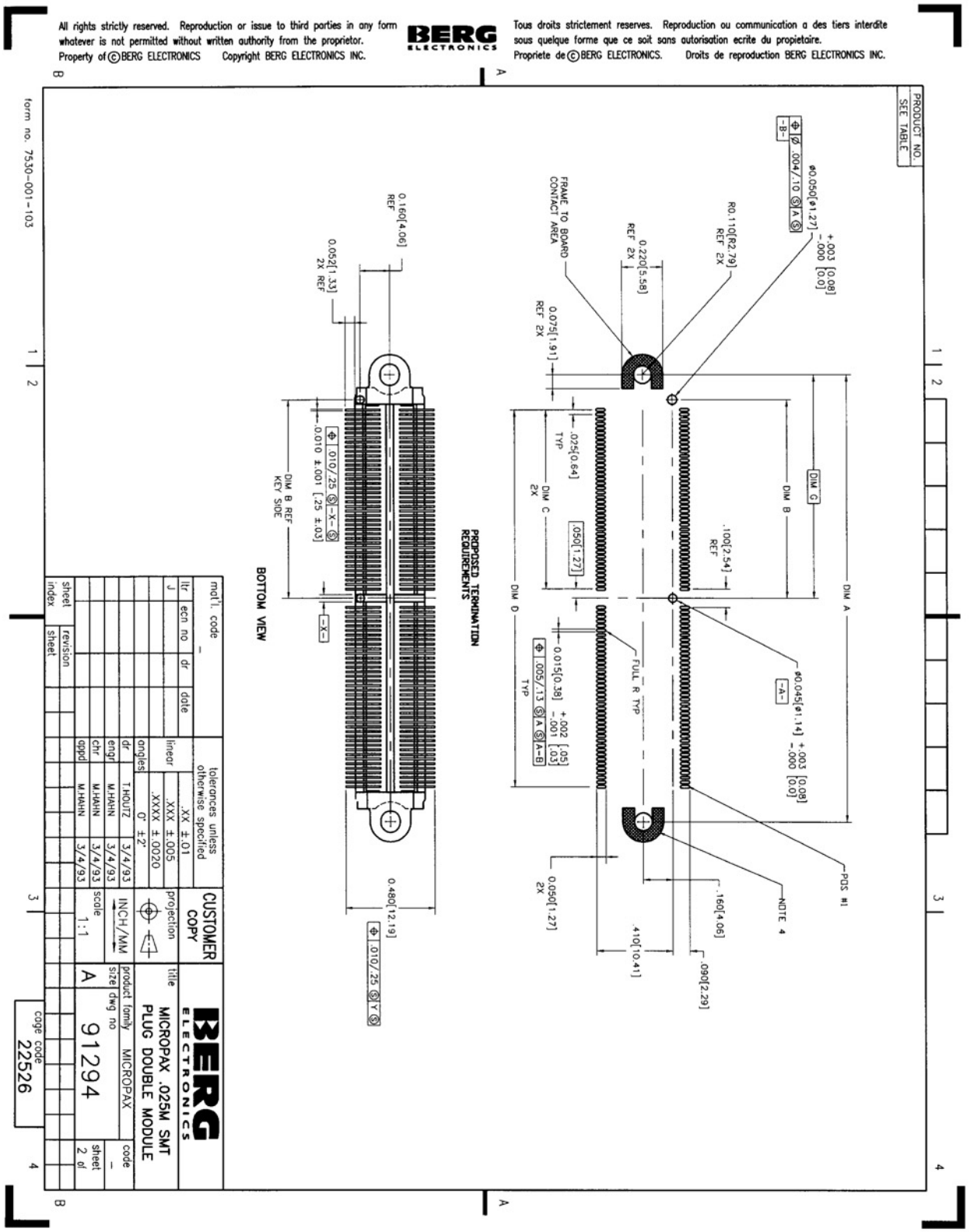
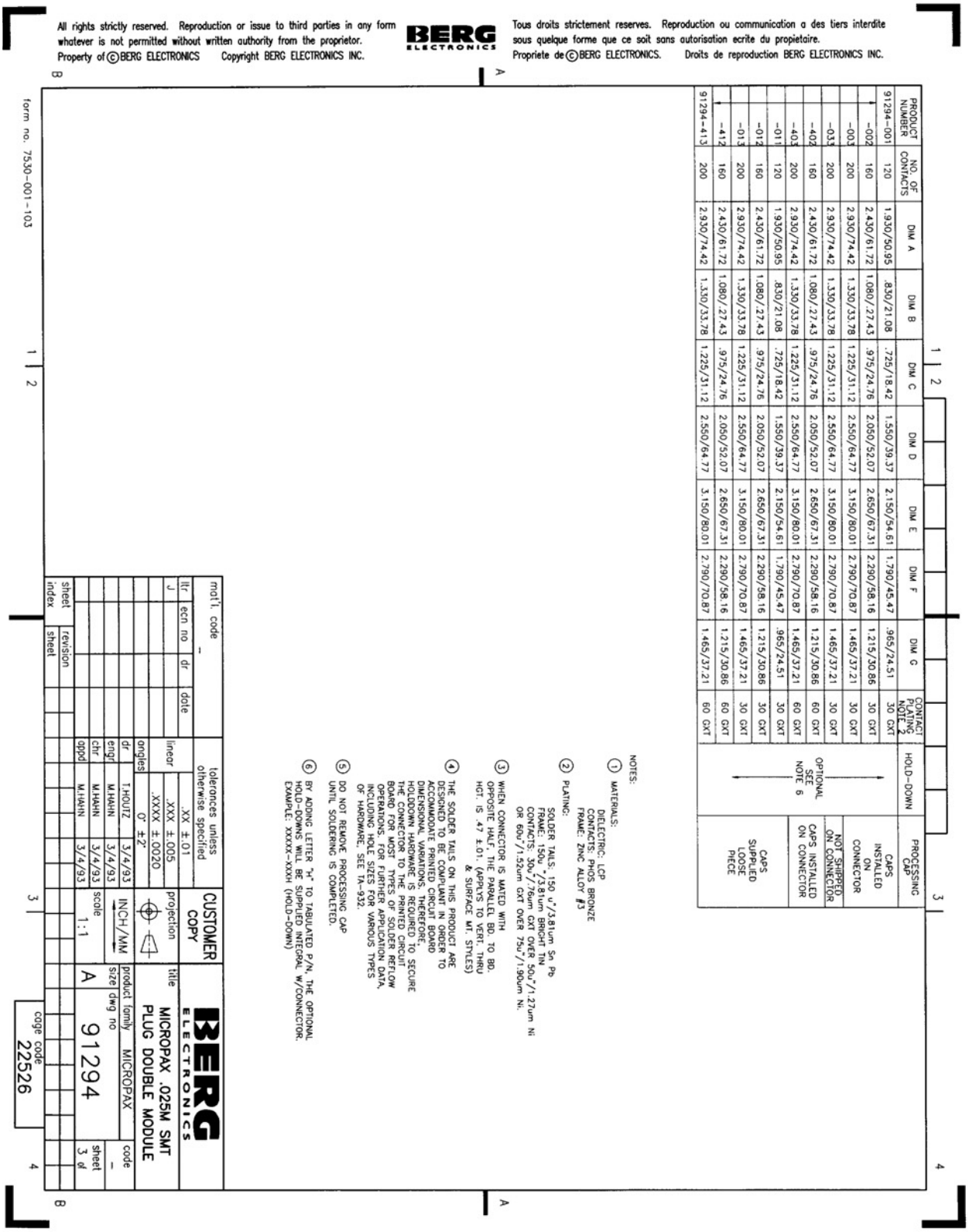


Figure A-4 Berg 91294-003 Datasheet Page 2 of 3



All rights strictly reserved. Reproduction or issue to third parties in any form whatever is not permitted without written authority from the proprietor.
Property of ©BERG ELECTRONICS Copyright BERG ELECTRONICS INC.

Tous droits strictement reserves. Reproduction ou communication a des tiers interdite sous quelque forme que ce soit sans autorisation ecrite du propietaire.
Propriete de ©BERG ELECTRONICS. Droits de reproduction BERG ELECTRONICS INC.

PRODUCT NUMBER	NO. OF CONTACTS	DIM A	DIM B	DIM C	DIM D	DIM E	DIM F	DIM G	CONTACT PLATING NOTE 2	HOLD-DOWN	PROCESSING CAP
91294-001	120	1.930/50.95	.830/21.08	.725/18.42	1.550/39.37	2.150/54.61	1.790/45.47	.965/24.51	30 GXT	OPTIONAL SEE NOTE 6	CAPS INSTALLED ON CONNECTOR
-002	160	2.430/61.72	1.080/27.43	.975/24.76	2.050/52.07	2.650/67.31	2.290/58.16	1.215/30.86	30 GXT		
-003	200	2.930/74.42	1.330/33.78	1.225/31.12	2.550/64.77	3.150/80.01	2.790/70.87	1.465/37.21	30 GXT		NOT SHIPPED ON CONNECTOR
-033	200	2.930/74.42	1.330/33.78	1.225/31.12	2.550/64.77	3.150/80.01	2.790/70.87	1.465/37.21	30 GXT		
-402	160	2.430/61.72	1.080/27.43	.975/24.76	2.050/52.07	2.650/67.31	2.290/58.16	1.215/30.86	60 GXT		CAPS INSTALLED ON CONNECTOR
-403	200	2.930/74.42	1.330/33.78	1.225/31.12	2.550/64.77	3.150/80.01	2.790/70.87	1.465/37.21	60 GXT		
-011	120	1.930/50.95	.830/21.08	.725/18.42	1.550/39.37	2.150/54.61	1.790/45.47	.965/24.51	30 GXT		CAPS SUPPLIED LOOSE PIECE
-012	160	2.430/61.72	1.080/27.43	.975/24.76	2.050/52.07	2.650/67.31	2.290/58.16	1.215/30.86	30 GXT		
-013	200	2.930/74.42	1.330/33.78	1.225/31.12	2.550/64.77	3.150/80.01	2.790/70.87	1.465/37.21	30 GXT		
-412	160	2.430/61.72	1.080/27.43	.975/24.76	2.050/52.07	2.650/67.31	2.290/58.16	1.215/30.86	60 GXT		
91294-413	200	2.930/74.42	1.330/33.78	1.225/31.12	2.550/64.77	3.150/80.01	2.790/70.87	1.465/37.21	60 GXT		

NOTES:

1. MATERIALS:
   DIELECTRIC: LCP
   CONTACTS: PHOS BRONZE
   FRAME: ZINC ALLOY #3
2. PLATING:
   SOLDER TAILS: 150 u"/3.81um Sn Pb
   FRAME: 150u"/3.81um BRIGHT TIN
   CONTACTS: 30u"/.76um GXT OVER 50u"/1.27um Ni
   OR 60u"/1.52um GXT OVER 75u"/1.90um Ni.
3. WHEN CONNECTOR IS MATED WITH OPPOSITE HALF, THE PARALLEL BD. TO BD. HGT. IS .47 ±.01. (APPLYS TO VERT. THRU & SURFACE MT. STYLES)
4. THE SOLDER TAILS ON THIS PRODUCT ARE DESIGNED TO BE COMPLIANT IN ORDER TO ACCOMMODATE PRINTED CIRCUIT BOARD DIMENSIONAL VARIATIONS. THEREFORE, HOLDDOWN HARDWARE IS REQUIRED TO SECURE THE CONNECTOR TO THE PRINTED CIRCUIT BOARD FOR MOST TYPES OF SOLDER REFLOW OPERATIONS. FOR FURTHER APPLICATION DATA, INCLUDING HOLE SIZES FOR VARIOUS TYPES OF HARDWARE, SEE TA-932.
5. DO NOT REMOVE PROCESSING CAP UNTIL SOLDERING IS COMPLETED.
6. BY ADDING LETTER "H" TO TABULATED P/N, THE OPTIONAL HOLD-DOWNS WILL BE SUPPLIED INTEGRAL W/CONNECTOR.
   EXAMPLE: XXXXX-XXXH (HOLD-DOWN)

form no. 7530-001-103

mat'l. code –

ltr	ecn no	dr	date
J			

sheet index

revision sheet	

tolerances unless otherwise specified

linear	.XX ±.01
	.XXX ±.005
	.XXXX ±.0020
angles	0° ±2°

dr	T.HOUTZ	3/4/93
engr	M.HAHN	3/4/93
chr	M.HAHN	3/4/93
appd	M.HAHN	3/4/93

CUSTOMER COPY

projection

scale 1:1

BERG ELECTRONICS

title MICROPAX .025M SMT PLUG DOUBLE MODULE

product family MICROPAX

code –

size A

dwg no 91294

sheet 3 of

cage code 22526

Figure A-5 Berg 91294-003 Datasheet Page 3 of 3

Glossary and Acronyms

*µ*P	microprocessor
BAR	Base Address Register
BGA	ball grid array
BIOS	Basic Input/Output Services
CMOS	complementary metal-oxide semiconductor
CPLD	Complex Programmable Logic Device
CSF	Configuration Settings File
DSP	digital signal processing
EEPROM	Electrically Erasable PROM
EIA	Electronic Industries Association
ESD	Electro-Static Discharge
FAQ	frequently asked questions
FAT	file allocation table
FPGA	field programmable gate array
FT	flowthrough
HDL	Hardware Description Language
I/O	input/output
IP	intellectual property
LED	light emitting diode
LSI	large scale integration
LVCMOS	low voltage complementary metal-oxide semiconductor
LVDS	Low-Voltage Differential Signaling
LVTTL	low voltage transistor-transistor logic
MDR	Mini D Ribbon
PCI	peripheral component interconnect
PCI-X	peripheral component interconnect (extended)
PL	pipelined
PLL	phase lock loop
PNP	plug-and-play
PWB	printed wire board
RBF	Raw Binary File
RISC	reduced instruction set computer
SDRAM	synchronous dynamic random access memory
SRAM	shadow random access memory
SSRAM	synchronous static random access memory
TTL	transistor-transistor logic
VHDL	VHSIC Hardware Description Language
VREF	reference voltage
ZBT	zero-bus-turnaround

Index

Symbols

A

B

C

D

Index (Continued)

E

F

G

H

I

J

Index (Continued)

L

M

O

P

Index (Continued)

Q

R

S

Index (Continued)

T

V

Index (Continued)