

# Navigator™ Motion Processor

---

## PCI Developer's Kit Manual



Performance Motion Devices, Inc.  
55 Old Bedford Rd  
Lincoln, MA 01773

Revision 1.1, October 2002

## **NOTICE**

This document contains proprietary and confidential information of Performance Motion Devices, Inc., and is protected by federal copyright law. The contents of this document may not be disclosed to third parties, translated, copied, or duplicated in any form, in whole or in part, without the express written permission of PMD.

The information contained in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form, by any means, electronic or mechanical, for any purpose, without the express written permission of PMD.

Copyright 1998, 1999, 2000 by Performance Motion Devices, Inc.  
**Navigator™** and **C-Motion™** are trademarks of Performance Motion Devices, Inc

## **Warranty**

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

Performance Motion Devices, Inc. (PMD) reserves the right to make changes to its products or to discontinue any product or service without notice, and advises customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

## **Safety Notice**

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage. Products are not designed, authorized, or warranted to be suitable for use in life support devices or systems or other critical applications. Inclusion of PMD products in such applications is understood to be fully at the customer's risk.

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

## **Disclaimer**

PMD assumes no liability for applications assistance or customer product design. PMD does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of PMD covering or relating to any combination, machine, or process in which such products or services might be or are used. PMD's publication of information regarding any third party's products or services does not constitute PMD's approval, warranty or endorsement thereof.



## Related Documents

---

### **Navigator Motion Processor User's Guide (MC2000UG)**

How to set up and use all members of the Navigator Motion Processor family.

### **Navigator Motion Processor Programmer's Reference (MC2000PR)**

Descriptions of all Navigator Motion Processor commands, with coding syntax and examples, listed alphabetically for quick reference.

### **Navigator Motion Processor Technical Specifications**

Five booklets containing physical and electrical characteristics, timing diagrams, pinouts, and pin descriptions of each series:

MC2100 Series, for brushed servo motion control (MC2100TS);

MC2300 Series, for brushless servo motion control (MC2300TS);

MC2400 Series, for microstepping motion control (MC2400TS);

MC2500 Series, for stepping motion control (MC2500TS);

MC2800 Series, for brushed servo and brushless servo motion control (MC2800TS).

### **Navigator Motion Processor Developer's Kit Manual (DK2000M)**

How to install and configure the DK2000 developer's kit PC board.



# Table of Contents

---

Warranty.....	iii
Safety Notice .....	iii
Disclaimer.....	iii
Related Documents.....	v
Table of Contents.....	vii
<b>1 Installation .....</b>	<b>9</b>
1.1 Installation Sequence.....	9
1.2 Components List.....	10
1.3 Required Hardware.....	10
1.4 Preparing the Board for Installation .....	10
1.4.1 Setting the Host I/O Mode.....	11
1.5 PCI DK2100 Connections Summary.....	11
1.6 PCI DK2300 Connections Summary.....	12
1.7 PCI DK2400 Connections Summary.....	13
1.8 PCI DK2500 Connections Summary.....	13
1.9 PCI DK2800 Connections Summary.....	14
1.10 Applying Power.....	15
1.11 Software Installation.....	15
1.12 First Time System Verification .....	15
1.12.1 Step #1: Set the Motor Amplifier Type .....	16
1.12.2 Step #2: Initialize the Commutation .....	17
1.12.3 Step #3: Check Commutation .....	18
1.12.4 Step #4: Set Filter Parameters.....	18
1.12.5 Step #5: Set the Motor Command.....	19
1.12.6 Step #6: Make a Trajectory Move .....	19
<b>2 Using Pro-Motion™ .....</b>	<b>20</b>
2.1 Communication .....	20
2.1.1 Parallel interface.....	21
2.1.2 Serial interface.....	21
2.2 Motion .....	22
2.2.1 Step #1 Amplifier Type .....	22
2.2.2 Step #2 Initialize Commutation .....	22
2.2.3 Step #3 Check Commutation .....	23
2.2.4 Step #4 Set Filter Parameters.....	23
2.2.5 Step #5 Motor Output Power .....	23
2.2.6 Step #6 Start motion .....	23
2.3 Command Window.....	24
2.3.1 Commands Available within the Command Window .....	26
<b>3 Developing Your Own Applications with C-Motion.....</b>	<b>27</b>

<b>4 PCI DK1 Electrical Reference.....</b>	<b>32</b>
4.1 PCI DK1 Layout.....	32
4.2 PCI DK1 Connectors.....	33
4.2.1 Serial Communications Connector (J1).....	33
4.2.2 Motion Peripherals Connector (J4).....	33
4.2.3 Connector Pin Layout.....	36
4.2.4 User-defined Digital I/O Connector (J5).....	37
4.2.5 Analog Input Connector (J3).....	37
4.2.6 Synch feature jumper (JP3).....	38
4.3 Host I/O Mode.....	39
4.3.1 PCI DK1 Configuration Jumpers and Switch Blocks.....	39
4.3.2 Configuration Jumper JP11.....	39
4.4 Serial Port.....	39
4.4.1 DIP Switch Block S1 and S2.....	39
4.4.2 S1: Transmission Parameters.....	39
4.4.3 S2: Serial Device Address.....	40
4.5 Outputs to Motor Amplifiers.....	41
4.5.1 Brushed Servo Motors (MC2100/2800 Series).....	41
4.5.2 Brushless Servo Motors (MC2300/2800 Series).....	41
4.5.3 Microstepping Motors (MC2400 Series).....	41
4.6 Encoder Inputs.....	42
<b>5 PCI DK1 Hardware Information.....</b>	<b>43</b>
5.1 Environmental and Electrical Ratings.....	43
5.2 PLX PCI Registers.....	43
5.3 PCI DK1 Schematics.....	44
5.3.1 PCI DK1 Assembly Drawing.....	44
5.3.2 PCI DK1 Overview Schematic.....	46
5.3.3 PCI DK1 CP and DPRAM Schematic.....	48
5.3.4 PCI DK1 I/O Schematic.....	50
5.3.5 PCI DK1 DAC Amplifiers Schematic.....	52
5.3.6 PCI DK1 Connector and Quadrature Schematic.....	54
5.3.7 PCI DK1 PLD Schematic.....	56
5.3.8 PCI DK1 Buffer Schematic.....	58
5.3.9 PCI DK1 PLX9030 Schematic.....	60
5.3.10 PCI DK1 PCI Edge Connector Schematic.....	62



# 1 Installation

---

The PMD Navigator motion processor developer's kit is an integrated board/software package that serves as an electrical and software design tool for building a Navigator-based system. The developer's kit supports all members of the Navigator motion processor family, as shown below.

Dev. Kit P/N	Navigator motion processors supported	Installed Motion processor
PCI DK2100	MC2140, MC2120, MC2110	MC2140
PCI DK2300	MC2340, MC2320, MC2310	MC2340
PCI DK2400	MC2440, MC2420, MC2410	MC2440
PCI DK2500	MC2540, MC2520, MC2510	MC2540
PCI DK2800	MC2840, MC2820	MC2840

All of the above developer's kit versions share the same physical PC card (called the PCI DK1). They differ in the specific type of chip that is installed in the motion card.

Because the installed chip for each Developer's kit version supports the maximum number of axes allowed, it can be used to develop systems based on motion processors with fewer axes simply by disabling the unused axes. Alternatively it is possible to have the developer's kit shipped with an alternate motion processor from the same family version (one with two or one axes). See your PMD representative for details.

In addition to the PC card the developer's kit also includes a diskette with the C-Motion™ source code library. C-Motion is a full-featured C language library, which simplifies the development of motion applications for the Navigator motion processors. Also included on the diskette is Pro-Motion, an interactive monitor program that supports all of the Navigator commands and motion processors. Pro-Motion is useful to test and exercise your system before you have started writing your own software.

## 1.1 Installation Sequence

For a normal installation of the developer's kit, you will need to configure the PCI DK1 board for the PC system and motor hardware that you will connect it to. Configuration of the PCI DK1 board is described in detail in section 1.4, "Preparing the board for installation."

Next you will need to connect your system's motors, encoders, amplifiers, and sensors as desired to operate your motion hardware. A description of the connections that are made for the various Navigator motion processors is found in section 1.5, DK2x00 Connection Summary.

Once this hardware configuration is complete, you should then install the software. Installation of the software is described in section 1.11, Software Installation.

The final step to finish the installation is to perform a functional test of the finished system. This is described in section 1.12, First Time System Verification.

Once all of the above has been accomplished installation is complete. You can now exercise your motion system, or simply use the developer's kit as a test bed while building your own board and software code.

## 1.2 Components List

The Navigator developer's kit contains the following components:

- 1) PCI DK1 4-axis PC Board
- 2) CD-ROM containing software and documentation in PDF format
- 3) 100 pin connector to dual 50-pin header converter cable
- 4) Documentation:
  - Navigator Motion Processor Developer's Kit Manual
  - Navigator Motion Processor User's Guide
  - Navigator Motion Processor Programmers Reference
  - Navigator Motion Processor Technical Reference (for either MC2100, MC2300, MC2400, MC2500, or MC2800 series of motion processors depending on the specific developer's kit you purchased)

If any of these components are missing, please contact PMD directly, or your PMD representative.

## 1.3 Required Hardware

To install the developer's kit, you will need the following hardware:

- 1) PC platform: the minimum platform consists of an Intel (or compatible) processor, 80486 or better, one available PCI slot, 5MB of available disk space, 16MB of available RAM, and a CDROM drive. The recommended platform is an Intel (or compatible) processor, Pentium or better, one available PCI slot, 5MB of available disk space, 32MB of available RAM, and a CDROM drive. The PC operating system required is Windows 9X/ME/NT/2000/XP. An asynchronous serial communications port is optional for both the minimum and recommended platforms.
- 2) 1 to 4 pulse and direction, PWM, or analog-input amplifiers. The type of amplifier depends on the developer's kit motion processor type.
- 3) 1 to 4 step motors or servo motors. These motors may or may not provide encoder position feedback signals depending on the type of motion processor being used.
- 4) Additional connectors as required to connect the PCI DK1 PC board to the amplifiers and the servo motors. Dual male 50 pin header-type connectors will be needed to interface to the PCI DK1 board's signal cable.

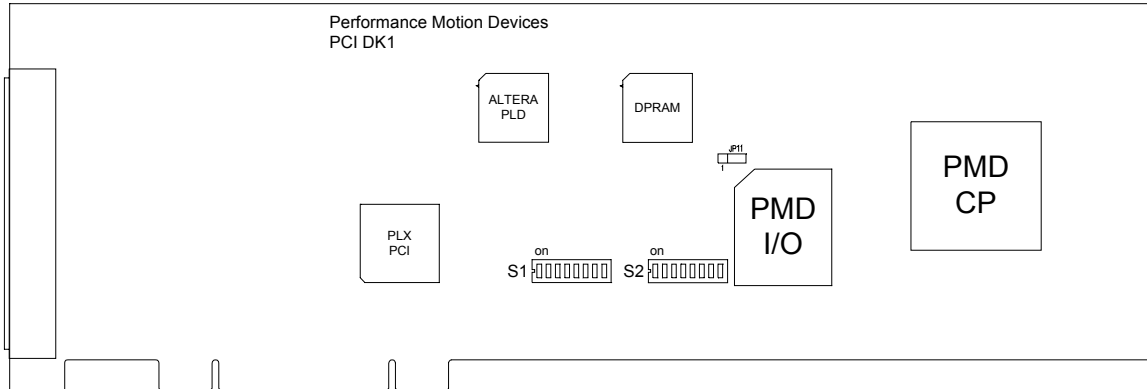
## 1.4 Preparing the Board for Installation

The board provides the following user-settable hardware options:

Option	Set using	Default
Host interface mode	jumper JP11	16/16 parallel

The host interface mode jumpers will not need to be changed unless it is desired that the card be operated in serial mode. If operated in serial mode, switch banks S1 and S2 set the serial configuration, which is described in section 4.4.1.

The following diagram shows the location of the switch banks S1, S2 and jumper JP11:



**Figure 1-1. Settable jumpers and switch blocks**

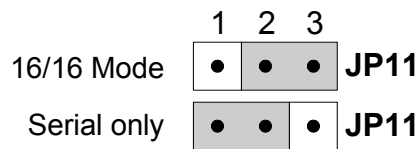
If you need to change the default setting values from the table above, or are not sure if they need to be changed, the following sections explain more about these settings.

### 1.4.1 Setting the Host I/O Mode

The PCI DK1 supports two different communication modes. This is shown in the table below:

Mode	Description
16-bit mode (16/16)	The motion processor accepts instructions and data as full 16-bit words, using the entire 16-bit data path.
Serial port	The motion processor accepts instructions through an asynchronous serial port.

The figure below shows how the JP11 jumper should be installed to select the host mode. 16/16-bit mode is the default. Shading indicates the location of the jumper.



## 1.5 PCI DK2100 Connections Summary

The following table summarizes the connections provided and expected by the PCI DK1 PC board when an MC2140 motion processor is installed. Although the MC2140 supports up to four axes, any number of axis between 1 and 4 may be connected.

Motion Processor:	MC2100 Series (brushed servo)
Maximum # of Axes:	4
Encoder Input Type:	Incremental encoder
Encoder Input Signals: (per axis)	A quadrature channel input B quadrature channel input Index pulse channel input
# Motor Output Channels (per axis:)	1
Amplifier Output Signals (per axis, if PWM sign, magnitude used)	PWM Direction PWM magnitude
Amplifier Output Signals (per axis, if PWM 50/50 used)	PWM magnitude
Amplifier Output Signals (per axis, if analog output used)	Analog out (DAC output)
Other Control Signals: (per axis)	Home signal channel input Positive limit switch input Negative limit switch input AxisIn input AxisOut output
Miscellaneous Signals:	GND +5 V (for encoder power)

For a complete description of the PC card connectors and interfacing requirements see Section 4, PCI DK1 Electrical Reference.

## 1.6 PCI DK2300 Connections Summary

The following table summarizes the connections provided and expected by the PCI DK1 PC board when an MC2340 motion processor is installed. Although the MC2340 supports up to four axes any number of axis between 1 and 4 may be connected.

Motion Processor	MC2300 Series (Brushless Servo)
Maximum # of Axes:	4
Encoder Input Type:	Incremental encoder
Encoder Input Signals: (per axis)	A quadrature channel input B quadrature channel input Index pulse channel input
# motor output channels (per axis:)	2 or 3 depending on motor output selected and # phases
Amplifier Output Signals (per axis, if PWM 50/50 used)	PWM magnitude (phase A) PWM magnitude (phase B) PWM magnitude (phase C)
Amplifier Output Signals (per axis, if analog output used)	Analog out (phase A) Analog out (phase B)
Hall inputs:	Hall (phase A) Hall (phase B) Hall (phase C)

Other Control Signals: (per axis)	Home signal channel input Positive limit switch input Negative limit switch input AxisIn input AxisOut output
Miscellaneous Signals:	GND +5 V (for encoder power)

For a complete description of the PC card connectors and interfacing requirements see Section 4, PCI DK1 Electrical Reference.

## 1.7 PCI DK2400 Connections Summary

The following table summarizes the connections provided and expected by the PCI DK1 PC board when an MC2440 motion processor is installed. Although the MC2440 supports up to four axes, any number of axis between 1 and 4 may be connected.

Motion Processor:	MC2400 Series (microstepping)
Maximum # of Axes:	4
Encoder Input Type:	Incremental encoder (optional)
Encoder Input Signals: (per axis)	A quadrature channel input B quadrature channel input Index pulse channel input
# Motor Output Channels (per axis:)	2 or 3 depending on motor output selected and # phases
Amplifier Output Signals (per axis, if PWM sign, magnitude used)	PWM Direction PWM magnitude
Amplifier Output Signals (per axis, if PWM 50/50 used)	PWM magnitude
Amplifier Output Signals (per axis, if analog output used)	Analog out (DAC output)
Other Control Signals: (per axis)	Home signal channel input Positive limit switch input Negative limit switch input AxisIn input AxisOut output
Miscellaneous Signals:	GND +5 V (for encoder power)

For a complete description of the PC card connectors and interfacing requirements see Section 4, PCI DK1 Electrical Reference.

## 1.8 PCI DK2500 Connections Summary

The following table summarizes the connections provided and expected by the PCI DK1 PC board when an MC2540 motion processor is installed. Although the MC2540 supports up to four axes, any number of axis between 1 and 4 may be connected.

Motion Processor:	MC2500 Series (stepping)
Maximum # of Axes:	4
Encoder Input Type:	Incremental encoder (optional)
Encoder Input Signals: (per axis)	A quadrature channel input B quadrature channel input Index pulse channel input
Amplifier Output Signals:	Pulse Direction
Other Control Signals: (per axis)	AtRest signal output Home signal channel input Positive limit switch input Negative limit switch input AxisIn input AxisOut output
Miscellaneous Signals:	GND +5 V (for encoder power)

For a complete description of the PC card connectors and interfacing requirements see Section 4, PCI DK1 Electrical Reference.

## 1.9 PCI DK2800 Connections Summary

The following table summarizes the connections provided and expected by the PCI DK1 PC board when an MC2840 motion processor is installed. Although the MC2340 supports up to four axes any number of axis between 1 and 4 may be connected.

Motion Processor	MC2800 Series (Brushed or Brushless Servo)
Maximum # of Axes:	4
Encoder Input Type:	Incremental encoder
Encoder Input Signals: (per axis)	A quadrature channel input B quadrature channel input Index pulse channel input
# motor output channels (per axis:)	1, 2 or 3 depending on motor output selected and # phases
Amplifier Output Signals (per axis, if PWM 50/50 used)	PWM magnitude (phase A) PWM magnitude (phase B) PWM magnitude (phase C)
Amplifier Output Signals (per axis, if analog output used)	Analog out (phase A) Analog out (phase B)
Hall inputs:	Hall (phase A) Hall (phase B) Hall (phase C)
Other Control Signals: (per axis)	Home signal channel input Positive limit switch input Negative limit switch input AxisIn input AxisOut output

Miscellaneous Signals:	GND +5 V (for encoder power)
------------------------	---------------------------------

**For a complete description of the PC card connectors and interfacing requirements see Section 4, PCI DK1 Electrical Reference.**

## 1.10 Applying Power

Once you have connected the PCI DK1 board to the desired number of external amplifiers and motor encoders, hardware installation is complete and the board is ready for operation.

Upon power up, the motion processor will be in a reset condition. In this condition no motor output will be applied until the motion processor is initialized (see next section on software for details). Therefore, the motors should remain stationary. If the motors do move or jump, power down the board and check the amplifier and encoder connections. If anomalous behavior is still observed, call PMD for application assistance.

## 1.11 Software Installation

Included in your developer's kit is a CDROM marked "Developer's Kit Software". This CD contains software to exercise your board and source code that will enable you to develop your own motion applications. The exercise software is designed to work with Windows 95/98/ME or Windows NT/2000.

If you have autorun enabled, the installation process will start when you insert the CDROM. The installation program will guide you through installing the software. Upon completion of the installation process, the following components will be installed:

- 1) Pro-Motion – an application for communicating to and exercising the installed developer's kit. Refer to Section 2 for operating instructions.
- 2) PMD Board Setup – used to set the base address for communication with the ISA based developer's kit. This application is not used with the PCI developer's kit.
- 3) C-Motion – source code that can be used for developing your own motion applications based on the Navigator motion processor. Refer to Section 3 for further information. These files are installed in the "C-Motion" folder, a sub-folder of the installation folder.
- 4) "PDF" versions of the developer's kit manual, programmer's reference and user's guide. The Adobe Acrobat Viewer is required for viewing these files. If the Adobe Acrobat Viewer is not installed on your computer, you can download it from <http://www.adobe.com>.
- 5) Reference schematics from the Technical Specification manuals. These files are installed in the "Schematics" folder, a sub-folder of the installation folder.
- 6) Schematics for the PCI DK1 DK board. These files are installed in the "Schematics" folder, a sub-folder of the installation folder.
- 7) Parts list for the board. The file "PCI DK LOM.wri" is installed in the "Schematics" folder, a sub-folder of the installation folder.

## 1.12 First Time System Verification

To verify that the PCI DK1 board and Pro-Motion software program have been properly installed, it is useful to have each axis of the system perform a short move. The instructions shown below are a

summary. Before executing these commands you should refer to the section of the *Navigator User's Guide* that is relevant to the installed motion processor.

For the MC2100 motion processor to perform this simple sequence it is necessary to specify two items; the motor amplifier type (PWM sign/mag, PWM 50/50, or analog), and the filter gains. For the MC2300 motion processor parts it is necessary to specify these two items as well as initialize the motor commutation.

The following table summarizes this. Note that the step numbers reference specific steps that are detailed in the next section.

Motion Processor	Step #	Operation
MC2100	1	Set amplifier type (PWM sign/mag, PWM 50/50, DAC)
	4	Set filter parameters
	6	Make a trajectory move
MC2300/2800	1	Set amplifier type (PWM 50/50, DAC)
	2	initialize commutation*
	3	Check commutation
	4	Set filter parameters
	6	Make a trajectory move
MC2400	1	Set amplifier type (PWM 50/50, DAC)
	5	Set the motor output power
	6	Make a trajectory move
MC2500	6	Make a trajectory move

\* Contact PMD for the "Phase Initialization" application note.

**Perform only the setup step sequences for the motion processor installed on your board.**

It is assumed that you will check out each axis of your system one at a time. This being the case after executing Pro-Motion type one of the axis setting commands (#axis 1, #axis 2, #axis 3, or #axis 4) in the Command window and go through the whole checkout for that axis. Then to check out other axes enter a new axis number and check that axis out entirely, etc.

**1.12.1 Step #1: Set the Motor Amplifier Type**

The motion processor must be told what type of motor output mode to use, PWM sign/mag, PWM 50/50, or DAC. This can be set using the command SetOutputMode. Assuming the axis you want to exercise is #1, you would type in the command "SetOutputMode" followed by the output mode; 0 for DAC, 1 for PWM sign/mag, and 2 for PWM 50/50. For example to specify the output mode as PWM 50/50 the following sequence would be typed in:

SetOutputMode 2

And this would be followed by an <ENTER> to process the command. Upon successfully accepting this command Pro-Motion will return a prompt ">". If there is some problem with the command Pro-Motion will indicate the nature of the error, then will be followed with a ">" prompt.



For more information on Pro-Motion, refer to Section 2.

### 1.12.2 Step #2: Initialize the Commutation

**NOTE: THIS SECTION APPLIES TO THE MC2300/2800 MOTION PROCESSORS ONLY.**

For the motor to be controlled properly using the MC2300, the motion processor must select and possibly initialize the commutation phasing. If you will be using Hall-based commutation then no initialization is necessary. Simply specify this to the motion processor using the command:

**SetCommutationMode 1**

No other commands are necessary and you may proceed to step #3.

If you are controlling a brushed type motor, set the number of phases to 1 as follows:

**SetNumberPhases 1**

No other commands are necessary and you may proceed to step #3.

If you will be commutating using a sinusoidal technique you must initialize the commutation phasing. There are two ways this can be done. You will need to decide whether to initialize using Hall-based or algorithmic methods. See the *Navigator Motion Processor User's Guide* for more information on this.

Each of these two phase initialization methods requires a separate sequence, as follows (note that // indicates a comment and should not be typed in):

*Hall-based initialization command sequence:*

```
SetNumberPhases x          // where x is 2 or 3 depending on type of motor
SetPhaseCounts yyyy       // yyyy is # of encoder counts per electrical cycle
SetPhaseInitializeMode 1   // set phase initialize mode to 'Hall-based'
InitializePhase
```

*Algorithmic-based initialization command sequence:*

```
SetNumberPhases x          // x is 2 or 3 depending on type of motor
SetPhaseCounts yyyy       // yyyy is # of encoder counts per electrical cycle
SetPhaseInitializeMode 0   // set phase initialize mode to 'algorithmic'
SetMotorMode 0            // places axis in open loop mode, required for algorithmic initialization.
SetPhaseInitializeTime zzzz // zzzz is # of motion processor cycles to initialize for
SetMotorCommand wwwww     // wwwww is motor command.
InitializePhase
```

To determine the values of x, yyyy, zzzz, and wwwww you should refer to the *Navigator Motion Processor User's Guide* MC2300 section.

For more information on Pro-Motion, refer to section 2.

**If your system has one or more of the following conditions present then the above sequence will need to be expanded. To handle such systems you will need to use the SetSignalSense command as well as the SetPhasePrescale command. Refer to the MC2300 section of the *Navigator User's Guide* for more information.**

- 1) One or more Hall signals must be inverted to commutate or initialize the commutation correctly
- 2) # of encoder counts per electrical cycle exceeds 32,767

### 1.12.3 Step #3: Check Commutation

**NOTE: THIS SECTION APPLIES TO THE MC2300/2800 MOTION PROCESSORS ONLY.**

After phase initialization has been completed it is useful to check the smoothness of the motor rotation in open loop mode to verify that the motor phasing initialization and commutation is correct. To do this use the following command sequence:

```
SetMotorMode 0           // set axis for open loop operation
SetMotorCommand xxxx    // xxxx is the motor command from 0 to 32,767 to output
Update
```

The 'xxxx' value represents the fraction of the value 32,768 of total power that will be applied to the motor. For example a value of 1,000 sends roughly 3 % (1000/32768) of the total power to the motor.

**When the motor mode is set off (SetMotorMode 0) the motor is not under servo control. Be aware that the motor may spin rapidly after a motor command value is applied. Use small values and increase slowly.**

After this command sequence the motor should smoothly spin in one direction or the other. The motor command is a signed number and the sign controls the rotation direction. When a positive motor command is given the motor should rotate in the positive (increasing encoder counts) direction. If the motor spins roughly, in the wrong direction, or if it moves a short distance and then abruptly stops there may be a problem with the commutation. Check your wiring and re-test. Once the motor is spinning smoothly in both directions under open loop control, re-enable closed-loop servo control by executing the following command:

```
SetMotorMode 1
```

### 1.12.4 Step #4: Set Filter Parameters

For motion to occur, some amount of feedback gain must be specified. Initially use just a proportional gain with a very low value between 1 and 25. Later you can add integral or derivative gains as well as feedforward gains if desired. The following sequence shows how to set the P, I, and D terms of the filter and how to 'update' them, making them active.

```
SetKp xxxx           // xxxx is the desired proportional gain
SetKd yyyy           // yyyy is the desired derivative gain
SetKi zzzz           // zzzz is the desired integral gain
SetIntegrationLimit aaaa // aaaa is the desired integration limit
Update               // make thee values active.
```

It is not necessary to specify all 3 gains. Just Kp, followed by an Update can be specified, just a Kd, etc.

**When exercising the motor use extreme caution. It is the responsibility of the user to observe safety precautions at all times.**

### 1.12.5 Step #5: Set the Motor Command

**NOTE: THIS SECTION APPLIES TO THE MC2400 MOTION PROCESSORS ONLY.**

In order for motion to occur, the magnitude of the output must be set. Refer to the *Navigator User's Guide* for more information. A value between 0 and 32767 represents an amplitude of 0 – 100%. A value of around 5000 should be satisfactory to start with.

Here is the command sequence to use:

```
SetMotorCommand xxxx          // Sets the motor output level
Update                        // execute the move
```

### 1.12.6 Step #6: Make a Trajectory Move

To test that the motor is being driven properly, set up and execute a small trapezoidal move. Specify a small distance of (for example) 5,000 counts, and a low velocity and acceleration of (for example) 10,000, and 10 respectively. With a cycle time of 400 uSec, these values correspond to roughly 381 counts/sec, and 954 counts/sec<sup>2</sup>, respectively.

**Whatever profile values you use, be sure that they are safe for your system.**

Here is the command sequence to use:

```
SetProfileMode 0              // Sets current profile mode to trapezoidal
SetPosition 5000              // 5000 is the desired destination position
SetVelocity 10000             // 10000 is the desired maximum velocity
SetAcceleration 10            // 10 is the desired acceleration
SetDeceleration 10           // 10 is the desired deceleration
Update                        // execute the move
```

After entering this sequence of commands you should see the axis smoothly move for about 15 seconds (if the suggested values are used and the cycle time of the motion processor is 400 usec)

If you do not see the axis moving, or if the axis jumps rapidly in one direction or the other, there may be a problem with the board or software settings. Re-check and review the board setup procedures, as well as the exerciser parameter settings.

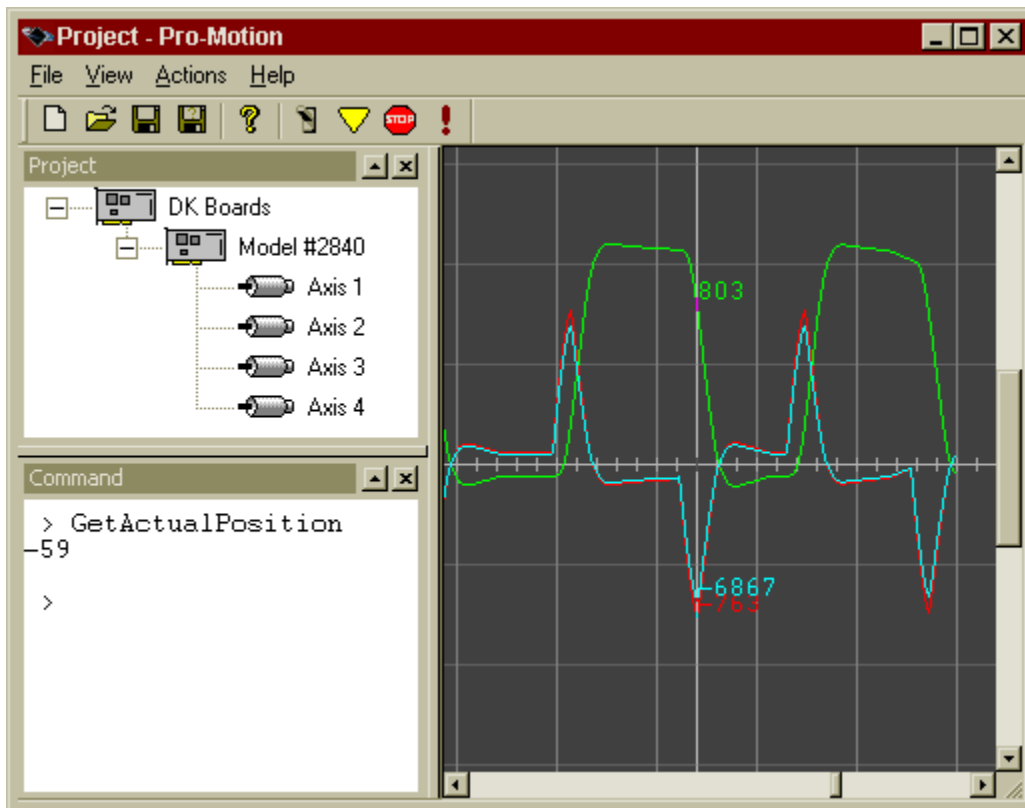
If you are still having problems after re-checking your system call PMD for applications assistance.

## 2 Using Pro-Motion™

The Pro-Motion program facilitates the exercising of the PMD motion processor installed in your *Navigator Motion Processor Developer's Kit*. All processor parameters can be viewed and modified via standard Windows controls.

Pro-Motion features:

- Project window for accessing processor parameters.
- Command window for direct text command entry. It also serves as a communications monitor that echoes all commands sent by Pro-Motion to the processor.
- Axis shuttle performs continuous back and forth motion between two positions
- Oscilloscope graphically displays processor parameters in real-time.



### 2.1 Communication

There are two interfaces to the processor. A serial and a parallel interface. Pro-Motion may use one or the other or both. When Pro-Motion is started (or File/New is selected) the DK board is interrogated by first attempting to connect via the parallel interface (ISA slot). If successful, Pro-Motion will display the processor version and the number of axes in the project window. If the parallel interface fails (i.e., when using the DK board in stand-alone mode), then the serial interface is attempted at the default baud of 57600. If no axes are displayed under the DK board item in the Project window, then there is a communication problem with the DK board. This may be caused by a mismatch of the parallel or serial interface parameters in Pro-Motion and the jumper setting on the

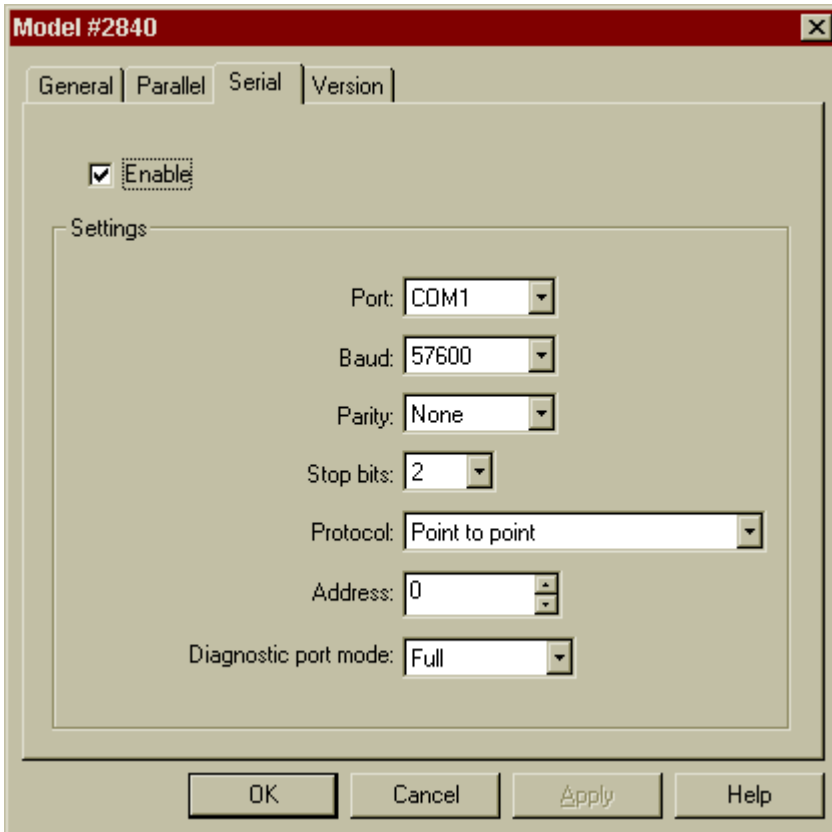
DK board. To change the interface parameters in Pro-Motion to match the DK board settings double-click on the second DK board icon in the Project window to display the DK board properties dialog and follow the directions below for the interface being used.

### 2.1.1 Parallel interface

To use the parallel interface the DK board must be installed in an available ISA slot. Run the PMD Board Setup program to set I/O address for Pro-Motion to match the board setting. In Pro-Motion, select the Parallel tab in the DK board properties dialog. Make sure that the Enable button is checked and the IO Mode is 16/16. When enabled, the parallel interface is the default interface whether the serial interface is enabled or not. If both parallel and serial interfaces are enabled, you may specify which interface an axis will use in the General page of the Axis properties dialog.

### 2.1.2 Serial interface

To use the serial interface the serial cable provided with the DK board must be connected to an available COM port. In Pro-Motion, select the Serial tab in the DK board properties dialog.



Set the serial interface parameters according to the settings on the DK board and the COM port that it is connected to on the PC. Check the Enable check box and set the Diagnostic mode to Full to allow all commands to be accepted by the processor. However, the Diagnostic mode can only be set to full via the parallel interface. Therefore, if the parallel interface is not enabled the following error message may appear in the Command window, "Command only available on primary port." In this case the Host Interface mode must be set to Serial only (see section 1.5 Stand-Alone mode). Note that some functions of the program may react slowly due to the slower data rate of the serial interface.

## 2.2 Motion

Once communications has been established with the DK board, motion may be performed to the connected motors by setting the appropriate Axis parameters and running the Motion Shuttle. Right clicking on an axis icon in the Project window accesses these dialogs.

The following table summarizes the steps necessary to obtain motion depending on the processor used.

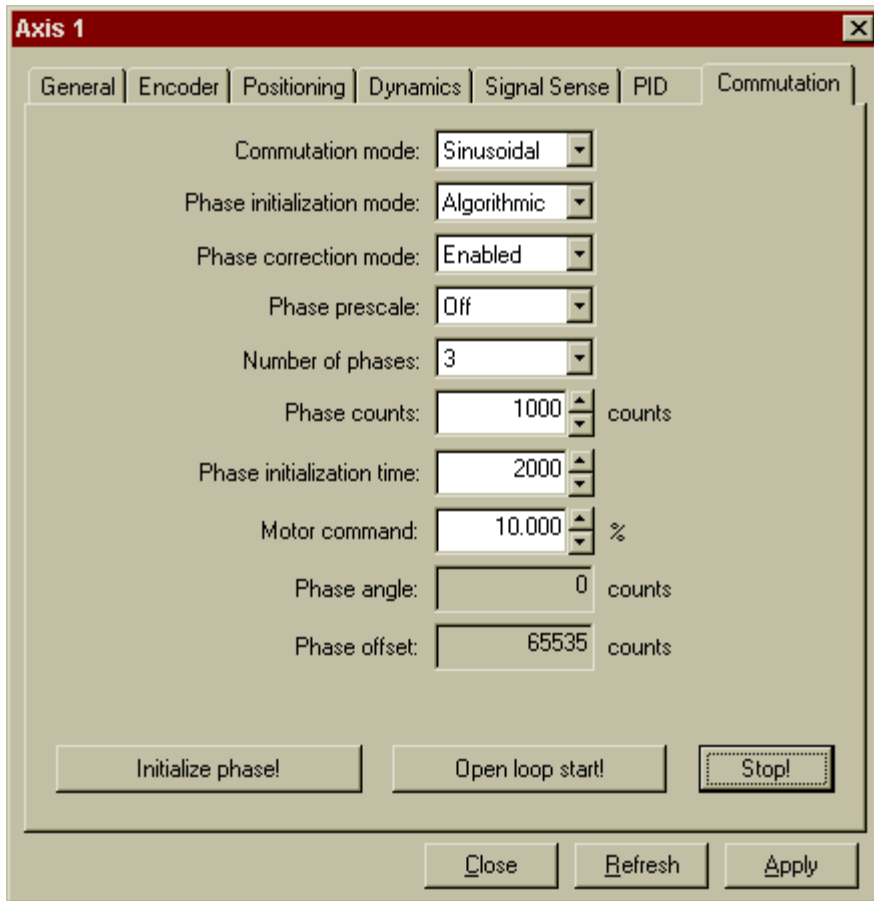
Motion Processor	Step #	Operation
MC2100	1	Set amplifier type (PWM sign/mag, PWM 50/50, DAC)
	4	Set filter parameters
	6	Start motion
MC2300/2800	1	Set amplifier type (PWM 50/50, DAC)
	2	initialize commutation
	3	Check commutation
	4	Set filter parameters
	6	Start motion
MC2400	1	Set amplifier type (PWM 50/50, DAC)
	5	Set the motor output power
	6	Start motion
MC2500	6	Start motion

### 2.2.1 Step #1 Amplifier Type

Select the General page of the Axis properties dialog and set the Motor output mode. Make sure the Motor mode is set to On and the Axis enabled button is checked. Click the Apply button to send the changes.

### 2.2.2 Step #2 Initialize Commutation

Select the Commutation page of the Axis properties dialog and set the Commutation parameters to the desired values. If the Commutation mode is set to Sinusoidal, then the Initialize phase button should be clicked after all the parameters have been correctly set and before any motion is attempted or the Open loop start button is pressed.



### 2.2.3 Step #3 Check Commutation

The commutation may be verified by running the motor in open loop mode. Set the motor command parameter to a low value between 1 and 15% keeping in mind the motor will start to move at a rate proportional to the motor command setting. Click the Open loop start button. If the motor is commutating properly, the motor should spin smoothly. Click the Stop button to stop motion.

### 2.2.4 Step #4 Set Filter Parameters

Select the PID page of the Axis properties dialog and set the  $K_p$  parameter to a low value between 1 and 25 initially to prevent any oscillations. Click the Apply button.

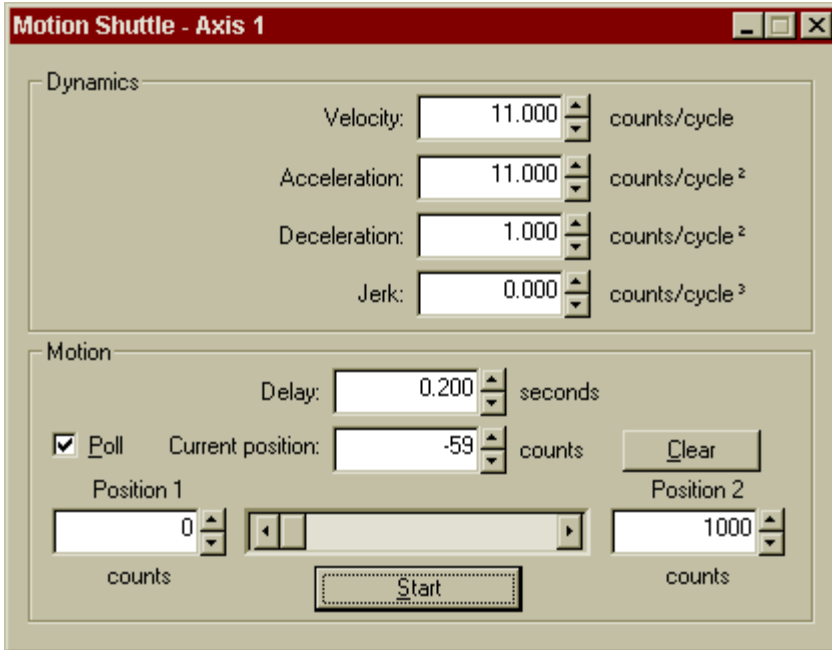
### 2.2.5 Step #5 Motor Output Power

Set the motor command parameter to a low value between 10 and 20% to start with. Click the Open loop start button to send this value because the Apply button does not send this particular value. If an attempt to start a trajectory move does not move the motor, try increasing this value.

### 2.2.6 Step #6 Start motion

Select the page of the Axis properties dialog and set the Profile mode to Trapezoidal. Set the velocity to a low value between 1 and 10 counts/cycle. Set the Acceleration and Deceleration parameters to a low value between 0.01 and 1 counts/cycle. Click the Apply button.

Right click on the desired axis and select Shuttle to display the Motion Shuttle dialog if it is not already displayed for the desired axis.

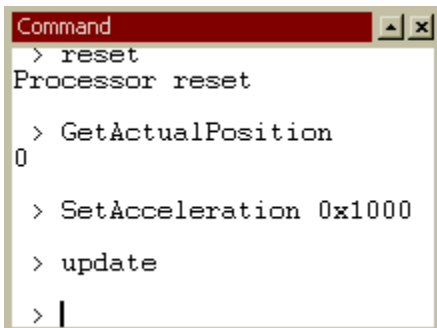


The Dynamics parameters should be set to the values previously set in the Dynamics page of the Axis properties dialog. Click the Start button. The motor should start moving back and forth between Position 1 and Position 2. If the motor does not begin to move return to step 1. If the motor starts moving, but does not look like its shuttling between the two points, verify that it is moving in the right direction and the current position is between the 2 position values. If it is not moving in the right direction, try changing the signal sense of one of the encoder signals if applicable. If the motor's current position was not between the two positions, it will move to Position 1 before shuttling will occur.

The motion can be monitored using the Oscilloscope.

The commands that Pro-Motion sends to the processor can be monitored by selecting the View/Monitor output menu item.

## 2.3 Command Window



The Command window in Pro-Motion allows you to issue commands directly to the PMD motion processor installed in your *Navigator Motion Processor Developer's Kit*. The window has a command line style interface that accepts all of the motion processor commands. The *Navigator Motion Processor Programmer's Reference* contains a full list of commands along with their required parameters. The command window presents you with the command prompt '>'. The sequence below shows a typical command session.

```
> Reset
```



```
Processor reset
> SetKp 25
> Update
> SetProfileMode 0
> SetVelocity 200000
> SetAcceration 256
> SetDeceleration 256
> SetPosition 123456
> Update
```

This sequence of commands:

- 1) Resets the Navigator motion processor;
- 2) Sets the K<sub>p</sub> PID filter parameter;
- 3) Sets up and executes a trapezoidal move.

The Command window is not case sensitive so commands can be entered in any combination of upper and lower case characters. As shown above, commands are entered as a sequence of command name followed by up to 2 numeric parameters. Parameters can represent a single 16-bit word of data or a 32-bit double word of data depending on what is required by the particular command.

```
> SetBreakpointValue 0 1000
```

In this example the first parameter represents a 16-bit word that contains the selected breakpoint number (0 or 1) and the second parameter represents a 32-bit word that contains the breakpoint value.

All of the “Get” commands display the value returned by the chipset.

```
> GetEventStatus
0x0309
```

Some “Get” commands require a parameter for selecting the desired value.

```
> GetBreakpointValue 0
0
```

In this example, 0 selects the breakpoint number for which the value is retrieved.

The Command window accepts numeric parameters in either decimal or hexadecimal format. Prefixing a numeric parameter with “0x” enters that number using hexadecimal format. For example,

```
> SetKp 0xA
```

sets the K<sub>p</sub> to 10 (decimal). When a numeric parameter is entered without any prefix it is assumed to be in decimal format.

The Command window automatically inserts the axis number into the command word before sending it to the Navigator motion processor. At startup, the axis number is set to 1. For motion processors that support more than one axis, you can select the axis number using the following commands:

```
> #axis 1          // selects axis # 1
> #axis 2          // selects axis # 2
> #axis 3          // selects axis # 3
> #axis 4          // selects axis # 4
```

The Command window is capable of executing sequences of commands stored in external files using the file-input command “<” followed by the name of the file to execute.

```
> <setup.txt
```

As an example, the file “setup.txt” could have the following command sequence that initializes DAC output mode and sets the servo parameters for the attached hardware.

```
Reset
SetOutputMode 0
SetKp 25
SetKd 200
Update
```

This file can then be used any time the system needs to be re-started. Any sequence of valid commands can be contained within the script file.

The tilde symbol ‘~’ can be prefixed to any line to make that line a comment which will not be executed.

### 2.3.1 Commands Available within the Command Window

CTRL-R:	Repeat the last command entered.
TAB:	Automatic command name completion. Press this key after having entered the first few characters of the command you wish to execute. If more than one command begins with these characters, a window containing a list of commands that match the starting characters will appear. Select the correct command using the keyboard or mouse.
UP-ARROW:	Sequentially scrolls back through previously entered commands.
DOWN-ARROW:	Sequentially scrolls forward through previously entered commands.
ESC:	Clears the command line.
<i>#axis number.</i>	Selects the current axis. (e.g., <i>#axis 3</i> selects axis number 3.) All future commands will be executed on the selected axis.
<i>&lt; filename</i>	Script file-input command. Executes the commands contained in the specified file.
TraceDownload <i>filename</i>	Reads trace data from the external memory and stores it to the specified <i>filename</i> . Prior to executing this command, an actual trace must have been started as described in section 7 of the <i>Navigator User's Guide</i> .

For more information click on Help in Pro-Motion.

## 3 Developing Your Own Applications with C-Motion

---

C-Motion is a “C” source code library that contains all the code required for communicating to the Navigator or Pilot motion processors using the parallel or serial interface. It is used in conjunction with the Pathfinder library for uploading data and controlling the motion processor. Alternately, if the required communication interface does not exist, C-Motion can be used as a basis for developing a custom communication interface.

C-Motion includes the following features:-

- Axis virtualization
- The ability to communicate to multiple PMD motion processors
- Can be easily linked to any “C/C++” application
- Supports 16/16, 8/16 and 8/8 parallel communication modes
- Supports serial communication
- Supports Windows ISA and PCI driver communication mode

The following files make up the C-Motion distribution:

C-Motion.h/C-Motion.c	Definition/declaration of the PMD Navigator command set
PMDpar.h/PMDpar.c	Parallel interface functions
PMDW32ser.h/PMDW32ser.c	Windows serial communication interface functions
PMDdrv.h/PMDdrv.c	Windows ISA driver communication interface functions
PMDpci.h/PMDpci.c	Windows PCI driver communication interface functions
PMDutil.h/PMDutil.c	General utility functions
PMDtrans.h/PMDtrans.c	Generic transport (interface) functions
PMDdecode.h	Defines the PMD Navigator and C-Motion error codes
PMDocode.h	Defines the control codes for Navigator commands
PMDtypes.h	Defines the basic types required by C-Motion
PMDdiag.h	Defines a string structure for each command code that can be used for debugging or diagnostics

C-Motion can be linked to your application code by including the above “C” source files in your application. Then, for any application source file that requires access to the motion processor `#include “C-Motion.h”`. In addition, the required interfaces need to be `#defined` as shown below. Only the required interfaces need to be included.

```
// use this for a standard parallel interface under DOS or Win9x
#define PMD_PARALLEL_INTERFACE
// use this for a standard serial interface under Win9x/NT/2000/XP
#define PMD_W32SERIAL_INTERFACE
```

```
// use this for a standard parallel interface under Win9x/NT/2000/XP
#define PMD_DRIVER_INTERFACE
// use this for a standard PCI parallel interface under Win9x/NT/2000/XP
#define PMD_PCI_INTERFACE
```

By customizing the base interface functions, C-Motion can be ported to virtually any hardware platform. An example would be a memory-mapped IO scheme that uses the parallel interface. This would be built using the PMDPar.c/.h source files as a basis.

The PCI DK board uses the PCI interface chip provided by PLX Technology. To fully understand the interface mechanism, or to write your own interface software you can purchase the PLX SDK. More information on the price and features can be found on the PLX website – <http://www.plxtech.com/> in the products, tools, software development kits area.

## Theory of Use

C-Motion is a set of functions that encapsulate the motion processor command set. Every command has as its first parameter an “axis handle.” The axis handle is a structure containing information about the interface to the motion processor and the axis number that the handle represents. Before communicating to the motion processor, the axis handle must be initialized using the following sequence of commands:

```
// the axis handle
PMDAxisHandle hXAxis;

PMDParallelIOTransportData transport_data;

// set the axis we are talking to with this handle
hXAxis.axis = PMDAxis1;

// the transport data is initialized first to set the defaults
// make sure the IO mode is set correctly
PMDParallel_InitData(&transport_data,PMDParallelIOMode_16_16);

// if required the transport data defaults can be changed here
// e.g. board IO address (dataPort,commandPort,statusPort)
// transport_data.dataPort = 0x560;

// assign the transport data structure we initialized to the axis handle
hXAxis.transport_data = (void*) &transport_data;

// initialize the transport (initializes parallel IO function pointers)
PMDParallel_Init(&hXAxis.transport);
```

The above is an example of initializing communication using the parallel communication interface. Each interface .c source file contains an example of initializing the interface.

Once the axis handle has been initialized, any of the motion processor commands can be executed. C-Motion.h includes the prototypes for all motion processor commands as implemented in C-Motion. Refer to this file for the required parameters for each command. The *Navigator or Pilot Programmer's Reference* is the primary source for information about the operation and purpose of each command.

As part of a normal startup procedure, the motion processor is often reset. This command only needs to be executed ONCE for each motion processor, and not for every axis. The following code demonstrates a chip reset using C-Motion.

```
PMDuint16 result;
PMDuint16 status;

// reset the PMD chip that this axis resides on
// if more than one chip(set) is present, all of them should be
// reset here
result = PMDReset(handle);

// in the serial interface mode if an error occurred it is returned
// immediately with no need to call GetHostIOError, so in this
// code we check for any error before continuing.
// with the parallel interface the result code will always be
// PMD_ERR_CommandError since that bit is set whenever a reset
// occurs. If it ISN'T set then there is some other error
if ( (result != PMD_ERR_ChipsetReset) && (result != PMD_ERR_CommandError) )
{
    printf("Error: %s\n", PMDGetErrorMessage(result));
    return 0;
}

// after the reset the chip will be in the PMDChipsetReset state
result = PMDGetHostIOError(handle, &status);

// the above command should execute without error but we need to check
if ( (result != PMD_ERR_OK) || (status != PMD_ERR_ChipsetReset) )
{
    printf("Error: %s\n", PMDGetErrorMessage(result));
    printf("Status: %s\n", PMDGetErrorMessage(status));
    return 0;
}
return 1;
```

An example of the reset procedure is also included in the PMDutil.c source file.

Every motion processor command returns a status code of type PMDuint16. The return code for every command executed should be checked before attempting to execute more commands.

```
PMDuint16 result,status;
```

```

result = PMDUpdate(&hXAxis);

if (result != PMD_ERR_OK)
{
    status = result;
    if (result == PMD_ERR_CommandError)
        PMDGetHostIOError( &hXAxis, &status );
    printf("Error: %s\n", PMDGetErrorMessage(status));
    return;
}

```

Internally, C-Motion checks the status of the HostIOError bit after issuing a command. If this bit is set it will return PMD\_ERR\_CommandError. The application should then call PMDGetHostIOError to clear the error bit and determine the cause of the error.

Many commands require additional parameters. Some standard values are defined by C-Motion and can be used with the appropriate commands. Refer to PMDtypes.h for a complete list of defined types. An example of calling one of the C-Motion functions with the pre-defined types is shown below.

```

PMDSetBreakpoint(&hXAxis, PMDBreakpoint1, PMDAxis2,
PMDBreakpointActionAbruptStop, PMDBreakpointActualPositionCrossed);

```

The following functions are provided by C-Motion in addition to the standard chip command set:

```

PMDuint16 PMDGetStatus(PMDAxisHandle* axis_handle);

```

returns the result of executing an IO status read operation. Only returns a valid result when the interface is parallel.

```

PMDuint16 PMDHasError(PMDAxisHandle* axis_handle);

```

returns 1 if the error bit is set in the IO status word and returns 0 if it is not set. Only returns a valid result when the interface is parallel.

```

PMDuint16 PMDIsReady(PMDAxisHandle* axis_handle);

```

returns 1 if the ready bit is set in the IO status word and returns 0 if it is not set. Only returns a valid result when the interface is parallel.

```

PMDuint16 PMDHasInterrupt(PMDAxisHandle* axis_handle);

```

returns 1 if the interrupt bit is set in the IO status word and returns 0 if it is not set. Only returns a valid result when the interface is parallel.

```

void PMDCloseAxisInterface(PMDAxisHandle* axis_handle);

```

should be called to terminate an interface connection

`char *PMDGetErrorMessage(PMDuint16 errorCode);`  
returns a character string representation of the corresponding PMD chip or C-Motion error code.

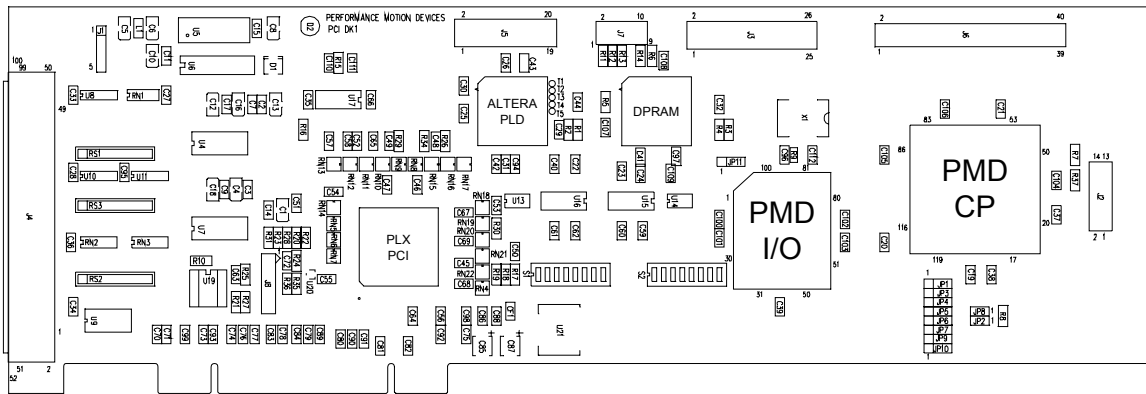
`void GetCMotionVersion(PMDuint8* MajorVersion, PMDuint8* MinorVersion);`  
returns the major and minor version number of C-Motion.

# 4 PCI DK1 Electrical Reference

## 4.1 PCI DK1 Layout

Figure 2-1 shows the locations of the principal components of the developer's kit board. The component side of the board is shown, with the ISA slot connector at the bottom. All component locations in this manual refer to this orientation. In Figure 4-1, the motion processor's CP chip and I/O chip are identified for reference. All other components of interest to the user are identified by their board label:

- Jumper JP11 sets the Host I/O mode
- Switch blocks S1 and S2 set the serial interface configuration and multi-drop address
- Connectors J1, J3, J4, and J5 are described in the second part of this manual
- Resistor packs RS1 – RS3 are set according to the type of encoder used



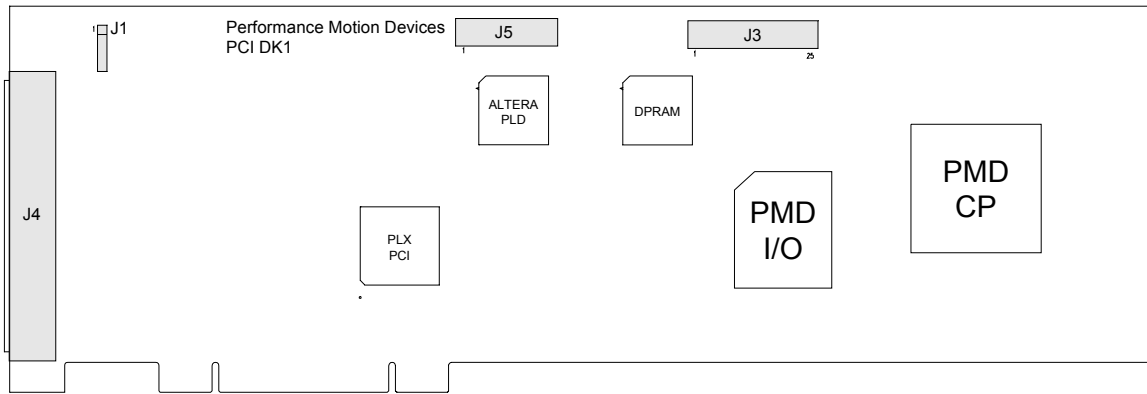
**Figure 4-1 Developer's kit board layout**



## 4.2 PCI DK1 Connectors

This section describes the pinouts for the following cable connectors on the Navigator developer's kit card (shaded areas in Figure 4-2 Developer's kit connector locations):

- J1** 5-pin serial communication connector
- J4** 100-pin main connector containing encoder input, Hall input, AxisIn signals, AxisOut signals, Motor output signals, and limit switch inputs
- J5** 20-pin user-defined digital I/O connector
- J3** 26-pin analog input connector



**Figure 4-2 Developer's kit connector locations**

### 4.2.1 Serial Communications Connector (J1)

*Location:* Upper left corner, near to the power indicator LED.

This is a 5-pin single row header (0.1" spacing).

Pin number	Signal Name
1	SrlXmt (CP pin 44)
2	SrlRcv (CP pin 43)
3	SrlEnable (CP pin 99)
4	GND
5	V <sub>cc</sub>

Note that if your board was shipped with an RS232 or RS485 serial driver board, these pins are under the board.

### 4.2.2 Motion Peripherals Connector (J4)

*Location:* On the right edge of the card.

This is a 100-pin high-density connector (2x50, 0.05" spacing). The accompanying cable assembly supplied with your developer's kit consists of two 36" flat ribbon cables terminating together at one end in the matching 100-pin connector. At the other end, each ribbon terminates in a 50-pin header

(2x25, 0.1” spacing). The ribbons are labeled **Hdr1** and **Hdr2**. Pins 1-50 on Hdr1 connect to pins 1-50 of J4. Pins 1-50 of Hdr2 connect to pins 51-100 of J4.

### Connector pinouts with an MC2100, MC2300 or MC2800 series processor installed

Header 1 (to J4 pins 1-50)		Header 2 (to J4 pins 51-100)					
Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	QuadA1+	26	QuadA2+	1	QuadA3+	26	QuadA4+
2	QuadA1-	27	QuadA2-	2	QuadA3-	27	QuadA4-
3	QuadB1+	28	QuadB2+	3	QuadB3+	28	QuadB4+
4	QuadB1-	29	QuadB2-	4	QuadB3-	29	QuadB4-
5	Index1+	30	Index2+	5	Index3+	30	Index4+
6	Index1-	31	Index2-	6	Index3-	31	Index4-
7	V <sub>cc</sub> (encoder)	32	V <sub>cc</sub> (encoder)	7	V <sub>cc</sub> (encoder)	32	V <sub>cc</sub> (encoder)
8	GND (encoder)	33	GND (encoder)	8	GND (encoder)	33	GND (encoder)
9	Hall1A	34	Hall2A	9	Hall3A	34	Hall4A
10	Hall1B	35	Hall2B	10	Hall3B	35	Hall4B
11	Hall1C	36	Hall2C	11	Hall3C	36	Hall4C
12	GND (Hall)	37	GND (Hall)	12	GND (Hall)	37	GND (Hall)
13	PosLim1	38	PosLim2	13	PosLim3	38	PosLim4
14	NegLim1	39	NegLim2	14	NegLim3	39	NegLim4
15	Home1	40	Home2	15	Home3	40	Home4
16	AxisIn1	41	AxisIn2	16	AxisIn3	41	AxisIn4
17	AxisOut1	42	AxisOut2	17	AxisOut3	42	AxisOut4
18	PWMMagA1	43	PWMMagA2	18	PWMMagA3	43	PWMMagA4
19	PWMMagB1	44	PWMMagB2	19	PWMMagB3	44	PWMMagB4
20	PWMMagC1	45	PWMMagC2	20	PWMMagC3	45	PWMMagC4
21	PWMSignA1	46	PWMSignA2	21	PWMSignA3	46	PWMSignA4
22	DACA1*	47	DACA2*	22	DACA3*	47	DACA4*
23	DACB1*	48	DACB2*	23	DACB3*	48	DACB4*
24	GND (DAC)	49	GND (DAC)	24	GND (DAC)	49	GND (DAC)
25	N.C.	50	N.C.	25	N.C.	50	N.C.

#### NOTE:

- Unused signals may be left unconnected.
- Default setting of the board is for differential encoder inputs.
- Each converter cable has labels to indicate HDR1 and HDR2.
- The above table shows pin assignment when the chipset is in PWM Sign/Magnitude mode single phase or PWM 5050 for 2 or 3-phase. For 2-phase PWM Sign/Magnitude pin assignments refer to the next table.
- \*DACA<sub>n</sub>, DACB<sub>n</sub> are mapped to two analog output signals for axis *n*. For single-phased motion processor products (for example MC2100) DACB1 will not be used.

## Connector pinouts with an MC2400 series processor installed

Header 1 (to J4 pins 1-50)		Header 2 (to J4 pins 51-100)					
Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	QuadA1+	26	QuadA2+	1	QuadA3+	26	QuadA4+
2	QuadA1-	27	QuadA2-	2	QuadA3-	27	QuadA4-
3	QuadB1+	28	QuadB2+	3	QuadB3+	28	QuadB4+
4	QuadB1-	29	QuadB2-	4	QuadB3-	29	QuadB4-
5	Index1+	30	Index2+	5	Index3+	30	Index4+
6	Index1-	31	Index2-	6	Index3-	31	Index4-
7	V <sub>cc</sub> (encoder)	32	V <sub>cc</sub> (encoder)	7	V <sub>cc</sub> (encoder)	32	V <sub>cc</sub> (encoder)
8	GND (encoder)	33	GND (encoder)	8	GND (encoder)	33	GND (encoder)
9		34		9		34	
10		35		10		35	
11		36		11		36	
12		37		12		37	
13	PosLim1	38	PosLim2	13	PosLim3	38	PosLim4
14	NegLim1	39	NegLim2	14	NegLim3	39	NegLim4
15	Home1	40	Home2	15	Home3	40	Home4
16	AxisIn1	41	AxisIn2	16	AxisIn3	41	AxisIn4
17	AxisOut1	42	AxisOut2	17	AxisOut3	42	AxisOut4
18	PWMMagA1	43	PWMMagA2	18	PWMMagA3	43	PWMMagA4
19	PWMMagB1	44	PWMMagB2	19	PWMMagB3	44	PWMMagB4
20	PWMSignB1	45	PWMSignB2	20	PWMSignB3	45	PWMSignB4
21	PWMSignA1	46	PWMSignA2	21	PWMSignA3	46	PWMSignA4
22	DACA1	47	DACA2	22	DACA3	47	DACA4
23	DACB1	48	DACB2	23	DACB3	48	DACB4
24	GND (DAC)	49	GND (DAC)	24	GND (DAC)	49	GND (DAC)
25		50		25		50	

### NOTE:

- Unused signals may be left unconnected.
- Default setting of the board is for differential encoder inputs.
- Each converter cable has labels to indicate HDR1 and HDR2.
- The above table shows pin assignment when the chipset is in 2-phase PWM Sign/Magnitude mode. For 3-phase PWM 5050 mode refer to the previous table.

## Connector pinouts with an MC2500 series processor installed

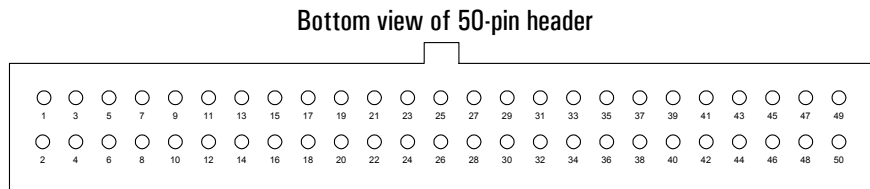
Header 1 (to J4 pins 1-50)				Header 2 (to J4 pins 51-100)			
Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	QuadA1+	26	QuadA2+	1	QuadA3+	26	QuadA4+
2	QuadA1-	27	QuadA2-	2	QuadA3-	27	QuadA4-
3	QuadB1+	28	QuadB2+	3	QuadB3+	28	QuadB4+
4	QuadB1-	29	QuadB2-	4	QuadB3-	29	QuadB4-
5	Index1+	30	Index2+	5	Index3+	30	Index4+
6	Index1-	31	Index2-	6	Index3-	31	Index4-
7	V <sub>cc</sub> (encoder)	32	V <sub>cc</sub> (encoder)	7	V <sub>cc</sub> (encoder)	32	V <sub>cc</sub> (encoder)
8	GND (encoder)	33	GND (encoder)	8	GND (encoder)	33	GND (encoder)
9		34		9		34	
10		35		10		35	
11		36		11		36	
12		37		12		37	
13	PosLim1	38	PosLim2	13	PosLim3	38	PosLim4
14	NegLim1	39	NegLim2	14	NegLim3	39	NegLim4
15	Home1	40	Home2	15	Home3	40	Home4
16	AxisIn1	41	AxisIn2	16	AxisIn3	41	AxisIn4
17	AxisOut1	42	AxisOut2	17	AxisOut3	42	AxisOut4
18	Pulse1	43	Pulse2	18	Pulse3	43	Pulse4
19		44		19		44	
20	AtRest1	45	AtRest2	20	AtRest3	45	AtRest4
21	Direction1	46	Direction2	21	Direction3	46	Direction4
22		47		22		47	
23		48		23		48	
24	GND (DAC)	49	GND (DAC)	24	GND (DAC)	49	GND (DAC)
25		50		25		50	

### NOTE:

- Unused signals may be left unconnected.
- Default setting of the board is for differential encoder inputs.
- Each converter cable has labels to indicate HDR1 and HDR2.

### 4.2.3 Connector Pin Layout

The following diagram shows the pin layout for the two 50-pin header cables used with the PCI DK1 board.



For testing purposes, this connector can be mated with a terminal block. PMD suggests Phoenix Contact (<http://www.phoenixcon.com/>), part number FLKM 50.

#### 4.2.4 User-defined Digital I/O Connector (J5)

*Location:* Along the upper edge, directly above the Altera PLD.

These general-purpose IO signals source up to 4mA and can sink up to 8mA. These pins are accessed using the Navigator commands **ReadIO/WriteIO**.

**WriteIO 0 xxxx**

the 8 LSB are output to PrlOut0-7

**ReadIO 0**

the 8 LSB are read from PrlIn0-7

This is a 20-pin header (2x10, 0.1" spacing).

Pin number	Signal Name	Pin number	Signal Name
1	PrlIn0	10	PrlOut4
2	PrlOut0	11	PrlIn5
3	PrlIn1	12	PrlOut5
4	PrlOut1	13	PrlIn6
5	PrlIn2	14	PrlOut6
6	PrlOut2	15	PrlIn7
7	PrlIn3	16	PrlOut7
8	PrlOut3	17, 19	GND
9	PrlIn4	18, 20	Vcc

#### 4.2.5 Analog Input Connector (J3)

*Location:* Along the upper edge, directly above the I/O chip and to the right of J5.

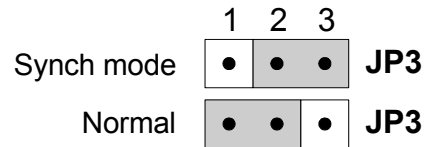
This is a 26-pin header (2x13, 0.1" spacing).

Pin number	Signal Name	Pin number	Signal Name
1	Analog1 (CP pin 74)	14	AnalogRefLow (CP pin 86)
2	Analog2 (CP pin 89)	15	AnalogGND (CP pin 87)
3	Analog3 (CP pin 75)	16	AnalogGND (CP pin 87)
4	Analog4 (CP pin 88)	17	AnalogVcc (CP pin 84)
5	Analog5 (CP pin 76)	18	GND
6	Analog6 (CP pin 83)	19	GND
7	Analog7 (CP pin 77)	20	Vcc
8	Analog8 (CP pin 82)	21	AxisOut1 (CP pin 94)
9	N.C. (CP pin 78)	22	AxisOut2 (CP pin 95)
10	N.C. (CP pin 79)	23	AxisOut3 (CP pin 96)
11	N.C. (CP pin 80)	24	AxisOut4 (CP pin 97)
12	N.C. (CP pin 81)	25	SrlEnable (CP pin 99)
13	AnalogRefHigh (CP pin 85)	26	~HostIntrpt (CP pin 98)

#### 4.2.6 Synch feature jumper (JP3)

*Location:* Towards the Right edge of the board, below the PMD CP chip.

For non synch-enabled chipsets this jumper should be set to “Normal” position. When used with the Synch-enabled chipsets this jumper should be set to the “Synch mode” position.



When multiple boards are connected for use with the Synch feature, pin 1 of JP3 must be connected across all boards in a daisy-chain fashion. For more information on the multi-chip synchronization feature, refer to the *Navigator User's Guide*.

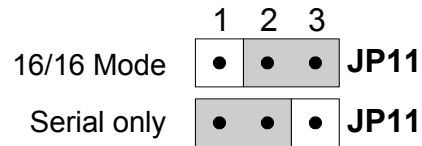
## 4.3 Host I/O Mode

### 4.3.1 PCI DK1 Configuration Jumpers and Switch Blocks

Settable jumpers and switch blocks are shown in Figure 1-1. **All other jumpers should be left in their default factory settings.**

### 4.3.2 Configuration Jumper JP11

These jumpers are used to set the host interface mode, as shown below. The shadowed area indicates the position of the jumper.



## 4.4 Serial Port

### 4.4.1 DIP Switch Block S1 and S2

These switch blocks are oriented horizontally on the developer's kit board; **on** is up.

### 4.4.2 S1: Transmission Parameters

Switch block S1 sets the transmission rate, parity, stop bits, and protocol. Switch block S2 selects the device address when using the multi-drop protocol. A blank space in the table indicates the switch should be off (to the left).

		S1-1	S1-2	S1-3	S1-4	S1-5	S1-6	S1-7	S1-8	S2-1
Transmission rate (bits per second)	1200	on	on	on	on					
	2400		on	on	on					
	9600	on		on	on					
	19200			on	on					
	57600	on	on		on					
	115200		on		on					
	250000	on			on					
	416667				on					
Parity	None					on	on			
	Odd						on			
	Even					on				
Stop bits	1							on		
	2									
Protocol	Point-to-point								on	on
	Address bit								on	
	Idle line									

### 4.4.3 S2: Serial Device Address

Switch block S3 sets the device address for multi-drop protocol systems. A blank space in the table indicates the switch should be off (to the left).

NOTE: S2-2 and S2-3 must be left in the ON position.

*Multi-drop Address Selection (S2)*

Address	S2-4	S2-5	S2-6	S2-7	S2-8
0	on	on	on	on	on
1		on	on	on	on
2	on		on	on	on
3			on	on	on
4	on	on		on	on
5		on		on	on
6	on			on	on
7				on	on
8	on	on	on		on
9		on	on		on
10	on		on		on
11			on		on
12	on	on			on
13		on			on
14	on				on
15					on
16	on	on	on	on	
17		on	on	on	
18	on		on	on	
19			on	on	
20	on	on		on	
21		on		on	
22	on			on	
23				on	
24	on	on	on		
25		on	on		
26	on		on		
27			on		
28	on	on			
29		on			
30	on				
31					



## 4.5 Outputs to Motor Amplifiers

The Navigator developer's kit supports three types of output to the motor amplifiers:

**DAC** Analog signals from the on-board D/A converters

**PWM 50/50** Pulse-width modulated square-wave signals with a 50% duty cycle

**PWM sign-magnitude** Pulse-width modulated signals with definable duty cycle and direction

These outputs should be connected from the designated J4 pins to the appropriate amplifier inputs, as shown in the following tables. The names of the input pins may vary among amplifiers; common names are shown.

### 4.5.1 Brushed Servo Motors (MC2100/2800 Series)

			<i>J4 connection (Header-pin)</i>			
	Signal name	Amplifier input	Axis 1	Axis 2	Axis 3	Axis 4
DAC	DACAn	Ref+ or V+	Hdr1-22	Hdr1-47	Hdr2-22	Hdr2-47
	GNDn	Ref- or Gnd	Hdr1-24	Hdr1-49	Hdr2-24	Hdr2-49
PWM sign/magnitude	PWMMagAn	PWM magnitude	Hdr1-18	Hdr1-43	Hdr2-18	Hdr2-43
	PWMSignAn	PWM direction	Hdr1-21	Hdr1-46	Hdr2-21	Hdr2-46

### 4.5.2 Brushless Servo Motors (MC2300/2800 Series)

			<i>J4 connection (Header-pin)</i>			
	Signal name	Amplifier input	Axis 1	Axis 2	Axis 3	Axis 4
DAC	DACAn	Ref1+ or V1+	Hdr1-22	Hdr1-47	Hdr2-22	Hdr2-47
	DACBn	Ref2+ or V2+	Hdr1-23	Hdr1-48	Hdr2-23	Hdr2-48
	GNDn	Ref- or Gnd	Hdr1-24	Hdr1-49	Hdr2-24	Hdr2-49
PWM 50/50	PWMMagAn	PWM phase 1	Hdr1-18	Hdr1-43	Hdr2-18	Hdr2-43
	PWMMagBn	PWM phase 2	Hdr1-19	Hdr1-44	Hdr2-19	Hdr2-44
	PWMMagCn	PWM phase 3	Hdr1-20	Hdr1-45	Hdr2-20	Hdr2-45

### 4.5.3 Microstepping Motors (MC2400 Series)

			<i>J4 connection (Header-pin)</i>			
	Signal name	Amplifier input	Axis 1	Axis 2	Axis 3	Axis 4
DAC	DACAn	Ref+ or V+	Hdr1-22	Hdr1-47	Hdr2-22	Hdr2-47
	GNDn	Ref- or Gnd	Hdr1-24	Hdr1-49	Hdr2-24	Hdr2-49
PWM sign/magnitude	PWMMagAn	PWM magnitude	Hdr1-18	Hdr1-43	Hdr2-18	Hdr2-43
	PWMSignAn	PWM direction	Hdr1-21	Hdr1-46	Hdr2-21	Hdr2-46
	PWMMagBn	PWM magnitude	Hdr1-19	Hdr1-44	Hdr2-19	Hdr2-44
	PWMSignBn	PWM direction	Hdr1-20	Hdr1-45	Hdr2-20	Hdr2-45

When using the MC2800 series, refer to the above tables in accordance to how you have each axis configured.

## 4.6 Encoder Inputs

Resistor packs RS1 – RS3

These three resistor packs are at the left end of the developer's kit board, next to the 100-pin connector J4. When using differential encoders, leave these packs in place. When using open-ended encoders, remove all three packs and connect encoder signals to the positive encoder input only. The negative input can be left unconnected. Encoder connections are shown below.

### Encoder connections when using differential encoder input

Signal	J4 pin connections			
	Axis 1	Axis 2	Axis 3	Axis 4
QuadAn+	Hdr1-1	Hdr1-26	Hdr1-51	Hdr1-76
QuadAn-	Hdr1-2	Hdr1-27	Hdr1-52	Hdr1-77
QuadBn+	Hdr1-3	Hdr1-28	Hdr1-53	Hdr1-78
QuadBn-	Hdr1-4	Hdr1-29	Hdr1-54	Hdr1-79
Indexn+	Hdr1-5	Hdr1-30	Hdr1-55	Hdr1-80
Indexn-	Hdr1-6	Hdr1-31	Hdr1-56	Hdr1-81
V <sub>cc</sub>	Hdr1-7	Hdr1-32	Hdr1-57	Hdr1-82
GND	Hdr1-8	Hdr1-33	Hdr1-58	Hdr1-83

### Encoder connections when using single-ended encoder input

Signal	J4 pin connections			
	Axis 1	Axis 2	Axis 3	Axis 4
QuadAn	Hdr1-1	Hdr1-26	Hdr1-51	Hdr1-76
QuadBn	Hdr1-3	Hdr1-28	Hdr1-53	Hdr1-78
Indexn	Hdr1-5	Hdr1-30	Hdr1-55	Hdr1-80
V <sub>cc</sub>	Hdr1-7	Hdr1-32	Hdr1-57	Hdr1-82
GND	Hdr1-8	Hdr1-33	Hdr1-58	Hdr1-83

## 5 PCI DK1 Hardware Information

---

### 5.1 Environmental and Electrical Ratings

*All ratings and ranges are for both the I/O and CP chips.*

<b>Dimensions</b>	4.25" x 12.25", PCI Adapter
<b>Storage Temperature</b>	-40 °C to 125 °C
<b>Operating Temperature</b>	0 °C to 70 °C*
<b>Power Consumption</b>	1A @ 5V; 83mA @ 12V
<b>Supply Voltage Limits</b>	-0.3V to +7.0V
<b>Supply Voltage Operating Range</b>	4.75V to 5.25V
<b>Analog Output Range</b>	-10.0V to 10.0V
<b>Analog Input Range</b>	0.0V to 5.0V

### 5.2 PLX PCI chip information

The developer's kit utilizes the PCI 9030 interface chip from PLX technology. For information on the operation of this device please refer to the PLX documentation available from <http://www.plxtech.com/>.

The following table lists the relevant PLX local configuration register values. For further information refer to the PLX documentation.

Space	Range	Remap	Descriptor	ChipSelect
0 (DPRAM)	0FFF0000	0001	00402081	0001
1 (CP)	0FFFFFE0	8001	00440480	8009

The following constants are used with the PlxBus\_xxx functions.

```
PCI_IOSPACE0_BASE = 8000
```

```
PCI_IOSPACE1_BASE = 800
```

```
PMD_data_port      = PCI_IOSPACE0_BASE + 0
```

```
PMD_cmd_status_port = PCI_IOSPACE0_BASE + 2
```

```
PMD_reset_port     = PCI_IOSPACE0_BASE + 10
```

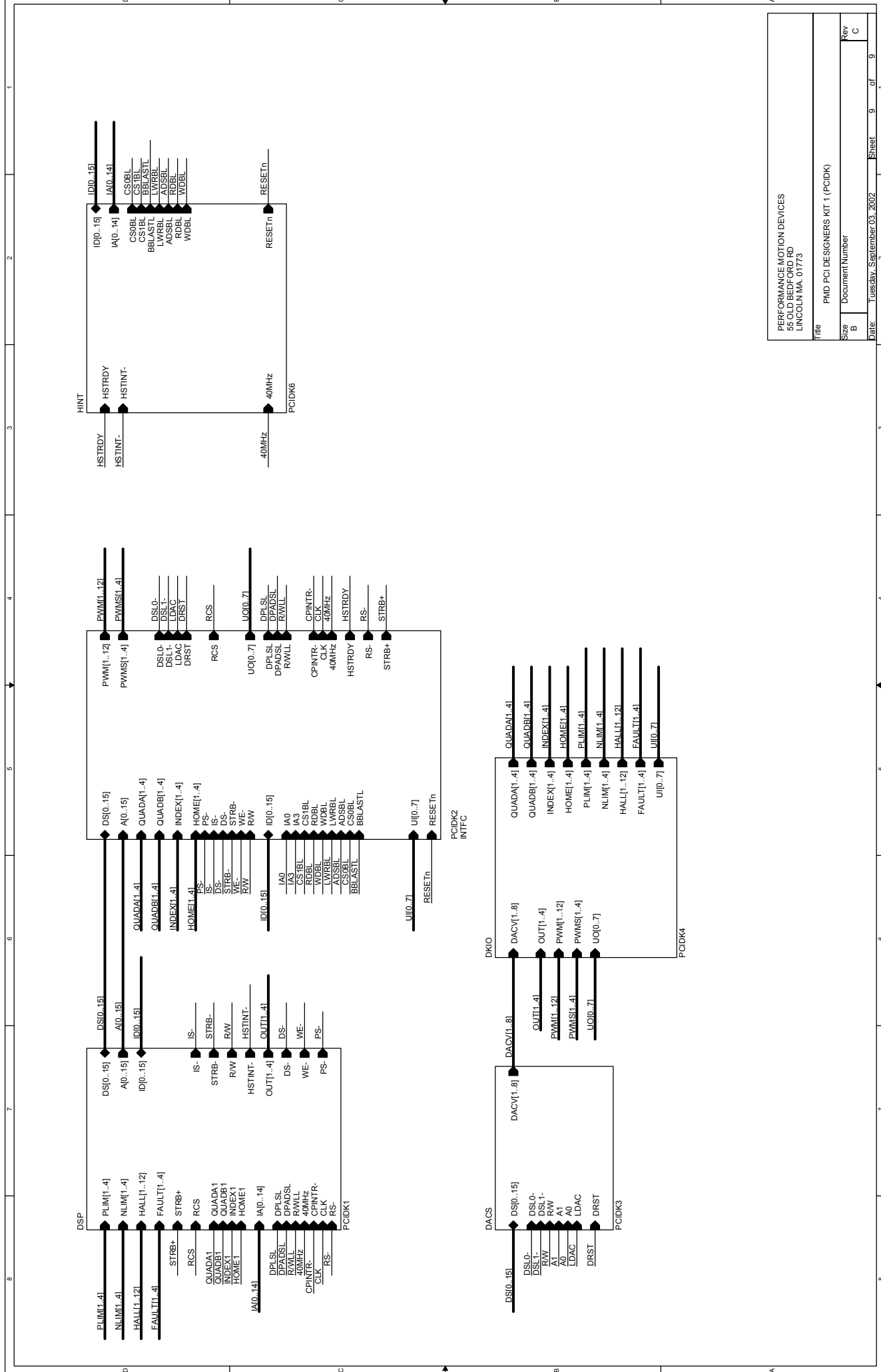
```
Dual_port_RAM      = PCI_IOSPACE1_BASE
```

## 5.3 PCI DK1 Schematics

### 5.3.1 PCI DK1 Assembly Drawing



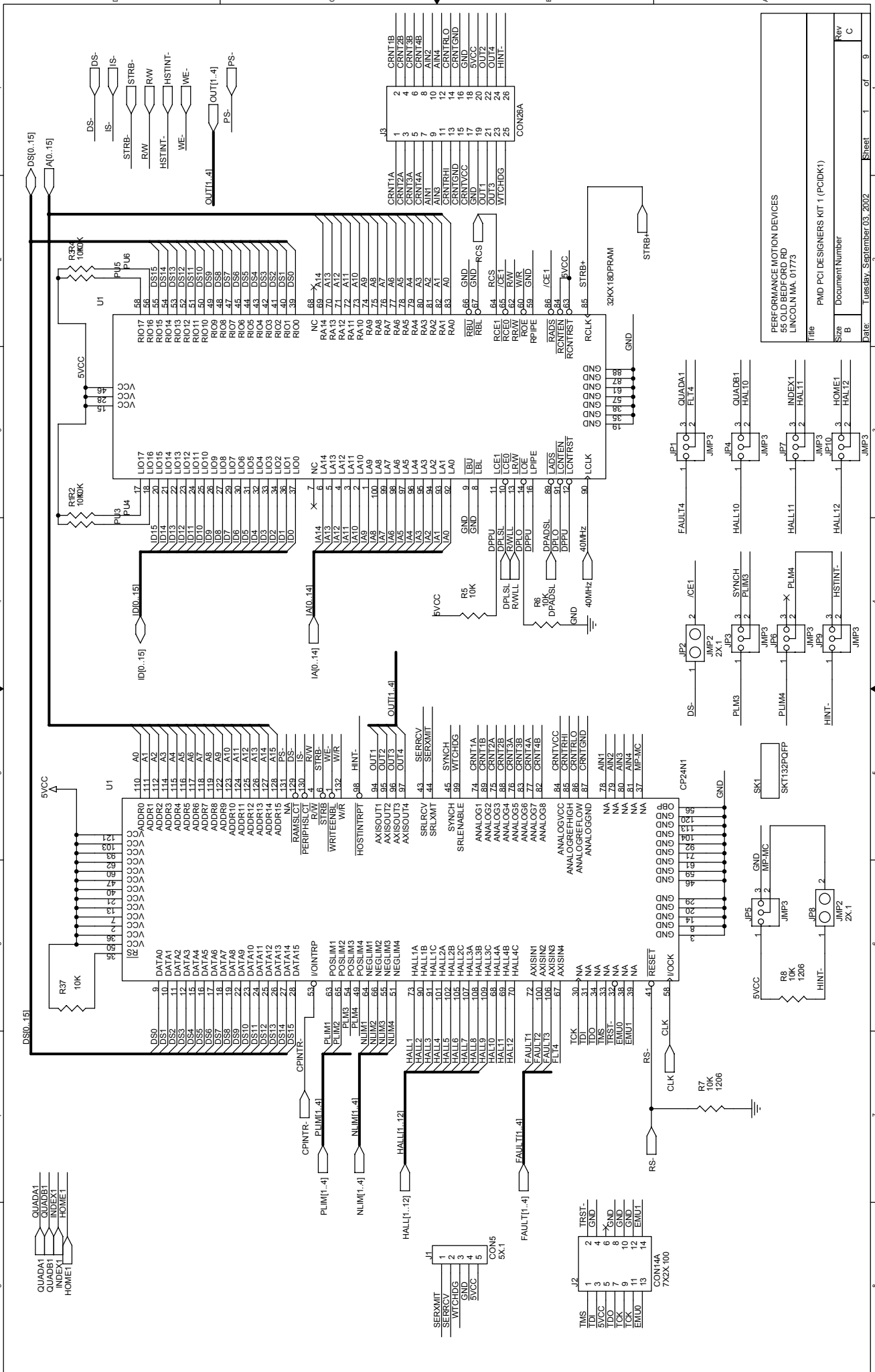
### 5.3.2 PCI DK1 Overview Schematic



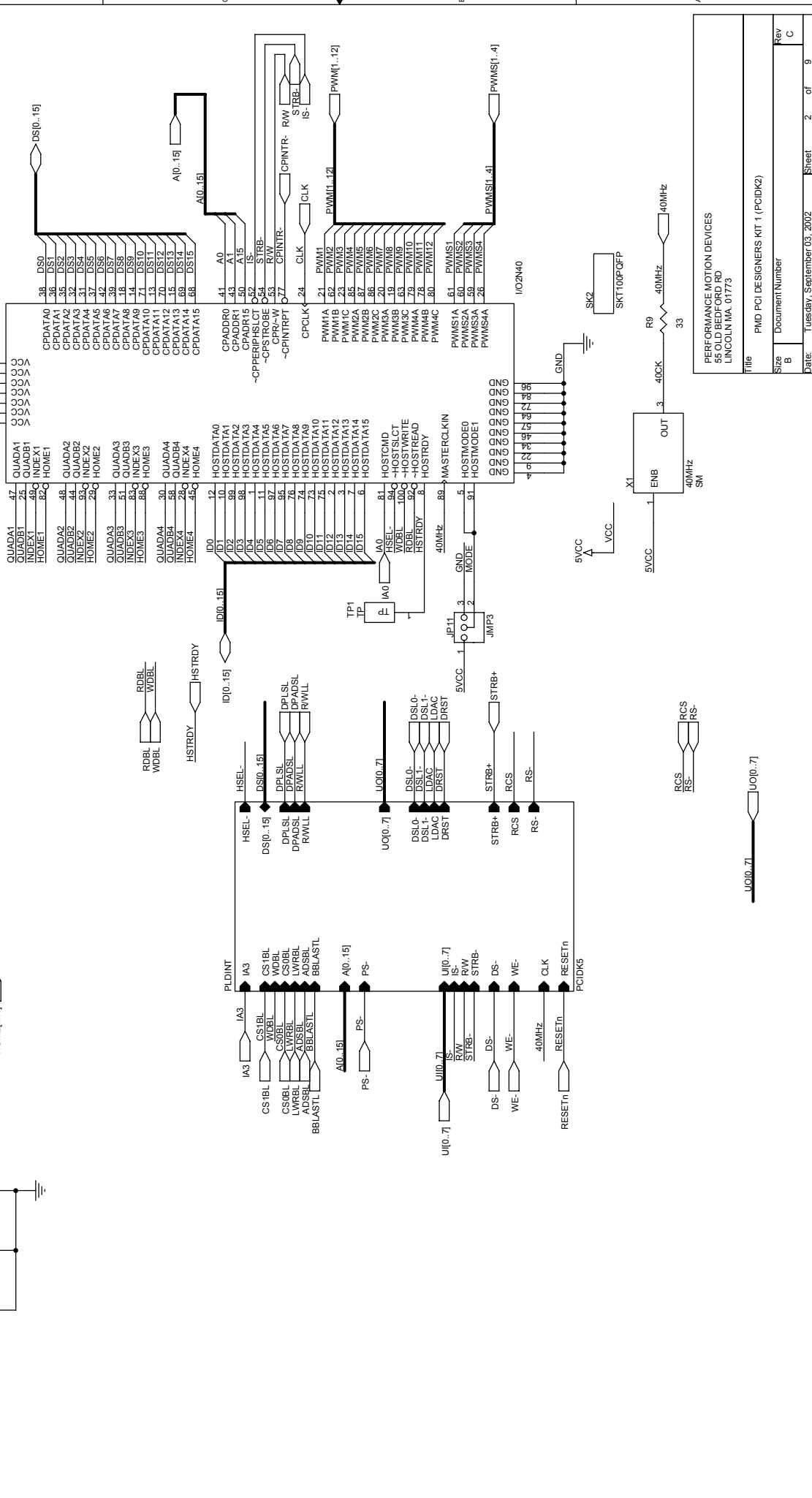
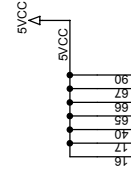
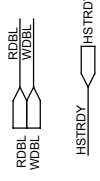
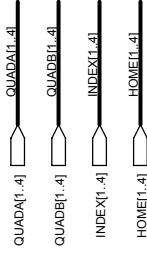
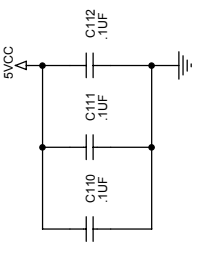
PERFORMANCE MOTION DEVICES 55 OLD BEDFORD RD LINCOLN, MA. 01773	
Title: PIMD PCI DESIGNERS KIT 1 (PCIDK)	
Size: B	Document Number
Date: Tuesday, September 03, 2002	Sheet: 9 of 9
Rev: C	

5.3.3 PCI DK1 CP and DPRAM Schematic

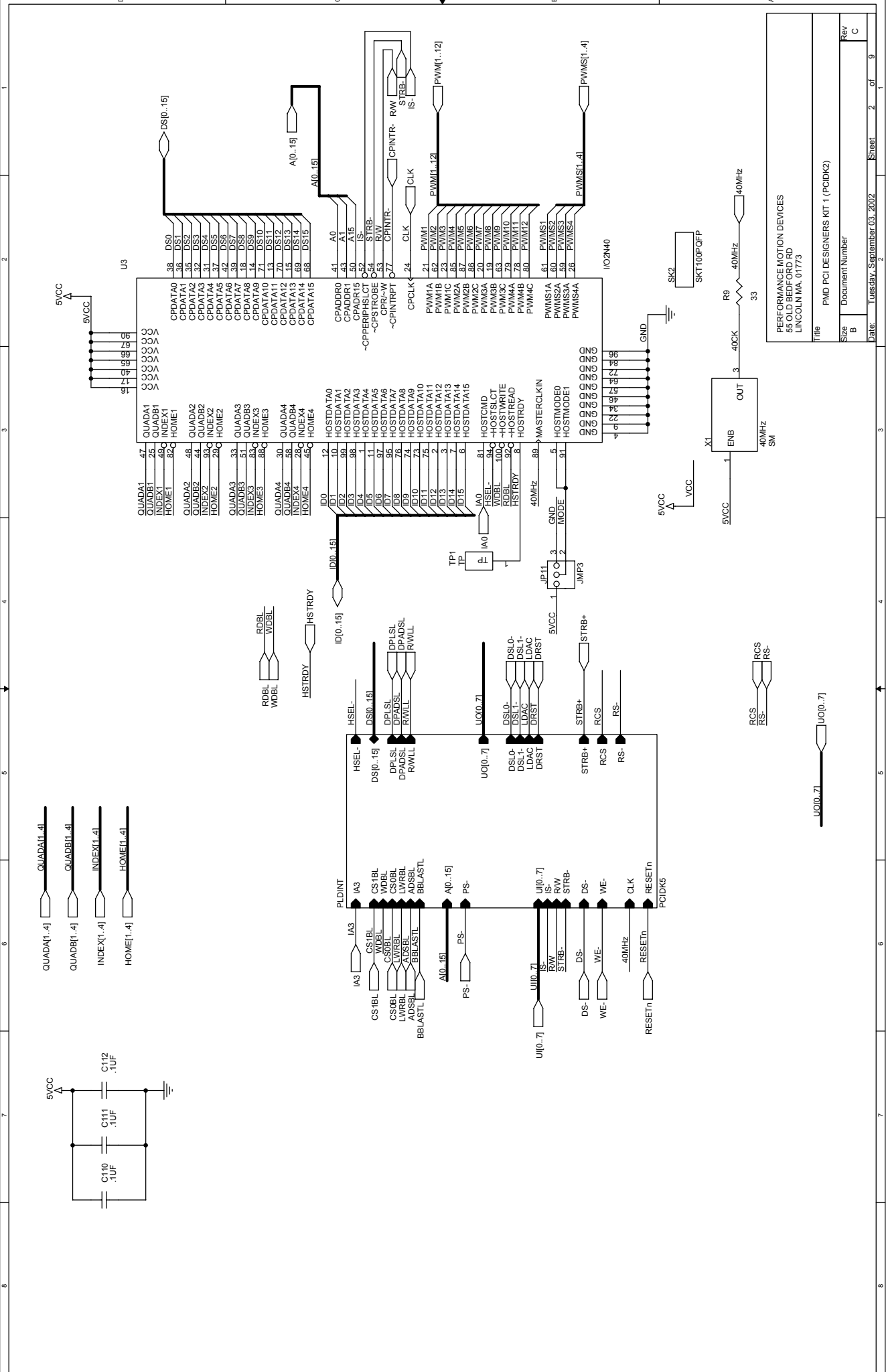




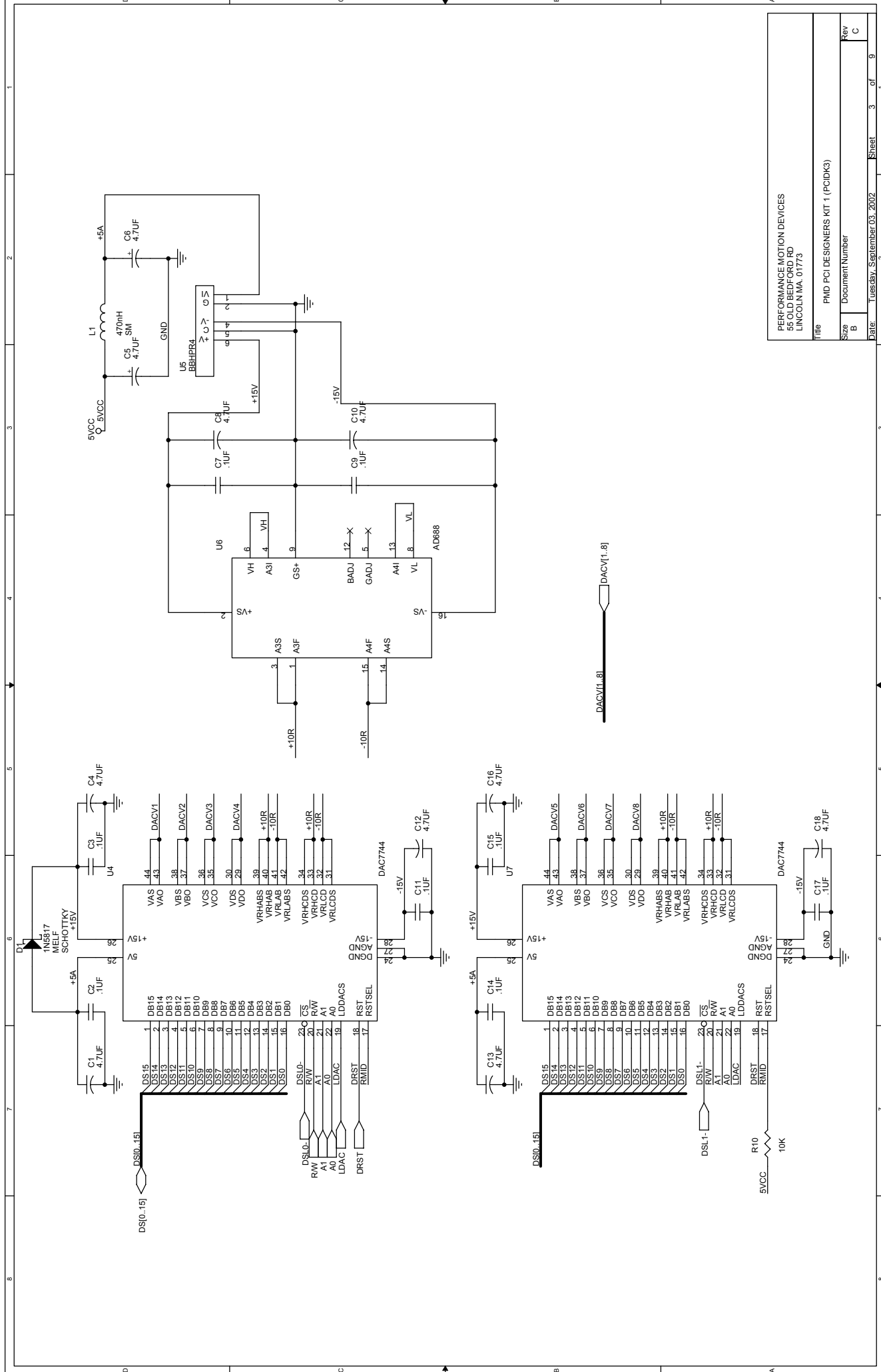
### 5.3.4 PCI DK1 I/O Schematic



PERFORMANCE MOTION DEVICES 55 OLD BEDFORD RD LINCOLN, MA 01773	
Title	PMD PCI DESIGNERS KIT 1 (PCIDK2)
Size	Document Number
Date	Tuesday, September 03, 2002
Sheet	2 of 9
Rev	C

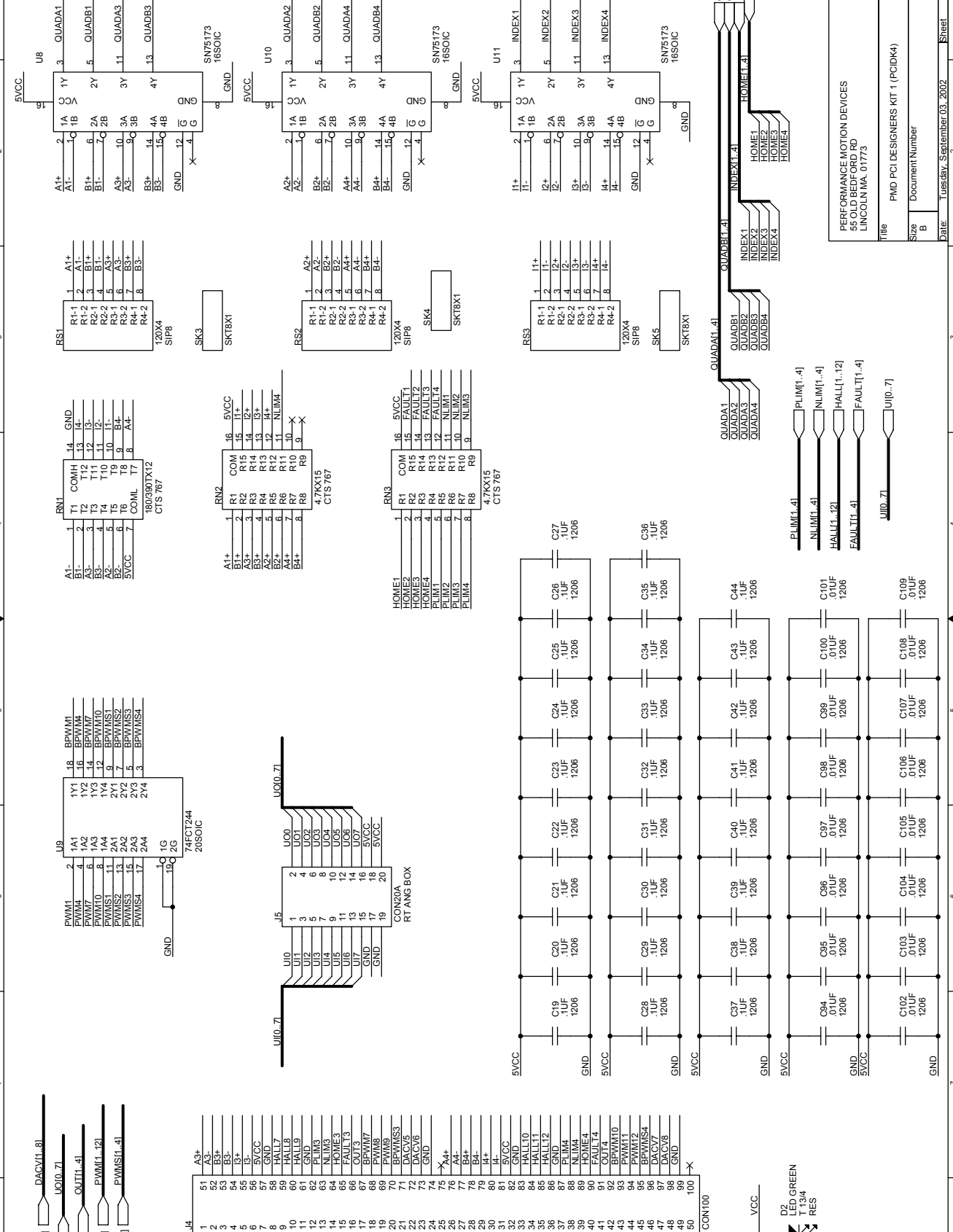


### 5.3.5 PCI DK1 DAC Amplifiers Schematic



PERFORMANCE MOTION DEVICES 55 OLD BEDFORD RD LINCOLN MA, 01773	
Title	PMD PCI DESIGNERS KIT 1 (PCIDK3)
Size	Document Number
B	
Date	Tuesday, September 03, 2002
Sheet	3 of 9
Rev	C

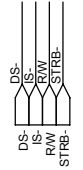
### 5.3.6 PCI DK1 Connector and Quadrature Schematic



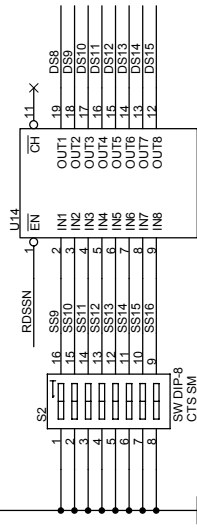
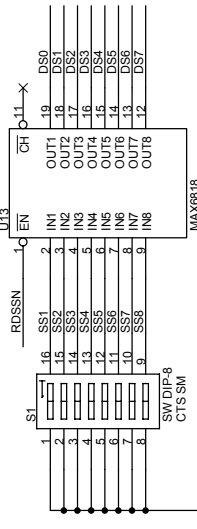
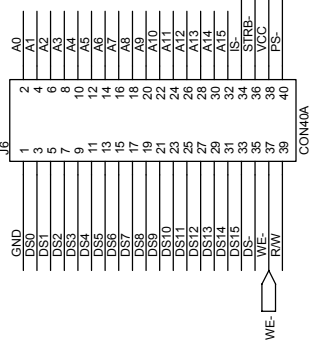
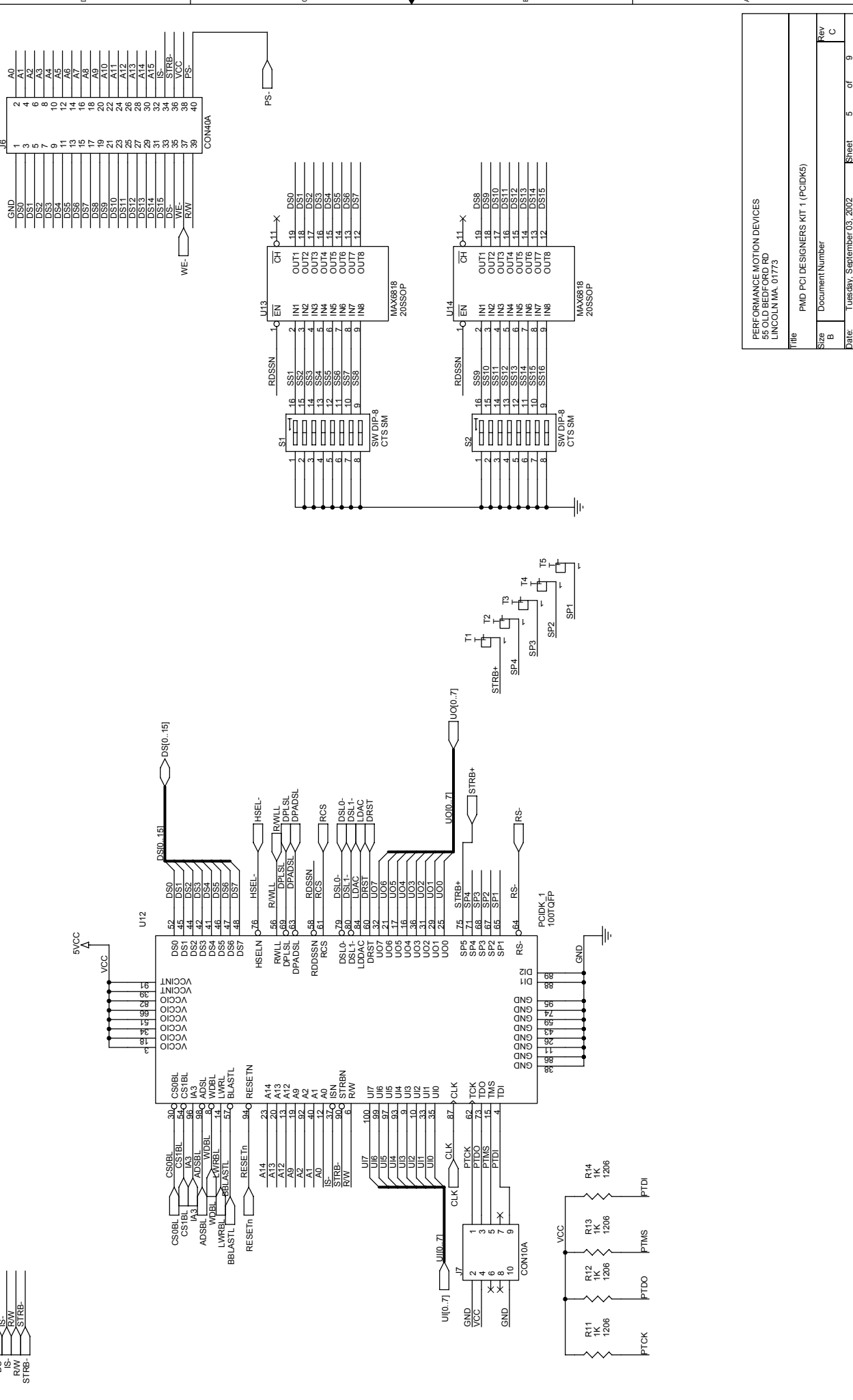
1	A1+
2	B1+
3	I1+
4	A2+
5	B2+
6	I2+
7	A3+
8	B3+
9	I3+
10	A4+
11	B4+
12	I4+
13	A5+
14	B5+
15	I5+
16	A6+
17	B6+
18	I6+
19	A7+
20	B7+
21	I7+
22	A8+
23	B8+
24	I8+
25	A9+
26	B9+
27	I9+
28	A10+
29	B10+
30	I10+
31	A11+
32	B11+
33	I11+
34	A12+
35	B12+
36	I12+
37	A13+
38	B13+
39	I13+
40	A14+
41	B14+
42	I14+
43	A15+
44	B15+
45	I15+
46	A16+
47	B16+
48	I16+
49	A17+
50	B17+
51	I17+
52	A18+
53	B18+
54	I18+
55	A19+
56	B19+
57	I19+
58	A20+
59	B20+
60	I20+
61	A21+
62	B21+
63	I21+
64	A22+
65	B22+
66	I22+
67	A23+
68	B23+
69	I23+
70	A24+
71	B24+
72	I24+
73	A25+
74	B25+
75	I25+
76	A26+
77	B26+
78	I26+
79	A27+
80	B27+
81	I27+
82	A28+
83	B28+
84	I28+
85	A29+
86	B29+
87	I29+
88	A30+
89	B30+
90	I30+
91	A31+
92	B31+
93	I31+
94	A32+
95	B32+
96	I32+
97	A33+
98	B33+
99	I33+
100	A34+

### 5.3.7 PCI DK1 PLD Schematic



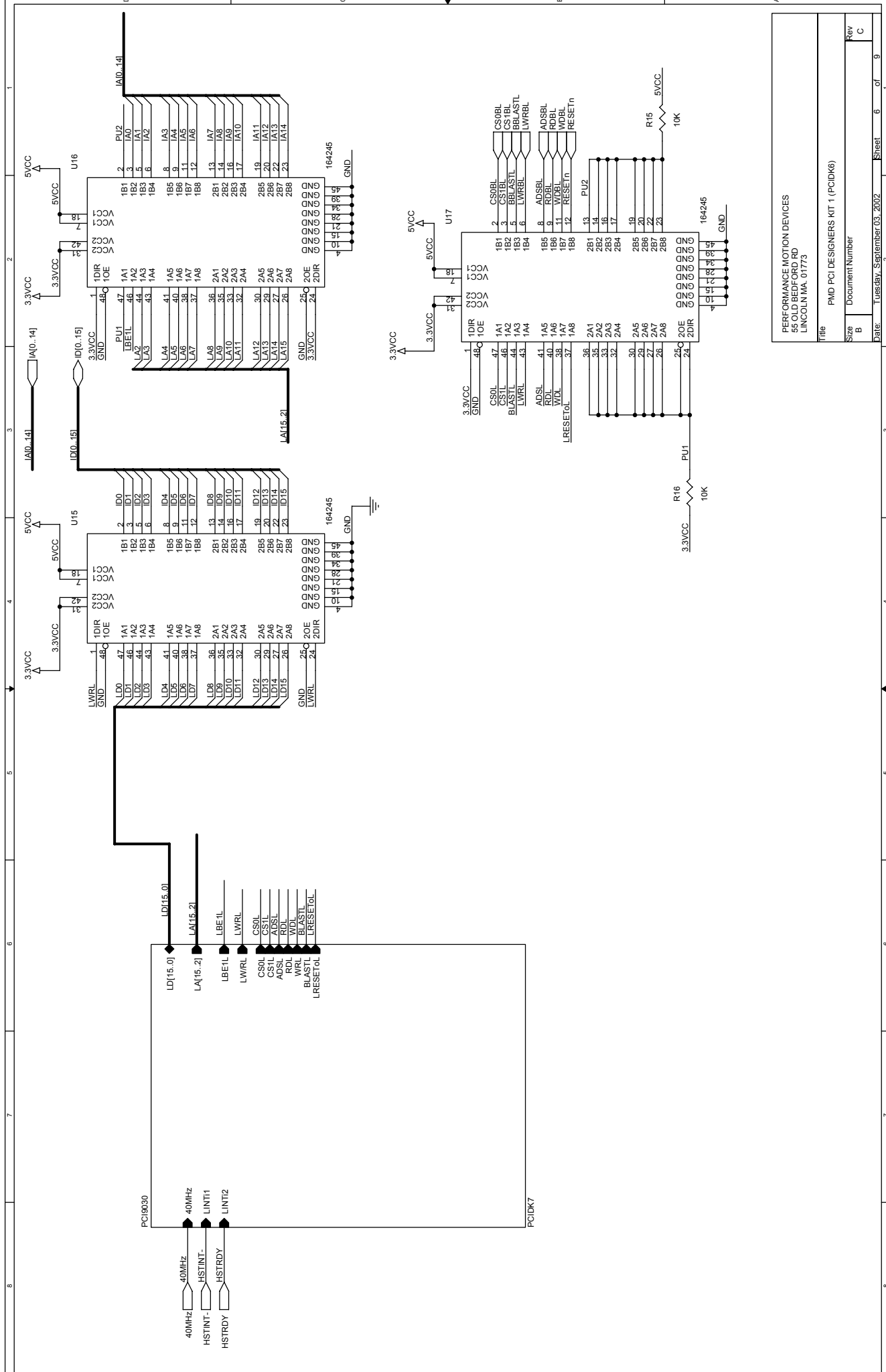


A10..15]



PERFORMANCE MOTION DEVICES 55 OLD BEDFORD RD LINCOLN MA, 01773	
Title	PMD PCI DESIGNERS KIT 1 (PCIDK5)
Size	Document Number
Date	Tuesday, September 03, 2002
Sheet	5 of 9
Rev	C

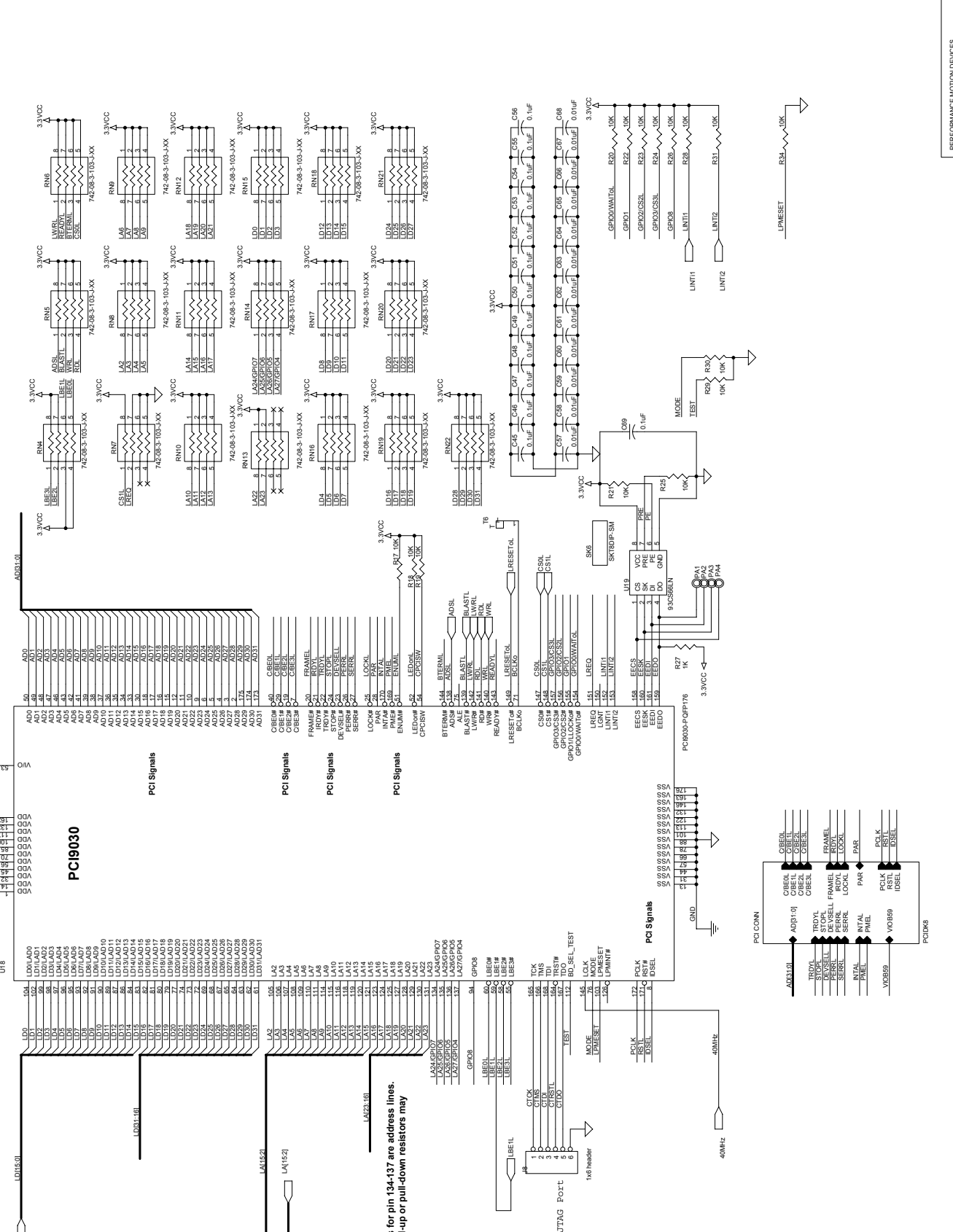
### 5.3.8 PCI DK1 Buffer Schematic



PERFORMANCE MOTION DEVICES  
 55 OLD BEDFORD RD  
 LINCOLN, MA, 01773

Title		PMD PCI DESIGNERS KIT 1 (PCIDK6)	
Size	B	Document Number	
Date	Tuesday, September 03, 2002	Sheet	6 of 9
Rev	C		

### 5.3.9 PCI DK1 PLX9030 Schematic

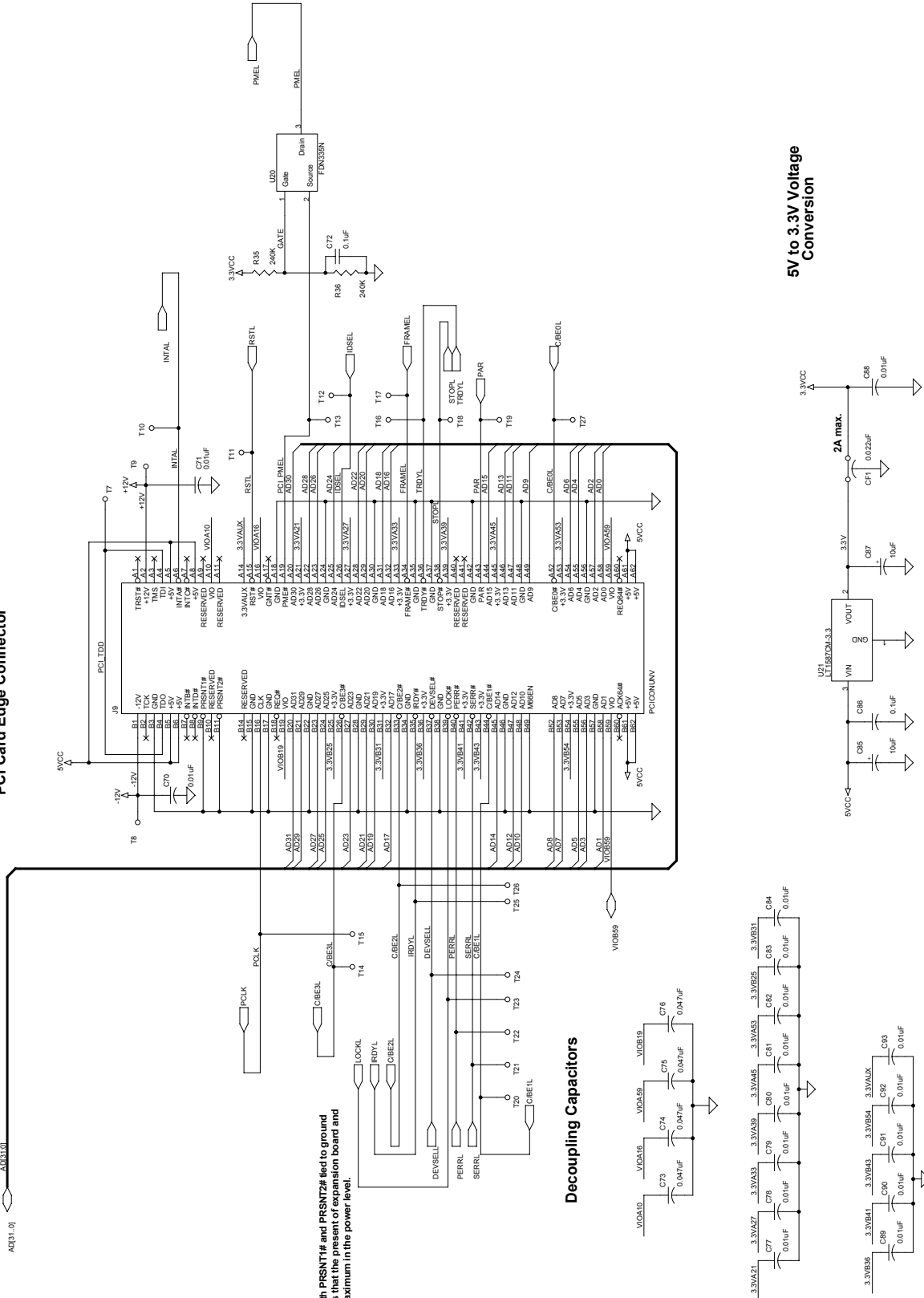


Default settings for pin 134-137 are address lines.  
 Otherwise, pull-up or pull-down resistors may  
 be required.

PERFORMANCE MOTION DEVICES 55 OLD BEDFORD RD LINCOLN, MA 01775		
File	PMD-PCI-DESIGNERS-KIT-1 (PCID47)	
Size	Document Number	
Rev		
Doc	1-03-2002	Sheet 7 of 9

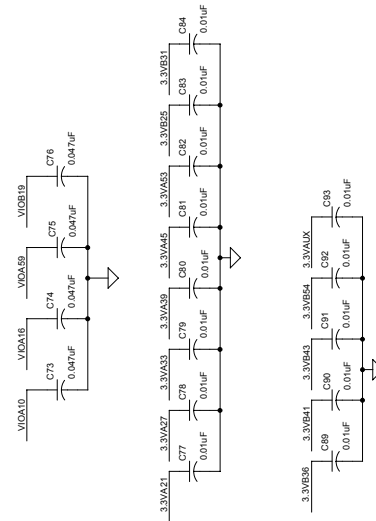
### 5.3.10 PCI DK1 PCI Edge Connector Schematic

## PCI Card Edge Connector

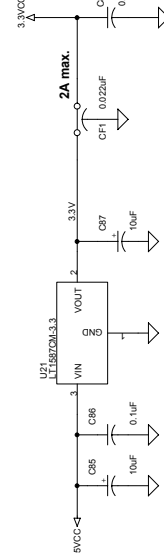


Note: both PRSNT1# and PRSNT2# led to ground indicates that the present of expansion board and 7.5 W maximum in the power level.

### Decoupling Capacitors



### 5V to 3.3V Voltage Conversion



PCI Card Edge Connector

PERFORMANCE MOTION DEVICES  
 7500 N. LITTLETON BLVD.  
 LITTLETON, CO. 80120

File	PMID_PCI DESIGNERS KIT 1 (PCIDK6)
Size	<Doc>
Doc	C
Date	TUESDAY, SEPTEMBER 03, 2002 3:02:00 PM