Pilot™ Motion Processor

PCI Developer's Kit Manual

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

Revision 1.2, November 2002

NOTICE

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

Pilot Motion Processor User's Guide (MC3000UG)

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

Pilot Motion Processor Programmer's Reference (MC3000PR)

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

Pilot Motion Processor Technical Specifications

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

MC3110, for brushed servo motion control (MC3110TS);
MC3310, for brushless servo motion control (MC3310TS);
MC3410, for micro-stepping motion control (MC3410TS);
MC3510, for stepping motion control (MC3510TS).

Pilot Motion Processor Developer's Kit Manual (PCIDK3000M)

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

Table of Contents

1 Installation

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

Dev. Kit P/N	Installed Motion Processor
PCIDK3110	MC3110
PCIDK3310	MC3310
PCIDK3410	MC3410
PCIDK3510	MC3510

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.

In addition to the PC card, the developer's kit also includes a CD-ROM 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 Pilot motion processors. Also included on the CD-ROM is Pro-Motion, an interactive monitor program that supports all of the Pilot commands and motion processors. Pro-Motion is useful to test and exercise your system before you start 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 Pilot motion processors is found in section 1.5 onwards.

Once this hardware configuration is complete, you should then install the software. Installation of the software is described in section 1.10, Software Installation.

The final step to finish the installation is to perform a functional test of the finished system. This is described in section 1.11, 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 Pilot developer's kit contains the following components:

1) PCI DK1 Board
2) CD-ROM containing software and documentation in PDF format
3) 100 pin connector to dual 50-pin header converter cable

4) Documentation:

Pilot Motion Processor Developer's Kit Manual
Pilot Motion Processor User's Guide
Pilot Motion Processor Programmers Reference
Pilot Motion Processor Technical Reference (for either MC3110, MC3310, MC3410, or MC3510 motion processor depending on the specific developer's kit you purchased)

If any of these components is 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 (RS232 or RS485) is optional for both the minimum and recommended platforms.
2) A pulse and direction, PWM, or analog-input amplifier. The type of amplifier depends on the developer's kit motion processor type.
3) A step motor or servo motor. This motor 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 amplifier and the servo motor. 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

If operated in serial mode, switch banks S1and S2 set the serial configuration, which is described in section 4.4.

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

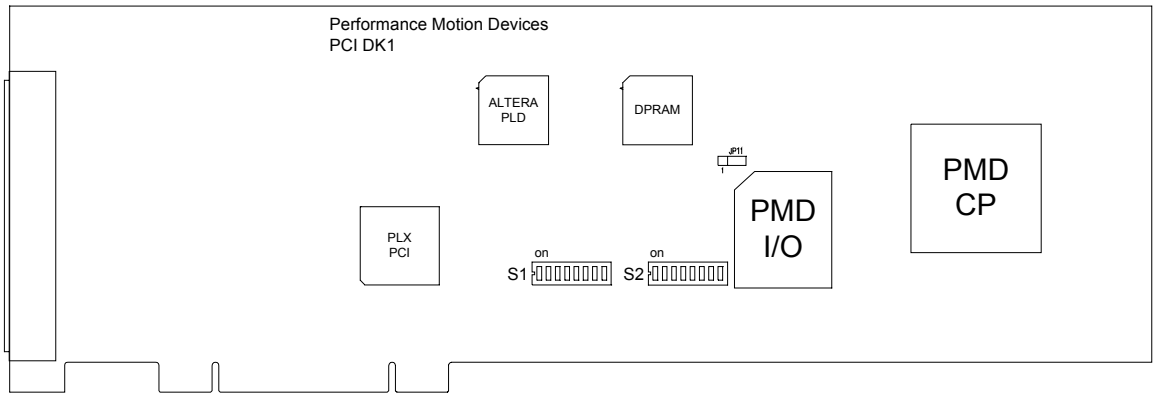


Figure 1-1. Settable switch blocks

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.

1.5 PCI DK3110 Connections Summary

The following table summarizes the connections provided and expected by the PCI DK1 PC board when an MC3110 motion processor is installed.

Motion Processor:	MC3110 Series (brushed servo)
Maximum # of Axes:	1
Encoder Input Type:	Incremental encoder
Encoder Input Signals:	A quadrature channel input B quadrature channel input Index pulse channel input
# Motor Output Channels	1
Amplifier Output Signals (if PWM sign, magnitude used)	PWM Direction PWM magnitude
Amplifier Output Signals (if PWM 50/50 used)	PWM magnitude
Amplifier Output Signals (if analog output used)	Analog out (DAC output)
Other Control Signals:	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 DK3310 Connections Summary

The following table summarizes the connections provided and expected by the PCI DK1 PC board when an MC3310 motion processor is installed.

Motion Processor	MC3310 (Brushless Servo)
Maximum # of Axes:	1
Encoder Input Type:	Incremental encoder
Encoder Input Signals:	A quadrature channel input B quadrature channel input Index pulse channel input
# motor output channels	2 or 3 depending on motor output selected and # phases
Amplifier Output Signals (if PWM 50/50 used)	PWM magnitude (phase A) PWM magnitude (phase B) PWM magnitude (phase C)
Amplifier Output Signals (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:	Home signal channel input Positive limit switch input Negative limit switch input AxisIn input AxisOut output
Miscellaneous Signals:	GND +5 V (for encoder power)

1.7 PCI DK3410 Connections Summary

The following table summarizes the connections provided and expected by the PCI DK1 PC board when an MC3410 motion processor is installed.

Motion Processor:	MC3410 (microstepping)
Maximum # of Axes:	1
Encoder Input Type:	Incremental encoder (optional)
Encoder Input Signals:	A quadrature channel input B quadrature channel input Index pulse channel input
# Motor Output Channels	2 or 3 depending on motor output selected and # phases
Amplifier Output Signals (if PWM sign, magnitude used)	PWM direction PWM magnitude

Amplifier Output Signals (if PWM 50/50 used)	PWM magnitude
Amplifier Output Signals (if analog output used)	Analog out (DAC output)
Other Control Signals:	Home signal channel input Positive limit switch input Negative limit switch input AxisIn input AxisOut output
Miscellaneous Signals:	GND +5 V (for encoder power)

1.8 PCI DK3510 Connections Summary

The following table summarizes the connections provided and expected by the PCI DK1 PC board when an MC3410 motion processor is installed.

Motion Processor:	MC3510 (stepping)
Maximum # of Axes:	1
Encoder Input Type:	Incremental encoder (optional)
Encoder Input Signals:	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)

1.9 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 motor should remain stationary. If the motor does 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.

1.10 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/XP.

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 Pilot 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.

1.11 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 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 *Pilot User's Guide* that is relevant to the installed motion processor.

For the MC3110 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 MC3310 motion processor, 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
MC3110	1	Set amplifier type (PWM sign/mag, PWM 50/50, DAC)
	4	Set filter parameters
	6	Make a trajectory move

Motion Processor	Step #	Operation
MC3310	1	Set amplifier type (PWM 50/50, DAC)
	2	initialize commutation
	3	Check commutation
	4	Set filter parameters
	6	Make a trajectory move
MC3410	1	Set amplifier type (PWM 50/50, DAC)
	5	Set the motor output power
	6	Make a trajectory move
MC3510	6	Make a trajectory move

Only perform the setup step sequences indicated above for the motion processor installed on your board.

1.11.1 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. 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 into the Command window in Pro-Motion:

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, and will then return you to the ">" prompt.

During the installation process, a shortcut to Pro-Motion was created on your start menu.

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

1.11.2 Step #2: Initialize the Commutation

NOTE: THIS SECTION APPLIES TO THE MC3310 MOTION PROCESSOR ONLY.

For the motor to be controlled properly using the MC3310, 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 *Pilot 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 wwww         // wwww is motor command.
InitializePhase
```

To determine the values of x, yyyy, zzzz, and wwww you should refer to the *Pilot Motion Processor User's Guide* MC3310 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 MC3310 section of the *Pilot 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

1.11.3 Step #3: Check Commutation

NOTE: THIS SECTION APPLIES TO THE MC3310 MOTION PROCESSOR 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

1.11.4 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.

1.11.5 Step #5: Set the Motor Command

NOTE: THIS SECTION APPLIES TO THE MC3410 MOTION PROCESSOR ONLY.

In order for motion to occur, the magnitude of the output must be set. Refer to the *Pilot 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
```

1.11.6 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)

2,500 and 2 respectively. With a cycle time of 100 uSec, these values correspond to roughly 381 counts/sec, and 1525 counts/sec^2, 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 2500          // 10000 is the desired maximum velocity
SetAcceleration 1         // 1 is the desired acceleration
SetDeceleration 1         // 1 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 100 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.

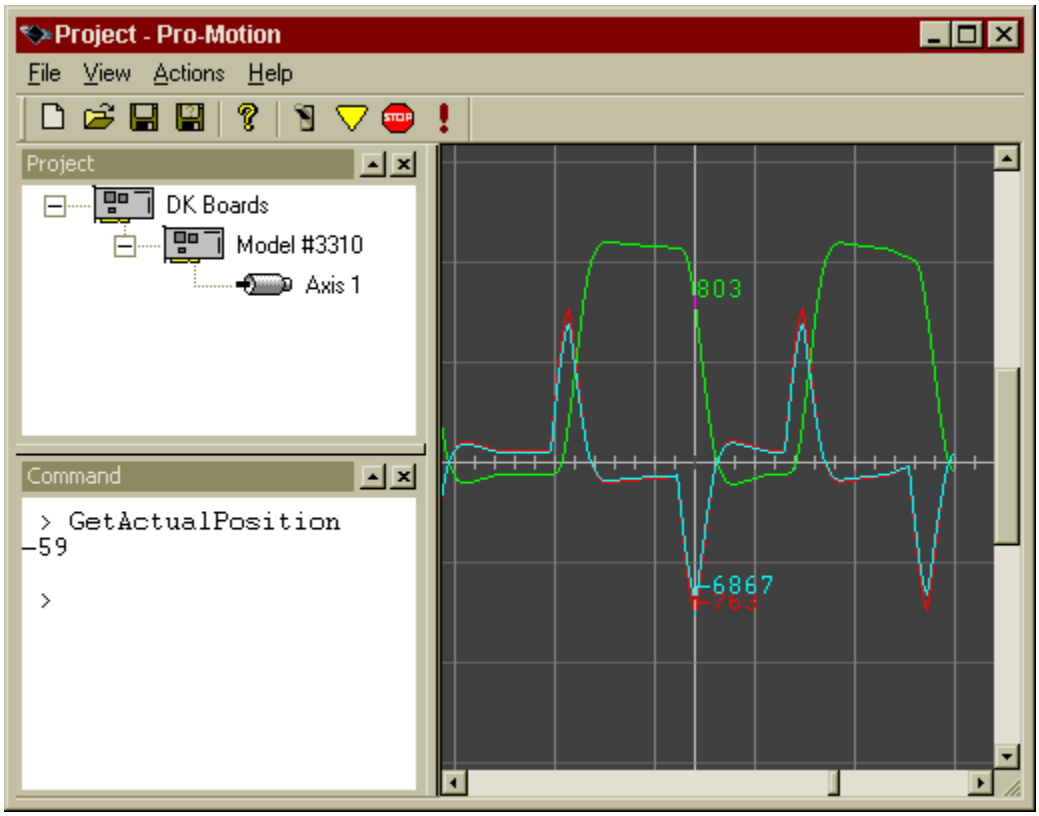
If you are still having problems after re-checking your system call PMD for applications assistance.

2 Using Pro-Motion™

The Pro-Motion program facilitates the exercising of the PMD motion processor installed in your *Pilot 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

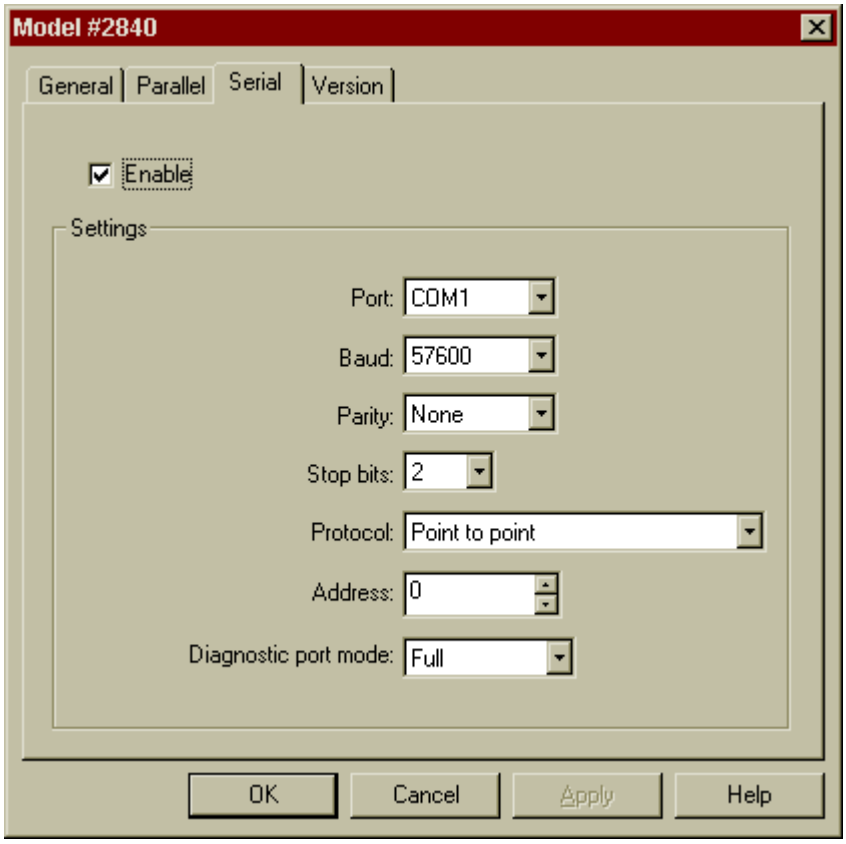
double-click on the second DK board icon in the Project window to display the DK board properties dialog and follow the directions below for the interface being used.

2.1.1 Parallel interface

To use the parallel interface the DK board must be installed in an available PCI slot. 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 is it 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. 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.

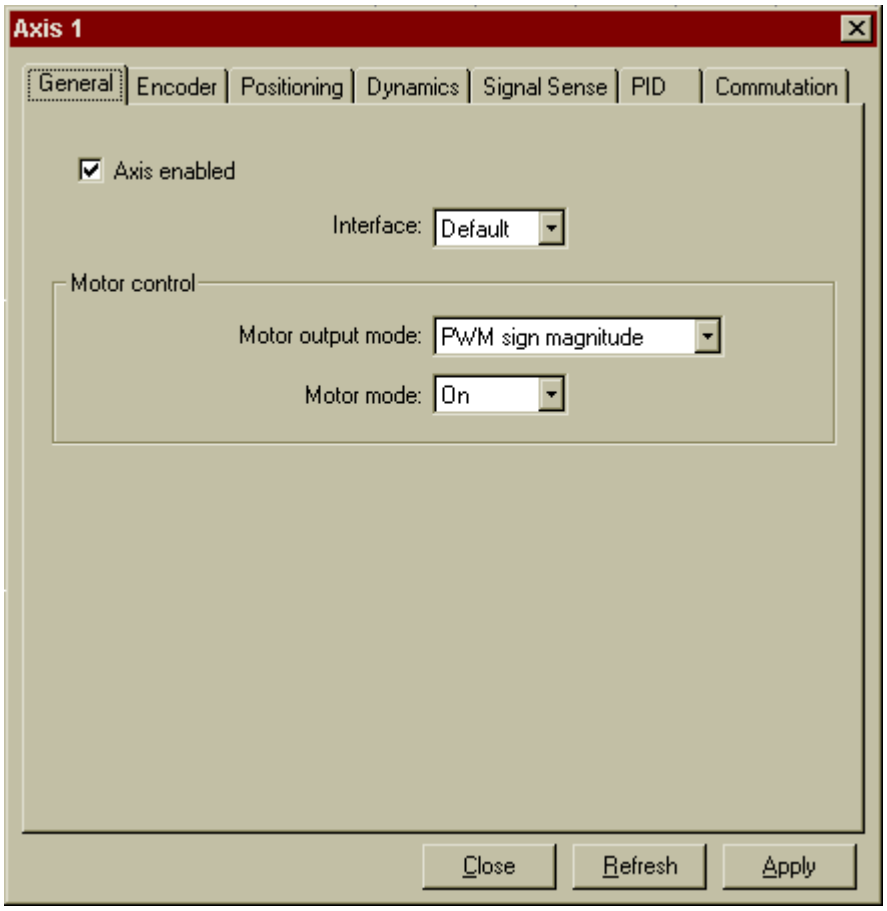
This section is similar to section 1.11, but has been modified for use with the Pro-Motion graphical user interface as opposed to the Command window. Refer to section 1.11 for a more detailed explanation of each step.

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

Motion Processor	Step #	Operation
MC3110	1	Set amplifier type (PWM sign/mag, PWM 50/50, DAC)
	4	Set filter parameters
	6	Make a trajectory move
MC3310	1	Set amplifier type (PWM 50/50, DAC)
	2	initialize commutation
	3	Check commutation
	4	Set filter parameters
	6	Make a trajectory move
MC3410	1	Set amplifier type (PWM 50/50, DAC)
	5	Set the motor output power
	6	Make a trajectory move
MC3510	6	Make a trajectory move

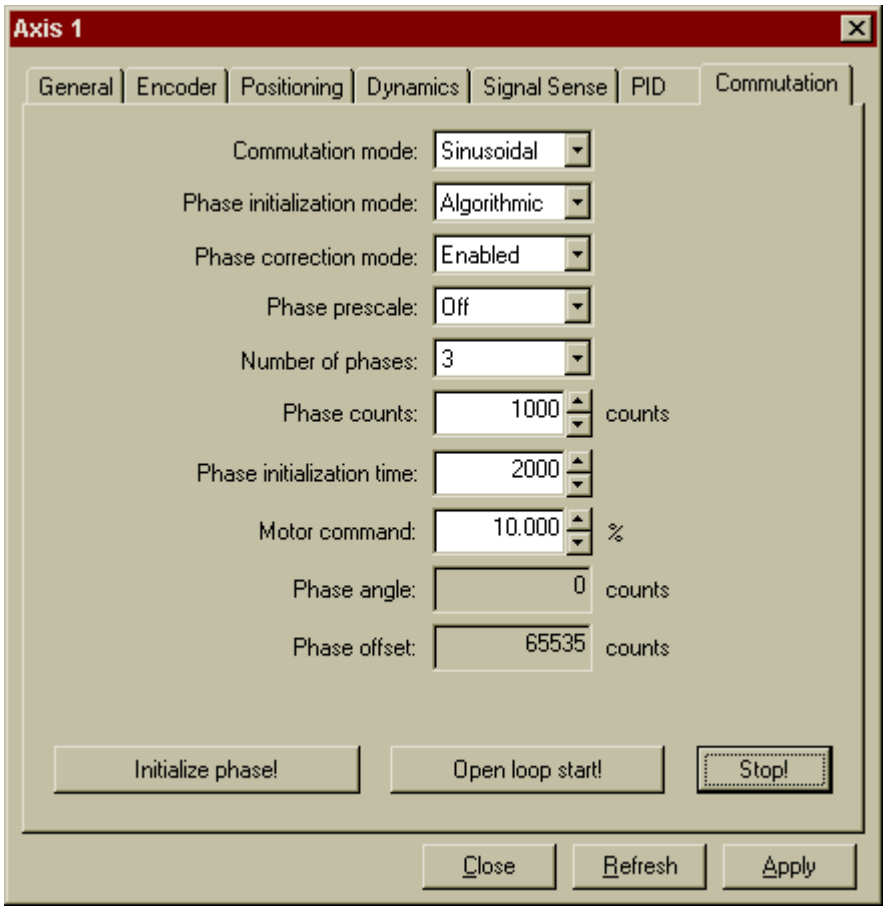
2.2.1 Step #1 Amplifier Type

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 pressed 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 Kp parameter to a low value between 1 and 25 initially to prevent any oscillations. Click the Apply button.

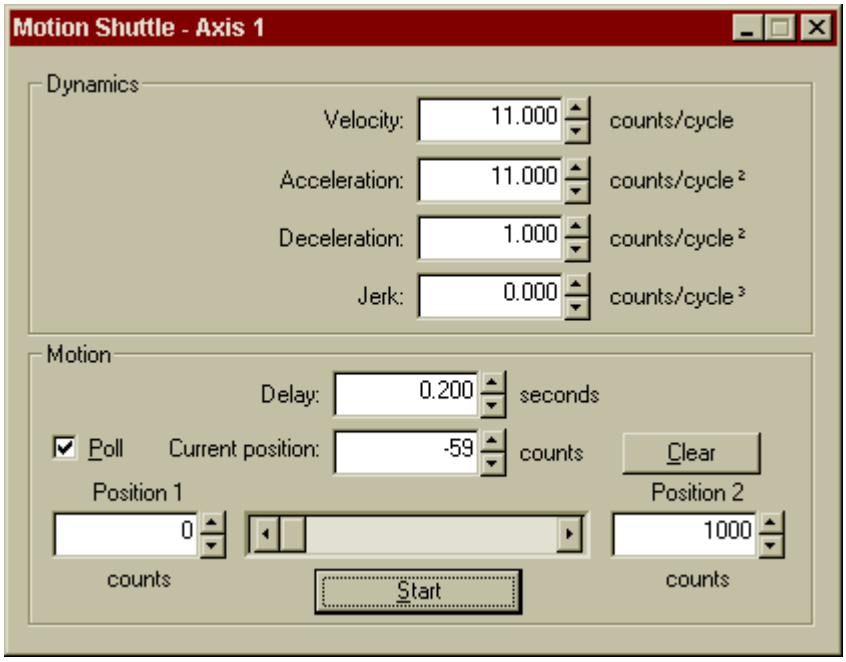
2.2.5 Step #5 Motor Output Power

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.

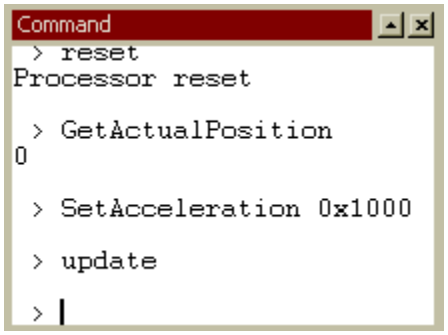


The Dynamics parameters should be set to the values previously set in the Dynamics page of the Axis properties dialog. Click the Start button. The motor should start moving back and forth between Position 1 and Position 2. If the motor does not begin to move return to step 1. If the motor starts moving, 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 in the Command window 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 *Pilot Motion Processor Developer's Kit.* The window has a command line style interface that accepts all of the motion processor commands. The *Pilot 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 Pilot motion processor;

2) Sets the Kp 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 exampl,e 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 Kp 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 Pilot 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.
#axis *number*:	Selects the current axis. (e.g., #axis 3 selects axis number 3.) All future commands will be executed on the selected axis.
< *filename*	Script file-input command. Executes the commands contained in the specified file.
TraceDownload *filename*	Reads trace data from the external memory and stores it to the specified *filename*. Prior to executing this command, an actual trace must have been started as described in section 7 of the *Pilot User's Guide*.

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
PMDecode.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 Pilot 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;

PMDW32SerialIOData 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
// serial defaults are COM1,57600,N,8,2.
PMDW32Serial_InitData(&transport_data);

// if required the transport data defaults can be changed here
// transport_data->multiDropAddress = 0;
// transport_data->protocol = Protocol_PointToPoint;
// transport_data->baud = 57600;
// default port is COM1
// transport_data->port = 1;

// 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)
PMDW32Serial_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 occured 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 PCI 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:

- Switch blocks S1 and S2 set the serial interface configuration and multi-drop address
- Connectors J1, J3, J4, and J5 are described below
- Resistor packs RS1 – RS3 are set according to the type of encoder used

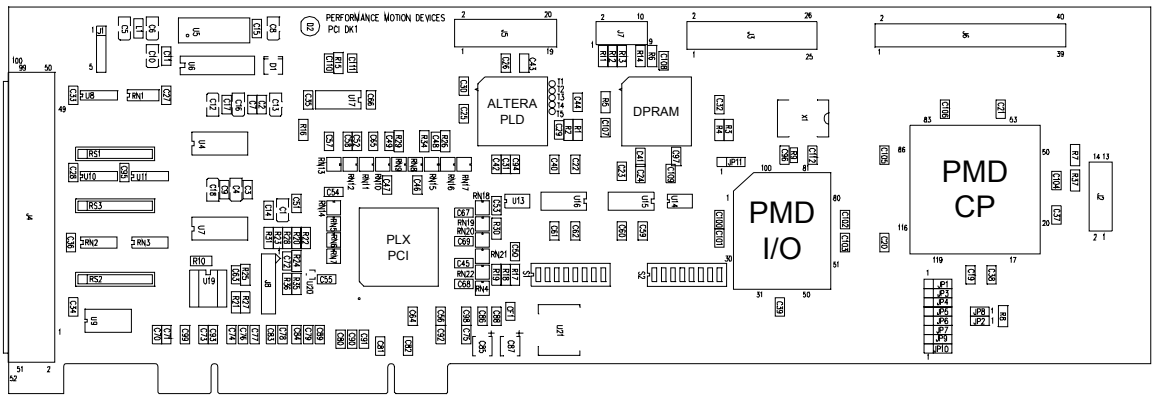


Figure 4-1 Developer's kit board layout

4.2 PCI DK1 Connectors

This section describes the pinouts for the following cable connectors on the Pilot 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

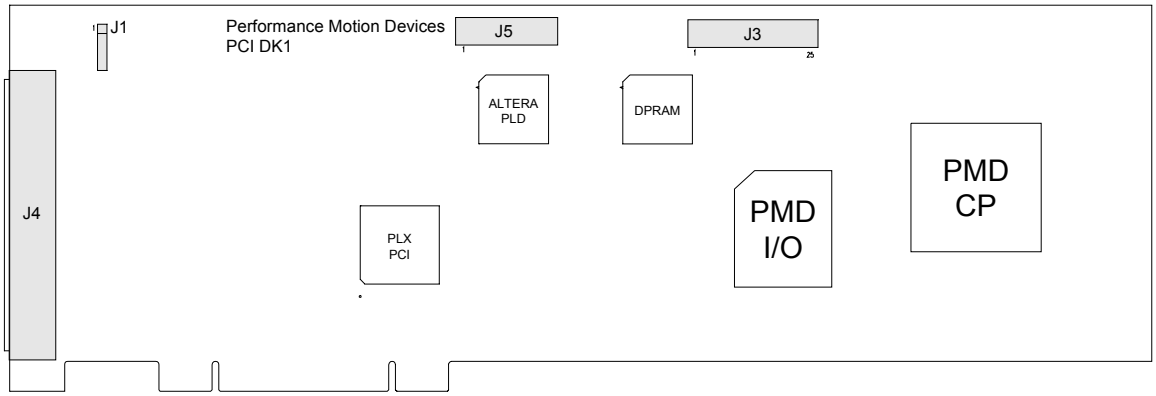


Figure 4-2 Developer's kit connector locations

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_{cc}

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 left 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 MC3110, MC3310 or MC3410 processor installed

Header 1 *(to J7 pins 1-50)*				**Header 2** *(to J7 pins 51-100)*			
Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	QuadA1+	26		1		26	
2	QuadA1-	27		2		27	
3	QuadB1+	28		3		28	
4	QuadB1-	29		4		29	
5	Index1+	30		5		30	
6	Index1-	31		6		31	
7	V_{cc} (encoder)	32		7		32	
8	GND (encoder)	33		8		33	
9	Hall1A	34	PWMMag2	9		34	
10	Hall1B	35	PWMMag3	10		35	
11	Hall1C	36		11		36	
12	GND (Hall)	37		12		37	
13	PosLim1	38		13		38	
14	NegLim1	39		14		39	
15	Home1	40		15		40	
16	AxisIn1	41	PWMMag1	16		41	
17	AxisOut1	42		17	PWMSign1	42	PWMSign2
18		43		18		43	
19		44		19		44	
20		45		20		45	
21		46		21		46	
22	DACA1*	47		22		47	
23	DACB1*	48		23		48	
24	GND (DAC)	49		24		49	
25	N.C.	50	N.C.	25	N.C.	50	N.C.

NOTE:

- Ununsed 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.
- *DACA*n*, DACB*n* are mapped to two analog output signals for axis *n*. For single-phased motion processor products (for example MC3110) DACB1 will not be used.

Connector pinouts with an MC3510 processor installed

Header 1 *(to J7 pins 1-50)*				**Header 2** *(to J7 pins 51-100)*			
Pin	Signal Name	Pin	Signal Name	Pin	Signal Name	Pin	Signal Name
1	QuadA1+	26		1		26	
2	QuadA1-	27		2		27	
3	QuadB1+	28		3		28	
4	QuadB1-	29		4		29	
5	Index1+	30		5		30	
6	Index1-	31		6		31	
7	V_{cc} (encoder)	32		7		32	
8	GND (encoder)	33		8		33	
9		34		9	AtRest	34	
10		35		10		35	
11		36	Direction1	11		36	
12		37		12		37	
13	PosLim1	38		13		38	
14	NegLim1	39		14		39	
15	Home1	40		15		40	
16	AxisIn1	41		16	Pulse1	41	
17	AxisOut1	42		17		42	
18		43		18		43	
19		44		19		44	
20		45		20		45	
21		46		21		46	
22		47		22		47	
23		48		23		48	
24		49		24		49	
25	N.C.	50	N.C.	25	N.C.	50	N.C.

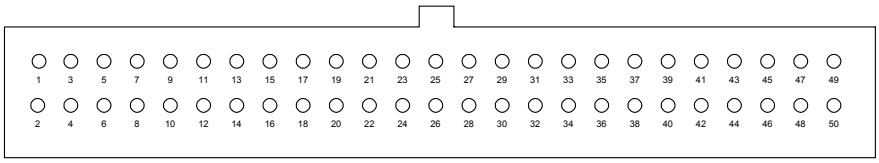
NOTE:

- Ununsed 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.

Bottom view of 50-pin header



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 Pilot 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	N.C. (CP pin 95)
10	N.C. (CP pin 79)	23	N.C. (CP pin 96)
11	N.C. (CP pin 80)	24	N.C. (CP pin 97)
12	N.C. (CP pin 81)	25	SrlEnable (CP pin 99)
13	AnalogRefHigh (CP pin 85)	26	~HostIntrpt (CP pin 98)

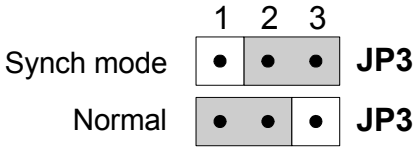
4.3 PCI DK1 Configuration Jumpers

Settable jumpers and switch blocks are shown in Figure 1-1. **All other jumpers should be left in their default factory settings. Notably, JP11 should be left in its factory default position.**

4.3.1 Synch feature jumper (JP3)

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

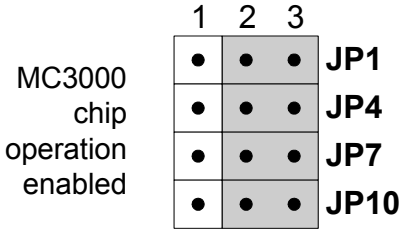
For non synch-enabled chipsets this jumper should be set to "Normal" position. When used with the Synch-enabled chipsets this jumper should be set to the "Synch mode" position.



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 *Pilot User's Guide*.

4.3.2 Configuration Jumpers JP1, JP4, JP7, JP10

For use with the MC3000 family of motion processors these jumpers must be placed as shown below, between pins 2 and 3. For all other chips the jumpers must be placed between pins 1 and 2.



4.4 Serial Port (DIP Switch Block S1 and S2)

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

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									

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 Pilot 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

Pulse and Direction Stepper motor amplifier

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 (MC3110)

			J4 connection (Header-pin)
	Signal name	Amplifier input	Axis 1
DAC	DACA*n*	Ref+ or V+	Hdr1-22
	GND*n*	Ref- or Gnd	Hdr1-24
PWM sign/magnitude	PWMMag1	PWM magnitude	Hdr1-41
	PWMSign1	PWM direction	Hdr2-17

4.5.2 Brushless Servo Motors (MC3310)

			J4 connection (Header-pin)
	Signal name	Amplifier input	Axis 1
DAC	DACA*n*	Ref1+ or V1+	Hdr1-22
	DACB*n*	Ref2+ or V2+	Hdr1-23
	GND*n*	Ref- or Gnd	Hdr1-24
PWM 50/50	PWMMag1	PWM phase 1	Hdr1-41
	PWMMag2	PWM phase 2	Hdr1-34
	PWMMag3	PWM phase 3	Hdr1-35

4.5.3 Microstepping Motors (MC3410)

			J4 connection (Header-pin)
	Signal name	Amplifier input	Axis 1
DAC	DACA*n*	Ref+ or V+	Hdr1-22
	GND*n*	Ref- or Gnd	Hdr1-24
PWM sign/magnitude	PWMMag1	PWM magnitude	Hdr1-41
	PWMSign1	PWM direction	Hdr2-17
	PWMMag2	PWM magnitude	Hdr1-34
	PWMSign2	PWM direction	Hdr2-42

4.5.4 Stepping Motors (MC3510)

			J4 connection (Header-pin)
	Signal name	Amplifier input	Axis 1
PWM sign/magnitude	Pulse	Step or Clock	Hdr2-16
	Direction	Direction or Forward/Reverse	Hdr1-36
	AtRest	Reduce current	Hdr2-9

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
QuadA*n*+	Hdr1-1
QuadA*n*-	Hdr1-2
QuadB*n*+	Hdr1-3
QuadB*n*-	Hdr1-4
Index*n*+	Hdr1-5
Index*n*-	Hdr1-6
V_{cc}	Hdr1-7
GND	Hdr1-8

Encoder connections when using single-ended encoder input

Signal	J4 pin connections
	Axis 1
QuadA*n*	Hdr1-1
QuadB*n*	Hdr1-3
Index*n*	Hdr1-5
V_{cc}	Hdr1-7
GND	Hdr1-8

5 PCI DK1 Hardware Information

5.1 Environmental and Electrical Ratings

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

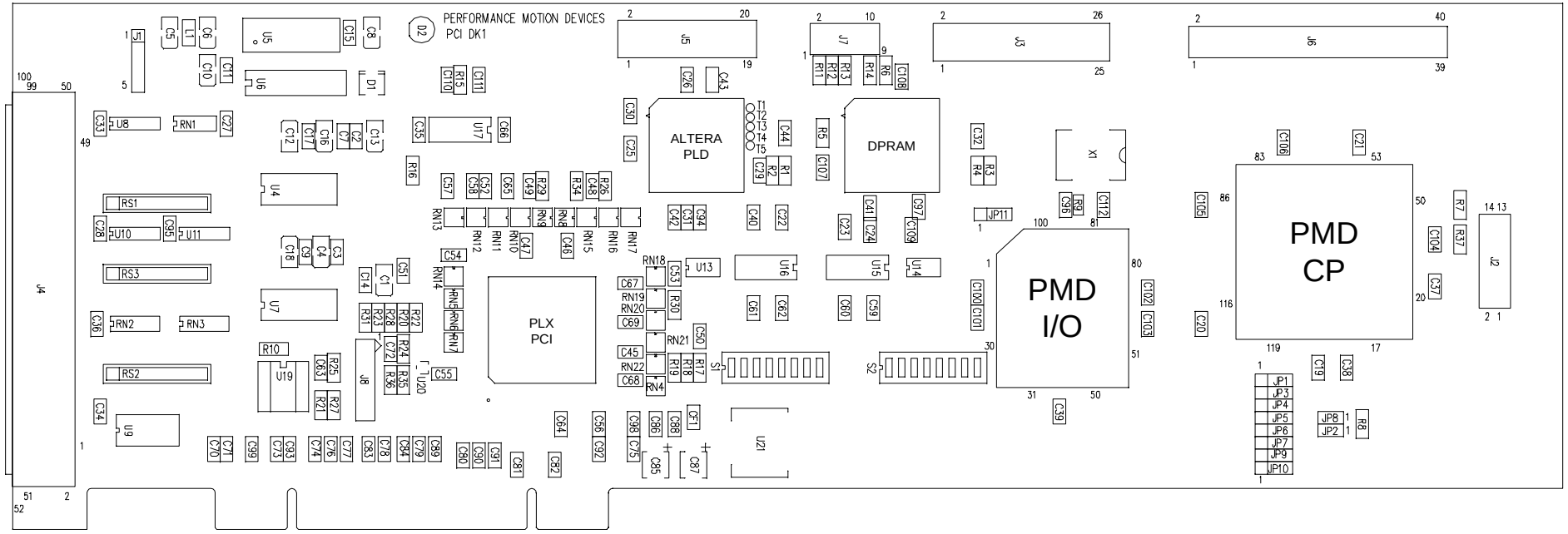
Space	Range	Remap	Descriptor	ChipSelect
0 (DPRAM)	0FFF0000	0001	00402081	0001
1 (CP)	0FFFFFE0	8001	00440480	8009

The following constants are used with the PlxBus_xxx functions.

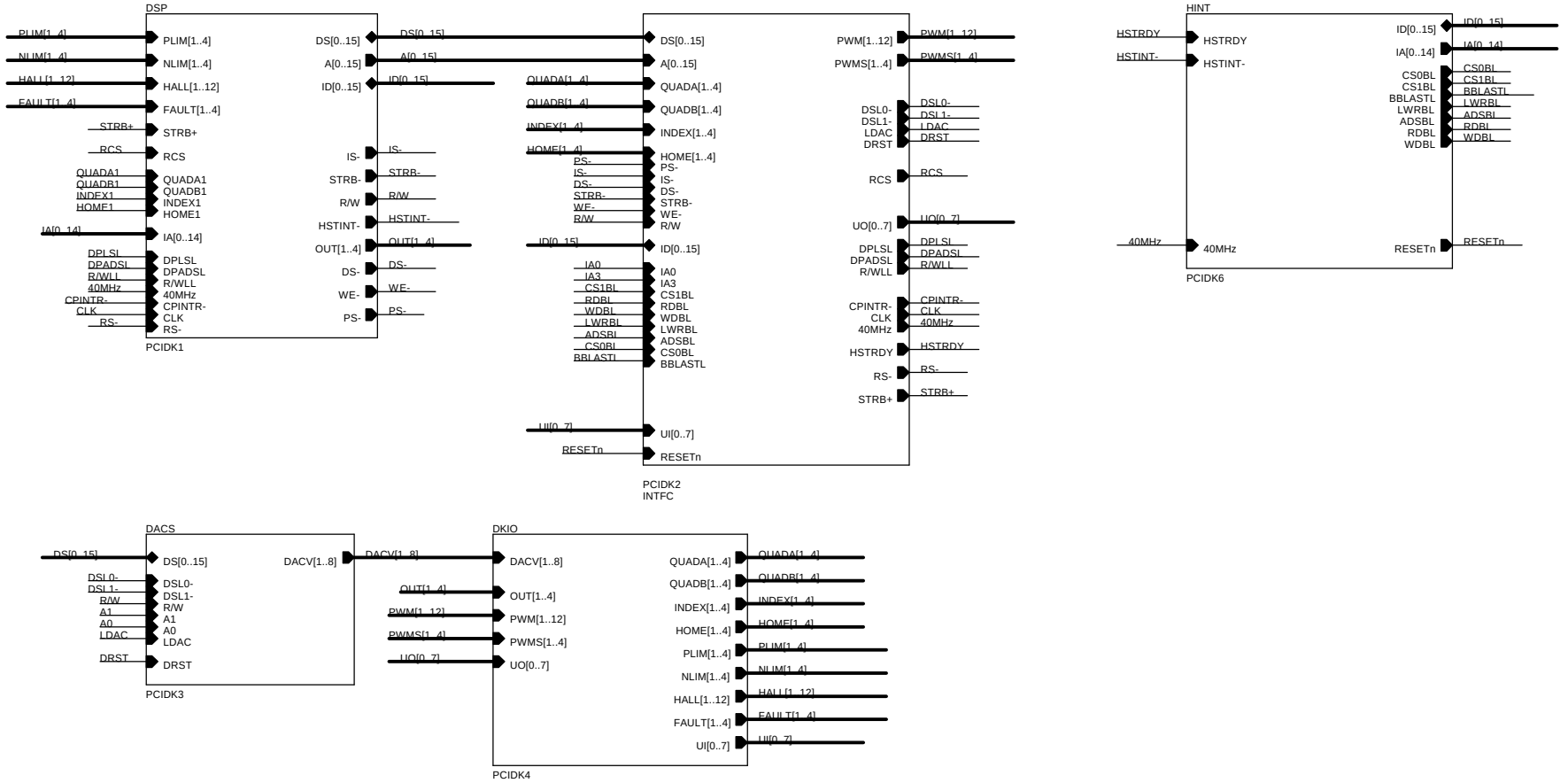
```
PCI_IOSPACE0_BASE = 8000
PCI_IOSPACE1_BASE = 800

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
```

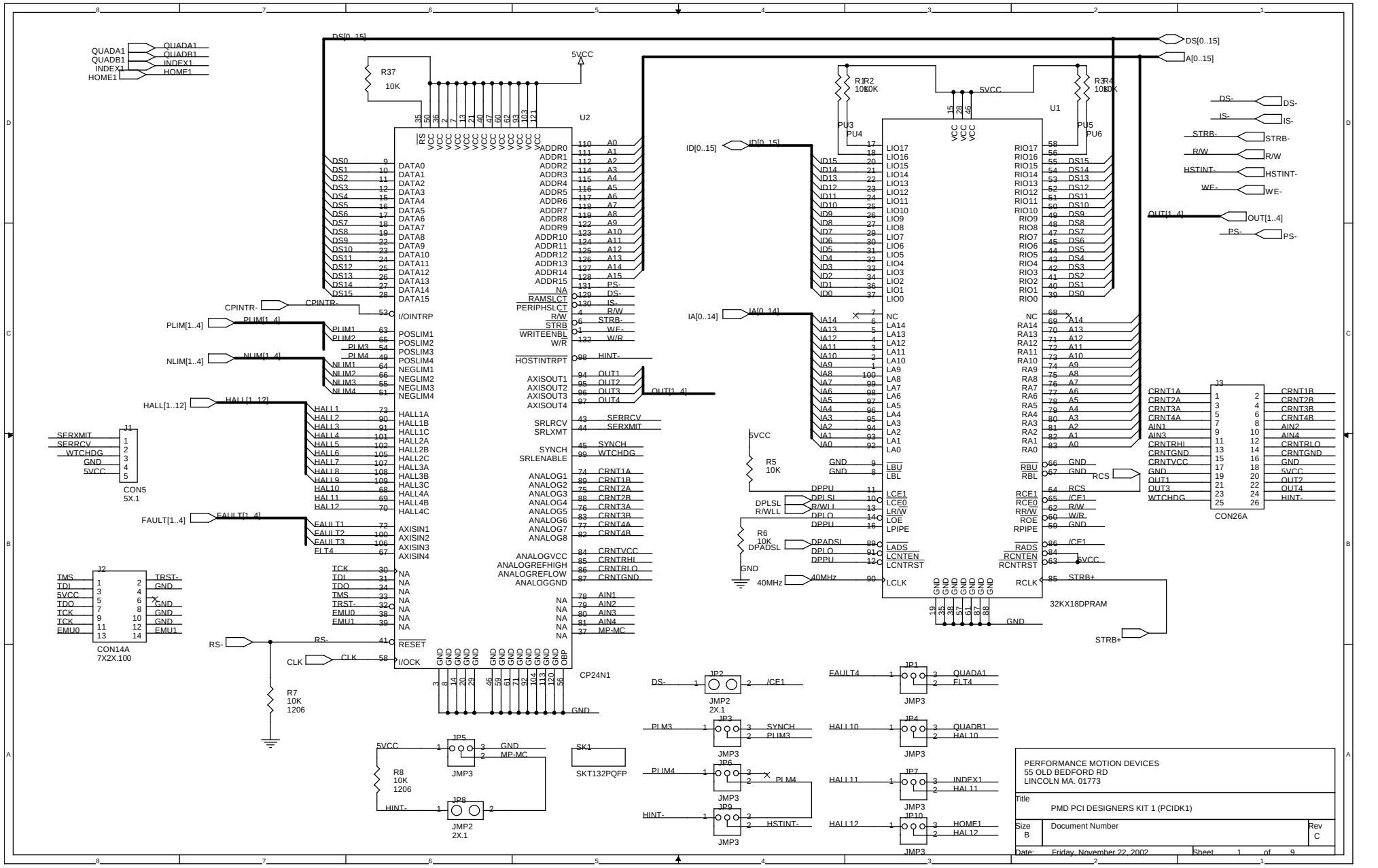

PERFORMANCE MOTION DEVICES
PCI DK1
ALTERA
PLD
DPRAM
PLX
PCI
PMD
I/O
PMD
CP

5.3.2 PCI DK1 Overview Schematic

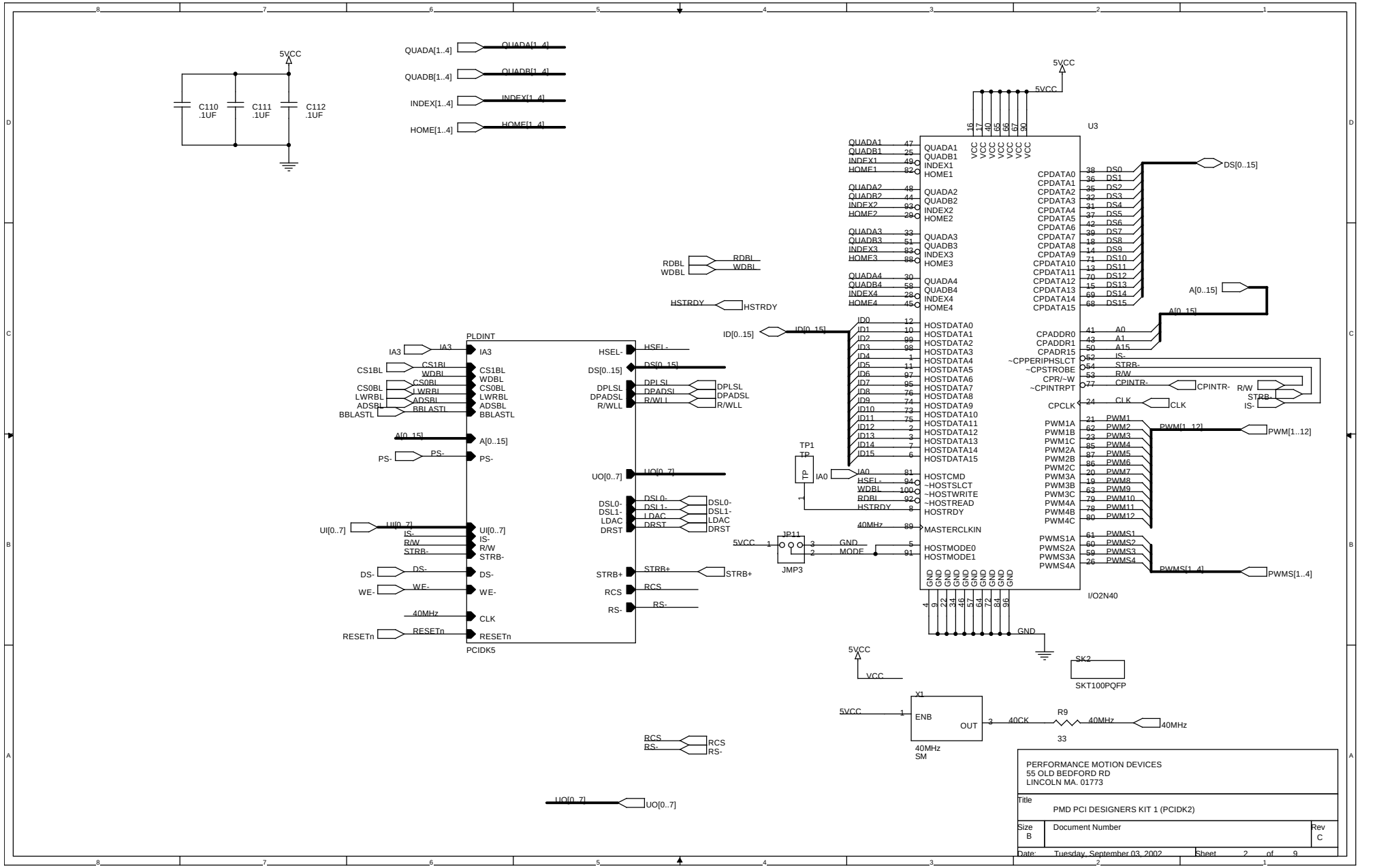


PERFORMANCE MOTION DEVICES 55 OLD BEDFORD RD LINCOLN MA. 01773	
Title	PMD PCI DESIGNERS KIT 1 (PCIDK)
Size B	Document Number — Rev C
Date: Tuesday, September 03, 2002	Sheet 9 of 9

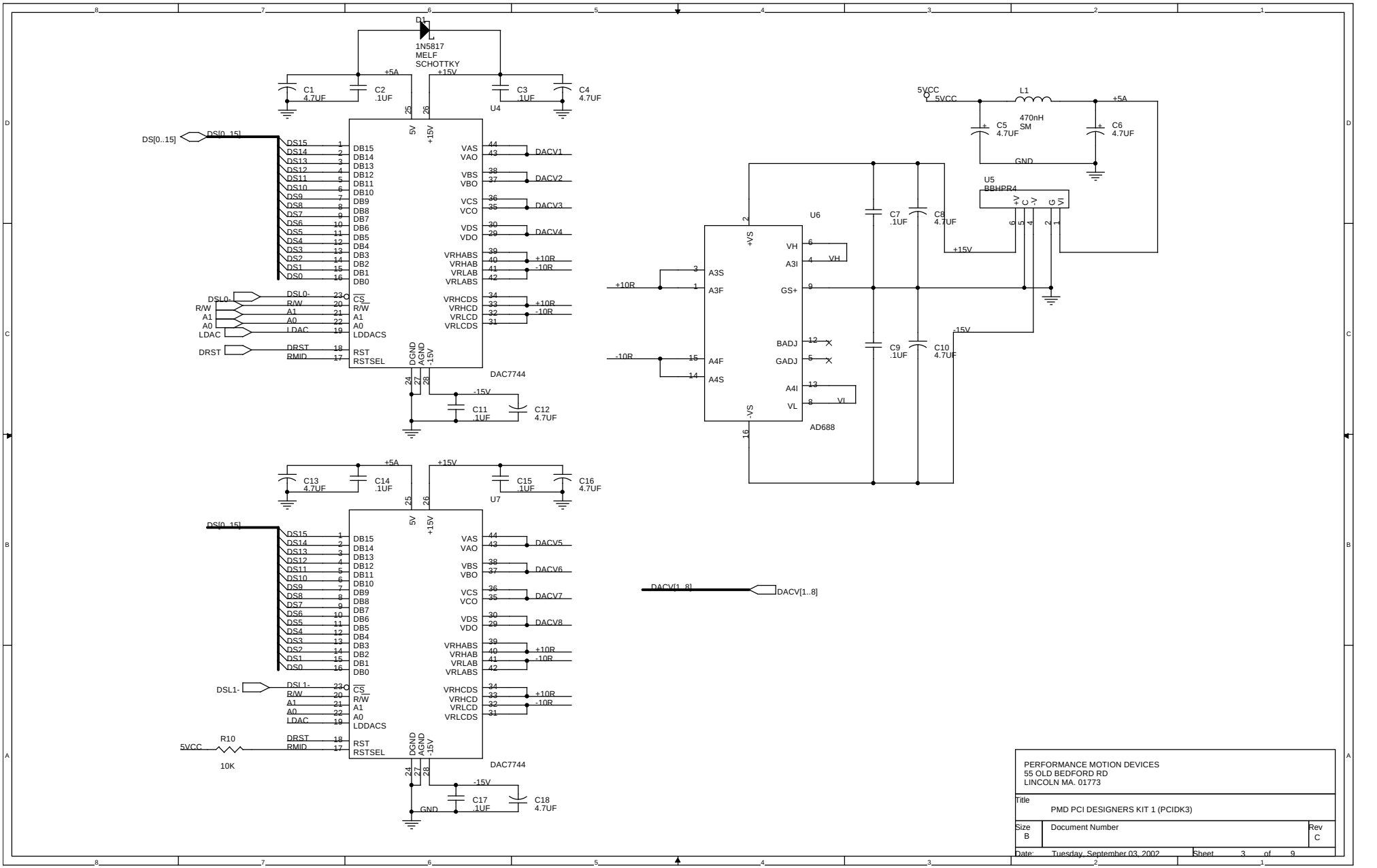
5.3.3 PCI DK1 CP and DPRAM Schematic


QUADA1
QUADB1
INDEX1
HOME1
DS[0..15]
5VCC
R37
10K
U2
DATA0
DATA1
DATA2
DATA3
DATA4
DATA5
DATA6
DATA7
DATA8
DATA9
DATA10
DATA11
DATA12
DATA13
DATA14
DATA15
I/OINTRP
POSLIM1
POSLIM2
POSLIM3
POSLIM4
NEGLIM1
NEGLIM2
NEGLIM3
NEGLIM4
HALL1A
HALL1B
HALL1C
HALL2A
HALL2B
HALL2C
HALL3A
HALL3B
HALL3C
HALL4A
HALL4B
HALL4C
AXISIN1
AXISIN2
AXISIN3
AXISIN4
RESET
I/OCK
ADDR0
ADDR1
ADDR2
ADDR3
ADDR4
ADDR5
ADDR6
ADDR7
ADDR8
ADDR9
ADDR10
ADDR11
ADDR12
ADDR13
ADDR14
ADDR15
PS-
DS-
IS-
RAMSLCT
PERIPHSLCT
R/W
STRB
WRITEENBL
W/R
HOSTINTRPT
HINT-
AXISOUT1
AXISOUT2
AXISOUT3
AXISOUT4
OUT1
OUT2
OUT3
OUT4
SRLRCV
SRLXMT
SYNCH
SRLENABLE
WTCHDG
ANALOG1
ANALOG2
ANALOG3
ANALOG4
ANALOG5
ANALOG6
ANALOG7
ANALOG8
CRNT1A
CRNT1B
CRNT2A
CRNT2B
CRNT3A
CRNT3B
CRNT4A
CRNT4B
ANALOGVCC
ANALOGREFHIGH
ANALOGREFLOW
ANALOGGND
CRNTVCC
CRNTRHI
CRNTRLO
CRNTGND
AIN1
AIN2
AIN3
AIN4
MP-MC
SERRCV
SERXMIT
TCK
TDI
TDO
TMS
TRST-
EMU0
EMU1
CP24N1
GND
OBP
NA
CPINTR-
PLIM[1..4]
NLIM[1..4]
HALL[1..12]
FAULT[1..4]
RS-
CLK
R7
10K
1206
J1
CON5
5X.1
SERXMIT
SERRCV
WTCHDG
GND
5VCC
J2
CON14A
7X2X.100
TMS
TDI
5VCC
TDO
TCK
TCK
EMU0
TRST-
GND
EMU1
U1
PU3
PU4
PU5
PU6
R1
R2
R3
R4
10K
ID[0..15]
IA[0..14]
LIO17
LIO16
LIO15
LIO14
LIO13
LIO12
LIO11
LIO10
LIO9
LIO8
LIO7
LIO6
LIO5
LIO4
LIO3
LIO2
LIO1
LIO0
RIO17
RIO16
RIO15
RIO14
RIO13
RIO12
RIO11
RIO10
RIO9
RIO8
RIO7
RIO6
RIO5
RIO4
RIO3
RIO2
RIO1
RIO0
NC
LA14
LA13
LA12
LA11
LA10
LA9
LA8
LA7
LA6
LA5
LA4
LA3
LA2
LA1
LA0
RA14
RA13
RA12
RA11
RA10
RA9
RA8
RA7
RA6
RA5
RA4
RA3
RA2
RA1
RA0
LBU
LBL
RBU
RBL
RCS
/CE1
R/W
WR
ROE
RPIPE
RADS
RCNTEN
RCNTRST
LADS
LCNTEN
LCNTRST
DPPU
DPLSL
R/WLL
DPLO
LOE
LPIPE
DPADSL
LCE1
LCE0
LR/W
LCLK
RCLK
STRB+
32KX18DPRAM
R5
10K
R6
10K
40MHz
DS[0..15]
A[0..15]
IS-
STRB-
R/W
HSTINT-
WE-
OUT[1..4]
PS-
J3
CON26A
CRNT1A
CRNT2A
CRNT3A
CRNT4A
AIN1
AIN3
CRNTRHI
CRNTGND
CRNTVCC
GND
OUT1
OUT3
WTCHDG
CRNT1B
CRNT2B
CRNT3B
CRNT4B
AIN2
AIN4
CRNTRLO
CRNTGND
GND
5VCC
OUT2
OUT4
HINT-
JP1
JP2
JP3
JP4
JP5
JP6
JP7
JP8
JP9
JP10
JMP3
JMP2
2X.1
QUADA1
FLT4
/CE1
SYNCH
PLM3
QUADB1
HAL10
INDEX1
HAL11
HOME1
HAL12
HSTINT-
PLM4
R8
10K
1206
SK1
SKT132PQFP
PERFORMANCE MOTION DEVICES
55 OLD BEDFORD RD
LINCOLN MA. 01773
Title
PMD PCI DESIGNERS KIT 1 (PCIDK1)
Size
B
Document Number
Rev
C
Date:
Friday, November 22, 2002
Sheet
1
of
9

5.3.4 PCI DK1 I/O Schematic


5VCC
C110 .1UF
C111 .1UF
C112 .1UF
QUADA[1..4]
QUADB[1..4]
INDEX[1..4]
HOME[1..4]
U3
I/O2N40
PLDINT
PCIDK5
SK2
SKT100PQFP
X1
ENB
OUT
40MHz SM
R9
33
JP11
JMP3
TP1
PERFORMANCE MOTION DEVICES
55 OLD BEDFORD RD
LINCOLN MA. 01773
Title
PMD PCI DESIGNERS KIT 1 (PCIDK2)
Size B
Document Number
Rev C
Date: Tuesday, September 03, 2002
Sheet 2 of 9

5.3.5 PCI DK1 DAC Amplifiers Schematic



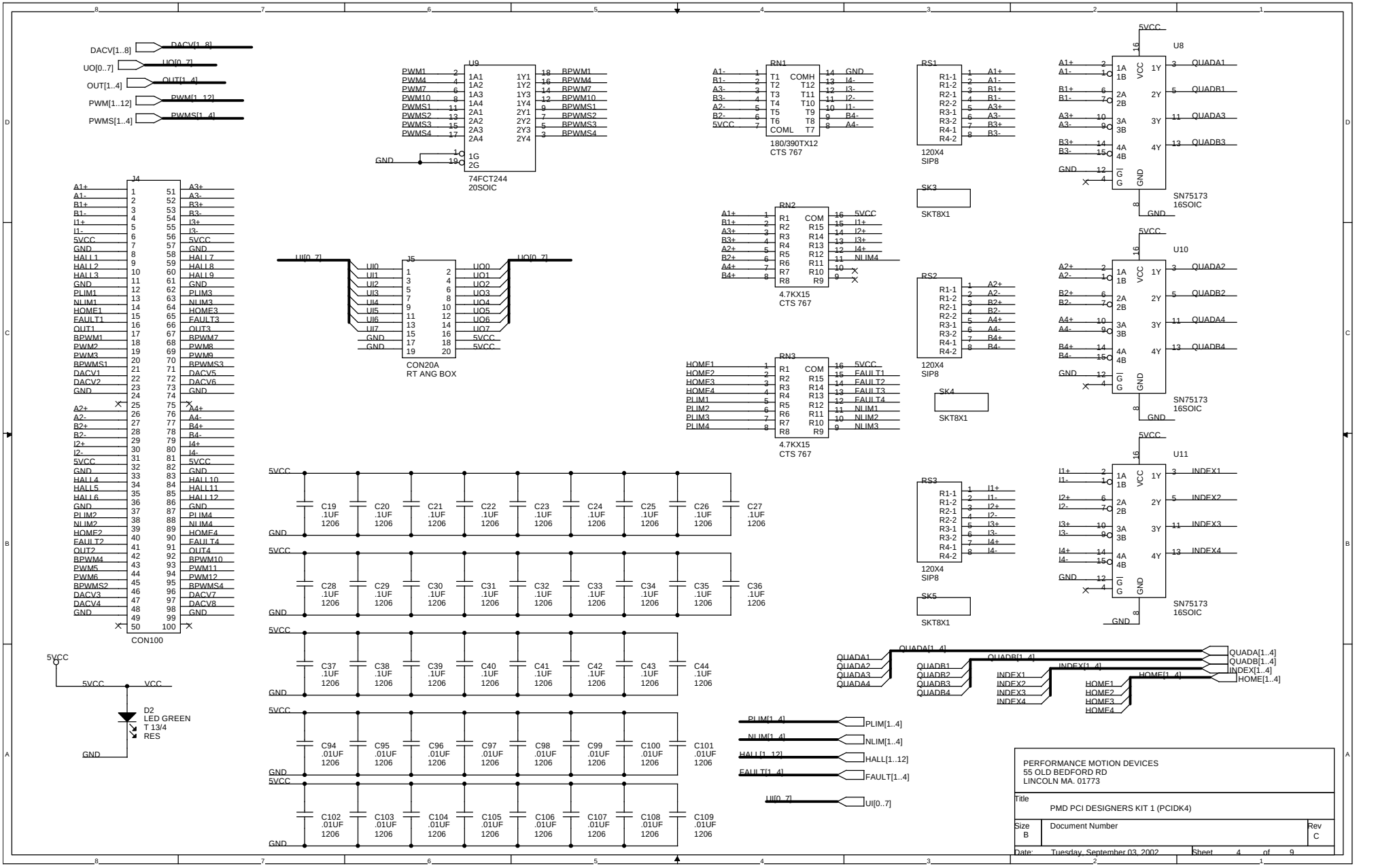
PERFORMANCE MOTION DEVICES
55 OLD BEDFORD RD
LINCOLN MA. 01773

Title: PMD PCI DESIGNERS KIT 1 (PCIDK3)

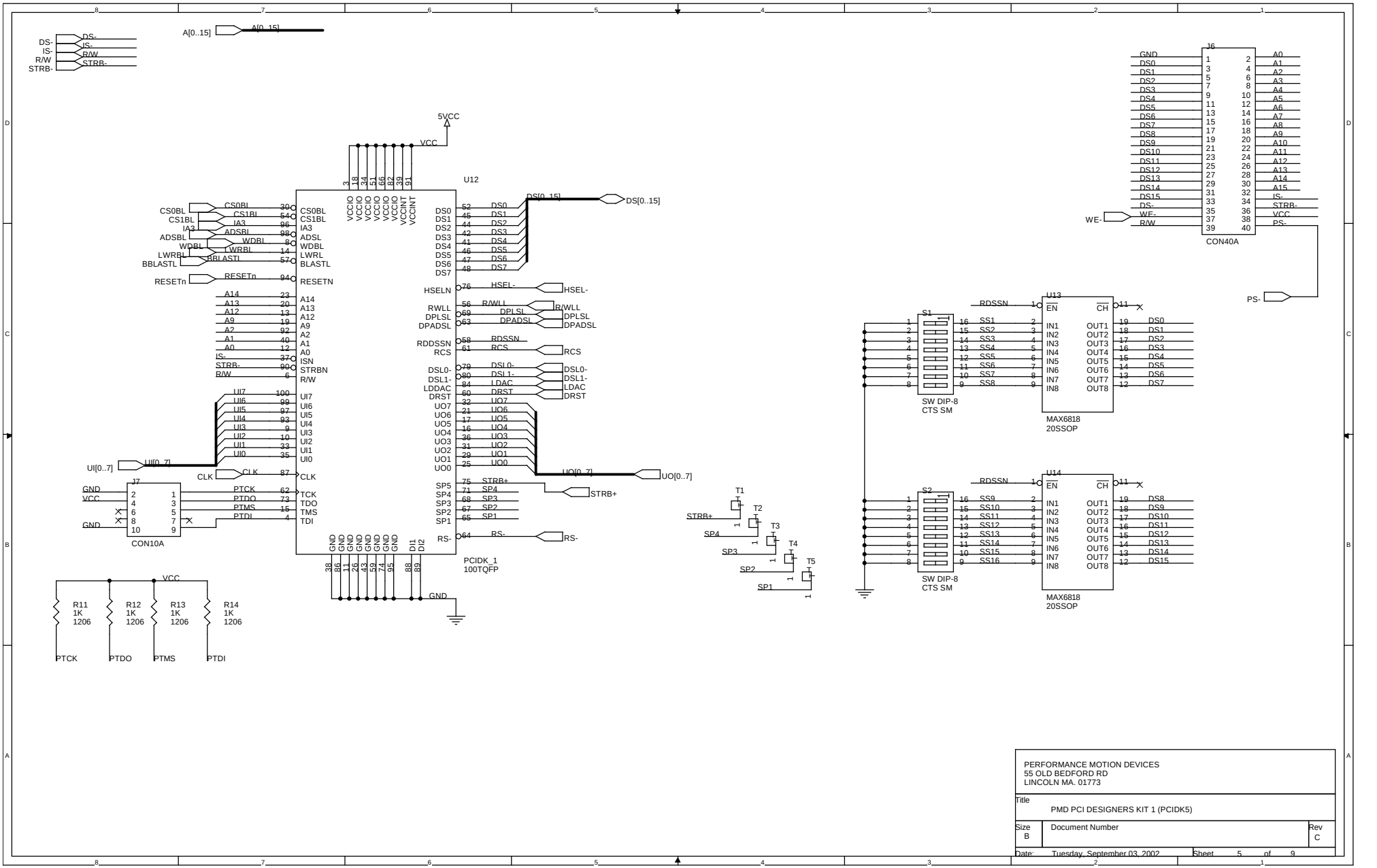
Size: B | Document Number | Rev: C

Date: Tuesday, September 03, 2002 | Sheet 3 of 9

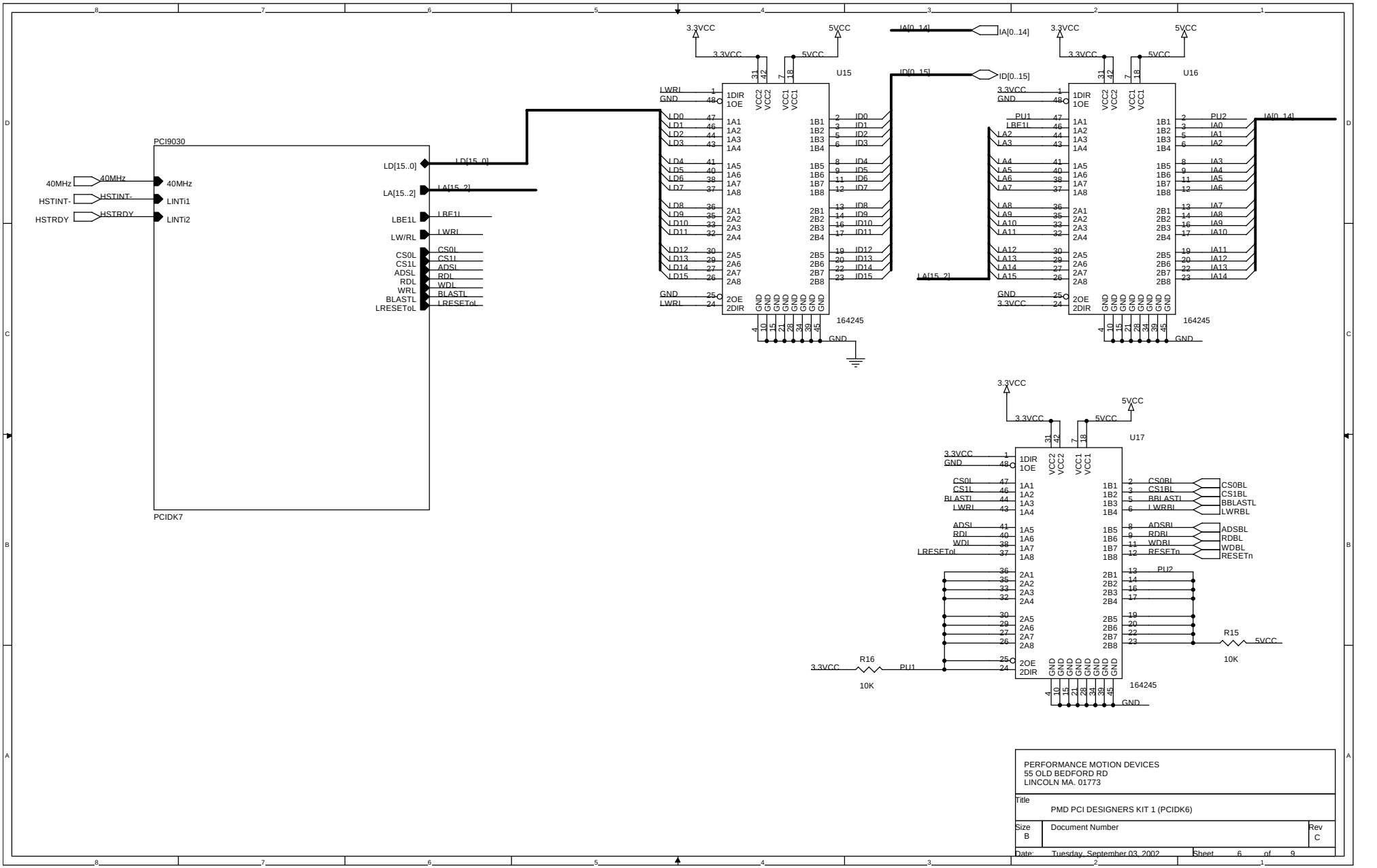
5.3.6 PCI DK1 Connector and Quadrature Schematic


DACV[1..8]
UO[0..7]
OUT[1..4]
PWM[1..12]
PWMS[1..4]
U9
74FCT244
20SOIC
RN1
180/390TX12
CTS 767
RS1
120X4
SIP8
SK3
SKT8X1
U8
SN75173
16SOIC
RN2
4.7KX15
CTS 767
RS2
SK4
U10
RN3
RS3
SK5
U11
J4
CON100
UI[0..7]
J5
CON20A
RT ANG BOX
5VCC
GND
C19 .1UF 1206
C20 .1UF 1206
C21 .1UF 1206
C22 .1UF 1206
C23 .1UF 1206
C24 .1UF 1206
C25 .1UF 1206
C26 .1UF 1206
C27 .1UF 1206
C28 .1UF 1206
C29 .1UF 1206
C30 .1UF 1206
C31 .1UF 1206
C32 .1UF 1206
C33 .1UF 1206
C34 .1UF 1206
C35 .1UF 1206
C36 .1UF 1206
C37 .1UF 1206
C38 .1UF 1206
C39 .1UF 1206
C40 .1UF 1206
C41 .1UF 1206
C42 .1UF 1206
C43 .1UF 1206
C44 .1UF 1206
C94 .01UF 1206
C95 .01UF 1206
C96 .01UF 1206
C97 .01UF 1206
C98 .01UF 1206
C99 .01UF 1206
C100 .01UF 1206
C101 .01UF 1206
C102 .01UF 1206
C103 .01UF 1206
C104 .01UF 1206
C105 .01UF 1206
C106 .01UF 1206
C107 .01UF 1206
C108 .01UF 1206
C109 .01UF 1206
D2
LED GREEN
T 13/4
RES
PLIM[1..4]
NLIM[1..4]
HALL[1..12]
FAULT[1..4]
UI[0..7]
QUADA[1..4]
QUADB[1..4]
INDEX[1..4]
HOME[1..4]
PERFORMANCE MOTION DEVICES
55 OLD BEDFORD RD
LINCOLN MA. 01773
Title
PMD PCI DESIGNERS KIT 1 (PCIDK4)
Size B
Document Number
Rev C
Date: Tuesday, September 03, 2002
Sheet 4 of 9

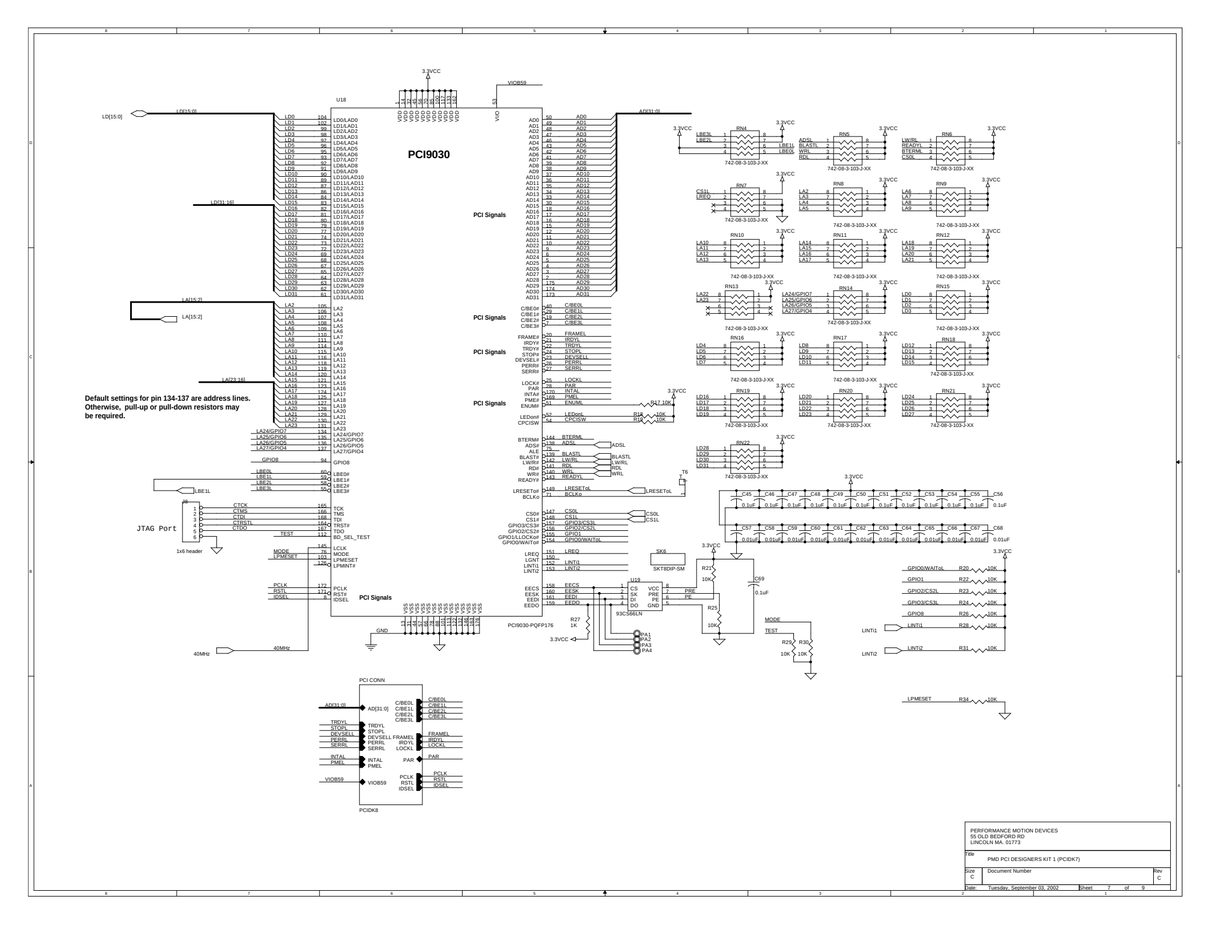
5.3.7 PCI DK1 PLD Schematic


A[0..15]
DS-
IS-
R/W
STRB-
5VCC
VCC
U12
PCIDK_1
100TQFP
DS[0..15]
UI[0..7]
UO[0..7]
CLK
RESETn
HSELN
RWLL
DPLSL
DPADSL
RDDSSN
RCS
DSL0-
DSL1-
LDAC
DRST
STRB+
RS-
CS0BL
CS1BL
IA3
ADSBL
WDBL
LWRBL
BBLASTL
A14
A13
A12
A9
A2
A1
A0
ISN
STRBN
R/W
UI7
UI6
UI5
UI4
UI3
UI2
UI1
UI0
UO7
UO6
UO5
UO4
UO3
UO2
UO1
UO0
SP5
SP4
SP3
SP2
SP1
TCK
TDO
TMS
TDI
GND
J7
CON10A
PTCK
PTDO
PTMS
PTDI
R11 1K 1206
R12 1K 1206
R13 1K 1206
R14 1K 1206
T1
T2
T3
T4
T5
J6
CON40A
WE-
PS-
U13
U14
S1
S2
SW DIP-8
CTS SM
MAX6818
20SSOP
PERFORMANCE MOTION DEVICES
55 OLD BEDFORD RD
LINCOLN MA. 01773
PMD PCI DESIGNERS KIT 1 (PCIDK5)
Size B
Document Number
Rev C
Date: Tuesday, September 03, 2002
Sheet 5 of 9

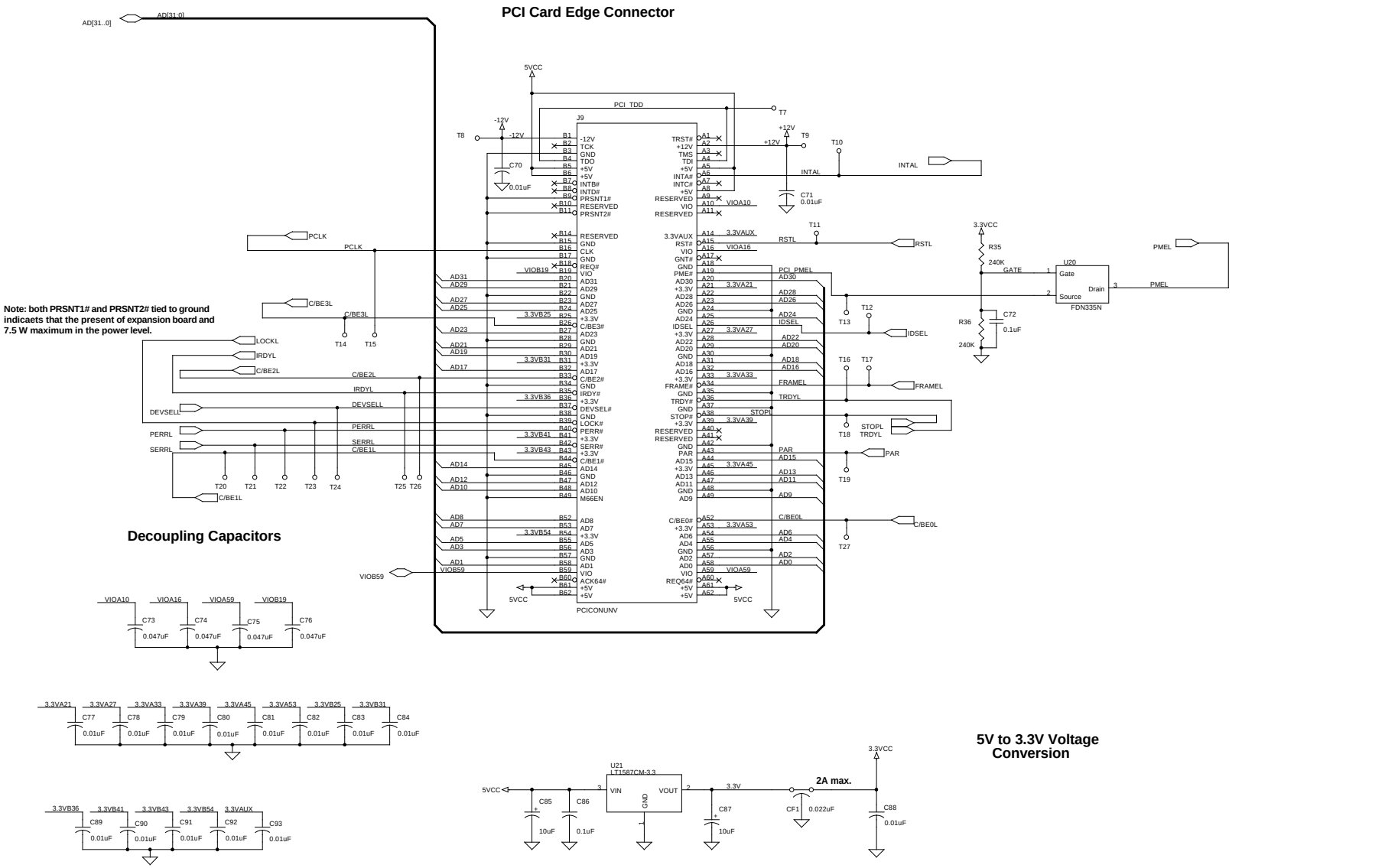
5.3.8 PCI DK1 Buffer Schematic


PCI9030
PCIDK7
40MHz
HSTINT-
HSTRDY
LINTi1
LINTi2
LD[15..0]
LA[15..2]
LBE1L
LW/RL
CS0L
CS1L
ADSL
RDL
WDL
WRL
BLASTL
LRESETol
3.3VCC
5VCC
U15
U16
U17
164245
GND
IA[0..14]
ID[0..15]
PU1
PU2
R15
R16
10K
PERFORMANCE MOTION DEVICES
55 OLD BEDFORD RD
LINCOLN MA. 01773
Title
PMD PCI DESIGNERS KIT 1 (PCIDK6)
Size B
Document Number
Rev C
Date: Tuesday, September 03, 2002
Sheet 6 of 9

5.3.9 PCI DK1 PLX9030 Schematic


PCI9030
U18
3.3VCC
VIOB59
LD[15:0]
LD[31:16]
LA[15:2]
LA[23:16]
AD[31:0]
PCI Signals
Default settings for pin 134-137 are address lines.
Otherwise, pull-up or pull-down resistors may
be required.
JTAG Port
1x6 header
J8
CTCK
CTMS
CTDI
CTRSTL
CTDO
TEST
TCK
TMS
TDI
TRST#
TDO
BD_SEL_TEST
LCLK
MODE
LPMESET
LPMINT#
PCLK
RSTL
IDSEL
40MHz
GND
PCI9030-PQFP176
U19
93CS66LN
SK6
SKT8DIP-SM
PA1
PA2
PA3
PA4
MODE
TEST
LINTi1
LINTi2
LPMESET
PCI CONN
PCIDK8
742-08-3-103-J-XX
RN4
RN5
RN6
RN7
RN8
RN9
RN10
RN11
RN12
RN13
RN14
RN15
RN16
RN17
RN18
RN19
RN20
RN21
RN22
C45
C46
C47
C48
C49
C50
C51
C52
C53
C54
C55
C56
C57
C58
C59
C60
C61
C62
C63
C64
C65
C66
C67
C68
0.1uF
0.01uF
R17 10K
R18 10K
R19 10K
R21 10K
R25 10K
R27 1K
R29 10K
R30 10K
R34 10K
C69 0.1uF
PERFORMANCE MOTION DEVICES
55 OLD BEDFORD RD
LINCOLN MA. 01773
Title
PMD PCI DESIGNERS KIT 1 (PCIDK7)
Size C
Document Number
Rev C
Date: Tuesday, September 03, 2002
Sheet 7 of 9

5.3.10 PCI DK1 PCI Edge Connector Schematic



PERFORMANCE MOTION DEVICES 55 OLD BEDFORD RD LINCOLN MA. 01773	
Title	PMD PCI DESIGNERS KIT 1 (PCIDK8)
Size C	Document Number <Doc> Rev C
Date: Tuesday, September 03, 2002	Sheet 8 of 9