PAMUX USER'S GUIDE

Form 726-060425 — April, 2006

OPTO 22

43044 Business Park Drive, Temecula, CA 92590-3614
Phone: 800-321-OPTO (6786) or 951-695-3000
Fax: 800-832-OPTO (6786) or 951-695-2712
www.opto22.com

Product Support Services:
800-TEK-OPTO (835-6786) or 951-695-3080
Fax: 951-695-3017
E-mail: support@opto22.com
Web: support.opto22.com

Pamux User's Guide
Form 726-060425 — April, 2006

TABLE OF CONTENTS

WELCOME

Opto 22's Pamux system provides low-cost, high-speed distributed control of both digital and analog I/O. A Pamux system can span up to 500 feet and can support up to 32 stations with a total of up to 512 I/O points.

This manual provides all the information you will need to set up and use a Pamux system. It features complete descriptions of all Pamux components, including the B4, B5, and B6 brain boards and the AC28 and AC36 adapter cards. It provides detailed instructions on setting up a Pamux system and programming the system with or without Opto 22's Pamux software driver. Several handy appendices at the back of the manual feature troubleshooting procedures and general tips, hardware specifications, frequently asked questions, a glossary of terms, and information on Opto 22 Product Support.

Note: Information in this manual relating to Pamux brain boards applies only to the revisions of the brain boards indicated below:

- AC28: Revision C or later. For revision B or earlier, request Opto 22 form #218.
- B4: Revision L or later. For revision K or earlier, request Opto 22 form #127.
- B5: Revision J or later. For revision I or earlier, request Opto 22 form #145.
- B6: Revision G or later. For revision F or earlier, request Opto 22 form #154.

ABOUT THIS MANUAL

This manual is organized as follows:

- **Chapter 1: Introduction** — Overview of the Pamux system, with descriptions and illustrations of all Pamux components: AC28, AC36, B4, B5, B6, TERM1, TERM2, and UCA4.
- **Chapter 2: System Setup** — Everything you need to know to design and construct a Pamux system. Includes details on laying out a system, setting up Pamux stations, connecting field wiring, connecting stations to the bus to form a system, powering the system, linking to an adapter card, and connecting field wiring.
- **Chapter 3: Programming with the Pamux Driver** — Complete information on how to use Opto 22's Pamux software driver, designed for use on a PC with an AC28 or AC36 adapter card. Covers installation of the driver under DOS or Windows, provides a complete command reference for use under both platforms, and includes examples of programming code.
- **Chapter 4: Programming without the Pamux Driver** — Details on all issues that must be addressed when communicating with Pamux without using the Pamux driver. Provides descriptions of the Pamux bus layout and timing, plus detailed information on how to read from and write to digital and analog Pamux stations. Includes extensive programming examples.
- **Appendix A: Troubleshooting** — Tips on resolving system setup and communication problems.

- **Appendix B: Specifications** — Detailed reference information on Pamux brain boards, cables, connectors, and power supplies.
- **Appendix C: Temperature Conversion Routines** — Complete algorithms required to convert readings from thermocouple and ICTD probe modules into temperatures.
- **Appendix D: Product Support** — Information on how to get help from Opto 22.
- **Appendix E: Glossary** — Definitions of commonly used terms.

DOCUMENT CONVENTIONS

- **Bold** typeface indicates text to be typed. Unless otherwise noted, such text may be entered in upper or lower case. (Example: "At the DOS prompt, type **cd \windows**.")
- *Italic* typeface indicates emphasis and is used for book titles. (Example: "See the *OptoControl User's Guide* for details.")
- File names appear in all capital letters. (Example: "Open the file TEST1.TXT.")
- Key names appear in small capital letters. (Example: "Press SHIFT.")
- Key press combinations are indicated by hyphens between two or more key names. For example, SHIFT-F1 is the result of holding down the SHIFT key, then pressing and releasing the F1 key. Similarly, CTRL-ALT-DELETE is the result of pressing and holding the CTRL and ALT keys, then pressing and releasing the DELETE key.
- "Press" (or "click") means press and release when used in reference to a mouse button.
- Menu commands are sometimes referred to with the Menuà Command convention. For example, "Select Fileà Run" means to select the Run command from the File menu.
- Numbered lists indicate procedures to be followed sequentially. Bulleted lists (such as this one) provide general information.

INTRODUCTION

WHAT IS PAMUX?

Pamux is a high-speed, high-density distributed I/O system that accommodates both digital and analog brain boards and I/O modules. A Pamux bus can extend up to 500 feet from a host computer or other programming device.

Pamux supports up to 32 distributable stations containing up to 512 I/O points. Each station is composed of a Pamux brain board and an I/O mounting rack; the final station on the Pamux bus must also include a terminator board. Digital (B4 and B5) and analog (B6) stations may be mixed on a single bus.

All Pamux stations within a system are daisy-chained via a 50-pin flat-ribbon cable, as shown in Figure 1-1.

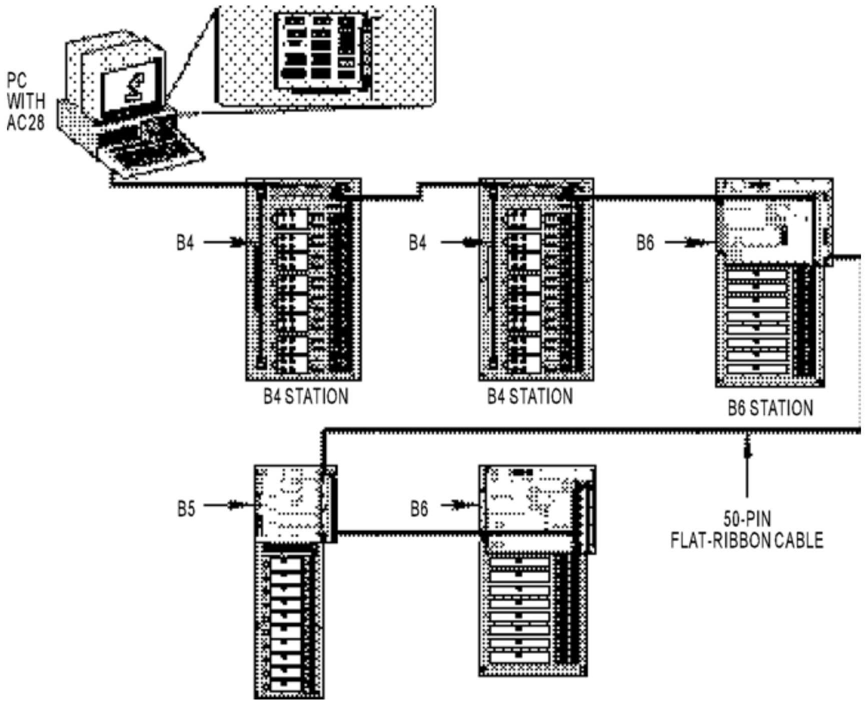


Figure 1-1: Typical Pamux System

A host computer with a standard 8-MHz I/O bus linked to a Pamux system can access eight digital I/O points in under three microseconds and 512 digital I/O points in under 200 microseconds. The host computer can also access 16 analog input points in under six milliseconds and 16 analog output points in about 15 milliseconds.

The Pamux product line is ideal for low-noise applications, such as:

- Robotics
- Numerical control
- Conveyor line control
- Sequence-of-events fault detection

Because it is an external high-speed parallel I/O system, Pamux is better suited to environments with low electromagnetic or radio frequency interference. Noisy applications are better suited to I/O systems that offer greater noise immunity, such as Opto 22's Mistic and Optomux systems.

PAMUX COMPONENTS

This section describes and illustrates each of the components of the Pamux family.

AC28 Adapter Card

Note: Information on the AC28 applies to revision C or later. For revision B or earlier, refer to Opto 22 form 218.

The AC28 is a high-speed adapter card designed to link the Pamux bus to IBM PC/AT or compatible computers. The AC28 is compatible with computers that feature a standard 8-MHz ISA bus.

Each AC28 can access up to 512 points of I/O. Four AC28s can be installed in one PC, supporting up to 2,048 points of I/O

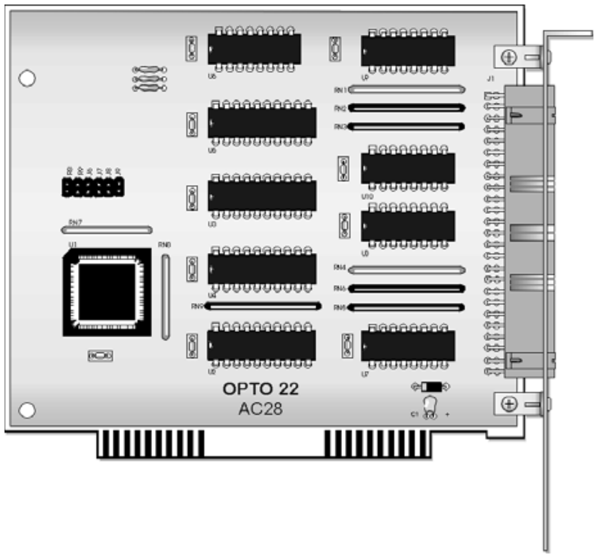


Figure 1-2: AC28 Adapter Card

PCI-AC51 Adapter Card

The PCI-AC51 adapter card links the Pamux bus to PCI-compatible computers. The PCI-AC51 is compatible with computers that feature a standard 33 MHz PCI bus. With the PCI-AC51 adapter card, your PCI computer can communicate with Opto 22 classic B4, B5, and B6 brain boards and with SNAP-B4 and SNAP-B6 brains.

- Each Pamux bus can access up to 32 remote brains.
- Each Pamux bus supports up to 512 points.
- Up to 32 PCI-AC51s are supported by the PCI-AC51 Toolkit.

The PCI-AC51 Toolkit is included with the PCI-AC51 adapter card and is also available on our Web site at www.opto22.com. This developer toolkit provides an interface between Pamux stations and application programs written in Microsoft Visual C++ and Visual Basic 6.0. The toolkit saves you time and effort that would otherwise be spent learning the intricacies of the Pamux bus structure.

For additional information about the PCI-AC51 and the PCI-AC51 Toolkit, see Opto 22 form #1459, the *PCI-AC51 User's Guide*.

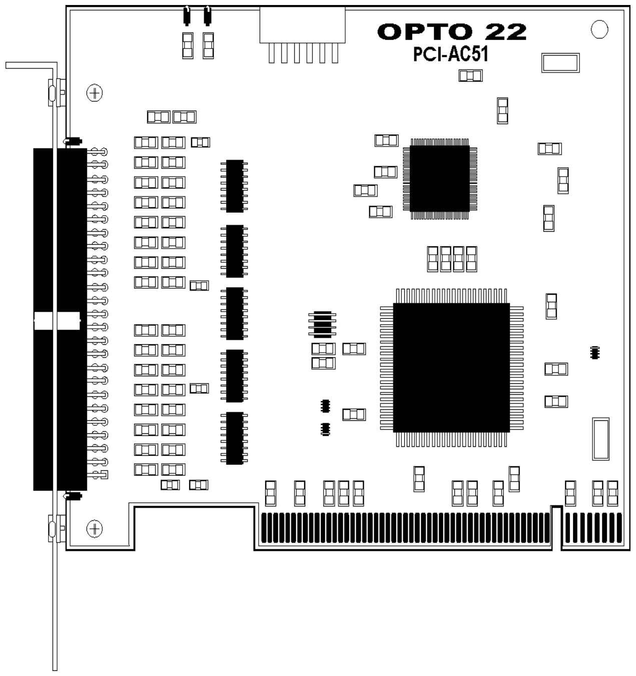


Figure 1-3: PCI-AC51 Adapter Card

AC36 Adapter Card

The AC36 is an adapter card that provides a Pamux bus interface for a TTL parallel port. It is compatible with parallel-port devices for MULTIBUS, STD bus, and VME bus products. (MULTIBUS is a patented Intel bus and a registered trademark of Intel Corp.)

Each AC36 can access up to 512 points of I/O.

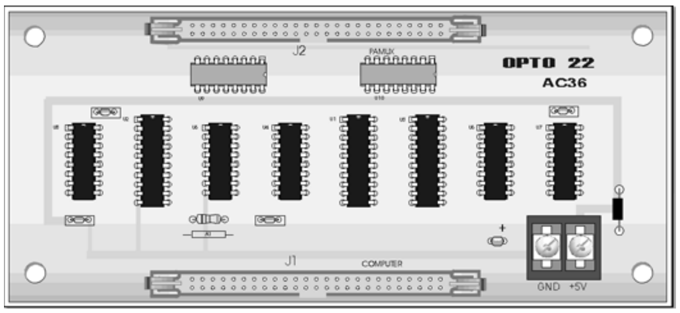


Figure 1-4: AC36 Adapter Card

B4 Brain Board

Note: Information on the B4 applies to revision L or later. For revision K or earlier, refer to Opto 22 form #127.

The Pamux B4 is an addressable digital brain board that can control up to 32 input or output points in distributed I/O applications. The B4 is designed for use with the G4PB32H mounting rack for single-point digital I/O, or the PB32HQ mounting rack for quad pak digital I/O (four points of I/O per module).

The B4 features a 70-pin header connector that attach to a digital I/O mounting rack. Up to 16 B4 brain boards may be linked on a single Pamux bus to control up to 512 points of digital I/O.

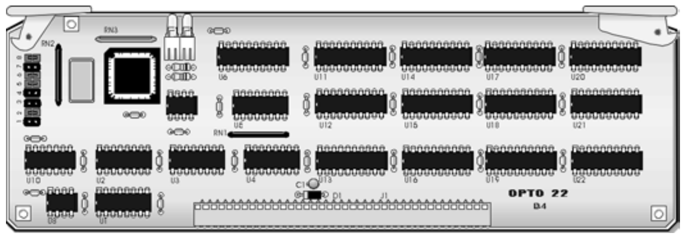


Figure 1-5: B4 Brain Board

B5 Brain Board

Note: Information on the B5 applies to revision J or later. For revision I or earlier, refer to Opto 22 form #145.

The Pamux B5 is an addressable digital brain board that can control up to 16 input or output points in distributed I/O applications. The B5 is designed for use with a variety of Opto 22 I/O mounting racks, including racks that accept single-point digital I/O, SNAP digital I/O, quad pak digital I/O, as well as racks with integrated digital I/O circuitry.

The B5 features a 50-pin female connector to attach to a mounting rack and two 50-pin male connectors to attach to the Pamux bus or a terminator board. Up to 32 B5 brain boards may be linked on a single Pamux bus to control up to 512 points of digital I/O.

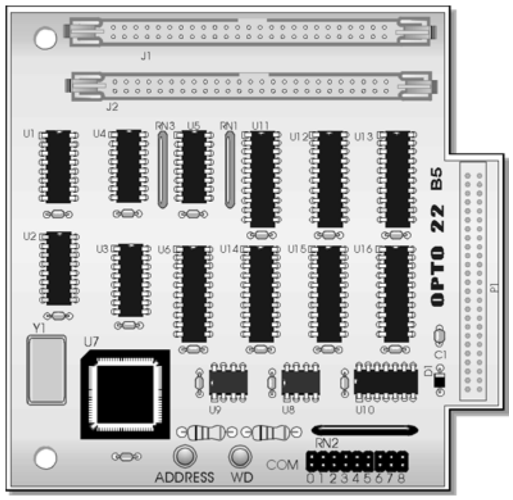


Figure 1-6: B5 Brain Board

B6 Brain Board

Note: Information on the B6 applies to revision G or later. For revision F or earlier, refer to Opto 22 form #154.

The Pamux B6 is an addressable analog brain board that can control up to 16 input or output points in distributed I/O applications. The B6 is designed for use with Opto 22 mounting racks that feature header connectors, including the PB4AH (four points of analog I/O), PB8AH (eight points), and PB16AH (16 points) using Opto 22 Classic analog modules.

The B6 features a 50-pin female connector to attach to a mounting rack and two 50-pin male connectors to attach to the Pamux bus or a terminator board. Up to 32 B6 brain boards may be linked on a single Pamux bus to control up to 512 points of analog I/O.

The B6 includes an on-board microprocessor that continually scans all I/O points on the mounting rack, performs necessary conversions, and then updates a dual-port RAM. The host computer transfers data along the Pamux bus by reading from or writing to the dual-port RAM.

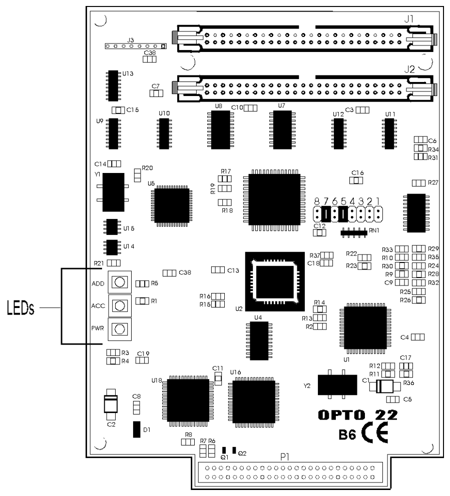


Figure 1-7: B6 Brain Board

SNAP-B6 Brain

The SNAP-B6 is a high-speed, addressable brain that can remotely control a mix of both analog and digital I/O modules, using the Pamux® protocol. Since the SNAP-B6 is designed for use with Opto 22's SNAP "B series" mounting racks, capable of handling eight, 12, or 16 I/O modules, it has a maximum capacity of 32 analog channels, or 16 analog and 32 digital channels.

The equivalent of two regular B6 analog brains and one regular B4 digital brain, the SNAP-B6 provides power and flexibility in a compact package. The SNAP-B6 includes an on-board microprocessor that continually scans all I/O points on the mounting rack, performs necessary conversions, and then updates a dual-port RAM. The host computer transfers data along the Pamux bus by reading from or writing to the dual-port RAM.

The SNAP-B6 includes an adapter cable with two 50-pin connectors to attach to the Pamux bus or a terminator board (purchased separately). Up to 16 SNAP-B6 brains can be linked on a single Pamux bus to control up to 512 points of analog or digital I/O. Each SNAP-B6 requires 5 VDC ±0.1 V @ 1.0 A (plus an additional 0.5 A if a terminator board is installed).

TERM1 Terminator Board

Both ends of a Pamux bus must be terminated. At the host computer end of the bus, an adapter card such as the AC28 includes terminator resistors. At the other end, the final Pamux brain board on the bus must be terminated with a TERM1 or TERM2 terminator board. See page 130 for an explanation of which one is appropriate for your application. The TERM1 must be used in systems with unshielded flat ribbon cable. These systems are limited to 500 feet.

Terminator boards plug directly into B5 and B6 brain boards. At a B4 station, a terminator board plugs into the I/O mounting rack such that the component side of the terminator board faces away from the modules.

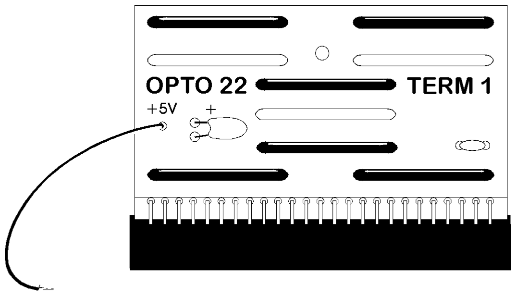


Figure 1-8: TERM1 Terminator Board

TERM2 Terminator Board

The TERM2 is identical to the TERM1 in size and function. See page 130 for an explanation of which one is appropriate for your application. The TERM2 must be used in systems with shielded flat ribbon cable. These systems are limited to 250 feet.

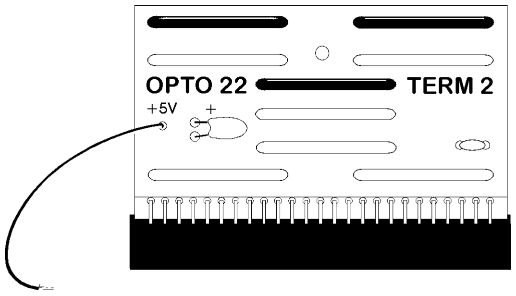


Figure 1-9: TERM2 Terminator Board

UCA4 Universal Communication Adapter

The UCA4 is a general-purpose adapter card used to connect any TTL device to the Pamux bus. Its purpose is to allow a user to build a custom interface to a Pamux system. For example, the UCA4 can be used to connect Pamux to a PDP-11, an Intel single-board computer, or any other device with a parallel I/O interface.

The UCA4 features terminators, bus drivers, a 50-pin header connector used to interface with the Pamux bus, and two 50-pin male connectors used to link with a TTL device. Because the incoming pins on the UCA4 are not defined, users can wire-wrap their own custom TTL logic on the board (a wiring schematic is provided). In addition, on-board SIP sockets allow users to incorporate additional logic circuitry directly on the board. All active parts are socketed to simplify replacement.

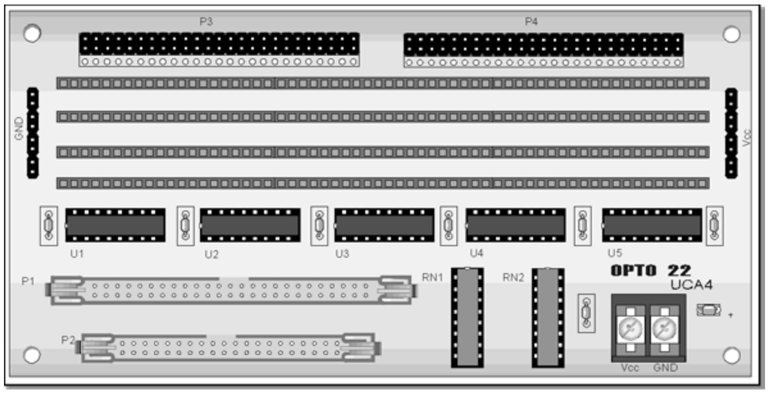


Figure 1-10: UCA4 Universal Communication Adapter

CHAPTER 2

SYSTEM SETUP

OVERVIEW

This chapter explains how to construct a Pamux system with B4, B5, and B6 stations. It covers general system configuration, installation of brain boards into I/O mounting racks, jumper configuration, cables and connectors, power supplies, and accessory devices.

DESIGNING A SYSTEM

Pamux is a distributed I/O system linked via a 50-pin flat-ribbon cable. As long as the total cable length does not exceed 500 feet, Pamux stations can be installed anywhere along the system. In addition, B4, B5, and B6 stations can be daisy-chained within a single Pamux system. This provides total flexibility in laying out your system configuration.

Each Pamux station consists of a brain board and an I/O mounting rack. Digital stations use B4 and B5 boards; analog stations use B6 boards. The B4 controls up to 32 points of digital I/O, the B5 controls up to 16 points of digital I/O, and the B6 controls up to 16 points of analog I/O.

In addition to a brain board and mounting rack, the last station on a Pamux bus must also include a terminator board (TERM1 or TERM2). See page 130 for selection criteria.

When laying out a system, keep in mind that the high cost of electrical wiring generally makes it best to place the control or monitoring I/O point as close to the controlled device as possible.

Note that round parallel ribbon cable should not be used with Pamux because it may introduce cross-talk between overlaying data lines. Use flat-ribbon cable only. (See Appendix B for cable specifications.)

Figures 2-1, 2-2, and 2-3 show three system configuration options: clustered, distributed, and distributed clustered. Clustered systems are used to accommodate several Pamux stations in a tight location. Distributed systems include single Pamux stations distributed over a larger area. Distributed clustered systems involve clusters of Pamux stations distributed over several locations. As long as the total cable length does not exceed 500 feet, the choice of system configuration is up to you.

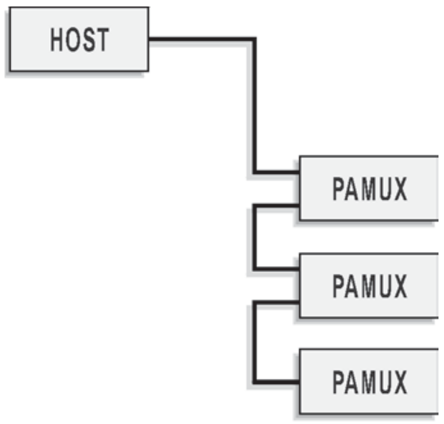


Figure 2-1: Pamux Clustered System

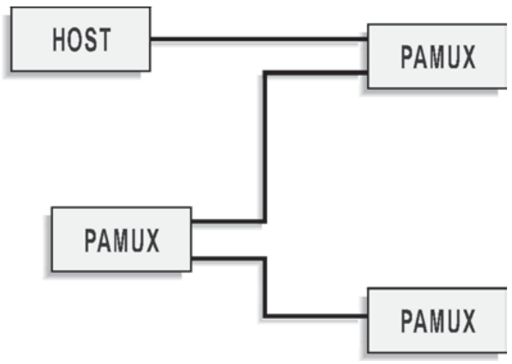


Figure 2-2: Pamux Distributed System

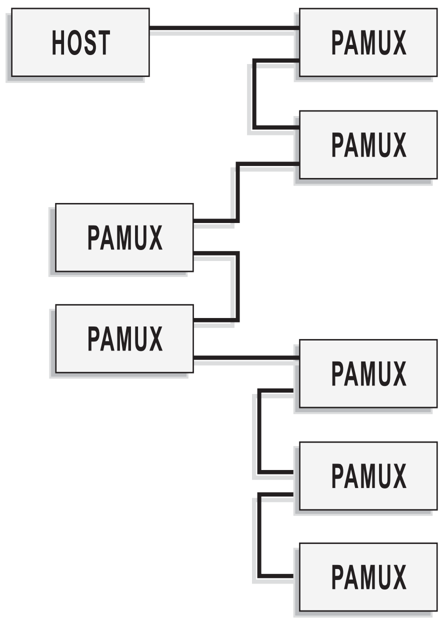


Figure 2-3: Pamux Distributed Clustered System

SETTING UP A B4 STATION

This section describes how to install the B4 digital I/O brain board on a compatible mounting rack. It also discusses B4 configuration issues, including how to set jumpers for the address, watchdog, and reset line. Finally, it addresses how to install a terminator board when a B4 station is at one end of a Pamux system and describes the LED indicators.

Installing the B4 on a Mounting Rack

The B4 brain board measures 9.25 by 2.9 inches. It includes a 70-pin box connector along its bottom edge to attach to a digital I/O mounting rack. Two levers are located on opposite corners of the board. These levers operate with the card guides on the mounting rack to hold the brain board in place or to help release it from the rack.

Figure 2-4 is a detailed illustration of the B4 along with its dimensions.

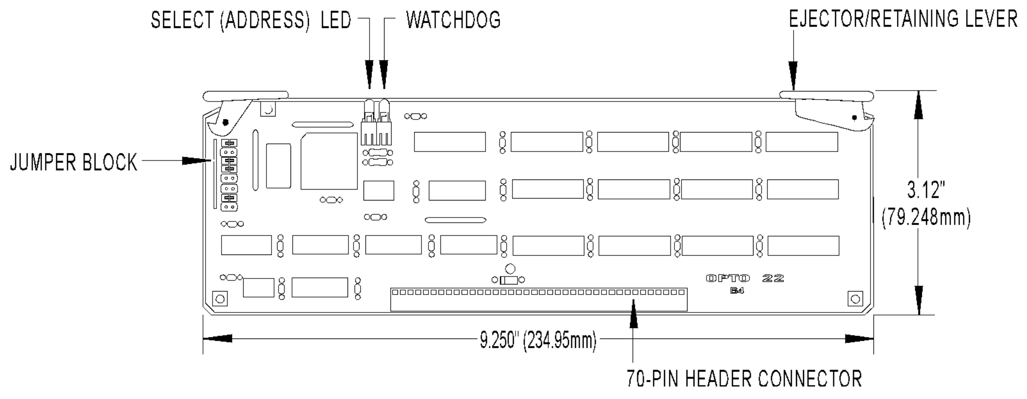


Figure 2-4: Dimensions of the B4 Brain Board

The B4 can be installed in either of two I/O mounting racks:

- G4PB32H — *32 channels of single-point G4 I/O*
- PB32HQ — *32 channels of quad pak I/O (8 modules, 4 points per module)*

The G4PB32H mounting rack uses single-point digital modules. It offers a dense footprint and point-by-point I/O configuration flexibility.

The PB32HQ also offers a dense footprint but uses Opto 22 quad pak modules. Each quad pak contains four discrete points of I/O in one package. The PB32HQ is thus configured in four-point increments.

Figure 2-5 shows how the B4 brain board is installed on either mounting rack.

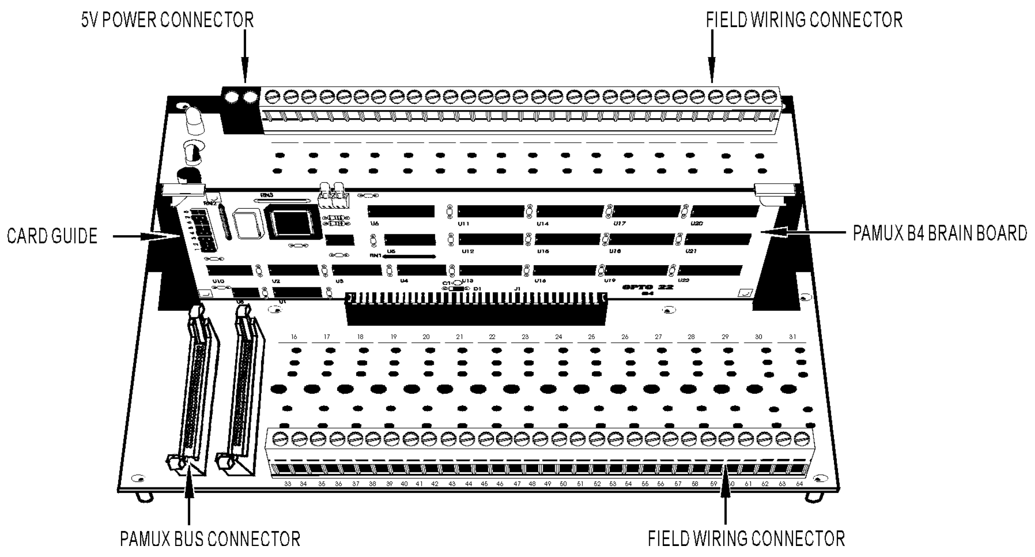


Figure 2-5: Installation of the B4 on a Mounting Rack

Figures 2-6 and 2-7 show the mounting dimensions of these racks.

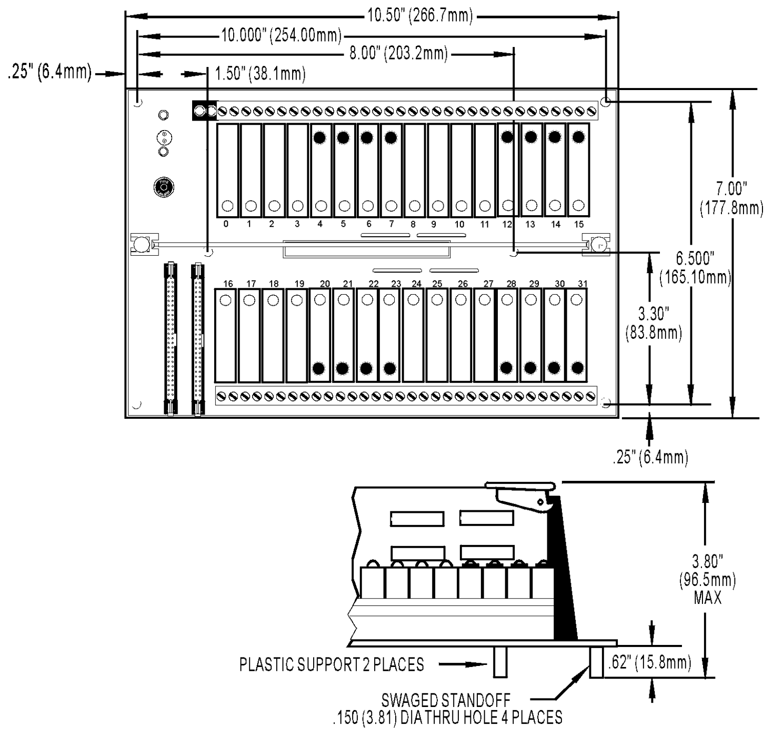


Figure 2-6: Mounting Dimensions of the G4PB32H

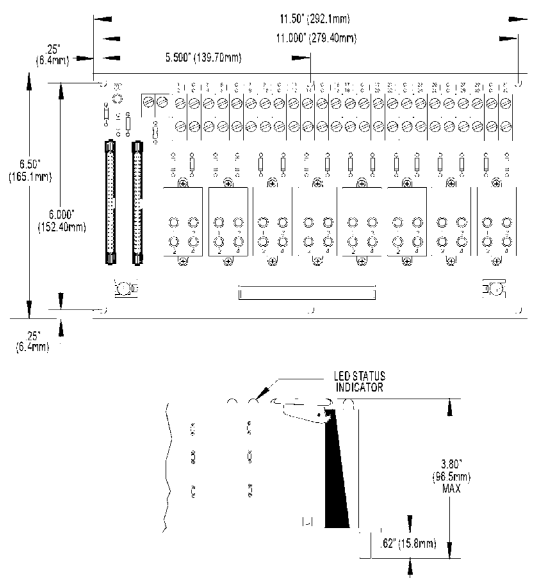


Figure 2-7: Mounting Dimensions of the PB32HQ

Figure 2-8 shows the vertical dimensions of the B4 mounted on either rack.

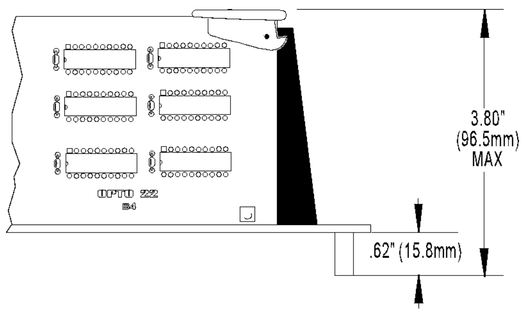


Figure 2-8: Vertical Dimensions of the B4 Mounted on a Rack

Setting B4 Jumpers

The B4 includes eight jumpers. Jumpers 1 through 4 set the address, jumpers 5 and 6 set the watchdog functionality, and jumpers 7 and 8 determine the behavior of the reset line.

Jumpers 1–4 (Address)

These jumpers configure the base address of the B4. Since the brain board controls 32 points of I/O, while the Pamux data bus is only eight bits wide, the B4 must be accessed as four consecutive banks of eight I/O channels each. Each bank has its own address.

The four banks on the B4 have contiguous addresses. The bank 0 address is the base address of the B4; the bank 1 address is the base address plus one; the bank 2 address is the base address plus two; and the bank 3 address is the base address plus three. Hence, only the base address needs to be configured. Refer to Table 2-1 to determine how to set this base address.

Note that each Pamux station on a bus must have a unique address.

Table 2-1: B4 Address Jumpers

Base Address	Jumper 4	Jumper 3	Jumper 2	Jumper 1
0	Out	Out	Out	Out
4	Out	Out	Out	In
8	Out	Out	In	Out
12	Out	Out	In	In
16	Out	In	Out	Out
20	Out	In	Out	In
24	Out	In	In	Out
28	Out	In	In	In
32	In	Out	Out	Out
36	In	Out	Out	In
40	In	Out	In	Out
44	In	Out	In	In
48	In	In	Out	Out
52	In	In	Out	In
56	In	In	In	Out
60	In	In	In	In

Jumpers 5 and 6 (Watchdog)

A watchdog timer shuts down a process when the host computer goes off line. The watchdog timer on the B4 depends on a periodic read or write strobe from the host processor. The individual B4 need not be addressed. The absence of a strobe for a specified time activates the watchdog function.

Using jumpers 5 and 6, you can configure the B4 to trigger one of four actions upon a timeout. Refer to Table 2-2.

Table 2-2: B4 Watchdog Jumpers

Watchdog Function	Jumper 6	Jumper 5
No action	In	In
Activate relay channel 0	In	Out
Deactivate all relay channels	Out	In
Activate relay channel 0 and deactivate relay channels 1-31	Out	Out

The watchdog timeout interval is set within the hardware and cannot be changed by the user. Refer to Table 2-3 for minimum, typical, and maximum watchdog timeout values.

Table 2-3: Watchdog Timeout Values

Minimum	Typical	Maximum
1.0 sec	1.6 sec	2.25 sec

Jumpers 7 and 8 (Reset)

One of the control lines on the Pamux bus is the reset line. This line is used for turning off the relays on all Pamux stations on the bus. Note that the reset is not intended to be used to shut off outputs upon a system communication error.

Two jumpers control how the reset line affects the B4. Jumper 7 determines the polarity of the reset line, either active high or active low, as shown in Table 2-4. In general, it does not matter which polarity you select as long as you are consistent throughout your Pamux system.

Table 2-4: B4 Reset Jumper

Reset Level	Jumper 7
Active High	In
Active Low	Out

Jumper 8 determines how the reset line affects the watchdog timer function of channel 0.

If jumper 8 is not installed, the reset line will not affect the watchdog timer function. Hence, if channel 0 activates due to a watchdog condition, an active reset line will have no effect on channel 0 (although it will deactivate any other channels that are on).

If jumper 8 is installed and the watchdog jumpers are configured to activate channel 0 upon a timeout, the state of channel 0 depends on whether or not the reset activates before the timeout occurs:

- If the reset line activates first, all outputs will deactivate. If a subsequent timeout occurs, no effect will take place until the reset line deactivates, at which time the watchdog function will take place and channel 0 will activate.
- If the timeout occurs first, the watchdog function takes place and channel 0 activates. If a subsequent reset occurs, channel 0 will not be affected and will remain active.

Terminating a B4 Station

For stations on a Pamux bus to operate correctly, **both ends of the bus must be terminated.** The host computer and the last Pamux station on the bus are the only devices that should be terminated. Note that if you are using an Opto 22 Pamux adapter card, the host computer is automatically terminated, since termination resistors are built into the card.

To terminate a B4 station, plug a Pamux bus terminator board (TERM1 or TERM2) into either connector on the I/O mounting rack. See page 130 for selection criteria. When the terminator board is installed correctly, its component side faces away from the I/O modules and its red wire connects to the +5V terminal on the rack.

Figure 2-9 illustrates the proper installation of the terminator board.

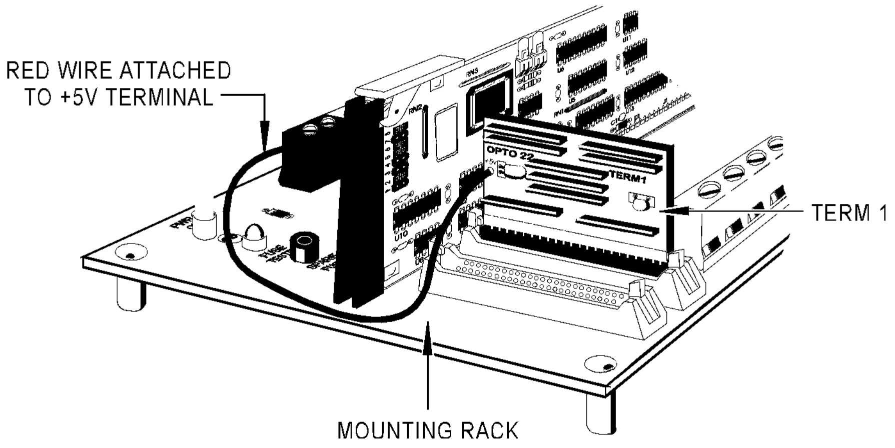


Figure 2-9: Terminator Board Installed on a B4-Compatible Mounting Rack

B4 LED Indicators

The B4 brain board includes the following LEDs:

- **Select (Address)** — This LED is on whenever the brain board is addressed (read from or written to) on the Pamux bus. It is off otherwise. For each operation the LED stays on for about 250 msec, so if the bus is very active the LED may appear constantly on.
- **Watchdog** — This LED stays on if the Pamux bus is idle (no strobe is present) for more than 1.2 seconds. It is off otherwise. Note that unlike the Select LED, this LED monitors overall bus activity.

SETTING UP A B5 STATION

This section describes how to install the B5 digital I/O brain board on a compatible mounting rack. It also discusses B5 configuration issues, including how to set jumpers for the address, watchdog, and reset line. Finally, it addresses how to install a terminator board when a B5 station is at one end of a Pamux system and describes the LED indicators.

Installing the B5 on a Mounting Rack

The B5 brain board measures 4.6 by 4.5 inches. It includes a 50-pin female connector to attach to a digital I/O mounting rack. At the top of the brain board are two 50-pin male header connectors used to link the brain board to the Pamux bus. For the last brain board on a Pamux bus, one of these connectors holds the terminator board.

Figure 2-10 is a detailed illustration of the B5 along with its dimensions.

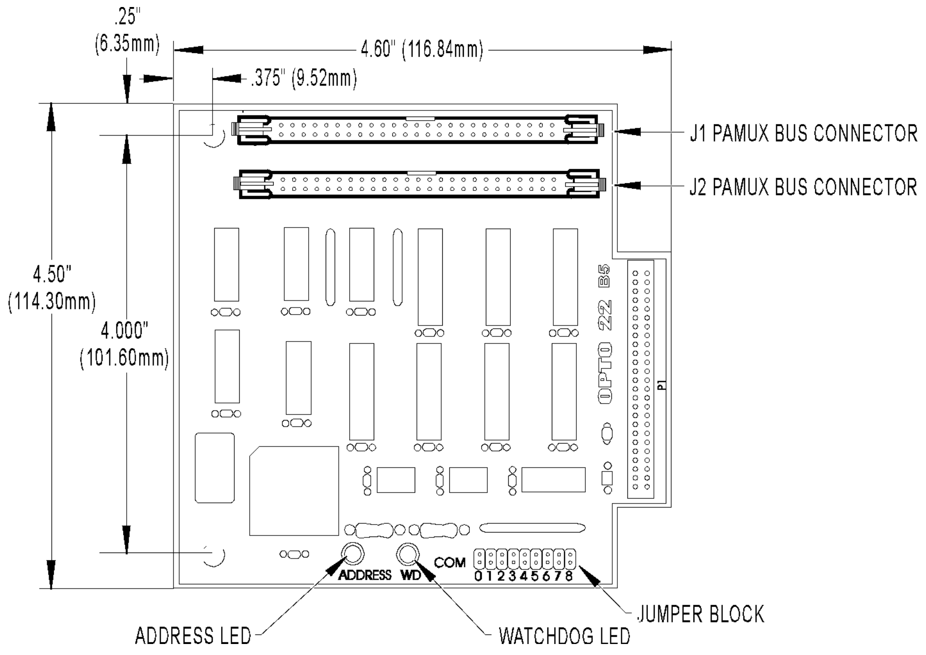


Figure 2-10: Dimensions of the B5 Brain Board

The following I/O mounting racks are available for the Pamux B5 brain board:

- G4PB8H — *8-channels of single-point G4 digital I/O*
- G4PB16H — *16-channels of single-point G4 digital I/O*
- G4PB16HC — *16-channels of single-point G4 digital I/O*
- G4PB16J/K/L — *16-channels of integrated single-point digital inputs*
- PB4H — *4-channels of single-point standard digital I/O*
- PB8H — *8-channels of single-point standard digital I/O*
- PB16H — *16-channels of single-point standard digital I/O*
- PB16HC — *16-channels of single-point standard digital I/O*
- PB16HQ — *4-channels of quad pak I/O (four points per module)*
- PB16J/K/L — *16-channels of integrated single-point digital inputs*
- SNAPD4M — *16-channels of SNAP digital I/O*
- SNAPD4MC — *16-channels of SNAP digital with tie points*
- SNAPD4MC-P — *16-channels of SNAP digital with pluggable connectors.*

Figure 2-11 shows how the B5 brain board mounts on these racks.

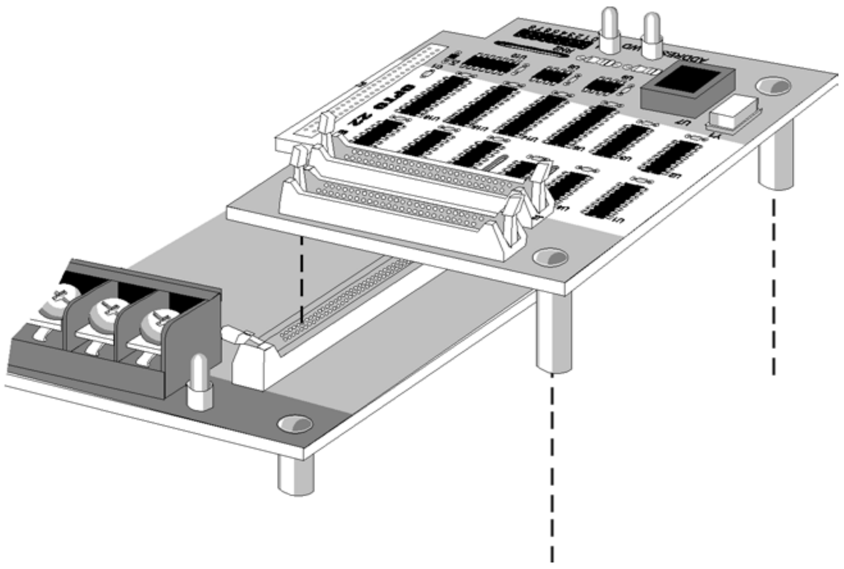


Figure 2-11: Installation of the B5 on a Mounting Rack

Figures 2-12 through 2-21 show the mounting dimensions of these racks with the B5 brain board installed.

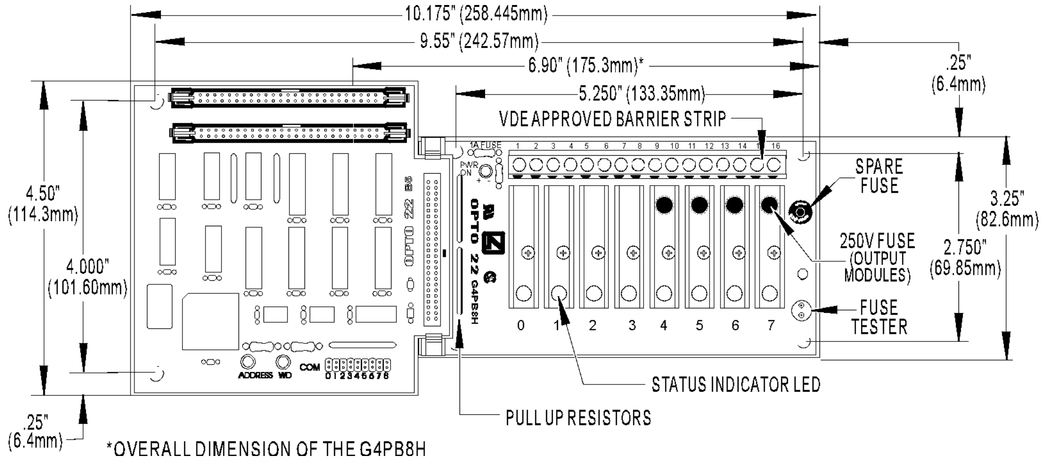


Figure 2-12: Mounting Dimensions of the G4PB8H with a B5 Installed

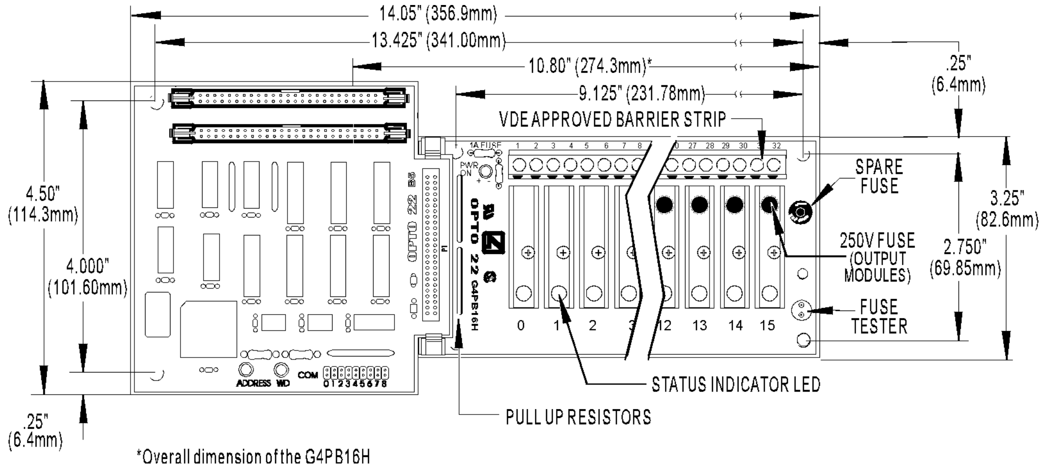


Figure 2-13: Mounting Dimensions of the G4PB16H with a B5 Installed

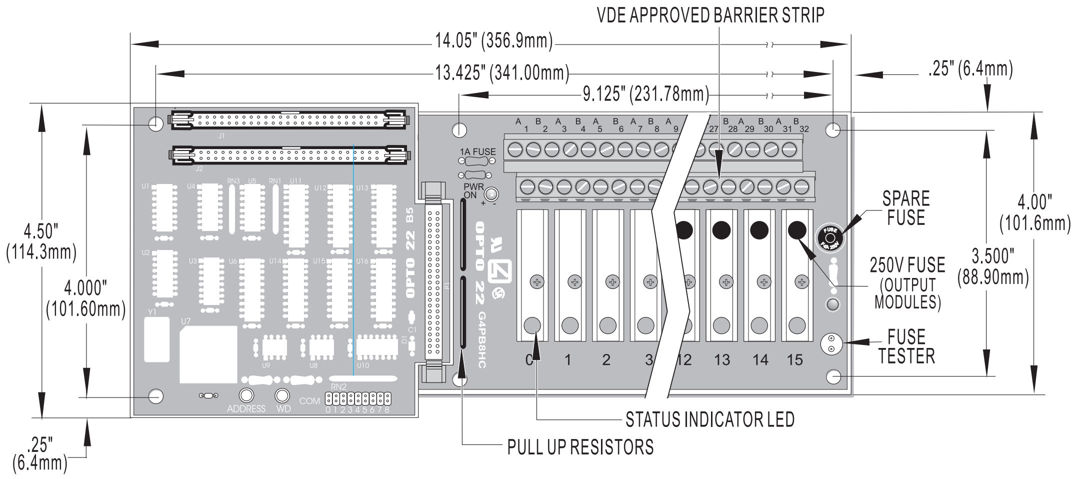


Figure 2-14: Mounting Dimensions of the G4PB16HC with a B5 Installed

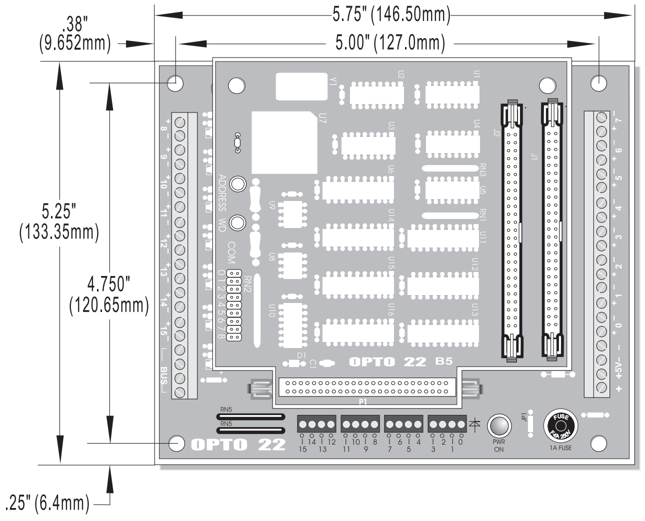


Figure 2-15: Mounting Dimensions of the G4PB16J/K/L with a B5 Installed

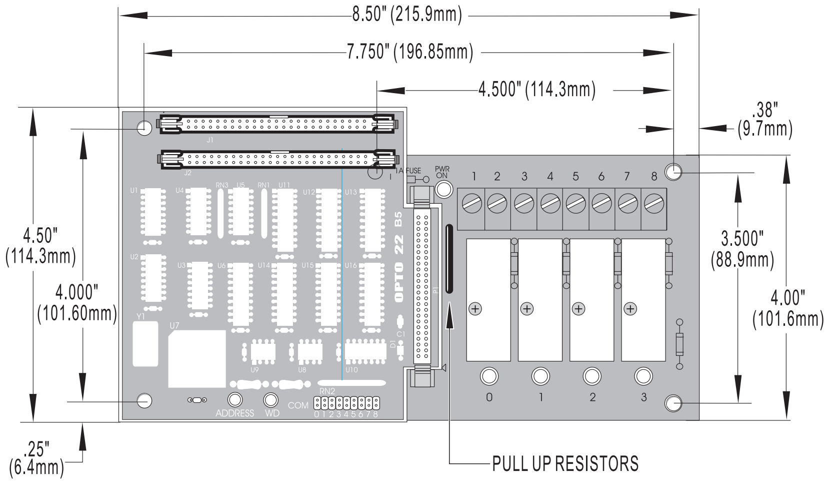


Figure 2-16: Mounting Dimensions of the PB4H with a B5 Installed

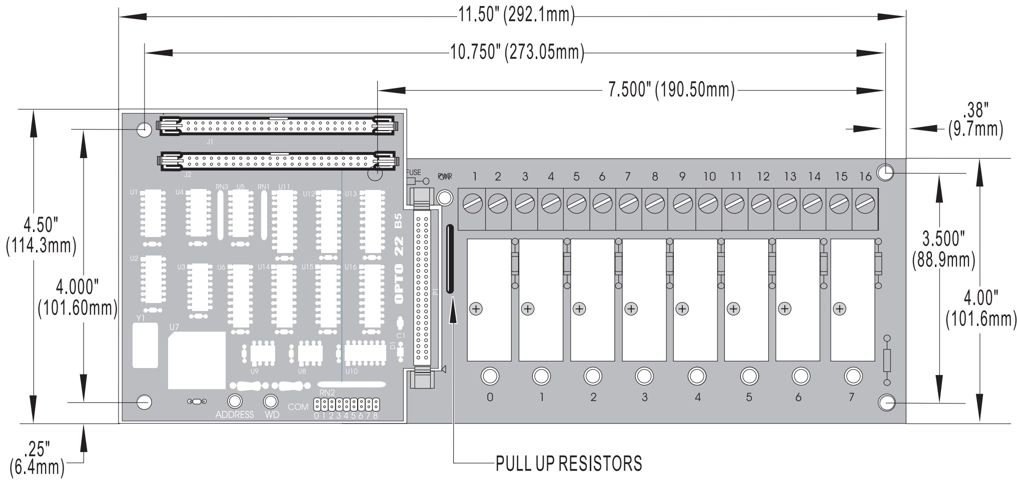


Figure 2-17: Mounting Dimensions of the PB8H with a B5 Installed

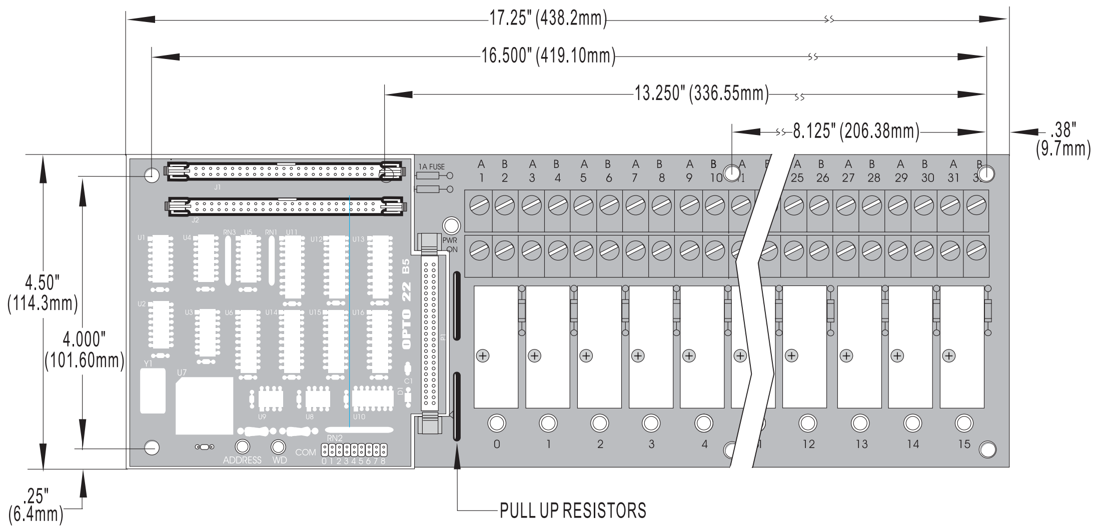


Figure 2-18: Mounting Dimensions of the PB16H with a B5 Installed

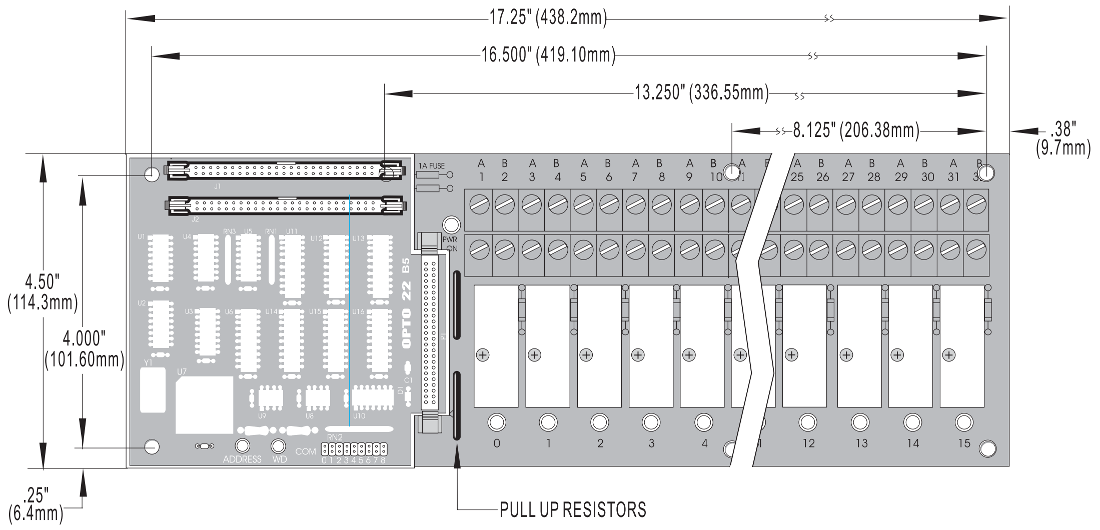


Figure 2-19: Mounting Dimensions of the PB16HC with a B5 Installed

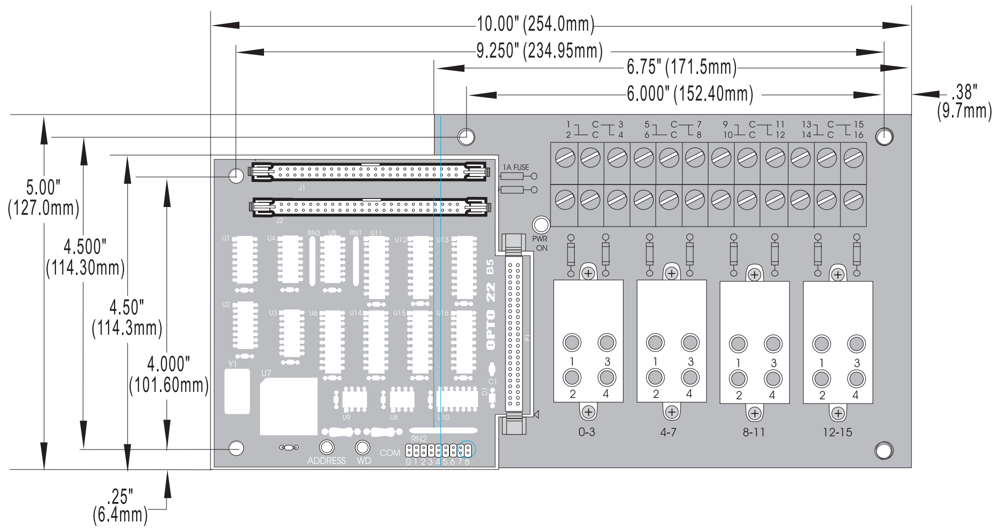


Figure 2-20: Mounting Dimensions of the PB16HQ with a B5 Installed

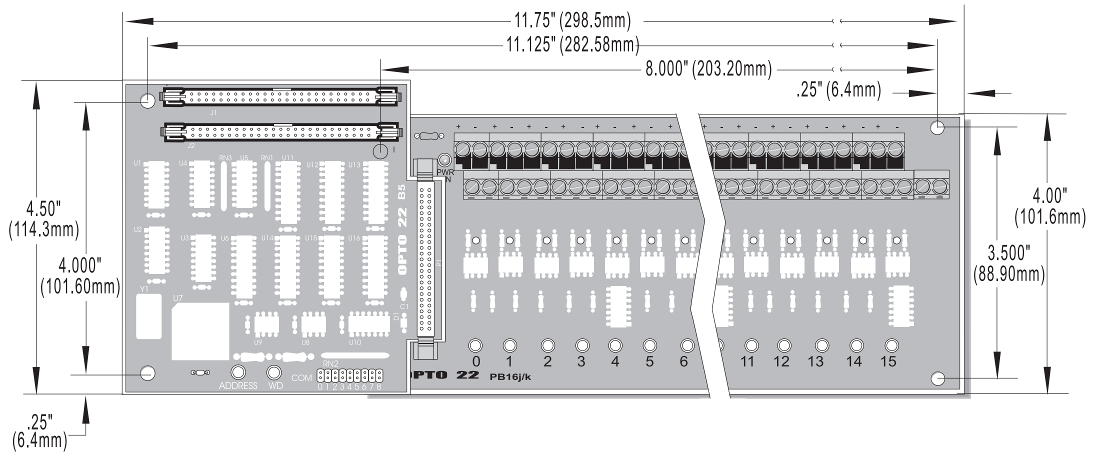


Figure 2-21: Mounting Dimensions of the PB16J/K/L with a B5 Installed

Figure 2-22 shows the vertical dimensions of the B5 mounted on any rack except the PB16HQ.

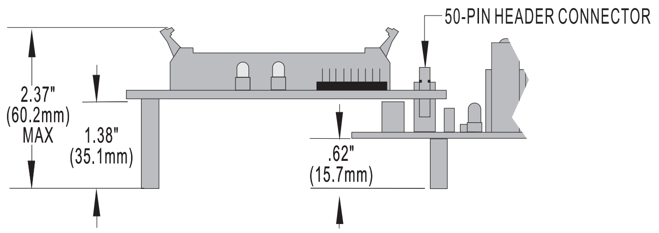


Figure 2-22: Vertical Dimensions of the B5 Mounted on Racks other than the PB16HQ

Figure 2-23 shows the vertical dimensions of the B5 mounted on the PB16HQ. This rack accepts quad pak modules, which are taller than standard digital I/O modules.

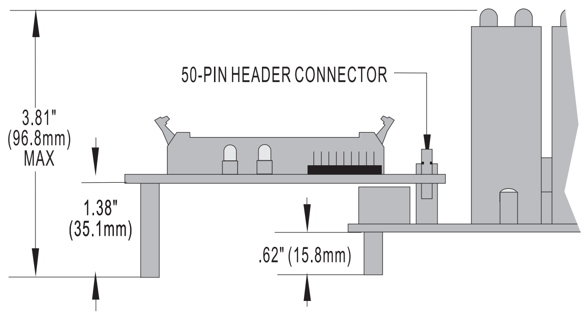


Figure 2-23: Vertical Dimensions of the B5 Mounted on a PB16HQ

Setting B5 Jumpers

The B5 includes nine jumpers. Jumpers 0 through 4 set the address, jumpers 5 and 6 set the watchdog functionality, jumper 7 sets the reset line polarity, and jumper 8 determines how the reset line affects the watchdog timer.

Jumpers 0–4 (Address)

These jumpers configure the base address of the B5. Since the brain board controls 16 points of I/O, while the Pamux data bus is only eight bits wide, the B5 must be accessed as two consecutive banks of eight I/O channels each. The least significant bit of the Pamux address bus is used to select which bank is accessed (0 = low bank, 1 = high bank). The other 5 bits of the Pamux bus address determine which Pamux station is active.

Refer to Table 2-5 on the following page to determine how to set the base address of the B5.

Note: That each Pamux station on a bus must have a unique address.

Jumpers 5 and 6 (Watchdog)

A watchdog timer shuts down a process when the host computer goes off line. The watchdog timer on the B5 depends on a periodic read or write strobe from the host processor. The individual B5 need not be addressed. The absence of a strobe for a specified time activates the watchdog function.

Using jumpers 5 and 6, you can configure the B5 to trigger one of four actions upon a timeout. Refer to Table 2-6.

Table 2-5: B5 Address Jumpers

Base Address	Jumper 4	Jumper 3	Jumper 2	Jumper 1	Jumper 0
0	Out	Out	Out	Out	Out
2	Out	Out	Out	Out	In
4	Out	Out	Out	In	Out
6	Out	Out	Out	In	In
8	Out	Out	In	Out	Out
10	Out	Out	In	Out	In
12	Out	Out	In	In	Out
14	Out	Out	In	In	In
16	Out	In	Out	Out	Out
18	Out	In	Out	Out	In
20	Out	In	Out	In	Out
22	Out	In	Out	In	In
24	Out	In	In	Out	Out
26	Out	In	In	Out	In
28	Out	In	In	In	Out
30	Out	In	In	In	In
32	In	Out	Out	Out	Out
34	In	Out	Out	Out	In
36	In	Out	Out	In	Out
38	In	Out	Out	In	In
40	In	Out	In	Out	Out
42	In	Out	In	Out	In
44	In	Out	In	In	Out
46	In	Out	In	In	In
48	In	In	Out	Out	Out
50	In	In	Out	Out	In
52	In	In	Out	In	Out
54	In	In	Out	In	In
56	In	In	In	Out	Out
58	In	In	In	Out	In
60	In	In	In	In	Out
62	In	In	In	In	In

The watchdog timeout interval is set within the hardware and cannot be changed by the user. Refer to Table 2-7 for minimum, typical, and maximum watchdog timeout values.

Table 2-6: B5 Watchdog Jumpers

Watchdog Function	Jumper 6	Jumper 5
No action	In	In
Activate relay channel 0	In	Out
Deactivate all relay channels	Out	In
Activate relay channel 0 and deactivate relay channels 1-15	Out	Out

Table 2-7: Watchdog Timeout Values

Minimum	Typical	Maximum
1.0 sec	1.6 sec	2.25 sec

Jumpers 7 and 8 (Reset)

One of the control lines on the Pamux bus is the reset line. This line is used for turning off the relays on all Pamux stations on the bus. Note that the reset is not intended to be used to shut off outputs upon a system communication error.

Two jumpers control how the reset line affects the B5. Jumper 7 determines the polarity of the reset line, either active high or active low, as shown in Table 2-8. In general, it does not matter which polarity you select as long as you are consistent throughout your Pamux system.

Table 2-8: B5 Reset Jumper

Reset Level	Jumper 7
Active High	In
Active Low	Out

Jumper 8 determines how the reset line affects the watchdog timer function of channel 0.

If jumper 8 is not installed, the reset line will not affect the watchdog timer function. Hence, if channel 0 activates due to a watchdog condition, an active reset line will have no effect on channel 0 (although it will deactivate any other channels that are on).

If jumper 8 is installed and the watchdog jumpers are configured to activate channel 0 upon a timeout, the state of channel 0 depends on whether or not the reset activates before the timeout occurs:

- If the reset line activates first, all outputs will deactivate. If a subsequent timeout occurs, no effect will take lace until the reset line deactivates, at which time the watchdog function will take place and channel 0 will activate.
- If the timeout occurs first, the watchdog function takes place and channel 0 activates. If a subsequent reset occurs, channel 0 will not be affected and will remain active.

Terminating a B5 Station

For stations on a Pamux bus to operate correctly, **both ends of the bus must be terminated.** The host computer and the last Pamux station on the bus are the only devices that should be terminated. Note that if you are using an Opto 22 Pamux adapter card, the host computer is automatically terminated, since termination resistors are built into the card.

To terminate a B5 station, plug a Pamux bus terminator board (TERM1 or TERM2) into either connector on the brain board. See page 130 for selection criteria. When the terminator board is installed correctly, its component side faces away from the brain board components and its red wire connects to the +5V terminal on the rack.

Figure 2-24 illustrates the proper installation of the terminator board.

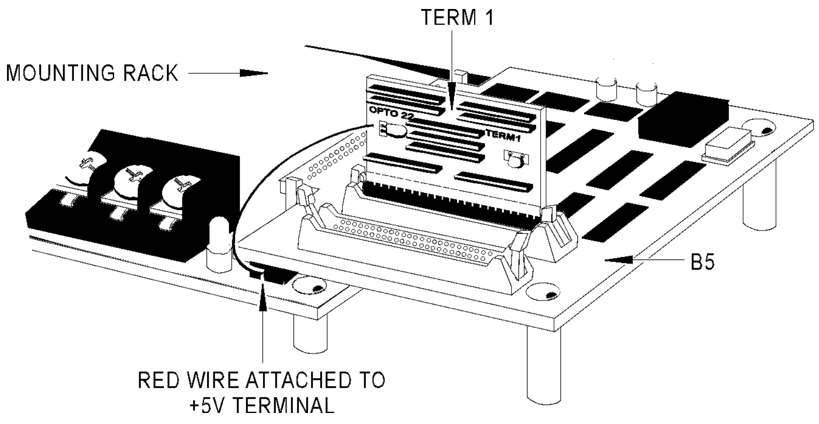


Figure 2-24: Terminator Board Installed on a B5-Compatible Mounting Rack

B5 LED Indicators

The B5 brain board includes the following LEDs:

- **Address** — This LED is on whenever the brain board is addressed (read from or written to) on the Pamux bus. It is off otherwise. For each operation the LED stays on for about 250 msec, so if the bus is very active the LED may appear constantly on.
- **Watchdog** — This LED stays on if the Pamux bus is idle (no strobe is present) for more than 1.2 seconds. It is off otherwise. Note that unlike the Select LED, this LED monitors overall bus activity.

SETTING UP A B6 STATION

This section describes how to install the B6 analog I/O brain board on a compatible mounting rack. It also discusses B6 configuration issues, including how to set jumpers for the address, watchdog, and reset line. Finally, it addresses how to install a terminator board when a B6 station is at one end of a Pamux system and describes the LED indicators.

Installing the B6 on a Mounting Rack

The B6 brain board measures 6.40 by 4.75 inches. It includes a 50-pin female connector to attach to an analog I/O mounting rack. At the top of the brain board are two 50-pin male header connectors used to link the brain board to the Pamux bus. For the last brain board on a Pamux bus, one of these connectors holds the terminator board.

Figure 2-25 is a detailed illustration of the B6 along with its dimensions.

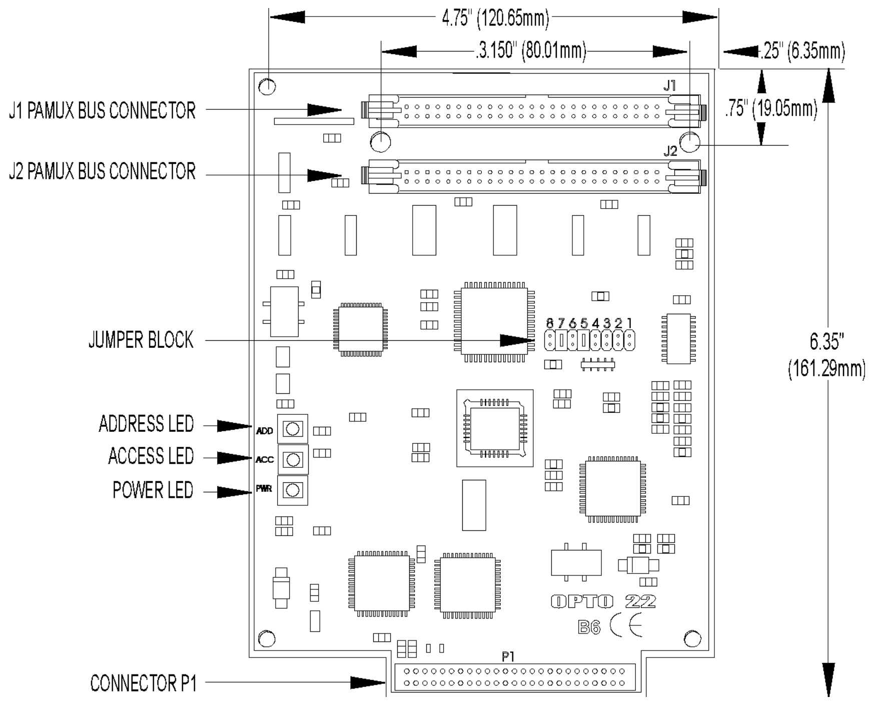


Figure 2-25: Dimensions of the B6 Brain Board

Three I/O mounting racks are available for the Pamux B6 brain board:

- PB4AH — *4-channels of single-point standard analog I/O*
- PB8AH — *8-channels of single-point standard analog I/O*
- PB16AH — *16-channels of single-point standard analog I/O*

Each mounting rack accommodates any combination of analog input or output modules and connects to the Pamux B6 brain board via a 50-pin header connection. The mounting rack includes a fuse for the 5-volt line.

Figures 2-26 through 2-28 show the mounting dimensions of these racks with the B6 brain board installed.

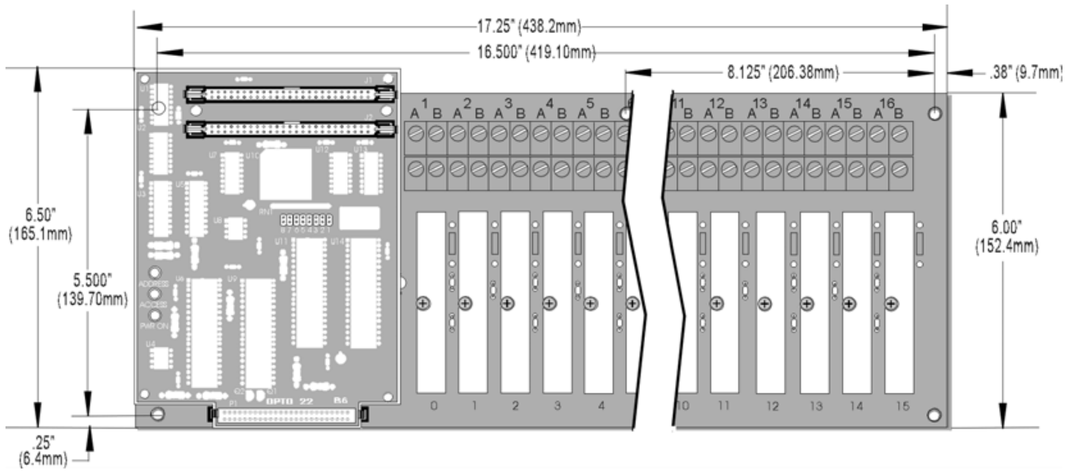


Figure 2-26: Mounting Dimensions of the PB4AH with a B6 Installed

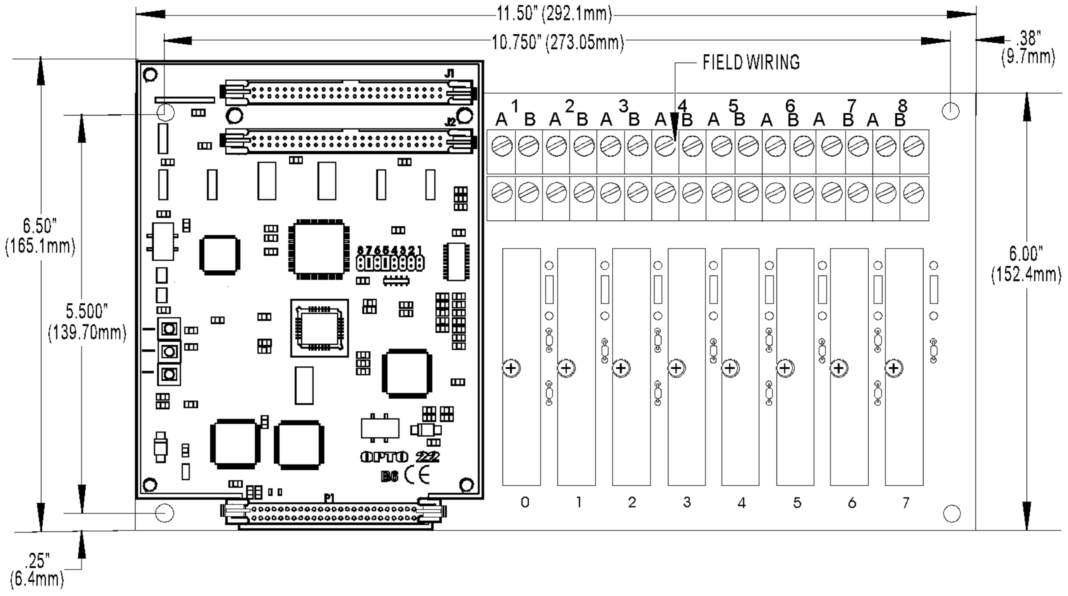


Figure 2-27: Mounting Dimensions of the PB8AH with a B6 Installed

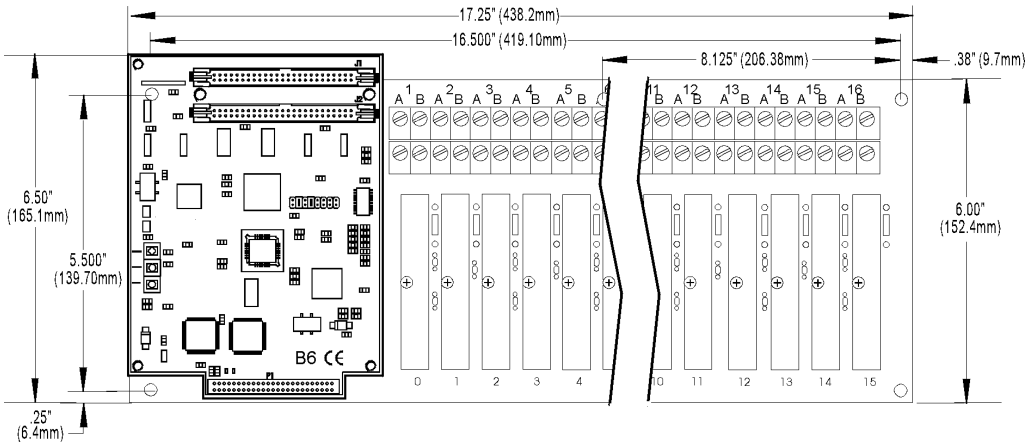


Figure 2-28: Mounting Dimensions of the PB16AH with a B6 Installed

Figure 2-29 shows the vertical dimensions of the B6 mounted on any rack.

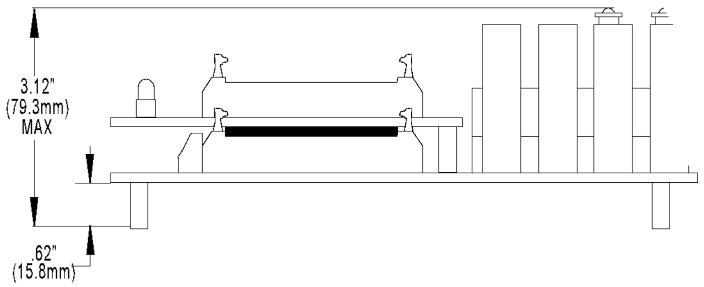


Figure 2-29: Vertical Dimensions of the B6 Mounted on a Rack

Setting B6 Jumpers

The B6 includes eight jumpers. Jumpers 1 through 5 set the address, jumper 6 is disabled, jumper 7 sets the reset line polarity, and jumper 8 sets the watchdog functionality.

Jumpers 1–5 (Address)

These jumpers configure the base address of the B6. The brain board can control 16 points of analog I/O. Data is passed to and from the host computer using one address register and one data register. Each B6 thus requires only two consecutive addresses.

Refer to Table 2-9 on the following page to determine how to set the base address of the B6.

Note that each Pamux station on a bus must have a unique address.

Table 2-9: B6 Address Jumpers

Base Address	Jumper 5	Jumper 4	Jumper 3	Jumper 2	Jumper 1
	Out	Out	Out	Out	Out
2	Out	Out	Out	Out	In
4	Out	Out	Out	In	Out
6	Out	Out	Out	In	In
8	Out	Out	In	Out	Out
10	Out	Out	In	Out	In
12	Out	Out	In	In	Out
14	Out	Out	In	In	In
16	Out	In	Out	Out	Out
18	Out	In	Out	Out	In
20	Out	In	Out	In	Out
22	Out	In	Out	In	In
24	Out	In	In	Out	Out
26	Out	In	In	Out	In
28	Out	In	In	In	Out
30	Out	In	In	In	In
32	In	Out	Out	Out	Out
34	In	Out	Out	Out	In
36	In	Out	Out	In	Out
38	In	Out	Out	In	In
40	In	Out	In	Out	Out
42	In	Out	In	Out	In
44	In	Out	In	In	Out
46	In	Out	In	In	In
48	In	In	Out	Out	Out
50	In	In	Out	Out	In
52	In	In	Out	In	Out
54	In	In	Out	In	In
56	In	In	In	Out	Out
58	In	In	In	Out	In
60	In	In	In	In	Out
62	In	In	In	In	In

Jumper 6

This jumper has been disabled and thus has no effect.

Jumper 7 (Reset)

One of the control lines on the Pamux bus is the reset line. This line is used to clear all analog outputs on a B6 station to zero scale, then to set the configuration of the B6 to input on all positions. Note that the reset is not intended to be used to shut off outputs upon a system communication error.

Jumper 7 determines the polarity of the reset line, either active high or active low, as shown in Table 2-10. In general, it does not matter which polarity you select as long as you are consistent throughout your Pamux system.

Table 2-10: B6 Reset Jumper

Reset Level	**Jumper 7**
Active High	In
Active Low	Out

Jumper 8 (Watchdog)

A watchdog timer shuts down a process when the host computer goes off line. The watchdog function of the B6 can be enabled or disabled with jumper 8. Since the B6 watchdog function is also under software control, the watchdog register must be written to **and** the jumper must be removed for the watchdog to be enabled.

Table 2-11 shows how jumper 8 affects the watchdog. For information on software configuration of the watchdog, see Chapter 3.

Table 2-11: B6 Watchdog Jumper

Reset Level	Jumper 7
Active High	In
Active Low	Out

Terminating a B6 Station

For stations on a Pamux bus to operate correctly, **both ends of the bus must be terminated.** The host computer and the last Pamux station on the bus are the only devices that should be terminated. Note that if you are using an Opto 22 Pamux adapter card, the host computer is automatically terminated, since termination resistors are built into the card.

To terminate a B6 station, plug a Pamux bus terminator board (TERM1 or TERM2) into either connector on the brain board. See page 130 for selection criteria. When the terminator board is installed correctly, its component side faces away from the brain board components and its red wire connects to the +5V terminal on the rack.

Figure 2-30 illustrates the proper installation of the terminator board.

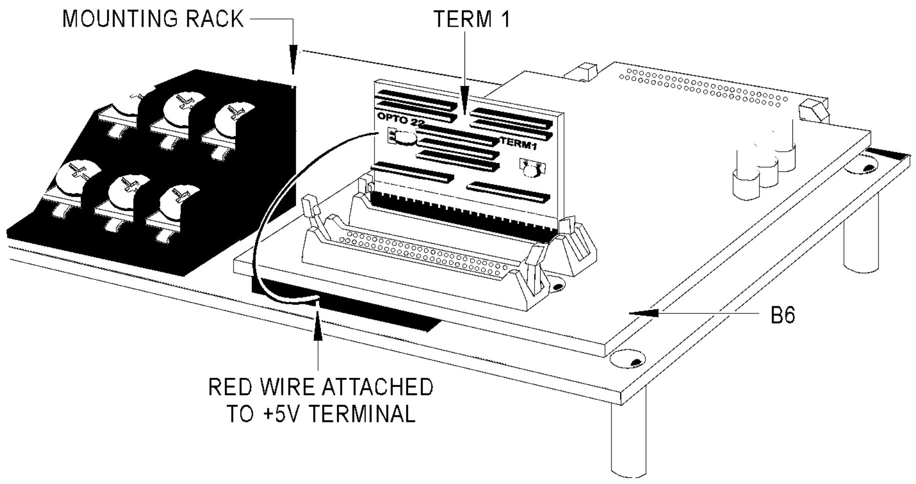


Figure 2-30: Terminator Board Installed on a B6-Compatible Mounting Rack

B6 LED Indicators

The B6 brain board includes the following LEDs:

- **Address** — This LED is on whenever the brain board is addressed (read from or written to) on the Pamux bus. It is off otherwise. For each operation the LED stays on for about 250 msec, so if the bus is very active the LED may appear constantly on.
- **Access** — This LED is on whenever access has been granted to the dual-port RAM. It remains on until access is released. (See Chapter 4 for details on getting and releasing access.)
- **Power** — This LED is on whenever power is connected to the board. It is off otherwise.

SETTING UP A SNAP-B6 STATION

This section describes how to install the SNAP-B6 brain on a SNAP B-series mounting rack. It also shows how to set jumpers and how to configure and address I/O modules.

Installing the SNAP-B6 on a Mounting Rack

The SNAP-B6 can be installed on any SNAP B-series mounting rack.

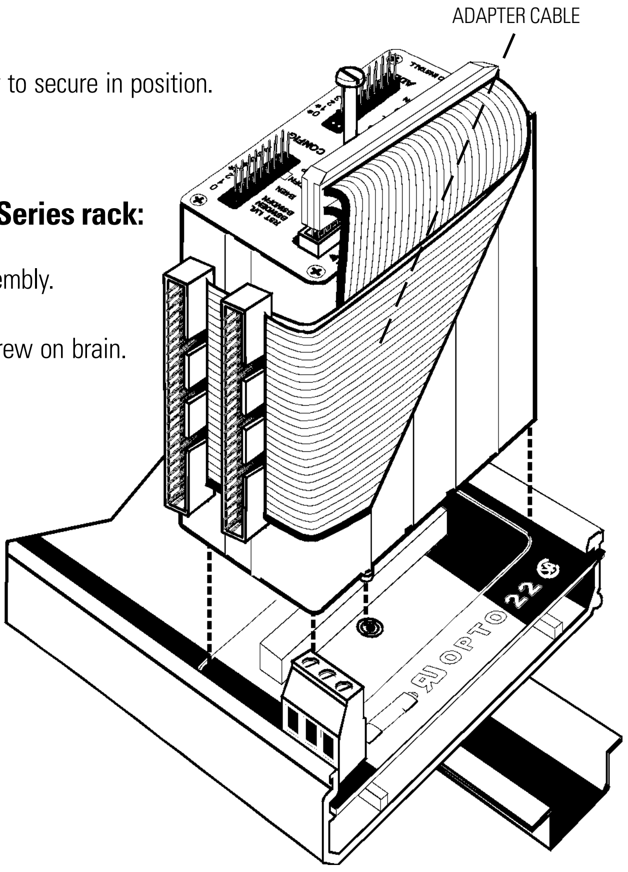
To install the SNAP-B6 brain on a B Series rack:

1. Remove power from rack assembly.
2. Align the brain connector with the mating connector on mounting rack.
3. Seat brain onto connector.
4. Use integral hold-down screw to secure in position.

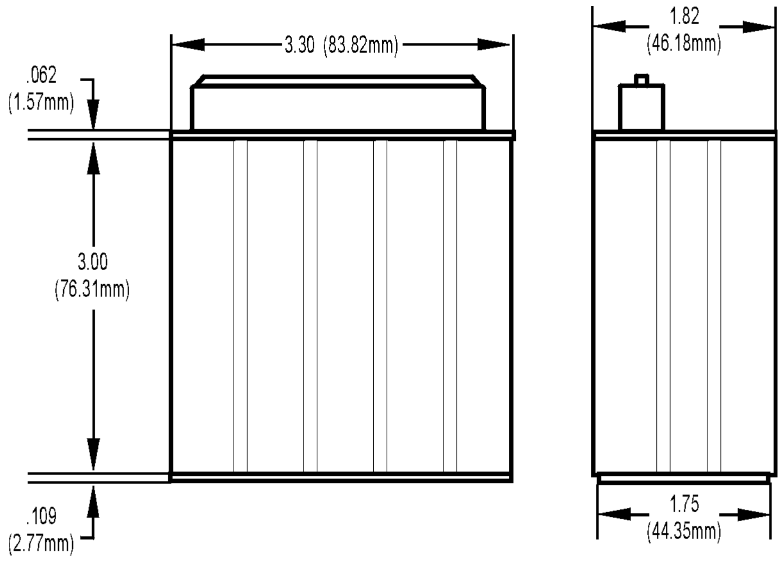
   DO NOT OVERTIGHTEN!

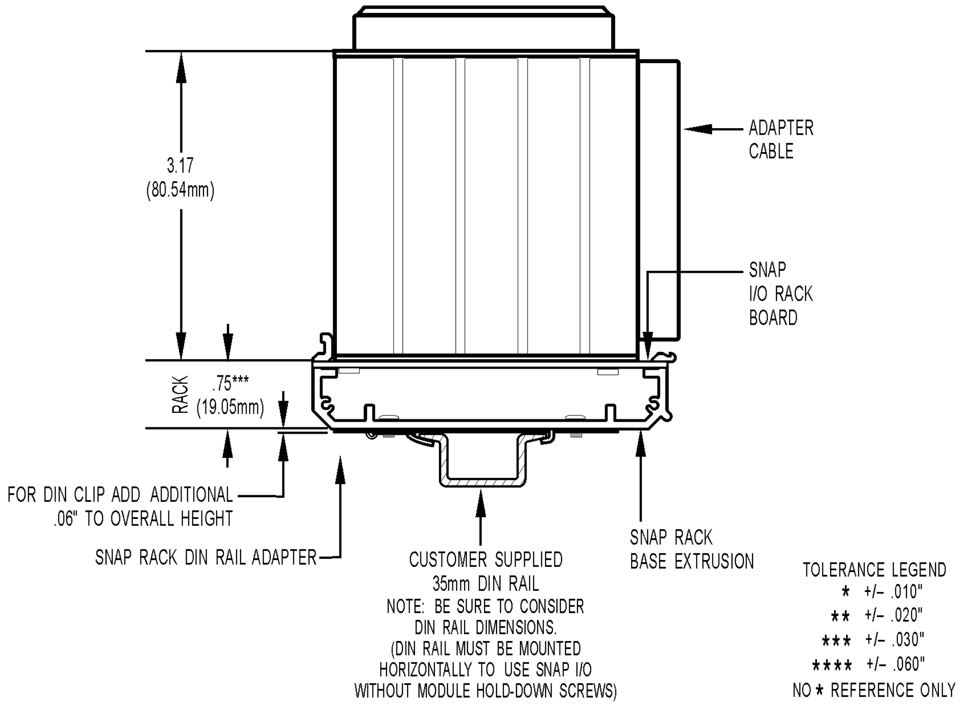
To remove the brain from a B Series rack:

1. Remove power from rack assembly.
2. Loosen integral hold-down screw on brain.
3. Pull up on brain.



Dimensions


3.30 (83.82mm)
.062 (1.57mm)
3.00 (76.31mm)
.109 (2.77mm)
1.82 (46.18mm)
1.75 (44.35mm)


3.17 (80.54mm)
ADAPTER CABLE
SNAP I/O RACK BOARD
RACK
.75*** (19.05mm)
FOR DIN CLIP ADD ADDITIONAL .06" TO OVERALL HEIGHT
SNAP RACK DIN RAIL ADAPTER
CUSTOMER SUPPLIED 35mm DIN RAIL
NOTE: BE SURE TO CONSIDER DIN RAIL DIMENSIONS.
(DIN RAIL MUST BE MOUNTED HORIZONTALLY TO USE SNAP I/O WITHOUT MODULE HOLD-DOWN SCREWS)
SNAP RACK BASE EXTRUSION
TOLERANCE LEGEND
* +/- .010"
** +/- .020"
*** +/- .030"
**** +/- .060"
NO * REFERENCE ONLY

Setting SNAP-B6 Jumpers

Table 1: Configuration Jumpers 0–7

See the next page for specific jumper settings.

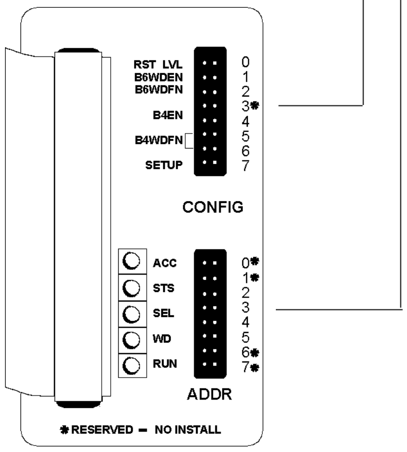
Jumper(s)	Description
0	Sets the reset line polarity
1 and 2	Set the analog watchdog
3	Not used
4	Enables digital functionality
5 and 6	Set the digital watchdog
7	Used for analog setup in a special boot mode

Table 2: Address Jumpers

These jumpers configure the base address of the SNAP-B6. Each Pamux station on a bus must have a unique address. The SNAP-B6 emulates two analog brains and, optionally, one digital brain. Each analog brain occupies two addresses, one for the data register and one for the control register. (For more information, see the *Pamux User's Guide*.) The digital brain occupies four addresses, one for every 8-bits (or bank) of I/O. If configuration jumper 4 is installed to enable digital function, address jumper 2 must not be installed.

Jumpers				With digital enabled			Without digital enabled	
J5	J4	J3	J2	An#1	An#2	Dig	An#1	An#2
Out	Out	Out	Out	00	02	04–07	00	02
Out	Out	Out	In				04	06
Out	Out	In	Out	08	10	12–15	08	10
Out	Out	In	In				12	14
Out	In	Out	Out	16	18	20–23	16	18
Out	In	Out	In				20	22
Out	In	In	Out	24	26	28–31	24	26
Out	In	In	In				28	30
In	Out	Out	Out	32	34	36–39	32	34
In	Out	Out	In				36	38
In	Out	In	Out	40	42	44–47	40	42
In	Out	In	In				44	46
In	In	Out	Out	48	50	52–55	48	50
In	In	Out	In				52	54
In	In	In	Out	56	58	60–63	56	58
In	In	In	In				60	62

Top View: SNAP-B6



Default settings are shown in **boldface**.

Table 3: Reset Jumper

Reset Level	Jumper 0
Active High	In
Active Low	**Out**

Table 4: Analog Watchdog Jumper

Watchdog	Jumper 1	Jumper 2
Disabled	**In**	--
Enabled	Out	--

Table 5: Digital Functionality Jumper

Digital Functionality	Jumper 4
Enable digital	**In**
Disable digital	Out

Table 6: Digital Watchdog Jumper

Watchdog	Jumper 5	Jumper 6
No action	In	In
Activate channel 0	Out	In
Deactivate all channels	**In**	**Out**
Activate channel 0 and deactivate channels 1–31	Out	Out

Table 7: Setup Jumper

Analog Setup	Jumper 7
Configuration mode	In
Normal operation	**Out**

Jumper 0 (Reset)

One of the control lines on the Pamux bus is the reset line. This line is used to clear all analog outputs on a SNAP-B6 station to zero scale and turn off all digital outputs, and then to set the configuration of the SNAP-B6 to input on all positions. Note that the reset is not intended to be used to shut off outputs upon a system communication error.

Jumper 0 determines the polarity of the reset line, either active high or active low, as shown in Table 4 at left. The default is active low. In general, it does not matter which polarity you select as long as you are consistent throughout your Pamux system.

Jumpers 1 and 2 (Analog Watchdog)

A watchdog timer shuts down a process when the host computer goes offline.
If the host computer does not access the SNAP-B6 analog or digital addresses for 1.6 seconds, the watchdog function is activated. As shown in Table 5, jumper 1, if installed, disables the watchdog for both analog addresses. The default is disabled. Jumper 2 is reserved for future use and has no effect.

Since the SNAP-B6 watchdog function is also under software control, the watchdog register must be written to **and** the jumper must be removed for the watchdog to be enabled. For information on software configuration of the watchdog, see Chapter 3 of the *Pamux User's Guide*.

Note: Disabling the analog watchdog does **not** disable the watchdog LED.

Jumper 3 (Not Used)

Jumper 4 (Digital Functionality)

To enable digital functionality, install jumper 4. See Table 6. The default is enabled.

Jumpers 5 and 6 (Digital Watchdog)

A watchdog timer shuts down a process when the host computer goes offline.
If the host computer does not access the SNAP-B6 analog or digital addresses for 1.6 seconds, the watchdog function is activated.

Jumpers 5 and 6 configure the digital address to take one of four actions when the watchdog is activated. The four actions are shown in Table 7. The default is all channels deactivated.

Note: Disabling the digital watchdog does **not** disable the watchdog LED.

Jumper 7 (Setup)

Because SNAP analog modules are multi-purpose, the SNAP-B6 must be set up using a special configuration mode in order to scale module readings properly. Use jumper 7, shown in Table 8, to set up this special mode before you turn the board on for the first time. (See next page for instructions.)

SNAP-B6 LED Indicators

LED	Description
ACC (Access)	LED is on whenever access has been granted to the dual-port RAM. It remains on until access is released (For more information on access, see chapter 4 of the Pamux User's Guide.)
STS (Status)	LED is on while booting (approximately two seconds) or while in reset. It flashes rapidly in configuration mode and blinks slowly to indicate an error condition such as an improperly configured module. LED is off for normal operation.
SEL (Selected)	LED flashes when analog or digital address is selected by the host computer.
WD (Watchdog)	LED is on if watchdog timer is tripped. LED is off for normal operation. Disabling the analog or digital watchdog does not disable the watchdog LED.
RUN	LED is on whenever power is connected to the board.

SNAP-B6 I/O Configuration

Setting Up Analog Modules

The disk that comes with the SNAP-B6 includes a utility called PamScan, a diagnostic tool for reading and writing to analog and digital I/O. PamScan can also be used to configure analog channel types. It reads channel types and lets you change types easily. For information on using PamScan, see its online help.

Follow these steps to set up analog modules:

1. Install configuration jumper 7 on the SNAP-B6 brain. The STS and RUN LEDs should be lit. (Ignore other LEDs.)
2. Watch for the STS LED to start flashing rapidly, indicating that the brain is in configuration mode.
3. Using the PamScan utility, choose the I/O Address/Bank that corresponds to the address jumpers on the SNAP-B6. Make sure the type of I/O is analog.
4. Scan the brain and read current channel types using the tables on this page. Values are decimal if you are using the DOS version of PamScan. If you are using the Win 32 version, you can choose whether to show decimal or hexadecimal values.
5. To change a channel type, write the appropriate value from the table on this page out to the channel. Channel types cannot be larger than 1 byte (values 0–255). You can configure the channels for both analog addresses in any order.
6. When you have finished, remove configuration jumper 7. The configuration is automatically saved, the STS LED stops flashing, and the SNAP-B6 goes through a normal power-up sequence.

Note: *The flash memory on the SNAP-B6 has programming life of approximately 100,000 write/erase cycles.*

* *For information on how temperature is reported, see page 140.*

Inputs

Dec	Hex	Channel Type
0	0	Generic Input Module (Bipolar)
1	1	Generic Input Module (Unipolar)
2	2	0–20 mA
3	3	4-20 mA
4	4	ICTD
5	5	Type J Thermocouple*
6	6	0-5 VDC
7	7	0-10 VDC
8	8	Type K Thermocouple*
9	9	-50-+50 mV
10	A	3-wire, 11-Ohm PT RTD*
11	B	-5-+ VDC
12	C	-10-+10 VDC
13	D	0-100 mV (±160 mA)
14-16	E-10	Unused
17	11	Type R Thermocouple*
18	12	Type T Thermocouple*
19	13	Type E Thermocouple*
20	14	Unused
21	15	Unused
22	16	0-1 VDC
23	17	Type S Thermocouple*
24	18	Type B Thermocouple*
25-29	19-1D	Unused
30	1E	Type N Thermocouple*
31	1F	Type G Thermocouple*
32	20	Type C Thermocouple*
33	21	Type D Thermocouple*
34-63	22-3F	Unused
64	40	-20-+20 mA
65	41	Unused
66	42	-150-+150 mV
67	43	-25-+25 mV
68	44	-75-+75 mV
69	45	AIRATE, 0-25,000 kHz
70-127	46-7F	Unused

Outputs (Bit 7 = 1 if Ouput)

Dec	Hex	Channel Type
128	80	Generic Output Module
129	81	Reserved
131	82	4–20 mA
132	83	0–5 VDC
133	84	0–10 VDC
134	85	-5–+5 VDC
135	86	-10–+10 VDC
136	88	0–20 mA
137–255	89–FF	Reserved Output Types

Note:

The default mode of SNAP analog modules with the SNAP B6 is the same as that of classic analog modules with the B6. Most SNAP analog input modules are capable of better resolution than that of the classic Pamux system. If you are designing a new system and would like to take advantage of the improved resolution of most SNAP analog inputs, then the brain must be configured for high-resolution mode.

As an example, the following code is what is executed behind a high-resolution button that has been added to the PCI version of Pamscan. When jumper 7 is installed and the button is pushed, the higher resolution mode is burned into the SNAP B6 brain’s flash memory.

```
f_PamuxErr%=PamuxDirectWrite(f_Ac28Handle&, 1, &H82) ‘request access
f_PamuxErr%=PamuxDirectWrite(f_Ac28Handle&, 1, &H7C) ‘select appropriate status register
f_PamuxErr%=PamuxDirectWrite(f_Ac28Handle&, 0, &H10) ‘write 10 hex to increase resolution
f_PamuxErr%=PamuxDirectWrite(f_Ac28Handle&, 1, &H82) ‘release access
f_PamuxErr%=PamuxDirectWrite(f_Ac28Handle&, 1, &H0)  ‘reinitialize
```

To return to standard Pamux resolution, a second button that executes the following code in the PCI pamscan utility will do the job. Install jumper 7 and push the button and the brain will return to standard resolution.

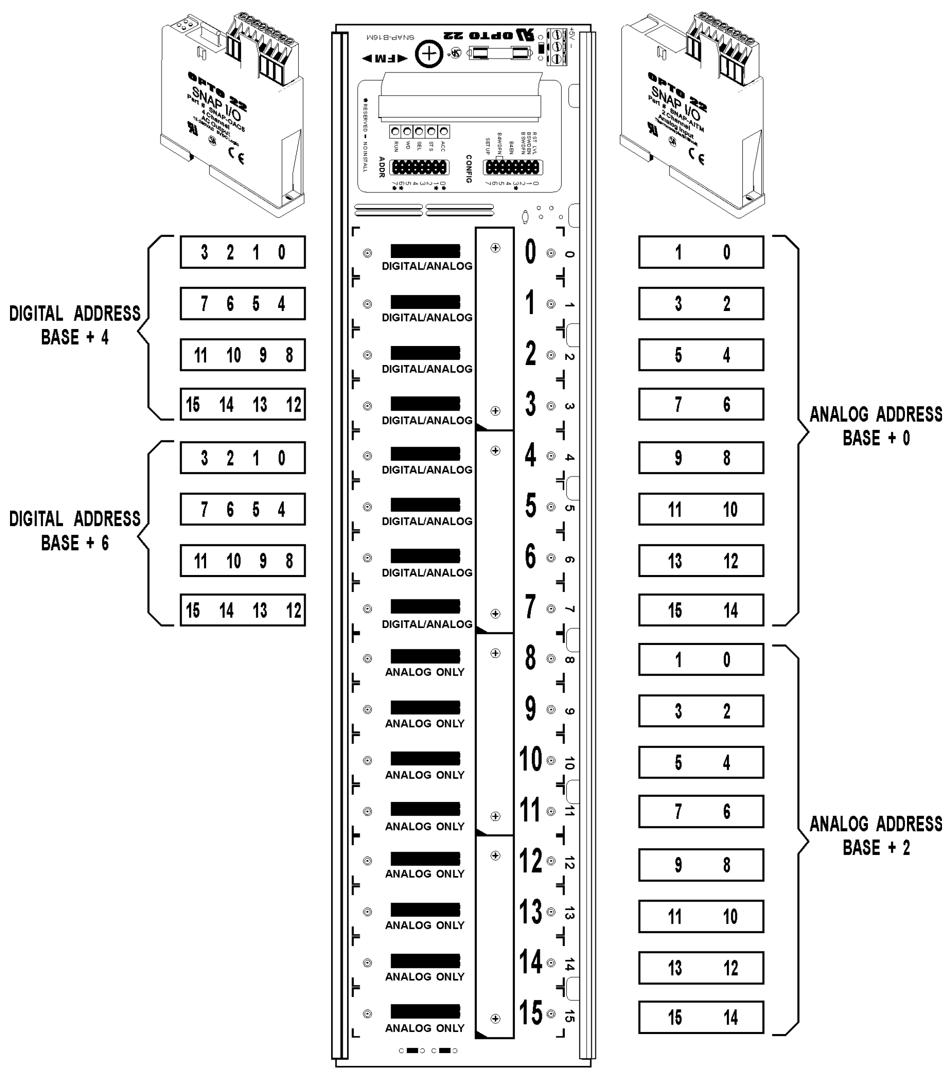
```
f_PamuxErr%=PamuxDirectWrite(f_Ac28Handle&, 1, &H82) ‘request access
f_PamuxErr%=PamuxDirectWrite(f_Ac28Handle&, 1, &H7C) ‘select appropriate status register
f_PamuxErr%=PamuxDirectWrite(f_Ac28Handle&, 0, &H0)  ‘write 0 hex to return to standard
                                                      resolution
f_PamuxErr%=PamuxDirectWrite(f_Ac28Handle&, 1, &H82) ‘release access
f_PamuxErr%=PamuxDirectWrite(f_Ac28Handle&, 1, &H0)  ‘reinitialize
```

SNAP-B6 I/O Mapping

The largest SNAP B Series I/O rack can contain a maximum of 16 modules. As shown below, the first eight modules can be either digital or analog. The last eight modules can be analog only. Because of the rack’s flexibility in handling both digital and analog inputs and outputs in many of the same module positions, you can choose where to install modules and how to use the points.

Since each digital module contains four points, up to 32 digital I/O points can be installed in the first eight module positions.

Analog input modules contain two points, but analog output modules can have either one or two points, depending on the module. Using all module positions, up to 32 analog I/O points can be installed in the rack.



CONNECTING AND POWERING A SYSTEM

Pamux stations are linked to each other and to a host computer or other device via a 50-pin flat-ribbon cable. Opto 22 provides a number of pre-made cables, and other vendors also offer compatible cables and connectors. Refer to Appendix B for a list of approved cables and connectors.

The minimum power requirements for each Pamux station are 5 VDC at 0.5 A. The final Pamux station on a bus requires an additional 1 A for the terminator board. In addition, B6 analog Pamux stations require +15 VDC and -15 VDC power supplies with current requirements that depend on the modules installed.

Make sure the 5V power supply is floating (not earth-grounded). Also, to benefit from optical isolation, make sure the 5V common is not tied to the ±15V common.

Appendix B details all power requirements for digital and analog Pamux stations along with a list of the current requirements for analog modules. Refer to this appendix for these requirements and for a list of vendors that provide compatible power supplies.

Note: Always check all power supply polarities before powering up any system.

CONNECTING TO AN ADAPTER CARD

A Pamux system must be programmed through a host computer or similar device connected to the Pamux bus via an interface card.

The most commonly used Pamux adapter card is the AC28, designed for IBM PC/AT systems. Up to four of these Opto 22 cards may be installed in a host computer, enabling control of up to four Pamux buses with a total of up to 2,048 points of I/O.

The PCI-AC51 is designed for PCI-compatible computers that feature a standard 33 MHz PCI bus. See form #1459, the *PCI-AC51 User's Guide*, for more information about using the card.

Other Opto 22 adapter cards inlude the AC36, used to connect a Pamux bus to a TTL parallel port, and the UCA4, a universal Pamux bus adapter card. These adapter cards are described and illustrated in Chapter 1.

Appendix B provides specifications on these and other Pamux components.

Setting AC28 Jumpers

The AC28 includes six jumpers. Jumpers J6, J7, J8, and J9 are used to set the AC28 base address. Jumpers R8 and R9 are used to set the reset port address. Tables 2-12 and 2-13 show the jumper configurations for these addresses.

Table 2-12: AC28 Base Address Jumpers

Base Address	Jumper J9	Jumper J8	Jumper J7	Jumper J6
100 hex	In	Out	In	In
140 hex	In	Out	In	Out
180 hex	In	Out	Out	In
280 hex	Out	In	Out	In

Table 2-13: AC28 Reset Port Address Jumpers

Reset Port Address	Jumper R9	Jumper R8
0E0 hex	In	In
1E0 hex	In	Out
2E0 hex	Out	In
3E0 hex	Out	Out

Note: The IBM PC can only use addresses 2E0 and 3E0.

AC28 Acknowledgment Data (Does not apply to PCI-AC51)

Pamux now includes a provision for identifying the last board communicated with via acknowledge lines. The acknowledgment data is read at the reset port address.

This feature works only with the following revisions of Pamux boards:

- AC28: Revision C or later
- B4: Revision L or later
- B5: Revision J or later
- B6: Revision G or later

Refer to Table 2-14 as a guide to using this feature.

Table 2-14: Communication Acknowledgment

Pamux Configuration	Data
Old AC28 with no connection or any board	FF hex
New AC28 with no connection or an old Pamux brain board	00 hex
New AC28 with new B4	01 hex
New AC28 with new B5	02 hex
New AC28 with new B6	03 hex

CONNECTING FIELD WIRING

This section provides detailed information on wiring digital, quad pak, and analog I/O modules to Pamux systems. It includes examples of how to wire all currently available Pamux-compatible modules. For a complete list of all modules that can be used with Pamux systems, see Appendix B.

Wiring Digital Modules

This section lists all Pamux-compatible digital modules and provides wiring diagrams for both input and output modules.

Each digital I/O channel has two terminals corresponding to each installed digital module. Terminals 1 and 2 correspond to a module in channel 0, terminals 3 and 4 correspond to a module in channel 1, and so on. For polarized modules, the positive connection goes to the first terminal of the pair and the negative connection goes to the second.

Input Modules

Use Figure 2-31 to wire the digital DC and AC input modules listed in Table 2-15. The diagram shows a DC input module wired to channel 0 and an AC input module wired to channel 1 on a G4PB16H rack.

For the digital input modules listed in Table 2-15, the input device may be wired to either terminal. The polarity of the power does not matter except for the G4IDC5K, G4IDC5D, and IDC5D.

Table 2-15: Digital DC and AC Input Modules

DC Input Modules	AC Input Modules
G4IAC5	G4IAC5
G4IAC5A	G4IAC5A
G4IAC5MA	G4IAC5MA
G4IDC5	G4IDC5
G4IDC5B	G4IDC5G
G4IDC5D	G4IDC5MA
G4IDC5G	IAC5
G4IDC5K	IAC5A
G4IDC5MA	IDC5
IDC5	IDC5G
IDC5B	SNAPIAC5
IDC5D	SNAPIAC5A
IDC5G	- - -
IAC5	- - -
IAC5A	- - -
SNAPIDC5	- - -

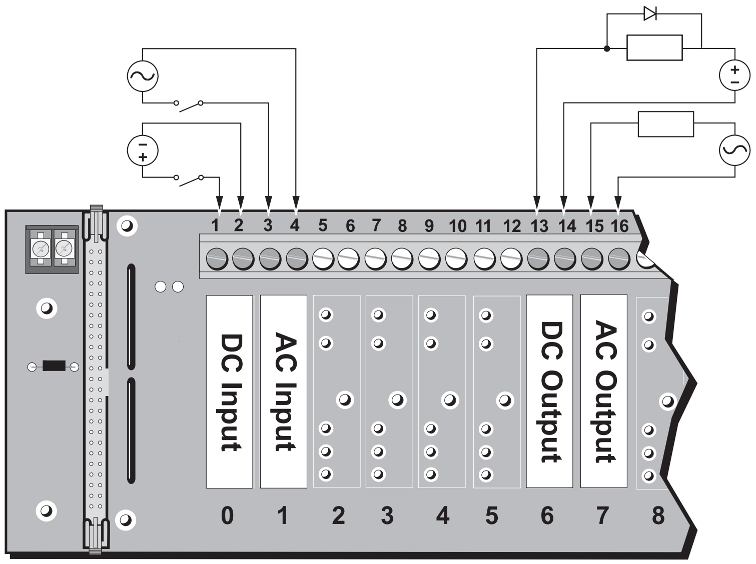


Figure 2-31: Wiring for DC and AC Input/Output Modules

Output Modules

Use Figure 2-31 on the previous page to wire the digital DC and AC output modules listed in Table 2-16. The diagram shows a DC output module wired to channel 6 and an AC output module wired to channel 7 on a G4PB16H rack.

For the digital output modules listed in Table 2-16, the load may be wired to either line. The polarity of the power does not matter except for the G4ODC5, G4ODC5A, G4ODC5MA, ODC5, ODC5A, SNAPODC5SRC, and SNAPODC5SNK.

For DC output modules used with inductive loads, add a commutating diode (typically a 1N4005) to the circuit as shown on the channel 6 connection to the G4PB16H rack.

Table 2-16: Digital DC and AC Output Modules

DC Output Modules	AC Output Modules
G4ODC5	G4OAC5
G4ODC5A	G4OAC5A
G4ODC5MA	G4OAC5A5
G4ODC5R	G4OAC5AMA
G4ODC5R5	G4OAC5MA
ODC5	G4ODC5R
ODC5A	G4ODC5R5
ODC5R	OAC5
SNAPODC5SRC,	OAC5A
SNAPODC5SNK	OAC5A5
- - -	SNAPOAC5

Wiring Quad Pak Modules

This section lists all quad pak modules and provides wiring diagrams for both input and output modules.

Each quad pak I/O module consists of two pairs of circuits. These circuit pairs share a common point.

A quad pak module connects to four numbered terminals and two common ("C") terminals, as follows:

- Terminal 1 and the upper C terminal correspond to a circuit in channel 0.
- Terminal 2 and the upper C terminal correspond to a circuit in channel 1.
- Terminal 3 and the lower C terminal correspond to a circuit in channel 2.
- Terminal 4 and the lower C terminal correspond to a circuit in channel 3.

For polarized modules, the positive connection goes to the C terminal and the negative connection goes to the numbered terminal.

Input Modules

Use Figure 2-32 on the following page, to wire the quad pak DC and AC input modules listed in Table 2-17. The diagram shows a DC input module wired to channels 16–19 and an AC input module wired to channels 0–3 on a PB32HQ rack.

For the quad pak input modules listed in Table 2-17, the input device may be wired to either terminal. The polarity of the power does not matter for any of the modules.

Table 2-17: Quad Pak DC and AC Input Modules

DC Input Modules	AC Input Modules
IDC5Q	IDC5Q
IDC5BQ	IAC5Q
IAC5Q	IAC5AQ
IAC5AQ	

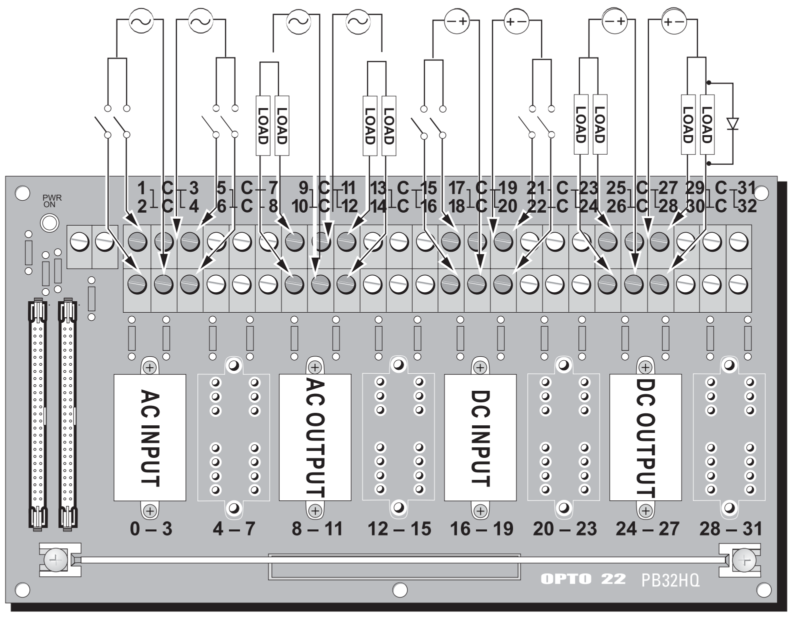


Figure 2-32: Wiring for Quad Pak Input and Output Modules

Output Modules

Use Figure 2-32 to wire the quad pak DC and AC output modules listed in Table 2-18. The diagram shows a DC output module wired to channels 24–27 and an AC output module wired to channels 8–11 on a PB32HQ rack.

For the quad pak output modules listed in Table 2-18, the load may be wired to either line.

Note that the 'C" terminal of DC output modules must be more positive when switching DC leads.

For DC output modules used with inductive loads, add a commutating diode (typically a 1N4005) to the circuit as shown on the channel 27 connection to the PB32HQ rack.

Table 2-18: Quad Pak DC and AC Output Modules

DC Output Modules	AC Output Modules
ODC5Q	OAC5Q
ODC5AQ	

Wiring Analog Modules

This section lists all Pamux-compatible analog modules and provides wiring diagrams for each module type.

Analog modules require up to four terminals per analog I/O channel. On the analog mounting rack, the lower A and B terminals are typically used to distribute a 24 VDC shared-loop source for use with 4–20 mA modules.

Note that two types of analog modules are available: those that provide transformer isolation and those that do not. Transformer-isolated modules include a “T” in the model number (e.g., AD3T, AD10T2) and provide channel-to-channel isolation. For a list of all analog modules, as well as their current requirements, see Appendix B.

IMPORTANT: Once connected to a rack, modules without transformer isolation create an electrical connection from the negative field terminal (which is connected to the upper A terminal on the rack) to the ±15 VDC common. Hence, these modules do not offer channel-to-channel isolation. If you require such isolation, select transformer-isolated modules only.

Voltage Input and Output Modules

Use Figure 2-33 on the following page, to wire all of the analog voltage input or output modules listed in Table 2-19 except the AD15T. The diagram shows a voltage output module wired to channel 0, and two voltage input modules wired to channels 2 and 4 on a PB16AH rack.

Note that for the AD9T and AD13T, there is no connection on the mounting rack (A and B terminals). Instead, the connections are made directly to the module.

To wire the AD15T, refer to Figure 2-36 on page 54.

Table 2-19: Voltage Input and Output Modules

Voltage Input Modules	Voltage Output Modules
AD6	DA4
AD6T	DA4T
AD6HS	DA5
AD7	DA6
AD9T	DA7
AD11	- - -
AD12	- - -
AD12T	- - -
AD13T	- - -
AD15T (AC only)	- - -

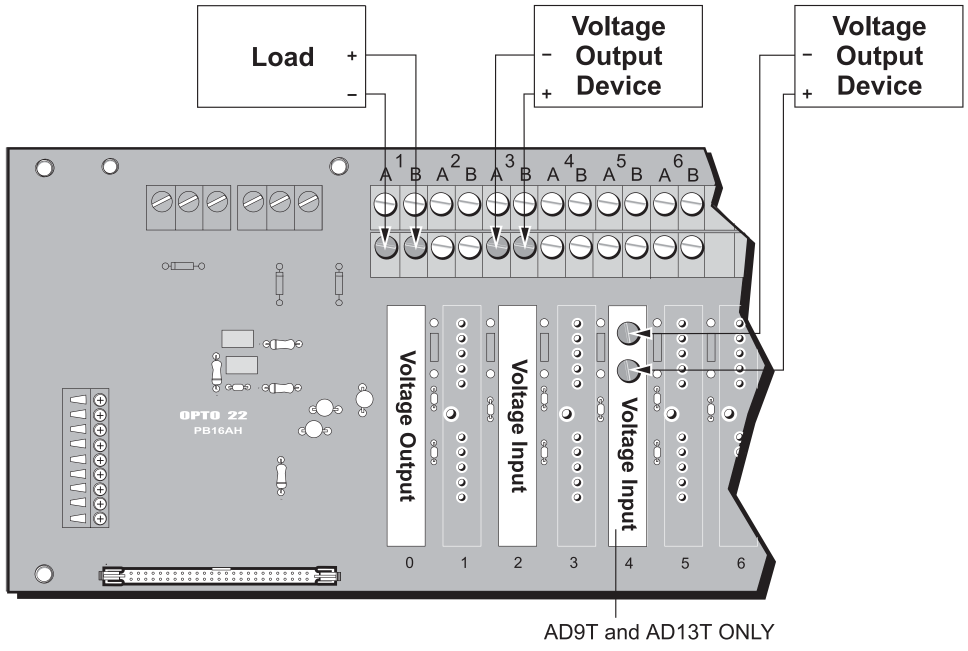


Figure 2-33: Wiring for Voltage Input and Output Modules

Milliamp Current Input and Output Modules

Use Figures 2-34 and 2-35 to wire the analog milliamp current input or output modules listed in Table 2-20. The examples show wiring to a PB16AH rack on channels 0, 1, and 3.

Figure 2-34 shows wiring with individual power supplies. Figure 2-35 shows wiring with a shared loop supply. In Figure 2-35, the wiring takes advantage of the fact that all lower A terminals on the mounting rack are tied together. These provide a convenient tie point for shared loop source return. Connect one lower A terminal to the shared loop source "–," then jumper each upper A to its respective lower A for each current module.

The current loop for an input or output current device must be powered by a user's external supply.

Table 2-20: Milliamp Current Input and Output

Milliamp Current Input Modules	Milliamp Current Output Modules
AD2T	DA3
AD3	DA3T
AD3T	DA8

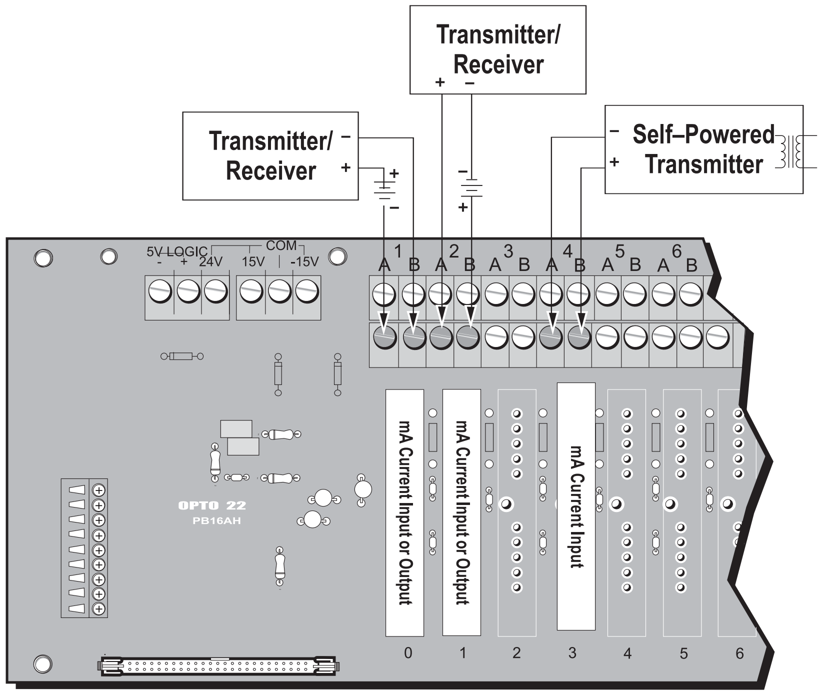


Figure 2-34: Wiring for Milliamp Current Input and Output Modules with Individual Power Supplies

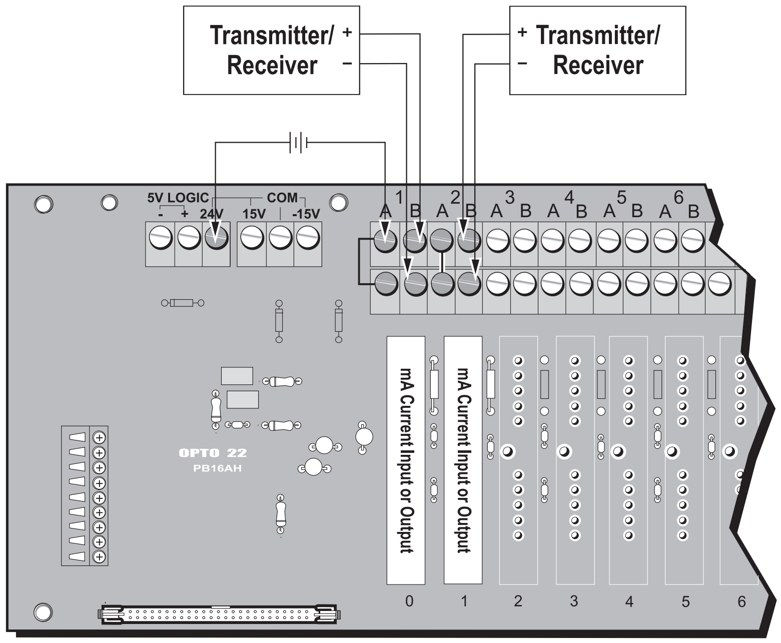


Figure 2-35: Wiring for Milliamp Current Input and Output Modules with Shared Power Supplies

0 to 5 Amp AC/DC Current Input Modules

Use Figure 2-36 to wire the 0–5A AC/DC current input module listed in Table 2-21. The example shows wiring of the AD16 to channel 0 on a PB16AH rack.

The 0–5A AC/DC current input module can be used to measure current directly or indirectly by using a standard current transformer. Applications include measuring or monitoring current through a field device such as a motor, solenoid, or lamp.

Table 2-21: 0 to 5 Amp AC/DC Current Input Module

0 to 5 Amp AC/DC Current Input Module
AD16T

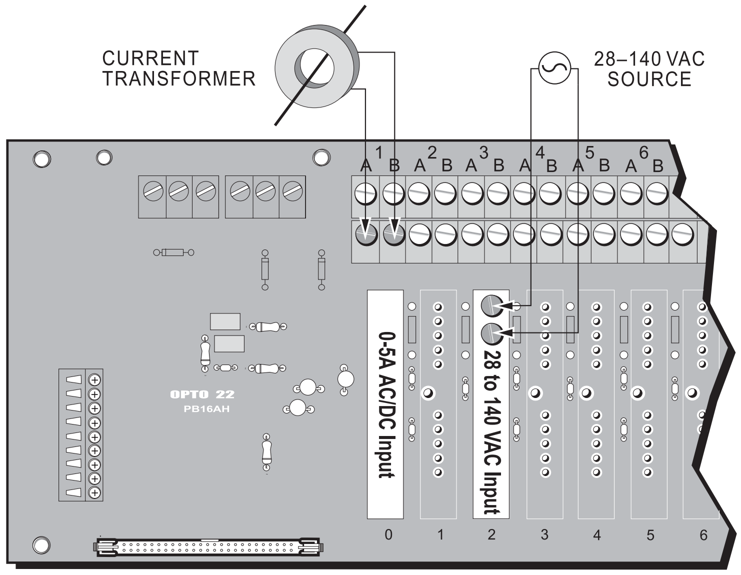


Figure 2-36: Wiring for 0–5A Current Input and 28–140 VAC Input Modules

Thermocouple Input Modules

Use Figure 2-37 to wire the analog thermocouple input modules listed in Table 2-22. The example shows a thermocouple input module wired to channel 3 on a PB16AH rack.

When wiring thermocouples, verify that you are using the proper polarity and wire color (see Table 2-22). Also, ensure that the wire type from the thermocouple to field terminals is consistent and does not introduce other thermocouples.

Thermocouple modules do not include a connection to the A and B terminals. Instead, the thermocouple wires are attached to screws located directly on the module.

Some thermocouple modules require a linearization algorithm to convert readings to temperatures. See Appendix C for details.

Table 2-22: Thermocouple Input Data

Modules	T/C Type	Polarity/Color	
		+	-
AD5, AD5T	J	White	Red
AD8, AD8T	K	Yellow	Red
AD17T	R	Black	Red
AD18T	T	Blue	Red
AD19T	E	Purple	Red
AD17T	S	Black	Red

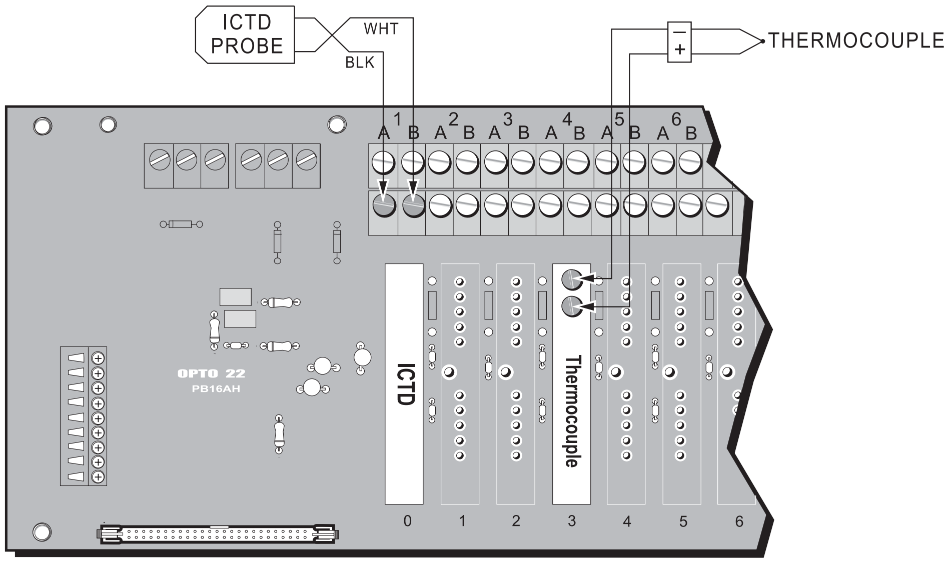


Figure 2-37: Wiring for Thermocouple and ICTD Temperature Input Modules

ICTD Temperature Input Modules

Use Figure 2-37 to wire the analog ICTD temperature input module listed in Table 2-23 to an Opto 22 ICTD probe. The example shows wiring to channel 0 on a PB16AH rack.

Note: That, like the thermocouple modules, ICTD modules also require a linearization algorithm to convert readings to temperatures. See Appendix C for details.

Table 2-23: ICTD Temperature Input Module

ICTD Temperature Input Module
AD4

RTD Input Modules

Use Figure 2-38 to wire the analog RTD input modules listed in Table 2-24. Wire colors may vary, but make sure two wires of the same color are connected as shown. The example shows a three-wire RTD probe connected to channel 0 on a PB16AH rack.

When using an AD10T2 with a four-wire RTD probe, do not connect the fourth wire. Connect three wires as shown for the three-wire RTD example.

For a two-wire RTD probe, add a second wire of the same type and gauge to one end, connecting it as you would a three-wire RTD.

Some RTD modules require a linearization algorithm to convert readings to temperatures. See Appendix C for details.

Table 2-24: RTD Input Modules

RTD Input Modules
AD10T2
AD14T

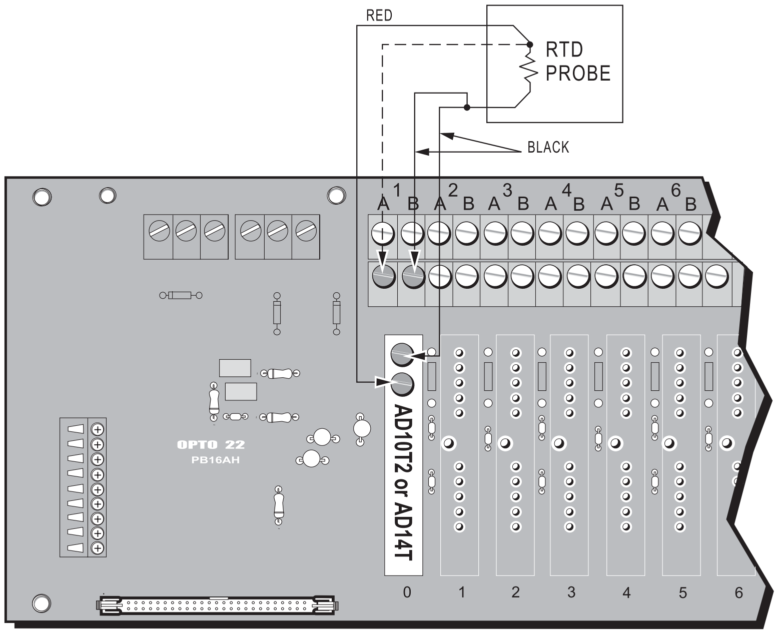


Figure 2-38: Wiring for RTD Input Modules

Rate Input Modules

Use Figure 2-39 to wire the analog rate modules listed in Table 2-25. The example shows the possible wiring options to channels 0 and 3 on a PB16AH rack. Note that the internal pull-up is between the + input post and the upper B terminal. Also, the – input and the upper A barrier terminal are connected directly to the analog common when the module is plugged into the analog rack.

Rate modules measure the frequency of an incoming signal and produce a count based on the number of cycles per second (Hertz). For example, a count value of 1,000 indicates a frequency of 1,000 Hz. This module is ideal for directly reading the frequency of a signal or a rotating disk for RPM calculations, for example.

Table 2-25: Rate Input Modules

Rate Input Modules
AD20

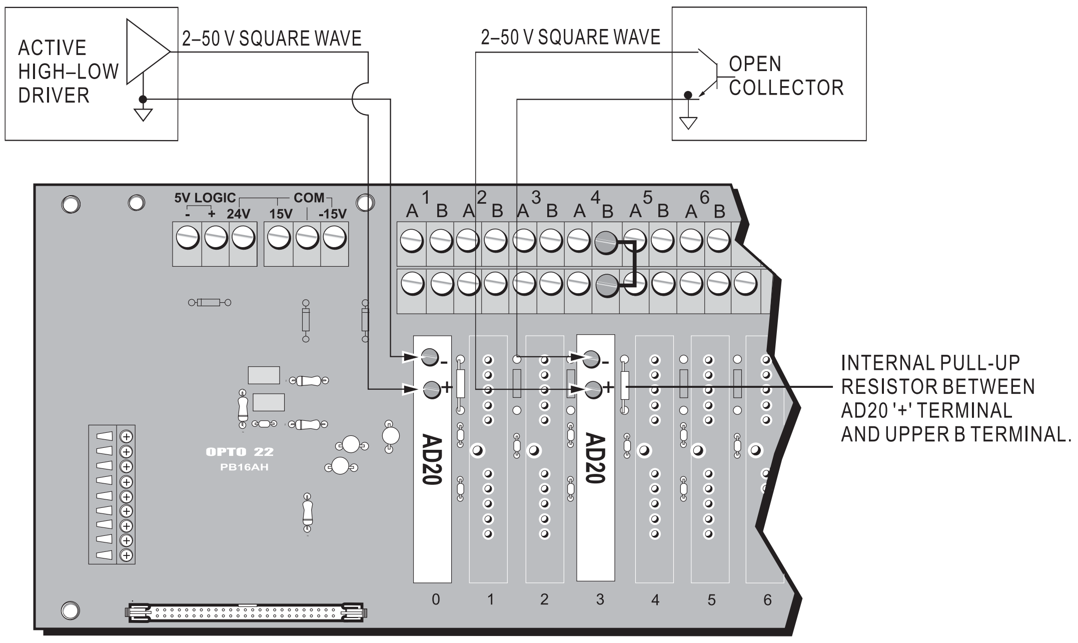


Figure 2-39: Wiring for Rate Input Modules

PROGRAMMING WITH THE PAMUX DRIVER

OVERVIEW

Many Pamux systems feature an AC28 adapter card installed in a personal computer with an ISA bus. If you are using an AC28, you can use Opto 22's Pamux driver to simplify communication to the Pamux bus. This chapter explains how to use the DOS and Windows versions of this driver with the AC28.

If you are using a PCI-AC51 adapter card installed in a PCI-compatible computer, you can use the PCI-AC51 Toolkit. For complete instructions, see Opto 22 form #1459, the *PCI-AC51 User's Guide*.

If you are not using an AC28 or PCI-AC28, or if you do not wish to use the Pamux software drivers, skip this chapter and go to Chapter 4, "Programming without the Pamux Driver."

WHAT IS THE PAMUX DRIVER?

The Pamux analog/digital driver is a subroutine that provides an interface between Pamux stations and application programs written in high-level languages, such as BASIC, C, C++, and Pascal. The driver allows the programmer to talk to Pamux by calling a single subroutine. This reduces the task of communicating to all analog and digital I/O down to understanding how to call the driver subroutine.

The software driver diskette includes DOS and Windows versions of the Pamux driver. The DOS version is called PDRIVER.OBJ or PDRIVER.COM (choose the file type you need). The Windows version is the dynamically linked library called Pamux.DLL.

The Pamux driver performs the following functions:

- Converts the data returned by Pamux to a form that is easily manipulated in a high-level language.
- Carries out all necessary handshaking with the Pamux bus.
- Transparently handles input masking on write operations for digital stations.
- Performs error checking and returns diagnostic codes. This saves you time and effort that would be spent becoming familiar with the intricacies of the Pamux bus structure.

To use the driver in your application program, you need to know the following:

- How to call a subroutine from the language you chose for your application
- How to tell the driver what Pamux command to send by assigning values to parameters
- How to interpret the data passed back by the driver

This chapter explains everything you need to know to use the driver in a DOS or Windows environment. If you are using Windows, you can skip the following section and proceed directly to page 88.

USING THE PAMUX DRIVER UNDER DOS

Installation

The Pamux driver is provided on one disk. You should make a backup copy of this disk before installing the driver. (For instructions on copying disks, refer to your DOS manual.)

The Pamux driver disk includes DOS and WIN directories. To use the driver under DOS, you will need to copy the contents of the DOS directory to an appropriate directory on your hard drive.

For complete installation instructions, view the README.TXT file on the driver disk.

Pamux Driver Parameters

Every call to the Pamux driver must include five parameters. These parameters provide the driver with all information needed to fully specify the command to be sent. Data and error codes are also passed back to the application program through these parameters.

Below is an example of a call to the driver in a Microsoft Interpreted BASIC program:

```
CALL Pamux(ERRCOD%, ADDRESS%, COMMAND%, POSITION%, VALUE%(0))
```

All parameters passed to the driver must be 16-bit integers. Notice that all variable names in the command above end with a percent sign (%). In BASIC, the % indicates that a variable is a 16-bit integer.

The five Pamux parameters are:

1. **ERRCOD** — An integer variable that contains an error code. A value of zero returned in this parameter indicates that no errors occurred and the command was successfully executed. A value less than zero indicates that the driver detected an error and the command was not executed. The value indicates the type of error encountered. For example, -1 represents "Configuration Required," indicating that a Pamux analog board has lost its configuration due to a reset or a power loss.
2. **ADDRESS** — An integer variable that contains the address of the intended Pamux unit (0–63 decimal).
3. **COMMAND** — An integer variable that contains the number of the desired command.
4. **POSITION** — An integer variable that contains either the module position or bank number to be affected by the command. A bank is a group of eight positions. Each bank takes one address position.
5. **VALUE ARRAY** — An eight-element integer array used to hold data values that help describe the command's execution. Also used to hold all returned data.

Using the Pamux Driver with Interpreted BASIC or QBasic

To call the Pamux driver subroutine from a program written in Interpreted BASIC or QBasic, the program must load the driver, name it, declare and initialize variables, and call the driver. These steps are described below.

Load the Pamux Driver

To use the driver in your application program, you must first load the subroutine into memory. This is done with two commands: DEF SEG and BLOAD.

DEF SEG is used to tell BASIC where to put the driver. The driver should be loaded above BASIC. This location depends on the size of DOS and the BASIC interpreter. A typical value for a system with 640 KB of memory is 3600 hex. The following statement can be used in most cases without modifications:

```
DEF SEG = &H3600
```

BLOAD is used to tell the BASIC interpreter to load the driver subroutine from disk to memory. This statement must follow the DEF SEG statement, since BASIC must know where to put the driver before it can load it. The following BLOAD statement instructs BASIC to load the driver at the currently defined segment:

```
BLOAD “PDRIVER.COM”, 0
```

The 0 in the statement above tells BASIC to start loading at an offset of 0 from the currently defined segment.

The file containing the version of the driver to be loaded under Interpreted BASIC or QBasic is PDRIVER.COM.

Name the Pamux Driver

The CALL statement calls the driver subroutine from a BASIC program. The CALL statement must include the name of a variable whose value indicates where the subroutine is located in relation to the currently defined segment. If you have loaded the driver as described previously, the variable must be equal to 0. The name Pamux must be used when using the BASIC compiler. Hence, the following simple assignment statement should appear in your program before the driver is called:

```
Pamux = 0
```

Declare and Initialize the Parameters

All variables passed to the driver subroutine must be declared and initialized before the driver can be called. The VALUE% array parameter must be declared using the DIM statement. Each element should then be set to a default value. The following program segment is an example:

```
200   DIM VALUE%(7)                      ‘ Dimension the array
210   FOR I% = 0 TO 7                    ‘ Fill all array elements with 0
220   VALUE%(I%) = 0
230   NEXT
240   ERRCOD% = 0                        ‘ Initialize all parameters
250   ADDRESS% = 0
260   COMMAND% = 0
270   POSITION% = 0
```

Call the Pamux Driver

The driver subroutine can be called from any place in your program any number of times. The CALL statement is used to call machine language subroutines. The following is a sample statement that calls the driver from Interpreted BASIC:

```
CALL  Pamux  (ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
```

The equivalent call in QBasic is slightly different:

```
CALL  ABSOLUTE  (ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0),0)
```

Note: that the VALUE% array is passed by including the first element of the array in the CALL statement.

The variables included in the CALL statement must be included in the statement in the order indicated above. The actual name of the variables can be different as long as the variables are the correct type (16-bit integers) and size. For example, the fifth parameter must be an eight-element integer array.

HINT: Placing the CALL statement within a subroutine in your program produces much more readable code and reduces the chance for errors.

Pamux Error Codes

Table 3-1 describes the errors returned by the Pamux driver under DOS:

Table 3-1: Pamux Driver Error Codes under DOS

Error	Description
-1	Configuration required (due to reset or loss of power)
-2	Invalid address (address > 63)
-3	Invalid bank (address + bank > 63)
-4	Invalid point (point + 8 * address > 511)
-5	Invalid command
-6	Invalid range
-7	Turnaround time-out (analog board did not respond)
-8	Analog watchdog time-out

Working with Banks

The concepts of Addresses and Banks can be a little confusing when dealing with a Pamux I/O system.

Every group of 8 channels in a Pamux system (a Bank) is related to an Address and a Bank Number. Every Pamux system is composed of 64 Banks (8 channels each).

A 16-channel Pamux brain board (B5 or B6) has two banks. Bank 0 refers to channels 0 to 7 and Bank 1 refers to channels 8 through 15.

The B4 Pamux brain board addresses 4 Banks.

Bank Number	Channels
0	0 to 7
1	8 to 15
2	16 to 23
3	24 to 31

When using the driver, Address% is set to the base address of the brain board, then the group of 8 channels is selected by setting the Bank (Position%) to 0, 1, 2, or 3.

One last bit of confusion…the Pamux driver allows an alternate addressing scheme. It is possible to leave Address always set to (0). The Bank of 8 channels is then selected by setting Bank to a value from 0 through 63! This technique may be easier in a system with a mixture of B4, B5, and B6 brain boards.

COMMAND REFERENCE

This section describes all analog and digital Pamux commands. These commands are passed to the driver by setting the COMMAND parameter to the correct command number, as defined below.

#	Command	#	Command
0	Set Base Address	12	Write Digital Point
1	Set Reset Address	13	Configure Analog Bank
2	Set Reset Level	14	Configure Analog Bank, Expanded
3	Reset	15	Configure Analog Point
4	Configure Digital Bank	16	Read Analog Bank
5	Configure Digital Bank, Expanded	17	Read Analog Point
6	Configure Digital Point	18	Write Analog Bank
7	Read Digital Bank	19	Write Analog Point
8	Read Digital Bank, Expanded	20	Set Analog Watchdog Timeout
9	Read Digital Point	21	Write Analog Watchdog Bank
10	Write Digital Bank	22	Write Analog Watchdog Point
11	Write Digital Bank, Expanded		

The format of each command description in this section is as follows:

COMMAND NAME

PURPOSE:

What the command does.

COMMAND TYPE:

"Driver" indicates the command affects only the driver configuration. "Digital," "Analog," and "Digital and analog" indicate the types of Pamux brain boards that will accept the command.

PARAMETERS:

Which of the five parameters passed to the driver are relevant to the command. While all five parameters must be passed to the driver each time it is called, only some of these parameters are required to fully specify the command and to store returned data.

REMARKS:

What the command does and how it should be used.

EXAMPLE:

Actual BASIC program segment that demonstrates the use of the command. Examples are in Microsoft's Interpreted BASIC (not QBasic) as implemented on a PC.

SET BASE ADDRESS 0

PURPOSE:

Configures the driver with the base address of the AC28 to use when sending future commands.

COMMAND TYPE:

Driver

PARAMETERS:

COMMAND	Contains the value (zero).
VALUE ARRAY	The first element of this array contains a single value that selects the address. Valid values are:

VALUE%(0)	Base Address
0	100 hex (default base address)
1	140 hex
2	180 hex
3	280 hex
8	2100 hex
9	2180 hex
10	2200 hex
11	2280 hex

REMARKS:

To properly initialize the driver with the correct address of the AC28 card, send this command before any other command.

The driver assumes a default AC28 base address of 100 hex. If a value greater than 12 is specified, the driver returns a -6 in the ERRCOD% variable.

This command needs to only be sent once during initialization.

This command has no effect when using the driver resident in the LC4.

EXAMPLE:

This example tells the driver to use 180 hex as the AC28 base address:

```
100   COMMAND% = 0              ‘ Set base address command
110   VALUE%(0) = 2             ‘ Value of 2 = 180 hex
120   GOSUB 1000                ‘ Call the driver
      .
      .
1000  CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010  IF ERRCOD% < 0 THEN GOTO 2000
1020  RETURN
```

SET RESET ADDRESS 1

PURPOSE:

Configures the driver with the reset address of the AC28 to use when sending a reset command.

COMMAND TYPE:

Driver

PARAMETERS:

COMMAND	Contains the value 1.
VALUE ARRAY	The first element of this array contains a single value that selects the reset address. Valid values are:

VALUE%(0)	Reset Address
0	0E0 hex (default reset address)
1	1E0 hex
2	2E0 hex
3	3E0 hex

REMARKS:

To properly initialize the driver with the correct reset address of the AC28 card, send this command before sending a reset command.

The driver assumes a default AC28 reset address of 0E0 hex. If a value greater than 12 is specified, the driver returns a -6 in the ERRCOD% variable.

This command need only be sent once during initialization.

This command has no effect when using the driver resident in the LC4.

EXAMPLE:

This example tells the driver to use 1E0 hex as the AC28 reset address.

```
100    COMMAND% = 1                ‘ Set reset address command
110    VALUE%(0) = 1               ‘ Value of 1 = 1E0 hex
120    GOSUB 1000                  ‘ Call the driver
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

SET RESET LEVEL 2

PURPOSE:

Configures the driver with the reset level to use when sending a reset command to Pamux.

COMMAND TYPE:

Driver

PARAMETERS:

COMMAND	Contains the value 2.
VALUE ARRAY	The first element of this array contains a single value that selects the reset level. Valid values are:
VALUE%(0)	Reset Level

0	Active Low
1	Active High (default)

REMARKS:

This command must be sent before sending a Reset command.

For proper operation of the Reset command, set the reset level to match the jumper setting of each Pamux brain board. All Pamux brain boards connected to the same AC28 must be configured to the same reset level.

The driver assumes a default active high level.

This command need only be sent once during initialization.

EXAMPLE:

This example tells the driver to use an active low value when a Reset command is sent.

```
100    COMMAND% = 2               ‘ Set Reset Level command
110    VALUE%(0) = 0              ‘ Value of 0 = active low
120    GOSUB 1000                 ‘ Call the driver
       .
       .
1000   CALL 
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

RESET 3

PURPOSE:

Resets all Pamux brain boards.

COMMAND TYPE:

Digital and analog

PARAMETERS:

COMMAND	Contains the value 3.

REMARKS:

For proper operation of this command, the Set Reset Address and Set Reset Level commands must already have been sent to set the proper AC28 reset address and level.

EXAMPLE:

This example resets all Pamux units.

```
100    COMMAND% = 3                ‘ Reset command
110    GOSUB 1000                  ‘ Call the driver
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

CONFIGURE DIGITAL BANK 4

PURPOSE:

Configures a bank of eight digital I/O points.

COMMAND TYPE:

Digital

PARAMETERS:

COMMAND	Contains the value 4.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the bank number.
VALUE ARRAY	The first element of this array contains a one-byte bitmask (0–255) specifying the configuration of the bank. Each bit corresponds to one module position. If a bit is set to 1, the corresponding module position is configured as an output. If a bit is set to 0, the corresponding module position is configured as an input.

REMARKS:

All output module positions must first be configured as outputs before they can be activated. The driver will ignore any write commands to positions configured as inputs and no errors will be returned.

EXAMPLE:

This example configures bank 1 at address 4 with the first four positions as inputs and the second four positions as outputs. Remember, bank 1 represents modules at positions 8 through 15, so in this example bit 0 of the bitmask corresponds to module position 8 and bit 7 of the bitmask corresponds to module position 15.

```
BIT POSITION  7 6 5 4 3 2 1 0
VALUE%(0)     1 1 1 1 0 0 0 0 = F0 hex or 240 decimal
```

```
100   COMMAND% = 4             ‘ Configure Digital Bank command
110   ADDRESS% = 4             ‘ Address of brain board
120   POSITION% = 1            ‘ Bank number
130   VALUE%(0) = &HF0         ‘ High nibble = outputs,
                                low nibble = inputs
140   GOSUB 1000               ‘ Call the driver
      .
      .
1000  CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010  IF ERRCOD% < 0 THEN GOTO 2000
1020  RETURN
```

CONFIGURE DIGITAL BANK, EXPANDED 5

PURPOSE:

Configures a bank of eight digital I/O points.

COMMAND TYPE:

Digital

PARAMETERS:

COMMAND	Contains the value 5.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the bank number.
VALUE ARRAY	Each element of this array corresponds to a module position within the bank. Element 0 corresponds to the first module position, element 7 corresponds to the last. If an element is set to 1, the corresponding module position is configured as an output. If an element is set to 0, the corresponding module position is configured as an input.

REMARKS:

All output module positions must first be configured as outputs before they can be activated. The driver will ignore any write commands to positions configured as inputs and no errors will be returned.

EXAMPLE:

This example configures bank 0 at address 5 with module positions 0, 2, 4, and 6 configured as inputs and positions 1, 3, 5, and 7 configured as outputs.

```
100    COMMAND% = 5             ‘ Configure Digital Bank, Expanded
110    ADDRESS% = 5             ‘ Address of brain board
120    POSITION% = 0            ‘ Bank number 0
130    VALUE%(0) = 0            ‘ Input module 0
140    VALUE%(1) = 1            ‘ Output module 1
150    VALUE%(2) = 0            ‘ Input module 2
160    VALUE%(3) = 1            ‘ Output module 3
170    VALUE%(4) = 0            ‘ Input module 4
180    VALUE%(5) = 1            ‘ Output module 5
190    VALUE%(6) = 0            ‘ Input module 6
200    VALUE%(7) = 1            ‘ Output module 7
210    GOSUB 1000               ‘ Call the driver
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

CONFIGURE DIGITAL POINT 6

PURPOSE:

Configures one digital point as an input or output.

COMMAND TYPE:

Digital

PARAMETERS:

COMMAND	Contains the value 6.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the point number. Legal values are 0 to 15 for B5 and B6, 0 to 31 for B4.
VALUE ARRAY	The first element of this array contains a 0 if the point is to be configured as an input, or anything else if the point is to be configured as an output.

REMARKS:

All output module positions must first be configured as outputs before they can be activated. The driver will ignore any write commands to positions configured as inputs and no errors will be returned.

The POSITION value is a point offset starting from the value in the ADDRESS parameter. The point number in the POSITION parameter can range from 0 to 511 if the ADDRESS parameter is 0. An error will be returned if the values in these parameters exceed the limits.

EXAMPLE:

This example configures point 0 at address 1 as an output.

```
100    COMMAND% = 6              ‘ Configure Digital Point command
110    ADDRESS% = 1              ‘ Address of brain board
120    POSITION% = 0             ‘ Point number 0
130    VALUE%(0) = 1             ‘ Configure as an output
140    GOSUB 1000                ‘ Call the driver
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

READ DIGITAL BANK 7

PURPOSE:

Reads a bank of eight digital I/O points.

COMMAND TYPE:

Digital

PARAMETERS:

COMMAND	Contains the value 7.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the bank number.
VALUE ARRAY	The first element of this array contains returned data in the form of a one-byte bitmask (0–255). This bitmask specifies the status of the bank. Each bit corresponds to one module position. If a bit is set to 1, the corresponding module position is active (on). If a bit is set to 0, the corresponding module position is inactive (off).

EXAMPLE:

This example reads back eight points at address 4 of bank 1 and then displays the resultant value bitmask.

```
100    COMMAND% = 7                ‘ Read Digital Bank command
110    ADDRESS% = 4                ‘ Address of brain board
120    POSITION% = 1               ‘ Bank number
130    GOSUB 1000                  ‘ Call the driver
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

READ DIGITAL BANK, EXPANDED 8

PURPOSE:

Reads a bank of eight digital I/O points.

COMMAND TYPE:

Digital

PARAMETERS:

COMMAND	Contains the value 8.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the bank number.
VALUE ARRAY	Each element of this array contains returned data corresponding to a module position within the bank. Element 0 corresponds to the first module position, element 7 corresponds to the last. If an element is set to -1, the corresponding module position is active (on). If an element is set to 0, the corresponding module position is inactive (off).

EXAMPLE:

This example reads the status of eight points at address 32 at bank 0 and then displays the status of each point.

```
100   COMMAND% = 8              ‘ Read Digital Bank, Expanded
110   ADDRESS% = 32             ‘ Address of brain board
120   POSITION% = 0             ‘ Bank number
130   GOSUB 1000                ‘ Call the driver
140   FOR I% = 0 TO 7
150   IF VALUE%(I%) = 0 THEN STAT$ = "OFF"
      ELSE STAT$ = " ON"        ‘ Update STAT$
160   PRINT "MODULE "; I%; " :"; STAT$
170   NEXT
      .
      .
1000  CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010  IF ERRCOD% < 0 THEN GOTO 2000
1020  RETURN
```

READ DIGITAL POINT 9

PURPOSE:

Reads a digital point.

COMMAND TYPE:

Digital

PARAMETERS:

COMMAND	Contains the value 9.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the point number.
VALUE ARRAY	The first element of this array contains returned data. The value will be -1 if the point is active (on), or 0 if the point is inactive (off).

REMARKS:

The POSITION value is a point offset starting from the value in the ADDRESS parameter. The point number in the POSITION parameter can range from 0 to 511 if the ADDRESS parameter is 0. An error will be returned if the values in these parameters exceed the limits.

EXAMPLE:

This example reads the state of the module at position 123 and displays the status.

```
100    COMMAND% = 9               ‘ Read Digital Point command
110    ADDRESS% = 0               ‘ Address of brain board
120    POSITION% = 123            ‘ Position 123
130    GOSUB 1000                 ‘ Call the driver
140    IF VALUE%(0) = 0 THEN PRINT "Module Is Off"
       ELSE PRINT "Module Is On"
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

Note: *This is an example of an alternate numbering scheme for I10 points. Both ADDRESS% and Bank (if used) are set to zero and I/O points are numbered linearly from 0 to 511.*

WRITE DIGITAL BANK 10

PURPOSE:

Writes to a bank of eight digital I/O points.

COMMAND TYPE:

Digital

PARAMETERS:

COMMAND	Contains the value 10.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the bank number.
VALUE ARRAY	The first element of this array contains a one-byte bitmask (0–255) specifying how the bank will be written to. Each bit corresponds to one module position. If a bit is set to 1, the corresponding module position will be activated (turned on). If a bit is set to 0, its corresponding module position will be deactivated (turned off).

REMARKS:

Only I/O points configured as outputs will be affected.

EXAMPLE:

This example turns on all eight points of bank 1 at address 3. The example assumes that these points have been previously configured as outputs.

```
100    COMMAND% = 10              ‘ Write Digital Bank command
110    ADDRESS% = 3               ‘ Address of brain board
120    POSITION% = 1              ‘ Bank number
130    VALUE%(0) = 255            ‘ Turn all points on
140    GOSUB 1000                 ‘ Call the driver
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

WRITE DIGITAL BANK, EXPANDED 11

PURPOSE:

Writes to a bank of eight digital I/O points.

COMMAND TYPE:

Digital

PARAMETERS:

COMMAND	Contains the value 11.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the bank number.
VALUE ARRAY	Contains the values to be written to each position in the specified bank. Each element of the array corresponds to one module position. Element 0 corresponds to the first module position, element 7 corresponds to the last. If an element is set to any number other than 0, the corresponding module position will be activated (turned on). If an element is set to 0, the corresponding module position will be deactivated (turned off).

REMARKS:

Only I/O points configured as outputs will be affected.

EXAMPLE:

This example activates modules on bank 0 at address 5, positions 0, 2, 4, and 6, and deactivates positions 1, 3, 5, and 7 at the same address. The example assumes that these positions have been previously configured as outputs.

```
100    COMMAND% = 11            ‘ Write Digital Bank, Expanded
110    ADDRESS% = 5             ‘ Address of brain board
120    POSITION% = 0            ‘ Bank number
130    VALUE%(0) = 1            ‘ Turn on module 0
140    VALUE%(1) = 0            ‘ Turn off module 1
150    VALUE%(2) = 1            ‘ Turn on module 2
160    VALUE%(3) = 0            ‘ Turn off module 3
170    VALUE%(4) = 1            ‘ Turn on module 4
180    VALUE%(5) = 0            ‘ Turn off module 5
190    VALUE%(6) = 1            ‘ Turn on module 6
200    VALUE%(7) = 0            ‘ Turn off module 7
210    GOSUB 1000               ‘ Call the driver
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

WRITE DIGITAL POINT 12

PURPOSE:

Writes to a digital point.

COMMAND TYPE:

Digital

PARAMETERS:

COMMAND	Contains the value 12.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the point number.
VALUE ARRAY	The first element of this array contains any value other than 0 if the point will be activated (turned on), or a 0 if the point will be deactivated (turned off).

REMARKS:

The position to be written to must have been previously configured as an output, otherwise this command will have no effect.

The POSITION value is a point offset starting from the value in the ADDRESS parameter. The point number in the POSITION parameter can range from 0 to 511 if the ADDRESS parameter is 0. An error will be returned if the values in these parameters exceed the limits.

EXAMPLE:

This example deactivates position 2 at address 32. The example assumes that this position has been previously configured as an output.

```
100    COMMAND% = 12              ‘ Write Digital Point command
110    ADDRESS% = 32              ‘ Address of brain board
120    POSITION% = 2              ‘ Position 2
130    VALUE%(0) = 0              ‘ Turn module off
140    GOSUB 1000                 ‘ Call the driver
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

CONFIGURE ANALOG BANK 13

PURPOSE:

Configures a bank of eight analog I/O points.

COMMAND TYPE:

Analog

PARAMETERS:

COMMAND	Contains the value 13.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the bank number.
VALUE ARRAY	The first element of this array contains a one-byte bitmask (0–255) specifying the configuration of the bank. Each bit corresponds to one module position. If a bit is set to 1, the corresponding module position will be configured as an output. If a bit is set to 0, the corresponding module position will be configured as an input.

REMARKS:

All output module positions must first be configured as outputs before values can be written to them. The driver will ignore any write commands to positions configured as inputs and no errors will be returned.

EXAMPLE:

This example configures bank 1 at address 4 with the first four positions as inputs and the second four positions as outputs. Remember, bank 1 represents modules at positions 8 through 15, so in this example bit 0 of the bitmask corresponds to module position 8 and bit 7 of the bitmask corresponds to module position 15.

```
BIT POSITION    7 6 5 4 3 2 1 0
VALUE%(0)       1 1 1 1 0 0 0 0 = F0 hex or 240 decimal
```

```
100   COMMAND% = 13          ‘ Configure Analog Bank command
110   ADDRESS% = 4           ‘ Address of brain board
120   POSITION% = 1          ‘ Bank number
130   VALUE%(0) = &HF0       ‘ High nibble = outputs,
                              low nibble = inputs
140   GOSUB 1000             ‘ Call the driver
      .
      .
1000  CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010  IF ERRCOD% < 0 THEN GOTO 2000
1020  RETURN
```

CONFIGURE ANALOG BANK, EXPANDED 14

PURPOSE:

Configures a bank of eight analog I/O points.

COMMAND TYPE:

Analog

PARAMETERS:

COMMAND	Contains the value 14.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the bank number.
VALUE ARRAY	Each element of this array corresponds to a module position within the bank. Element 0 corresponds to the first module position, element 7 corresponds to the last. If an element is set to 0, the corresponding module position will be configured as an input. If an element is set to 1, the corresponding module position will be configured as an output.

REMARKS:

All output module positions must first be configured as outputs before values can be written to them. The driver will ignore any write commands to positions configured as inputs and no errors will be returned.

EXAMPLE:

This example configures bank 0 at address 5 with module positions 0, 2, 4, and 6 configured as inputs and positions 1, 3, 5, and 7 configured as outputs.

```
100   COMMAND% = 14            ‘ Configure Analog Bank, Expanded
110   ADDRESS% = 5             ‘ Address of brain board
120   POSITION% = 0            ‘ Bank number
130   VALUE%(0) = 0            ‘ Input module 0
140   VALUE%(1) = 1            ‘ Output module 1
150   VALUE%(2) = 0            ‘ Input module 2
160   VALUE%(3) = 1            ‘ Output module 3
170   VALUE%(4) = 0            ‘ Input module 4
180   VALUE%(5) = 1            ‘ Output module 5
190   VALUE%(6) = 0            ‘ Input module 6
200   VALUE%(7) = 1            ‘ Output module 7
210   GOSUB 1000               ‘ Call the driver
      .
      .
1000  CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010  IF ERRCOD% < 0 THEN GOTO 2000
1020  RETURN
```

CONFIGURE ANALOG POINT 15

PURPOSE:

Configures the position of an analog point as an input or output.

COMMAND TYPE:

Analog

PARAMETERS:

COMMAND	Contains the value 15.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the point number.
VALUE ARRAY	The first element of this array contains a 0 if the point will be configured as an input, or anything else if the point will be configured as an output.

REMARKS:

All output module positions must first be configured as outputs before values can be written to them. The driver will ignore any write commands to positions configured as inputs and no errors will be returned.

The POSITION value is a point offset starting from the value in the ADDRESS parameter. The point number in the POSITION parameter can range from 0 to 511 if the ADDRESS parameter is 0. An error will be returned if the values in these parameters exceed the limits.

EXAMPLE:

This example configures point 0 at address 1 as an output.

```
100    COMMAND% = 15              ‘ Configure Analog Point command
110    ADDRESS% = 1               ‘ Address of brain board
120    POSITION% = 0              ‘ Point number 0
130    VALUE%(0) = 1              ‘ Configure as an output
140    GOSUB 1000                 ‘ Call the driver
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

READ ANALOG BANK 16

PURPOSE:

Reads a bank of eight analog I/O points.

COMMAND TYPE:

Analog

PARAMETERS:

COMMAND	Contains the value 16.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the bank number.
VALUE ARRAY	Each element of this array contains returned data corresponding to a module position within the bank. Element 0 corresponds to the lowest module position, element 7 corresponds to the highest.

EXAMPLE:

This example reads the status of eight points at address 32 of bank 0, and then displays the value of each point.

```
100   COMMAND% = 16             ‘ Read Analog Bank command
110   ADDRESS% = 32             ‘ Address of brain board
120   POSITION% = 0             ‘ Bank number
130   GOSUB 1000                ‘ Call the driver
140   FOR I% = 0 TO 7
150   PRINT "MODULE ";I%;": ";VALUE%(I%)
160   NEXT
      .
      .
1000  CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010  IF ERRCOD% < 0 THEN GOTO 2000
1020  RETURN
```

READ ANALOG POINT 17

PURPOSE:

Reads an analog point.

COMMAND TYPE:

Analog

PARAMETERS:

COMMAND	Contains the value 17.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the point number.
VALUE ARRAY	The first element of this array contains returned data read from the point.

REMARKS:

The POSITION value is a point offset starting from the value in the ADDRESS parameter. The point number in the POSITION parameter can range from 0 to 511 if the ADDRESS parameter is 0. An error will be returned if the values in these parameters exceed the limits.

EXAMPLE:

This example reads the state of the module at position 8 and displays the status.

```
100    COMMAND% = 17             ‘ Read Analog Point command
110    ADDRESS% = 0              ‘ Address of brain board
120    POSITION% = 8             ‘ Position 8
130    GOSUB 1000                ‘ Call the driver
140    PRINT VALUE%(0)‘ Display the analog value
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

WRITE ANALOG BANK 18

PURPOSE:

Writes to a bank of eight analog I/O points.

COMMAND TYPE:

Analog

PARAMETERS:

COMMAND	Contains the value 18.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the bank number.
VALUE ARRAY	Contains the values to be written to each position in the bank. Each element of the array corresponds to a module position within the bank. Element 0 corresponds to the first module position, element 7 corresponds to the last.

REMARKS:

Only I/O points configured as outputs will be affected.

EXAMPLE:

This example writes the value 3,048 to module positions 4, 5, and 6 and writes the value 0 to all other module positions at address 6 of bank 1. The example assumes that these positions have been previously configured as outputs.

```
100   COMMAND% = 18           ‘ Write Analog Bank command
110   ADDRESS% = 6            ‘ Address of brain board
120   POSITION% = 1           ‘ Bank number
130   VALUE%(0) = 0           ‘ Module 0
140   VALUE%(1) = 0           ‘ Module 1
150   VALUE%(2) = 0           ‘ Module 2
160   VALUE%(3) = 0           ‘ Module 3
170   VALUE%(4) = 3048        ‘ Module 4
180   VALUE%(5) = 3048        ‘ Module 5
190   VALUE%(6) = 3048        ‘ Module 6
200   VALUE%(7) = 0           ‘ Module 7
210   GOSUB 1000              ‘ Call the driver
      .
      .
1000  CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010  IF ERRCOD% < 0 THEN GOTO 2000
1020  RETURN
```

WRITE ANALOG POINT 19

PURPOSE:

Writes to an analog point.

COMMAND TYPE:

Analog

PARAMETERS:

COMMAND	Contains the value 19.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the point number.
VALUE ARRAY	The first element of this array contains the value to be written to the point.

REMARKS:

All output module positions must first be configured as outputs before values can be written to them. The driver will ignore any write commands to positions configured as inputs and no errors will be returned.

The POSITION value is a point offset starting from the value in the ADDRESS parameter. The point number in the POSITION parameter can range from 0 to 511 if the ADDRESS parameter is 0. An error will be returned if the values in these parameters exceed the limits.

EXAMPLE:

This example writes the value 98 to position 15 at address 30. The example assumes that this position has been previously configured as an output.

```
100    COMMAND% = 19               ‘ Write Analog Point command
110    ADDRESS% = 30               ‘ Address of brain board
120    POSITION% = 15              ‘ Position 15
130    VALUE%(0) = 98              ‘ Value to be written
140    GOSUB 1000                  ‘ Call the driver
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

SET ANALOG WATCHDOG TIMEOUT 20

PURPOSE:

Sets the analog watchdog.

COMMAND TYPE:

Analog

PARAMETERS:

COMMAND	Contains the value 20.
ADDRESS	Contains the address of the Pamux brain board.
VALUE ARRAY	The first element of this array contains the value of the watchdog timeout interval for the analog brain board watchdog.

REMARKS:

The value in the first element of the VALUE array can be an integer from 0 to 65,535. This represents a delay length in units of 10 milliseconds, resulting in a range from 10 milliseconds to 10.92 minutes. A value of 0 disables the watchdog feature.

Use the Write Analog Watchdog command to set the values to be assigned to the outputs should a watchdog timeout occur.

EXAMPLE:

This example sets the analog watchdog delay for the board at address 12 to 8 minutes (480 seconds). This results from writing the value 48,000 to address 12. Note that integer values above 32,767 must be entered in hex when using BASIC.

```
100    COMMAND% = 20              ‘ Set Analog Watchdog Timeout
110    ADDRESS% = 12              ‘ Address of brain board
120    VALUE%(0) = &HBB80         ‘ 48000 decimal = BB80 hex
130    GOSUB 1000                 ‘ Call the driver
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

WRITE ANALOG WATCHDOG BANK 21

PURPOSE:

Writes to a bank of eight analog watchdog registers.

COMMAND TYPE:

Analog

PARAMETERS:

COMMAND	Contains the value 21.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the bank number.
VALUE ARRAY	Contains the values to be written to each watchdog register in the bank. Each element of this array corresponds to the watchdog register of one module position within the bank. Element 0 corresponds to the first module position's watchdog register, element 7 corresponds to the last module position's watchdog register.

REMARKS:

When a watchdog timeout occurs, the values in the watchdog registers are written to their corresponding output module positions. This task is performed automatically by the analog Pamux brain board if a watchdog time has been previously set with the Set Analog Watchdog Timeout command.

Only I/O points configured as outputs are affected.

EXAMPLE:

This example writes the value 1,024 to the watchdog registers of all module positions at address 10 of bank 1. The example assumes that these positions have been previously configured as outputs.

```
100    COMMAND% = 21              ‘ Write Analog Watchdog Bank
110    ADDRESS% = 10              ‘ Address of brain board
120    POSITION% = 1              ‘ Bank number
130    FOR I% = 0 TO 7            ‘ Fill all VALUE% elements with
                                    1024
140    VALUE%(I%) = 1024
150    NEXT
160    GOSUB 1000                 ‘ Call the driver
       .
       .
1000   CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010   IF ERRCOD% < 0 THEN GOTO 2000
1020   RETURN
```

WRITE ANALOG WATCHDOG POINT 22

PURPOSE:

Writes to an analog watchdog register.

COMMAND TYPE:

Analog

PARAMETERS:

COMMAND	Contains the value 22.
ADDRESS	Contains the address of the Pamux brain board.
POSITION	Contains the point number.
VALUE ARRAY	The first element of this array contains the value to be written to the watchdog register of the point.

REMARKS:

The POSITION value is a point offset starting from the value in the ADDRESS parameter. The point number in the POSITION parameter can range from 0 to 511 if the ADDRESS parameter is 0. An error will be returned if the values in these parameters exceed the limits.

When a watchdog timeout occurs, the values in the watchdog registers are written to their corresponding output module positions. This task is performed automatically by the analog Pamux brain board if a watchdog time has been previously set with the Set Analog Watchdog Timeout command.

Only I/O points configured as outputs are affected.

EXAMPLE:

This example writes the value 90 to the watchdog register of position 15 at address 36. The example assumes that this position has been previously configured as an output.

```
100   COMMAND% = 22              ‘ Write Analog Watchdog Point
110   ADDRESS% = 36              ‘ Address of brain board
120   POSITION% = 15             ‘ Position 15
130   VALUE%(0) = 90             ‘ Value to be written
140   GOSUB 1000                 ‘ Call the driver
      .
      .
1000  CALL
Pamux(ERRCOD%,ADDRESS%,COMMAND%,POSITION%,VALUE%(0))
1010  IF ERRCOD% < 0 THEN GOTO 2000
1020  RETURN
```

USING THE PAMUX DRIVER UNDER WINDOWS

Installation

The Pamux driver is provided on one disk. You should make a backup copy of this disk before installing the driver.

The Pamux driver disk includes DOS and WIN directories. To use the driver under Windows, you will need to copy files from the WIN directory to appropriate directories on your hard drive.

For complete installation instructions, view the README.TXT file on the driver disk.

Briefly, you will need to copy the file Pamux.DLL to your application directory or your Windows System directory. Note that Windows searches for DLLs first in memory, then in the current working directory, and finally in the Windows and Windows System directories.

In addition, you will need to copy header files (Pamux.H for C and Pamux.BAS for BASIC) to the directory in which your application source code is located.

The driver disk also includes the import library file Pamux.LIB. This file tells the linker that the Pamux APIs are part of a DLL. If this library file is used, the Pamux APIs do not need to be included in the imports section of the application's DEF file. Import libraries are not required for Visual Basic; the Pamux.BAS file serves this purpose.

Architecture

The Pamux driver is implemented as a Microsoft Windows DLL that allows Windows applications (such as Visual Basic) to communicate with Pamux I/O. The driver, Pamux.DLL, provides a set of APIs in the C language. This driver may be used with any Windows language that supports DLLs, such as Visual Basic, C, or C++.

The Pamux driver allows up to four AC28 cards to be used. Before an application can access an AC28 using the driver, the PamuxCardOpen() function must be used to acquire a handle to the AC28.

The driver allows multiple applications to access the AC28s simultaneously, although it does not provide a means of locking I/O for exclusive access by one application.

The driver keeps track of inputs and outputs to prevent a “1” from being written to a digital input. Writing a “1” to a digital input causes a “1” to be read from the input regardless of the actual state of the input module.

Pamux APIs

Required API Calls

For many applications, only four Pamux APIs are required. These perform the following functions:

- Open an AC28 to configure the AC28 and get a handle.
- Configure outputs.
- Read and write I/O.
- Close the AC28 when the application is about to end.

Naming Conventions

API names in the Pamux library start with “Pamux.” Example: “PamuxDigPointRead.”

API names follow the object-operation format, with the object first and the operation second. Example: “PamuxDigPointRead,” where “PamuxDigPoint” (the object) is first and “Read” (the operation) follows.

Utility functions, provided primarily for Visual Basic, start with “PamuxUtil.” Example: “PamuxUtilBitEqual.”

Banks and Points

Any I/O point can be addressed multiple ways. A 16-channel I/O board has two banks. Point 0, the first point, is accessed using a bank number of 0 and a point number of 0. Point 8 can be accessed in two ways:

1. A bank number of 0 and a point number of 8, or
2. A bank number of 1 and a point number of 0.

Common API Parameters and Return Values

`int hAc28`	Handle to an AC28 card. Handles are acquired using PamuxCardOpen().
`int Bank`	A bank number (0 to 63).
`int Point`	A point or channel number on a rack starting with zero.
`OutputMask`	A “1” bit represents an output. Used to configure outputs.

Return Values

All functions in the Pamux.DLL return an error value unless otherwise noted. A non-zero value indicates an error has occurred. An example of a function that does **not** return an error value is PamuxDigBankReadFast().

API Command Reference

The APIs listed in this section include function prototypes and descriptions. In general, most APIs return an integer error number. Zero indicates no error.

AC28 Operations

PamuxCardOpen

PROTOTYPE:

```
int PamuxCardOpen(int far* phAc28, int far* Reserved, int BaseIoPort, int
ResetIoPort, BOOL ResetLevel=FALSE);
```

DESCRIPTION:

Opens access to an AC28 card. `phAc28` gets a handle. `Reserved` always returns 0. Three parameters must be specified: the I/O Port addresses for the AC28 base port, its reset port, and the reset level. These three parameters must correspond to jumper settings on the AC28 card.

PamuxCardClose

PROTOTYPE:

```
int PamuxCardClose(int hAc28);
```

DESCRIPTION:

Releases the handle.

PamuxCardReset

PROTOTYPE:

```
int PamuxCardReset(int hAc28);
```

DESCRIPTION:

Resets the AC28 card, turning off all outputs. This is not called automatically when the AC28 is opened.

PamuxReadType

PROTOTYPE:

```
int PamuxReadType(int hAc28, int Bank, int* pType);
```

DESCRIPTION:

Reads brain board type. the AC28 reset register, when read, has the board type for the last brain board accessed. The values are: 1 for B4, 2 for a B5, and 3 for B6. A type of 0 indicates an unsuccessful operation. If Bank is -1, the type most recently accessed brain board is returned. This is a means of determining if the most recent operation was successful. If Bank is a valid bank number then this function will automatically access the brain board specified by ‘Bank’ by performing a ‘read’, then the board type will be determined.

Digital Bank Operations

The term “bank” refers to groups of eight digital I/O points. A 32-channel Pamux board with a B4 brain board has four banks. It is faster to read a bank all at once rather than to read each point individually.

Note that channel 0 corresponds to the least significant bit. For example when reading a bank with channels 0, 3, and 4 on and all other channels off, the returned value would be 19 hex (11001 binary, or 25 decimal).

PamuxDigBankConfig, PamuxDigBank16Config, PamuxDigBank32Config

PROTOTYPES:

```
int PamuxDigBankConfig(int hAc28, int Bank, int OutputMask, int Byte
Qty);

int PamuxDigBank16Config(int hAc28, int Bank, UINT OutputMask);

int PamuxDigBank32Config(int hAc28, int Bank, long OutputMask);
```

DESCRIPTION:

Configures a bank of digital I/O points as either inputs or outputs. A “1” in the mask indicates an output. Use `PamuxDigBankConfig` for configuring between 8 and 32 points, `PamuxDigBank16Config` for 16 points, or `PamuxDigBank32Config` for 32 points.

The `Byte Qty` parameter in `PamuxDigBankConfig` indicates how many banks to configure (`Byte Qty =` 4 for 32 points).

PamuxDigBankRead, PamuxDigBank16Read, PamuxDigBank32Read

PROTOTYPES:

```
int PamuxDigBankRead(int hAc28, int Bank, int far* pData);

int PamuxDigBank16Read(int hAc28, int Bank, int far* pData);

int PamuxDigBank32Read(int hAc28, int Bank, long far* pData);
```

DESCRIPTION:

Reads inputs and outputs and places the result in `pData.` Use `PamuxDigBankRead` for reading eight points, `PamuxDigBank16Read` for 16 points, or `PamuxDigBank32Read` for 32 points.

PamuxDigBankWrite, PamuxDigBank16Write, PamuxDigBank32Write

PROTOTYPES:

```
int PamuxDigBankWrite(int hAc28, int Bank, int Data);

int PamuxDigBank16Write(int hAc28, int Bank, int Data);

int PamuxDigBank32Write(int hAc28, int Bank, long Data);
```

DESCRIPTION:

Writes outputs using the value in `Data.` Inputs are not affected if written to. Use `PamuxDigBankWrite` for writing to eight points, `PamuxDigBank16Write` for 16 points, or `PamuxDigBank32Write` for 32 points.

Digital Point Operations

PamuxDigPointConfig

PROTOTYPE:

```
int PamuxDigPointConfig(int hAc28, int Bank, int Position, BOOL bOutput);
```

DESCRIPTION:

Configures a point as either an input or output. A non-zero value in `bOutput` will configure the point
as an output.

PamuxDigPointRead

PROTOTYPE:

```
int PamuxDigPointRead(int hAc28, int Bank, int Point, BOOL far* pData);
```

DESCRIPTION:

Reads the value of a point and puts the value in `pData.` The value will be either 1 for on or 0 for off.

PamuxDigPointWrite

PROTOTYPE:

```
int PamuxDigPointWrite(int hAc28, int Bank, int Point, BOOL Data);
```

DESCRIPTION:

Writes to a point using the value in `Data.` A non-zero value for `Data` will turn the point on.

Digital “Fast” Operations

For high-speed applications, these APIs can be used to bypass some error-checking and port calculations. The configure functions should be used to configure outputs.

PamuxDigIoPortGet

PROTOTYPE:

```
int PamuxDigIoPortGet(int hAc28, int far* pBank, int far* pPoint, int
far* pIoPort);
```

DESCRIPTION:

Provides the I/O port needed given the AC28 handle, the bank number, and the point number.

PamuxDigBankReadFast

PROTOTYPE:

```
int PamuxDigBankReadFast(int hAc28, int IoPort);
```

DESCRIPTION:

Reads one byte (eight bits) from the specified I/O port. Does nothing more than call the function `_inp()`, and hence may be used in Visual Basic as a general-purpose INP function. The return value is the value read rather than a Pamux error code.

PamuxDigBankWriteFast

PROTOTYPE:

```
void PamuxDigBankWriteFast(int hAc28, int IoPort, int Data);
```

DESCRIPTION:

Writes one byte (eight bits) to the specified port. Does nothing more than call the function `_out()`, and hence may be used in Visual Basic as a general-purpose OUT function.

Analog Bank Operations

PamuxAnaBank16Config

PROTOTYPE:

```
int PamuxAnaBank16Config(int hAc28, int Bank, UINT OutputMask);
```

DESCRIPTION:

Configures a bank of analog I/O points as either inputs or outputs. A 1 in the mask indicates an output.

PamuxAnaBank16Read

PROTOTYPE:

```
int PamuxAnaBank16Read(int hAc28, int Bank, int far DataArray16[])
```

DESCRIPTION:

Reads a bank of 16 analog points and places the values in the `DataArray` (channel 0 in element 0 and channel 15 in element 15).

PamuxAnaBank16Write

PROTOTYPE:

```
int PamuxAnaBank16Write(int hAc28, int Bank, int far DataArray16[])
```

DESCRIPTION:

Writes values to a bank of analog points (channel 0 value in element 0).

Analog Point Operations

PamuxAnaPointConfig

PROTOTYPE:

```
int PamuxAnaPointConfig(int hAc28, int Bank, int Point, BOOL bOutput);
```

DESCRIPTION:

Configures an analog point as either an input or output. A non-zero value to `bOutput` will configure the point as an output.

PamuxAnaPointRead

PROTOTYPE:

```
int PamuxAnaPointRead(int hAc28, int Bank, int Point, int far* pData)
```

DESCRIPTION:

Reads the value of an analog point.

PamuxAnaPointWrite

PROTOTYPE:

```
int PamuxAnaPointWrite(int hAc28, int Bank, int Point, int Data)
```

DESCRIPTION:

Writes the value in Data to an analog point.

Analog Watchdog Operations

PamuxAnaWatchdogSet

PROTOTYPE:

```
int PamuxAnaWatchdogSet(int hAc28, int Bank, int Time100, int Data 16[])
```

DESCRIPTION:

Sets up the watchdog timer for an analog point. Time100 units are hundredths of a second.

PamuxAnaWatchdogTime

PROTOTYPE:

```
int PamuxAnaWatchdogTime(int hAc28, int Bank, int Time100)
```

DESCRIPTION:

Sets the value of the watchdog timer in units of hundredths of a second. Setting `Time100` to 0 will disable the watchdog. Setting the time to 0 can also be used to reset the watchdog error flag if it has tripped. This and any other analog function can be used to “tickle” the watchdog to prevent it from tripping.

Analog Status Operations

PamuxAnaStatusGetAsError

PROTOTYPE:

```
int PamuxAnaStatusGetAsError(int hAc28, int Bank)
```

DESCRIPTION:

Gets an analog board’s status and returns an equivalent error. Analog read functions also return the same result after performing their read. An analog configure function can be used to clear the “power-up” error. The “old data” error is cleared by the B6 processor as it updates inputs. This function could be used to wait for fresh input data.

Pamux Utility Operations

The following utility functions are provided primarily for Visual Basic applications. These are not Pamux-specific functions.

Bit Operations (PamuxUtilBitSetTo, PamuxUtilBitSet, PamuxUtilBitClr, PamuxUtilBitTest)

PROTOTYPES:

```
void PamuxUtilBitSetTo(UINT far* pData, UINT BitNumber0, BOOL bBitValue);

void PamuxUtilBitSet(UINT far* pData, UINT BitNumber0);

void PamuxUtilBitClr(UINT far* pData, UINT BitNumber0);

BOOL PamuxUtilBitTest(UINT Data, UINT BitNumber0);
```

DESCRIPTION:

These bit operations are useful in Visual Basic applications to access individual bits within an integer. The BitSetTo function either sets or clears the specified bit based on the value of `bBitValue.` Any non-zero value for `Data` will turn on the bit. The BitSet and BitClr functions set and clear the specified bit. Bit numbers start at zero for the least significant bit (LSB). The zero at the end of the parameter name "BitNumber0" serves as a reminder of this fact for anyone looking through the function definitions in either the .BAS file or the .H header files. **PamuxUtilBitTest** returns true if bit number **BitNumber0** in **Data is** set.

Scaling Operations (PamuxUtilScaleI2I, PamuxUtilScaleI2F, PamuxUtilScaleF2F, PamuxUtilScaleF2I)

PROTOTYPES:

```
int PamuxUtilScaleI2I(int X1, int X2, int Y1, int Y2, int Xin);

float PamuxUtilScaleI2F(int X1, int X2, float Y1, float Y2, int Xin);

float PamuxUtilScaleF2F(float X1, float X2, float Y1, float Y2, float
Xin);

int PamuxUtilScaleF2I(float X1, float X2, int Y1, int Y2, float Xin);
```

DESCRIPTION:

These interpolation functions are useful for converting between engineering units and raw analog counts. Pamux analog input and output values range between 0 and FFF hex (4,095 decimal). These values typically correspond to engineering units, such as pH and psi. For example, to convert raw counts (from 0 to FFF hex) to a percentage, use:

```
float  fPercent=PamuxUtilScaleI2F(0,0xFFF,0F,100.F,nRawCount);
```

Pack/Unpack Utility Operations (PamuxUtilBitPackArray2I, PamuxUtilBitUnPackI2Array)

PROTOTYPE:

```
Void PamuxUtilBitPackArray2I(UINT far* DestInt, int far SourceArray[],
int Qty);

Void PamuxUtilBitUnPackI2Array(int far DestArray[], UINT SourceInt, int
Qty);
```

DESCRIPTION:

PamuxUtilBitPackArray2I converts an array of boolean (0 or not 0) values to a bit packed integer.

PamuxUtilBitUnPackI2Array converts a bit packed integer to an array of boolean (1 or 0) values. Qty indicates the number of bits to be packed/unpacked.

Status/Error Codes

A 0 returned as a status or error code by a Pamux API indicates no error. The most common error codes are listed below. The complete set of status and error codes can be found in the Pamux header files (Pamux.H or Pamux.BAS).

Table 3-2: Status/Error Codes Returned by Pamux Driver APIs

Error	Enum Name in Pamux.H or Constant in Pamux.BAS	Description
0	PamuxNoError	No error.
-1	(not used)	Not an error code used by the Pamux driver.
-2	PamuxErrorAc28Handle	A bad AC28 handle parameter was used.
-3	PamuxErrorBank	A bad bank number parameter was used.
-4	PamuxErrorIoPort	A bad I/O port parameter was used.
-5	PamuxErrorNoMoreHandle	No more handles could be allocated. An attempt was made to open too many AC28s.
-6	PamuxErrorAc28Conflict	An attempt was made to open an AC28 with parameters that conflicted with an AC28 already in use.
-7	PamuxErrorPoint	A bad point or channel parameter was used.

PROGRAMMING WITHOUT THE PAMUX DRIVER

OVERVIEW

This chapter covers everything you will need to know to program a Pamux system without using the Pamux software driver. It begins by providing information on Pamux bus pin definitions and bus timing. It then addresses all programming issues for both digital and analog Pamux stations, providing specific examples written in MS-DOS QBasic.

If you are using the Pamux driver, you may skip this chapter or review it as a background reference to acquire a deeper understanding of Pamux communication.

BUS PIN DEFINITIONS – NOT USING AC28

To implement the host Pamux bus interface without using an AC28, you will need a UCA4 adapter card and a 17-line parallel port. The port is configured with eight data lines, six address lines, a read strobe line, a write strobe line, and a reset line. AC36 is a preconfigured adapter to connect Pamux to an Intel Multibus® format parallel port. Figure 4-1 details the Pamux bus pin-out:

Table 4-1: Pamux Bus Pin Definitions

Pin	Signal Function	Pin	Signal Function
1	Address line 0	33	Data line 7
3	Address line 1	35	Data line 6
5	Address line 2	37	Data line 5
7	Address line 3	39	Data line 4
9	Address line 4	41	Data line 3
11	Address line 5	43	Data line 2
13	Write strobe line	45	Data line 1
15	Read strobe line	47	Data line 0
		49	Reset line

NOTE: All even-numbered pins on the connector are connected to logic ground.

BUS TIMING

Write Timing

Pamux write timing is simple to generate. First, apply the Pamux address and data to the bus, wait for at least 100 nanoseconds, then activate the write strobe for at least 2000 nanoseconds. When you deactivate the write strobe, the data will be latched on the addressed Pamux station. The address lines must be held stable for a minimum of 100 nanoseconds after the write strobe is brought low.

Figure 4-1 shows the minimum bus timing for a Pamux bus write.

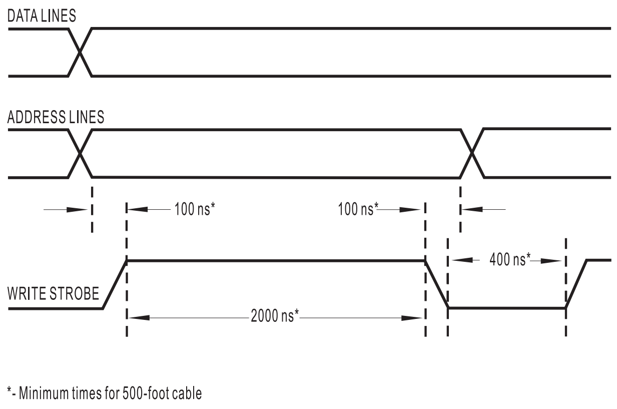


Figure 4-1: Write Timing on the Pamux Bus

Read Timing

To generate the Pamux bus read cycle, first apply the Pamux address to the bus, wait for at least 100 nanoseconds, then activate the read strobe. After 2000 nanoseconds, the data will be available. Read the data and *then* deactivate the read strobe.

Figure 4-2 shows the minimum bus timing for a read cycle on a Pamux bus with 500 feet of cable.

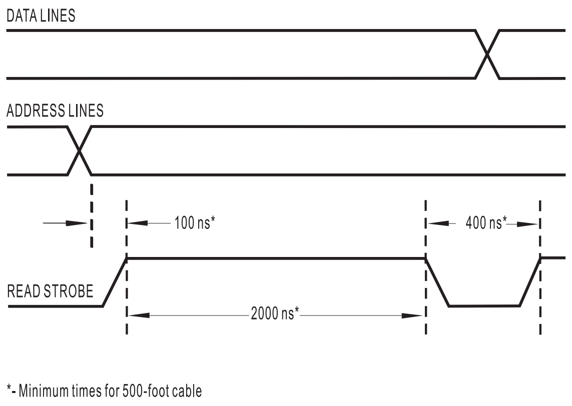


Figure 4-2: Read Timing on the Pamux Bus

DIRECT PROGRAMMING OF PAMUX STATIONS WITH AC28

The remainder of this chapter presents important information on reading from and writing to Pamux stations. Several simplified examples illustrate the basics of Pamux driver programming using BASIC's INP function and OUT statement. These examples are written in MS-DOS QBasic, available on most PCs configured with MS-DOS 5.0 and above.

For more complete examples that handle error checking and provide a set of functions, see the Pamux.DLL source code, which is written in C/C++. This source code is found on the Pamux driver disk (in the WIN\SOURCE directory) and on the Opto 22 BBS.

Variables Used in Examples

This variable is the I/O port address for the AC28 data port. This value corresponds to the "A" jumpers on the AC28.

```
CONST MyAc28IoPort% = &H100
```

This variable is the I/O port address for the AC28 reset port. This value corresponds to the "R" jumpers on the AC28.

```
CONST MyAc28ResetPort% = &H1E0        ' Base I/O address for reset
register
```

This variable determines the reset level of the AC28. This value corresponds to the jumper setting on the brain boards. This example assumes an active-high reset level.

```
CONST MyAc28ResetLevel% = -1          ' Turn reset on
```

Address of digital brain board.

```
CONST MyDigitalBoard% = 0      ' Assume digital B4 or B5 at address 0
```

Address of analog brain board.

```
CONST MyAnalogBoard% = 4       ' Assume analog B6 at address 4
```

Address of a nonexistent brain board. This is used to demonstrate the use of the reset register to determine if a board is present and functioning.

```
CONST MyNonExistBoard% = 10    ' Address of board that doesn't exist
```

This example assumes a digital board with inputs on channels 0, 3, 9, 11 and 15. Channel 0 corresponds to the least significant bit and channel 15 corresponds to the most significant bit.

```
CONST MyDigitalOutMask% = &H75F6 ' Assume inputs at chs. 0, 3, 9, 11, 15
```

Analog input channel used in these examples.

```
CONST MyAnalogInputChannel% = 4
```

Analog output channel used in these examples.

```
CONST MyAnalogOutputChannel% = 1
```

Analog output mask used in these examples. Assumes channels 0, 1, 3 are outputs.

```
CONST MyAnalogOutputMask% = &HB
```

Utility Subroutines Used in Examples

The examples in this chapter make use of the following subroutines:

- MyKillTime: Pauses for a short interval.
- GetAccess: Gains access to a B6 analog brain board.
- Release Access: Releases access to the B6 brain board.
- CombineUpperAndLowerBytes: Takes two bytes and combines them into a 16-bit integer.
- GetUpperAndLowerBytes: Takes a 16-bit integer and breaks it into two bytes.

The code for the GetAccess and ReleaseAccess subroutines can be found on page 109. The code for the remaining subroutines is presented at the end of the chapter.

TURN OFF RESET

Before reading from or writing to any Pamux station, you must make sure a reset is not being requested. To do so you will need to turn off the reset if it is configured on the brain board as active high, or turn on the reset if it is configured as active low. (See Chapter 3 to review the positions of the reset jumpers on the brain boards.)

```
OUT MyAc28ResetPort%, NOT MyAc28ResetLevel%   ‘ Restore reset line
```

Resetting the System

The following example pulses the reset line. First, it asserts the reset line by turning it on if it is active high, turning it off it is active low. Then, after a brief time interval, it returns the reset line to its normal state. The subroutine MyKillTime is used to pause the routine (see “Common Subroutines” at the end of this chapter for specifics).

```
DECLARE SUB MyKillTime ()
OUT MyAc28ResetPort%, MyAc28ResetLevel%           ‘ Reset I/O
CALL MyKillTime                                   ‘ Wait at least 20 ms
OUT MyAc28ResetPort%, NOT MyAc28ResetLevel%
```

READING FROM AND WRITING TO DIGITAL PAMUX STATIONS

The I/O on digital B4 and B5 Pamux stations is accessed eight bits at a time. Channel 0 is the least significant bit.

Reading Digital Inputs

The following example reads 16 digital input channels and prints the hex representation on the screen. During the loop, the address LED should be on almost continuously and the watchdog LED should be off.

```
DIM UpperByte%
DIM LowerByte%
DO WHILE 1                                 ' Loop until ctrl-break
  UpperByte% = INP(MyAc28IoPort% + MyDigitalBoard% + 1)
                                           ' Read bank 1
  LowerByte% = INP(MyAc28IoPort% + MyDigitalBoard%)
                                           ' Read bank 0
  PRINT HEX$( UpperByte%); HEX$( LowerByte%)
LOOP
```

Writing Digital Outputs

You must be especially careful when writing to Pamux stations. When writing to a bank of eight modules that includes both inputs and outputs, you **must** be sure to write 0's to input module positions. If you write a 1 to an input module, the module will read back as being active (on) even if it is inactive (off).

The following example writes to 16 digital output channels using dummy data that implements a ring counter, which is used to shift the bits of the register to the left one bit at a time. Note that an output mask is used to prevent input channels from being turned on. If an input is turned on, it will read “on” regardless of its actual input state.

```
DIM a&: a& = 1
DIM OutputData%
                     ' Prevents a 1 from being written to an input
DO WHILE 1           ' Loop until ctrl-break
  a& = a& * 2: IF a& > &HFFFF& THEN a& = 1   ' Generate some dummy data
  OutputData% = a& AND MyDigitalOutMask%
  CALL GetUpperAndLowerBytes(OutputData%, UpperByte%, LowerByte%)
  OUT MyAc28IoPort% + MyDigitalBoard% + 1, UpperByte%
  OUT MyAc28IoPort% + MyDigitalBoard%, LowerByte%
LOOP
```

READING FROM AND WRITING TO ANALOG PAMUX STATIONS

Each Pamux B6 analog brain board interfaces to the Pamux bus through a dual-port RAM device. The on-board microprocessor constantly reads from and writes to this dual-port RAM. For the host computer to access data, the host must first request permission to access the contents of the RAM. This handshaking scheme prevents contention problems.

Each analog Pamux brain board takes up two byte locations on the bus. The upper address is the control register, the lower address is the data register. The control and data registers are at the following locations:

• Control Register Address	=	AC28 Base Address + Pamux Address + 1
• Data Register Address	=	AC28 Base Address + Pamux Address

The Pamux address is set by configuration jumpers on the B6 brain board. Configuration jumpers on the AC28 determine its base address.

Pamux Internal Registers

To read or write analog Pamux data, you must access each brain board's internal registers (dual-port RAM locations).

The following is a map of the Pamux analog internal registers. All values are in hex and can be read from or written to, since all registers are RAM locations.

Table 4-2: Pamux Internal Registers

Register	Description	Register	Description
0	Analog Channel 0 (LSB)	23	Watchdog Channel 1 (MSB)
1	Analog Channel 0 (MSB	24	Watchdog Channel 2 (LSB)
2	Analog Channel 1 (LSB)	25	Watchdog Channel 2 (MSB)
3	Analog Channel 1 (MSB)	26	Watchdog Channel 3 (LSB)
4	Analog Channel 2 (LSB)	27	Watchdog Channel 3 (MSB)
5	Analog Channel 2 (MSB)	28	Watchdog Channel 4 (LSB)
6	Analog Channel 3 (LSB)	29	Watchdog Channel 4 (MSB)
7	Analog Channel 3 (MSB)	2A	Watchdog Channel 5 (LSB)
8	Analog Channel 4 (LSB)	2B	Watchdog Channel 5 (MSB)
9	Analog Channel 4 (MSB)	2C	Watchdog Channel 6 (LSB)
A	Analog Channel 5 (LSB)	2D	Watchdog Channel 6 (MSB)
B	Analog Channel 5 (MSB)	2E	Watchdog Channel 7 (LSB)
C	Analog Channel 6 (LSB)	2F	Watchdog Channel 7 (MSB)
D	Analog Channel 6 (MSB)	30	Watchdog Channel 8 (LSB)
E	Analog Channel 7 (LSB)	31	Watchdog Channel 8 (MSB)
F	Analog Channel 7 (MSB)	32	Watchdog Channel 9 (LSB)
10	Analog Channel 8 (LSB)	33	Watchdog Channel 9 (MSB)
11	Analog Channel 8 (MSB)	34	Watchdog Channel 10 (LSB)
12	Analog Channel 9 (LSB)	35	Watchdog Channel 10 (MSB)
13	Analog Channel 9 (MSB)	36	Watchdog Channel 11 (LSB)
14	Analog Channel 10 (LSB)	37	Watchdog Channel 11 (MSB)
15	Analog Channel 10 (MSB)	38	Watchdog Channel 12 (LSB)
16	Analog Channel 11 (LSB)	39	Watchdog Channel 12 (MSB)
17	Analog Channel 11 (MSB)	3A	Watchdog Channel 13 (LSB)
18	Analog Channel 12 (LSB)	3B	Watchdog Channel 13 (MSB)
19	Analog Channel 12 MSB)	3C	Watchdog Channel 14 (LSB)
1A	Analog Channel 13 (LSB)	3D	Watchdog Channel 14 (MSB)
1B	Analog Channel 13 MSB)	3E	Watchdog Channel 15 (LSB)
1C	Analog Channel 14 (LSB)	3F	Watchdog Channel 15 (MSB)
1D	Analog Channel 14 MSB)	7A	Watchdog Timer (LSB)
1E	Analog Channel 15 (LSB)	7B	Watchdog Timer (MSB)
1F	Analog Channel 15 (MSB)	7C	Status Register
20	Watchdog Channel 0 (LSB)	7E	Configuration Byte (Lower 8 Channels)
21	Watchdog Channel 0 (MSB)	7F	Configuration Byte (Upper 8 Channels)
22	Watchdog Channel 1 (LSB)	82	Semaphore Register

MSB= Most significant byte LSB= Least significant byte

Gaining and Releasing Access

Before reading from or writing to a Pamux analog brain board, you must request access to its internal registers. This prevents data values from being accessed simultaneously by both an AC28 (or equivalent) and a brain board.

Gaining Access

First, write the hex value 82 to the control register (AC28 base address + Pamux address + 1). Then read the data register (AC28 base address + Pamux address) and test bit 7 of the byte just read. Bit 7 is the semaphore bit. When the value of bit 7 is 1, the microprocessor on the Pamux brain board is updating the contents of the RAM. Once the value of bit 7 becomes 0, you have gained access.

Once you have gained access, the value of bit 7 will continue to be 1 until you release access (as described below).

The following function illustrates how to gain access to an analog brain board. The GetAccess function returns True if access is gained, False otherwise. This example tries a fixed number of times before "giving up," but a better approach would be to try for a fixed amount of time as well. Note that access may not be granted if there is a hardware failure.

```
FUNCTION GetAccess% (DataPortArg%)
  CONST SemaphoreRegister% = &H82
  CONST MaxTryQty% = 30          ' Quantity of times to retry
  DIM ControlPort%: ControlPort% = DataPortArg% + 1 ' 2nd register on B6
  OUT ControlPort%, SemaphoreRegister% ' Point to the semaphore register
  DIM TryQty%                    ' Loop counter -- don't try forever
  DIM IsBusy%                    ' Non-zero if busy
  FOR TryQty% = 1 TO MaxTryQty%
    IsBusy% = INP(DataPortArg%) AND &H80      ' Get bit 7
    IF 0 = IsBusy% THEN
      GetAccess% = -1            ' Return True to indicate success
      EXIT FUNCTION              ' Return
    END IF
  NEXT
  GetAccess% = 0                 ' Return False to indicate failure
END FUNCTION
```

Releasing Access

This step is crucial. Once you are finished reading from or writing to the Pamux analog brain board, you must release access back to the brain board. **If you do not release access, inputs and outputs will never be updated!**

To release access, write a hex 82 to the control register, then write anything to the data register. The function below illustrates how to do this in QBasic. The ReleaseAccess function releases access by simply writing any value to the semaphore register.

```
SUB ReleaseAccess (DataPortArg%)
  CONST SemaphoreRegister% = &H82
  CONST Anything% = &HFF
  DIM ControlPort%: ControlPort% = DataPortArg% + 1 ‘ 2nd register on B6
  OUT ControlPort%, SemaphoreRegister%
  OUT DataPortArg%, Anything%
END SUB
```

Reading an Analog Channel

Remember that before reading an analog channel, you must first gain access to the internal registers, as described previously.

After gaining access, write the position to be read to the control register, then read the contents of the data register to get the least significant byte. Increment the contents of the control register, then read the data register to get the most significant byte. Remember, I/O instructions are done one byte at a time, and an analog value is a 12-bit value (0–4095).

The following example continuously reads an analog input and prints the value to the screen.

```
DIM HaveAccess%                          ‘ Non-zero if we have access
DIM DataPort%: DataPort% = MyAc28IoPort% + MyAnalogBoard%
DIM ControlPort%: ControlPort% = DataPort% + 1
DIM LowerByte%
DIM UpperByte%
DIM AnalogValue%
DO WHILE 1                               ‘ Loop until ctrl-break
  HaveAccess% = GetAccess%(DataPort%)
  IF HaveAccess% THEN
    OUT ControlPort%, MyAnalogInputChannel% * 2  ‘ Select a channel
    LowerByte% = INP(DataPort%)          ‘ Read lower byte
    OUT ControlPort%, MyAnalogInputChannel% * 2 + 1
    UpperByte% = INP(DataPort%)          ‘ Read upper byte
    AnalogValue% = CombineUpperAndLowerBytes(UpperByte%, LowerByte%)
    PRINT AnalogValue%                   ‘ Print result
  ELSE
    PRINT “No Access”
  END IF
  CALL ReleaseAccess(DataPort%)
LOOP
```

IMPORTANT: Don’t forget to release access to the brain board if this is the last operation you will be doing on this pass.

Normal under-range values are returned as a 16-bit twos-complement number (i.e., FFFF hex = -1, FFFE = -2, FFFD = -3, etc.). Input channels that do not have modules installed will return -4096 (F000 hex). Input modules that receive signals too far under range (below 1.25% of span) will stop relaying data to the CPU. When this happens, the data returned will be -4096. This can occur when a 4–20 milliampere signal goes below 3.8 milliamperes, or when a thermocouple input becomes an open circuit.

Configuring and Writing to Analog Channels

The Pamux analog brain board configures all channels as inputs on power-up or after a reset. Thus, if you are using output modules, they must be configured before they can be turned on.

To configure a channel as an output, you must set the module position's bit to 1 in the configuration register. The configuration register is found at location 7E and 7F Hex.

For example, if module positions 0, 1, and 3 are outputs, the configuration byte would be 0B, as indicated below:

```
Position:  7 6 5 4 3 2 1 0
Bits:      0 0 0 0 1 0 1 1 = 0B Hex
```

With this information, you are ready to configure the channels. First you must gain access, as described previously. Next, write the address of the configuration register to the control register. Finally, write the configuration byte to the data register.

Once you have configured the channels, you are ready to write a value to an analog position. Writing is the reverse of reading. The steps are as follows:

1. Write the position of the analog channel's least significant byte to the control register.
2. Write the least significant byte to the data register.
3. Increment the control register.
4. Write the most significant byte to the data register.

The following example configures an analog output and, within a loop, continuously writes a changing value to the output.

The following example configures an analog output and, within a loop, continuously writes a changing value to the output.

```
DIM HaveAccess%       ' Non-zero if we have access
DIM DataPort%: DataPort% = MyAc28IoPort% + MyAnalogBoard%
DIM ControlPort%: ControlPort% = DataPort% + 1
DIM LowerByte%
DIM AnalogValue%
DIM AnalogPercent%
                      ' Configure analog outputs
HaveAccess% = GetAccess%(DataPort%)
IF HaveAccess% THEN
   CALL GetUpperAndLowerBytes(MyAnalogOutputMask%, UpperByte%, LowerByte%)
   OUT ControlPort%, OutConfigRegister%
   OUT DataPort%, LowerByte%
   OUT ControlPort%, OutConfigRegister% + 1
   OUT DataPort%, UpperByte%
ELSE
   PRINT "No Access"
END IF
CALL ReleaseAccess(DataPort%)
                      ' Done configuring analog outputs
DO WHILE 1            ' Loop until ctrl-break
   ' Generate dummy data in AnalogValue% using QBasic's SIN and TIMER
                      ' Data fluctuates between 0 and &HFFF
   AnalogValue% = &HFFF * (SIN(TIMER) + 1) / 2
   AnalogPercent% = AnalogValue% / &HFFF * 100
   PRINT AnalogValue%, AnalogPercent%; "%"
   HaveAccess% = GetAccess%(DataPort%)
   IF HaveAccess% THEN
      CALL GetUpperAndLowerBytes(AnalogValue%, UpperByte%, LowerByte%)
      OUT ControlPort%, MyAnalogOutputChannel% * 2' Select a channel
      OUT DataPort%, LowerByte%
      OUT ControlPort%, MyAnalogOutputChannel% * 2 + 1
      OUT DataPort%, UpperByte%
   ELSE
      PRINT "No Access"
   END IF
   CALL ReleaseAccess(DataPort%)
LOOP
```

IMPORTANT: Don't forget to release access to the brain board if this is the last operation you will be doing on this pass.

The following example shows how to configure and write an output on a Pamux B6 brain board, using the DOS Debug utility. The Debug utility uses "o" to command an output to a port, and "i" to read data at a port. The same port reading and writing structure is used in a high-level language to perform the same operation. To run Debug, type 'debug' at the DOS prompt.

For an AC28 configured at port 140h, and a B6 at Pamux address 20:>Data Register = AC28 Port + Address Offset = 140h + 20 = 160h>Control Register = AC28 Port + Address Offset + 1 = 140 + 20 + 1 = 161h"

Table 4-3: Instructions — Pamux Access With DOS Debug

	Type the Following		Reason
CONFIGURATION	1.	o161 82	Request access to B6 dual-port RAM by requesting semaphore (signaling) register. "Semaphore" comes from the maritime flag-waving method of communication between ships — this register serves as the flag that is "waved" to get the attention of the B6 brain board.
	2.	i160 00	Test for access. A 00 is actually a "no access" code, but the act of testing for access will cause access to be granted. If this step is repeated, the board will return C0, indicating access. This second test is not required, however.
	3.	o161 7E	Writing 7E to control register gains access to lower configuration byte per Table 4-2; any data written to the data register will now be placed in the lower configuration byte.
	4.	o160 FF	Writing FF to the data register configures the lower byte as all outputs (FFh = 11111111b).
	5.	o161 82	Request the semaphore register again.
	6.	o106 FF	Writing any data to the semaphore register after access has been granted releases access; FF is convenient. The board is now configured.
READ	7.	o161 82	Now that the board is configured (only needs to be done on power-up), get access to write output channels.
	8.	i160 00	
	9.	o161 00	Writing 00 to the control register sets the B6 to receive the low data byte for output channel 0per Table 4-2; remember, a 12-bit number must be expressed as two bytes.
	10.	o160 FF	Write data for channel 0 low byte. FF is the less significant byte of 0FFFh, or 4095 in decimal. This is equivalent to a full-scale output.
	11.	o161 01	Set up to write the upper byte of data to output channel 0.
	12.	o160 0F	Write high data byte. 0Fh is the more significant byte of 0FFFh.
	13.	o161 82	Release access to board. As soon as data has been written to the semaphore register, the board is released and any output module on channel 0 of this B6 should go to full-scale.
	14.	o160 FF	

Watchdog Registers

The watchdog feature allows all outputs to be set to a predetermined value upon loss of communication from the host computer. Each analog output on a B6 can have a watchdog value. Only output module positions are affected. Note that the jumpers on the B6 must be configured to enable watchdog operation (see Chapter 3 for details).

The watchdog can be used to set a time interval that a Pamux analog brain board will wait. If the brain board is not accessed by the host within the interval, the board will enter a watchdog state. In this condition the analog brain board will set bit 2 in the status register, then output all values contained in the watchdog registers to their corresponding module positions.

The watchdog registers include a watchdog timer register (low byte and high byte), starting at hex 7A in RAM, and 16 watchdog value registers (low byte and high byte), starting at location hex 20. Each 16-bit watchdog register corresponds to a specific module position. On power-up, the watchdog feature is disabled (watchdog timer register = 0).

To use the watchdog, a value from 1 to 65,535 must be loaded into the watchdog timer register. This value represents a watchdog time in units of 10 milliseconds (hence, a value of 100 = 1 second). A value of 0 in the watchdog timer RAM location disables the watchdog.

After loading the watchdog timer register, you should load all watchdog value registers with the values to output if a watchdog condition occurs. The default value for each location is 0.

As with any internal RAM location, you must first gain access before you can write to the watchdog registers. You must also release access after you are finished.

IMPORTANT: On a power-up or reset condition, all watchdog registers are set to 0 and must be reinitialized!

The following example configures the watchdog for an analog output. This example assumes that the channel has been configured previously as an output. The example sets the watchdog time to one second. It then defines an arbitrary watchdog value of half-scale (50% or &HFFF/2). Such a value makes it more likely that the watchdog will be noticed when tripped; however, most applications would use a watchdog value of 0.

To verify that the watchdog works properly, combine this example with the previous, then trip the watchdog by interrupting the program (hit PAUSE or CTRL-BREAK). This should cause all outputs configured with a watchdog to assume their half-scale values.

Example:

```
CONST WatchdogTimeRegister% = &H7A
CONST MyAnalogOutputChannel% = 1
CONST MyWatchdog10MilSec% = 100‘ Watchdog time: 1 sec = 100*10 msec
CONST MyWatchdogValue% = &H7FF‘ Watchdog value: half scale
HaveAccess% = GetAccess%(DataPort%)
IF HaveAccess% THEN
  ‘ Configure watchdog timer
  CALL GetUpperAndLowerBytes(MyWatchdog10MilSec%, UpperByte%, LowerByte%)
  OUT ControlPort%, WatchdogTimeRegister%
  OUT DataPort%, LowerByte%
  OUT ControlPort%, WatchdogTimeRegister% + 1
  OUT DataPort%, UpperByte%
  ‘ Set watchdog value
  DIM WatchdogReg%
  WatchdogReg% = MyAnalogOutputChannel% * 2 + &H20
  CALL GetUpperAndLowerBytes(MyWatchdogValue%, UpperByte%, LowerByte%)
  OUT ControlPort%, WatchdogReg%
  OUT DataPort%, LowerByte%
  OUT ControlPort%, WatchdogReg% + 1
  OUT DataPort%, UpperByte%
ELSE
  PRINT “No Access”
END IF
CALL ReleaseAccess(DataPort%)
```

Status Register

The status register is at internal RAM location hex 7C. The register includes three bits:

- Bit 0 — Power-up/reset flag
- Bit 1 — Fresh data flag
- Bit 2 — Watchdog timeout flag

Bit 0 is set to 1 if the Pamux B6 brain board has been reset due to a power-up condition or a reset signal from the host. Bit 1 is set to a 1 every time the microprocessor transfers data to and from the dual-port RAM and can be used to indicate if the current data is fresh. Bit 2 is set to a 1 if a watchdog timeout occurs.

Bit 0 is important because it can warn the host that the brain board needs to be reconfigured due to a loss of power. Bit 1 can be used as a warning that the host is polling too quickly and that the brain board’s microprocessor has not had sufficient time to update the RAM.

To make use of these flags, Bit 0 should be cleared by the host after the brain board has been configured. The flag should then be checked periodically to see if the brain board has lost its configuration and needs to be reconfigured.

Bit 1 should be checked after gaining access to see if the data will be fresh, then the bit must be cleared before releasing access. It is up to the host to clear this flag, or the flag will be meaningless.

Bit 2 should also be checked after gaining access to see if a watchdog timeout has occurred, then all analog outputs should be set to new values and bit 2 of the status register cleared.

Performing Diagnostics

Check Board Type

This example uses the capability of newer Pamux products to identify an I/O brain board type and to determine if a brain board was actually accessed. You can verify that a brain board was successfully accessed by reading the reset register. Successful operations are indicated by a 1 for a B4, a 2 for a B5, and a 3 for a B6. A 0 indicates an unsuccessful operation. With an older AC28 that does not provide ACK support, an FF will be read whether or not the operation is successful.

The following example assumes that one digital and one analog brain board are connected. Each is accessed by simply doing an INP at its address. If the board actually responds, its type can be read from the reset register.

At the end of the routine, an example is provided of an unsuccessful brain board access. A nonexistent board is accessed to demonstrate that the board type read from the reset register is 0.

```
DIM BoardType%
i% = INP(MyAc28IoPort% + MyDigitalBoard%)
BoardType% = INP(MyAc28ResetPort%)
PRINT BoardType%;                ‘ Should print 1 for B4 or 2 for B5
i% = INP(MyAc28IoPort% + MyAnalogBoard%)
BoardType% = INP(MyAc28ResetPort%)
PRINT BoardType%;                ‘ Should print 3 for B6
i% = INP(MyAc28IoPort% + MyNonExistBoard%)
BoardType% = INP(MyAc28ResetPort%)
PRINT BoardType%   ‘ Should print 0 for a board that doesn’t exist
```

Check if AC28 Is Working

To determine if an AC28 is working properly, perform the diagnostic above under “Check Board Type” and also implement the routine under “Reading Digital Inputs” (see page 4-5).

Determine Suitable I/O Addresses for an AC28

If an I/O address is unused, the I/O address should read &HFF or all 1’s. If the AC28 is not installed and the Reset I/O address in not already in use, the following statement should print “255”:

```
PRINT INP( MyAc28ResetPort% )
```

The following loop will check a range of I/O port addresses for conflicts before the AC28 is installed. The AC28 I/O address is assumed to be in the variable MyAc28IoPort%.

```
DIM i%
DIM InpByte%
CONST MaxPamuxAddress% = 63
DIM MaxPort%: MaxPort% = MyAc28IoPort% + MaxPamuxAddress%
FOR i% = MyAc28IoPort% TO MaxPort%
  InpByte% = INP(i%)
  IF &HFF <> InpByte% THEN
    EXIT FOR                       ‘ Error -- a port is in use
  END IF
NEXT
IF &HFF <> InpByte% THEN
  ‘ Print something like “Port &H100 is in use.”
  PRINT “Port &H”; HEX$(i%); “ is in use.”
  BEEP
ELSE
  ‘ Print something like “Ports &H100 thru &H13F are OK to use.”
  PRINT “Ports &H”; HEX$(MyAc28IoPort%); “ thru &H”; HEX$(MaxPort%);
  PRINT “ are OK to use.”
END IF
```

Common Subroutines

MyKillTime

This function implements a pause when running QBasic under DOS. It uses the BIOS timer to detect a “timer tick.” Environments other than DOS, such as Windows or real-time kernels, typically offer better timing capability with better resolution.

```
SUB MyKillTime
  CONST BiosTimerLocation% = &H46C
  DIM TickCounter%: TickCounter% = 3
  DIM OldTick%: OldTick% = PEEK(BiosTimerLocation%)
  DIM NewTick%
  DEF SEG = 0
  DO WHILE TickCounter%
    NewTick% = PEEK(BiosTimerLocation%)
    IF OldTick% <> NewTick% THEN
      TickCounter% = TickCounter% - 1
      OldTick% = NewTick%
    END IF
  LOOP
  EXIT SUB
END SUB
```

CombineUpperAndLowerBytes

This function takes two bytes and combines them into a 16-bit integer. QBasic requires an “if” condition to prevent an overflow exception. This is used to read and combine two eight-bit port values into one 16-bit integer such as when reading analog inputs.

```
FUNCTION CombineUpperAndLowerBytes% (UpperByte%, LowerByte%)
  DIM Temp%
  IF UpperByte% AND &H80 THEN      ‘ Prevent overflow if negative
    Temp% = ((UpperByte% AND &H7F) * &H100) OR &H8000
  ELSE
    Temp% = UpperByte% * &H100     ‘ Shift left 8 bits
  END IF
  CombineUpperAndLowerBytes% = Temp% OR LowerByte%
END FUNCTION
```

GetUpperAndLowerBytes

This function takes a 16-bit integer and breaks it into two separate bytes. This is used to write analog outputs, write 16 digital outputs (two banks), write watchdog values, and write watchdog times.

```
SUB GetUpperAndLowerBytes (Combined%, Upper%, Lower%)
  Upper% = Combined% \ &H100       ‘ Shift right 8 bits
  Lower% = Combined% AND &HFF
END SUB
```

TROUBLESHOOTING AND TIPS

OVERVIEW

This appendix contains several procedures and notes to help you identify and resolve any troubles you may experience with a Pamux system. If implementing the suggestions in this appendix does not eliminate your difficulties, please contact Opto 22 Product Support (see Appendix D).

PRELIMINARY PROCEDURES

1. Check the 5V power at each Pamux mounting rack (check at the unit, not at the power supply). The voltage should be between 4.9 VDC and 5.1 VDC. The 5 Volt "-" terminal must not be grounded. It is best to adjust the voltage toward the high end of the 4.9–5.1 VDC range.
2. Check the +15V and -15V power at each analog Pamux mounting rack. Each power supply has a tolerance
   of ±0.25 VDC.
3. Ensure that the address, reset level, and watchdog jumpers are installed to meet your specifications. Each Pamux unit must have a unique address. All units must have the same reset level.
4. Use only flat-ribbon cable (see Appendix B). Do not use rolled-ribbon or bundled-wire cable.
5. Verify that the last Pamux unit has the appropriate terminator board (TERM1 or TERM2) installed and that the terminator board is properly connected to 5 VDC. See page 130 for selection criteria.

GENERAL TROUBLESHOOTING

1. It is a good idea to disable the watchdog timer while troubleshooting. On B4 and B5 boards, install jumpers 5 and 6 to do so. On B6 boards, install jumper 8 to disable the watchdog.
2. Apply power to the Pamux unit to be tested. If the power LED doesn't light, check the fuse on the mounting rack. Note that the power LED may be on the brain board or on the mounting rack.
3. Run a small program to continuously read one Pamux board. (The Opto 22 BBS includes utilities that will do this; search for PMXSCAN.ZIP or PAMSCN.EXE.) While communicating, does the watchdog LED go out? If not, check the AC28's base address jumpers. Does the respective address LED blink? If not, check the address jumpers on the Pamux board.
4. If a B4 or B5 board reads inputs properly but does not write outputs correctly, check the board's reset level (jumper 7).
5. The Pamux driver (used in PMXSCAN.EXE and PAMSCA.ZIP) will return an error code of -7 if the driver cannot gain access to a B6. Should this occur while testing, check the B6 board's 5V power, check its reset level (jumper 7), and make sure the terminator board is inserted and properly attached to 5V.
6. On the B6 board, if communication is working correctly but the board does not read proper values, check the ±15 VDC power. Remember that the ±15 VDC power supplies must be within ±0.25 VDC.

NOTES ON LEDS

1. The address LED on the B4, B5, and B6 boards will blink on when the board decodes a match between the address lines on the Pamux bus (see “Bus Pin Definition” in Chapter 4) and the address jumpers on the board. The state of the reset line has no effect on the address LED.
2. The watchdog LED on the B4 and B5 boards will stay on as long as no communication is taking place (the read and write lines on the Pamux bus are not strobing). Once the read or write line strobes, the watchdog LED will go off. If no other communication takes place, the watchdog LED will stay off for approximately 1.6 seconds and then turn back on. If the read or write line strobes before 1.6 seconds has elapsed, the watchdog LED will remain off for at least another 1.6 seconds. As long as the system is reading or writing faster than once every 1.6 seconds, the watchdog LED will remain off. The state of the reset line has no effect on the watchdog LED.
3. The access LED on the B6 brain board will light when the host device has access to the dual-port RAM and the B6 processor wants access. (See “Reading from and Writing to Analog Pamux Stations” in Chapter 4.)

USEFUL TIPS

- The microprocessor on the B6 constantly scans and converts analog channels, then transfers input data to the dual-port RAM (if it has access). Output data is read by the processor from the dual-port RAM. The value read is then converted to output information that analog modules translate into voltages or currents.
- Polling one B6 continuously may not give the brain board's processor much of a chance to update the RAM. Remember, while you have access, the brain board's processor cannot read from or write to the RAM.
- It is OK to access more than one channel once you have access, but if you spend too much time on one board, the data may no longer be fresh (for the reason stated above).
- After a power-up or reset condition, it is important to reconfigure the analog brain board by writing to the configuration registers. The B6 brain board clears all dual-port RAM locations upon a power-up or reset. Since the configuration register is a RAM location, it will also be cleared.

SPECIFICATIONS

PAMUX PRODUCT SPECIFICATIONS

AC28 ADAPTER CARD

Description	Pamux bus adapter card for the IBM PC/AT bus
Power requirements	5 VDC ±0.1 V @ 1.5 A
Operating temperature	0°C to 60°C
Relative humidity	0% to 95%, non-condensing
Connectors	50-pin male header connector
Range	Up to 500 feet total length

AC36 ADAPTER CARD

Description	Pamux bus adapter card for a TTL parallel port using the Intel multibus port configuration.
Power requirements	5 VDC ±0.1 V @ 1 A
Operating temperature	0°C to 70°C
Relative humidity	0% to 95%, non-condensing
Connectors	Two 50-pin male header connectors (accepts HH series ribbon cable)
Range	Up to 500 feet total length
Configuration jumper	Reset

B4 BRAIN BOARD

Description	32-point digital brain board
Power requirements	5 VDC ±0.1 V @ 0.5 A (5 VDC ±0.1 V @ 1.0 A for brain board plus terminator board; terminator board is installed on last station only)
Operating temperature	0°C to 70°C
Relative humidity	0% to 95%, non-condensing
Connectors	70-pin box connector to I/O mounting rack 50-pin male connector located on mounting rack to link to Pamux bus
Data rates	2.2 µsec bus cycle for 8 positions (read or write) at 500 feet
Range	Up to 500 feet total length Up to 16 B4 brain boards per Pamux bus
LED indicators	Address, watchdog
Configuration jumpers	Address, reset, watchdog

B5 BRAIN BOARD

Description	16-point digital brain board
Power requirements	5 VDC ±0.1 V @ 0.5 A (5 VDC ±0.1 V @ 1.0 A for brain board plus terminator board; terminator board is installed on last station only)
Operating temperature	0°C to 70°C
Relative humidity	0% to 95%, non-condensing
Connectors	50-pin female connector to I/O mounting rack Two 50-pin male connectors to Pamux bus
Data rates	2.2 µsec bus cycle for 8 positions (read or write) at 500 feet
Range	Up to 500 feet total length Up to 32 B5 brain boards per Pamux bus
LED indicators	Address, watchdog
Configuration jumpers	Address, reset, watchdog

B6 BRAIN BOARD

Description	16-point analog brain board
Power requirements	5 VDC ±0.1 V @ 0.5 A (5 VDC ±0.1 V @ 1.0 A for brain board plus terminator board; terminator board is installed on last station only)
Operating temperature	0°C to 70°C
Relative humidity	0% to 95%, non-condensing
Connectors	50-pin female connector to I/O mounting rack Two 50-pin male connectors to Pamux bus
Data rates	8.8 µsec bus cycle per analog channel (read or write) at 500 feet
Range	Up to 500 ft total length Up to 32 B6 brain boards per Pamux bus
LED indicators	Access, address, power
Configuration jumpers	Address, reset, watchdog

SNAP-B6 BRAIN BOARD

Description	Analog and digital brain
Power requirements	5 VDC ±0.1 V @ 0.5 A (5 VDC ±0.1 V @ 1.0 A for brain board plus terminator board; terminator board is installed on last station only)
Operating temperature	0°C to 70°C
Relative humidity	0% to 95%, non-condensing
Communications Interface	50-pin Pamux bus
Connectors	Adapter cable with two 50-pin connectors for Pamux bus or terminator board
Analog Read/Write Access Time	70 μsec per channel, 1.12 msec per 16 channels (channels accessed individually)
Digital Read/Write Access Time	2 μsec per channel, 2 μsec per 8 channels (channels accessed in banks of 8)
Update time	With digital functionality enabled: Analog channels updated every 20 msec Digital channels updated every 1.25 msec Without digital functionality enabled: Analog channels updated every 2 msec
Range: Multidrop	Up to 500 feet
LED indicators	ACC (Access), STS (Status), SEL (Address selected), WD (Watchdog), and RUN (Power On)
Configuration jumpers	Address Watchdog Reset Enable digital (B4) Analog configuration mode
Software Included	PamScan Utility (Dos and Windows 32-bit versions), used for troubleshooting and for configuring analog modules. Online help in PamScan tells how to use the utility. Pamux Drivers, used to allow PamScan and third-party software to talk to the AC28 adapter card.

TERM1 TERMINATOR BOARD

Description	Pamux bus terminator board for systems using unshielded flat ribbon cable. These systems are limited to 500 feet.
Power requirements	5 VDC ±0.1 V @ 0.5 A
Operating temperature	0°C to 70°C
Relative humidity	0% to 95%, non-condensing
Connector	50-pin female connector

TERM2 TERMINATOR BOARD

Description	Pamux bus terminator board for systems using shielded flat ribbon cable. These systems are limited to 250 feet.
Power requirements	5 VDC ±0.1 V @ 0.5 A
Operating temperature	0°C to 70°C
Relative humidity	0% to 95%, non-condensing
Connector	50-pin female connector

UCA4 UNIVERSAL COMMUNICATION ADAPTER

Description	Universal TTL-to-Pamux bus adapter card
Power requirements	5 VDC ±0.1 V @ 1.5 A
Operating temperature	0°C to 70°C
Relative humidity	0% to 95%, non-condensing
Connectors	50-pin male connector 40-pin male connector

PAMUX-COMPATIBLE I/O MOUNTING RACKS

Table B-1: Pamux-Compatible I/O Mounting Racks

Brain Board	Compatible Racks	I/O that Can Be Used
B4 (digital)	G4PB32H	32 channels of single-point I/O
	PB32HQ	8 channels of quad pak I/O (four points per module)
B5 (digital)	G4PB8H	8 channels of single-point digital I/O
	G4PB16H	16 channels of single-point digital I/O
	G4PB16HC	16 channels of single-point digital I/O
	G4PB16J/K	16 channels of integrated single-point digital inputs[1]
	G4PB16L	16 channels of integrated single-point digital outputs[1]
	PB4H	4 channels of single-point digital I/O
	PB8H	8 channels of single-point digital I/O
	PB16H	16 channels of single-point digital I/O
	PB16HC	16 channels of single-point digital I/O
	PB16J/K	16 channels of integrated single-point digital inputs[1]
	PB16L	16 channels of integrated single-point digital outputs[1]
	SNAPD4M	16 channels of SNAP digital I/O
B6 (analog)	PB4AH	4 channels of single-point analog I/O
	PB8AH	8 channels of single-point analog I/O
	PB16AH	16 channels of single-point analog I/O

[1]Rack features built-in I/O and therefore does not accept I/O modules.

PAMUX-COMPATIBLE ANALOG I/O MODULES

Table B-2 on the following page lists all the analog I/O modules currently available for Pamux systems.

Please note that this list is subject to change. If you do not see a particular type of module, or if you require further information on any module, contact Opto 22.

For the power requirements for analog modules used with B6 stations, refer to Table B-6.

Table B-2: Pamux-Compatible I/O Modules

Module	Type	Description	Compatible Racks[1]
AD2T	Analog Input	0 to 20 mA[2]	Standard Analog
AD3	Analog Input	4 to 20 mA	Standard Analog
AD3T	Analog Input	4 to 20 mA[2]	Standard Analog
AD4	Analog Input	ICTD	Standard Analog
AD5	Analog Input	Type J Thermocouple	Standard Analog
AD5T	Analog Input	Type J Thermocouple[2]	Standard Analog
AD6	Analog Input	0 to 5 VDC	Standard Analog
AD6HS	Analog Input	0 to 5 VDC High-Speed	Standard Analog
AD6T	Analog Input	0 to 5 VDC[2]	Standard Analog
AD7	Analog Input	0 to 10 VDC	Standard Analog
AD8	Analog Input	Type K Thermocouple	Standard Analog
AD8T	Analog Input	Type K Thermocouple[2]	Standard Analog
AD9T	Analog Input	0 to 50 mV[2]	Standard Analog
AD10T2	Analog Input	100-Ohm RTD[2]	Standard Analog
AD11	Analog Input	-5 to +5 VDC	Standard Analog
AD12	Analog Input	-10 to +10 VDC	Standard Analog
AD12T	Analog Input	-10 to +10 VDC[2]	Standard Analog
AD13T	Analog Input	0 to 100 mV[2]	Standard Analog
AD14T	Analog Input	10-Ohm RTD[2]	Standard Analog
AD15T	Analog Input	28 to 140 VDC[2]	Standard Analog
AD16T	Analog Input	0 to 5 Amp AC/DC[2]	Standard Analog
AD17T	Analog Input	Type R/S Thermocouple[2]	Standard Analog
AD18T	Analog Input	Type T Thermocouple[2]	Standard Analog
AD19T	Analog Input	Type E Thermocouple[2]	Standard Analog
AD20	Analog Input	Rate	Standard Analog

[1] Standard analog racks are the PB4AH, PB8AH, and PB16AH.
[2] Transformer-isolated module.

Table B-2: Pamux-Compatible I/O Modules *(cont'd)*

Module	Type	Description	Compatible Racks[1]
DA3	Analog Output	4 to 20 mA	Standard Analog
DA3T	Analog Output	4 to 20 mA[2]	Standard Analog
DA4	Analog Output	0 to 5 VDC	Standard Analog
DA4T	Analog Output	0 to 5 VDC[2]	Standard Analog
DA5	Analog Output	0 to 10 VDC	Standard Analog
DA6	Analog Output	-5 to +5 VDC	Standard Analog
DA7	Analog Output	-10 to +10 VDC	Standard Analog
DA8	Analog Output	0 to 20 mA	Standard Analog

[1] Standard analog racks are the PB4AH, PB8AH, and PB16AH.
[2] Transformer-isolated module.

PAMUX COMPATIBLE DIGITAL I/O MODULES

All 5-Volt logic I/O modules from the SNAP, G4, Classic, and QuadPak families are compatible with Pamux digital brainboards.

CABLES AND CONNECTORS

Cables

The Pamux bus connects to a host computer or other device via a 50-pin flat-ribbon cable. The maximum length of the Pamux bus is 500 feet.

Opto 22 provides a number of pre-made cables for the Pamux system, as listed in Table B-3. Contact Opto 22 Product Support at 800-835-6786 for more information.

Table B-3: Opto 22 Cables for Pamux

Cable Length (feet)	Opto 22 Part Number
1.5	HH1.5
2	HH2
4	HH4
6	HH6
8	HH8
10	HH10

If the cables offered by Opto 22 do not meet your needs, several other cables are also available. Table B-4 presents Pamux-compatible cables offered by vendors 3M and Alpha. Contact these suppliers directly for more information.

Table B-4: Third-Party Pamux-Compatible Cables

Cable Type	3M Part Number	Alpha Part Number
Regular	3365/50	3580/50 or 3583/50
Ground planed	3353/50 or 3584/50	3469/50 or 3476/50
Jacketed	3603/50	3589/50
Jacketed and ground planed	3517/50	3590/50

Pamux Cabling and Termination

Table B-5: Regular (Unshielded) and Jacketed Cable

Cable Type	Max. Distance	Recommended Terminator
Regular (flat ribbon)	500 ft.	TERM1
jacketed (plastic jacket)	500 ft.	TERM1

Note: The jacketed ribbon cable is covered with a vinyl/plastic cover that is used simply to prevent abrasion or mechanical damage to the cable. The vinyl/plastic cover does not provide any RFI/EMI shielding.

Table B-6: Shielded or Ground Planed Cable

Cable Type	Max. Distance	Recommended Terminator
Ground Planed	250 ft.	TERM2
Jacketed and Ground Planed	250 ft.	TERM2

Note: The "ground planed ribbon cable" is also known as "shielded ribbon cable." The shielded cable is used to prevent RFI/EMI problems.

Table B-7: Bundled or Rolled Cable

Cable Type	Max. Distance	Recommended Terminator
Bundled or Rolled	N/A	N/A
(Not recommended for use with any Opto 22 products)		

Connectors

The connectors listed in Table B-5 will work with any of the ribbon cables above.

Table B-8: Pamux Connectors

Manufacturer	Connector Part Number
3M	3425-7000
Circuit Assembly	CA-50IDSB

POWER SUPPLIES

Each non-terminated Pamux station requires a power supply of 5 VDC at 0.5 A. The final Pamux station on a bus requires an additional 0.5 A for a TERM1 or TERM2 terminator board, bringing the total current requirements for terminated stations to 1 A. To assure proper operation, the voltage at each Pamux station should be between 4.9 and 5.1 VDC.

In addition, the analog I/O modules at each B6 analog station require additional power supplies of +15 VDC and -15 VDC (±0.25 VDC). The requirements for current for these analog Pamux stations depend on the quantity and type of modules used. Table B-6 on the following page lists the current required for each type of analog I/O module.

Although switching power supplies are typically acceptable, Opto 22 recommends that linear power supplies be used to power Pamux systems. Linear power supplies offer tight regulation (typically ±0.05%) and low noise (typically 5 mV peak-to-peak).

The following manufacturers provide linear power supplies that meet Pamux requirements. Contact these vendors directly for more information.

Condor Electronics
580 Weddell Drive
Sunnyvale, CA 94089
408/745-7141

Power-One Linear Products
740 Calle Plano
Camarillo, CA 93012
800/678-9445

Sola/Heavy Duty
1717 Busse Road
Elk Grove Village, IL 60007
800/879-7652

Note: Always check all power supply polarities before powering up any system.

Table B-9: Current Requirements for Pamux-Compatible Analog Modules

Module	Description	+15 VDC Current (mA)	-15 VDC Current (mA)
Input Modules			
AD2T	0 to 20 mA	35	35
AD3	4 to 20 mA	3	7.5
AD3T	4 to 20 mA	35	35
AD4	ICTD	16	11
AD5	Type J Thermocouple	17	12
AD5T	Type J Thermocouple	45	45
AD6	0 to 5 VDC	16	11
AD6HS	0 to 5 VDC High-Speed	16	11
AD6T	0 to 5 VDC	35	35
AD7	0 to 10 VDC	16	11
AD8	Type K Thermocouple	17	12
AD8T	Type K Thermocouple	45	45
AD9T	0 to 50 mV	45	45
AD10T2	100-Ohm RTD	45	45
AD11	-5 to +5 VDC	12	8
AD12	-10 to +10 VDC	12	8
AD12T	-10 to +10 VDC	35	35
AD13T	0 to 100 mV	45	45
AD15T	28 to 140 VDC	35	35
AD16T	0 to 5 Amps AC/DC	35	35
AD17T	Type S Thermocouple	30	30
AD18T	Type T Thermocouple	30	30
AD19T	Type E Thermocouple	30	30
AD20	Rate	25	15
Output Modules			
DA3	4 to 20 mA	16	1
DA4	0 to 5 VDC	17	1
DA4T	0 to 5 VDC	35	35
DA5	0 to 10 VDC	17	1
DA6	-5 to +5 VDC	17	1
DA7	-10 to +10 VDC	17	1
DA8	0 to 20 mA	20	10

TEMPERATURE CONVERSION ROUTINES

OVERVIEW

Readings from analog input modules are returned as raw counts, nominally in the range of 0–4095. To convert this data to temperatures in degrees C, the algorithms in this appendix must be used. To accommodate the non-linearity of the temperature devices, some of these algorithms require various coefficients.

Throughout these algorithms, VALUE% is a decimal number representing the value read from the module. Value is:

- less than 0 when the module is below zero scale;
- equal to 0 when the module is at zero scale;
- equal to 4095 when the module is at full scale;
- greater than 4095 when the module is above full scale.

The data returned from a direct reading of an analog input will include an offset of 4096 to accommodate over- and under-range indications. If you are using the Pamux driver, this offset is subtracted automatically. If you are not using the driver, you must first subtract 4096 before using the reading in an algorithm.

For example, if a B6 reads the data from an analog input as 4110, VALUE% is 4110 - 4096 = 14, indicating the input is slightly above zero scale. If the B6 reads the data as 4082, VALUE% is 4082 - 4096 = -12, indicating the input is slightly below zero scale.

Temperature Reporting on the SNAP-B6

Customers who have used Classic B6 brains will notice a difference in the new SNAP-B6: how the brain reports temperature values. Classic B6 brains return a non-linear raw count between 0 and 4095, which you must linearize to derive temperature.

The SNAP-B6, however, does the linearization on the brain and returns a temperature value in degrees C multiplied by 10. For example, room temperature would be returned as 250 (25^{o} C x 10).

ALGORITHMS

These algorithms should be used for all Pamux-compatible analog input modules that produce data to be converted to temperatures. The algorithms are listed by input module.

AD4 ICTD

TEMP = (0.08262 * VALUE%) - 188.4

AD5, AD5T TYPE J THERMOCOUPLE

TEMP = (VALUE% - A0) * A1 + A2

where A0, A1, and A2 depend on VALUE% and can be derived from the following table:

VALUE%	A0	A1	A2	Temp Range (°C)
< 1	104	0.1923067	20.15	-20 to -1
1 to 161	0	0.1863354	0.10	0 to 30
162 to 354	161	0.1813472	30.19	31 to 65
355 to 551	354	0.1776649	65.10	66 to 100
552 to 867	551	0.1740506	100.15	101 to 155
868 to 1766	867	0.1724137	155.10	156 to 310
1767 to 2546	1766	0.1730769	310.00	311 to 445
2547 to 2895	2546	0.1719197	445.05	446 to 505
2896 to 3191	2895	0.1689189	505.10	506 to 555
3192 to 3464	3191	0.1654411	555.10	556 to 600
3465 to 3743	3464	0.1612903	600.20	601 to 645
3744 to 3901	3743	0.1572327	645.17	646 to 670
> 3901	3901	0.1546391	670.10	> 670

Accurate to ±0.2°C from -20°C to 700°C.

AD8, AD8T TYPE K THERMOCOUPLE

TEMP = (VALUE% - A0) * A1 + A2

where A0, A1, and A2 depend on VALUE% and can be derived from the following table:

VALUE%	A0	A1	A2	Temp Range (°C)
< 96	0	0.3167899	-99.6	< -70
96 to 198	95	0.2892961	-69.6	-70 to -40
199 to 348	198	0.2675585	-39.8	-39 to 0
349 to 578	348	0.2518454	0.3	1 to 58
579 to 909	578	0.2478090	57.9	59 to 140
910 to 1365	909	0.2541073	140.2	141 to 256
1366 to 1870	1365	0.2456905	256.2	257 to 380
1871 to 3084	1870	0.2405867	380.1	381 to 672
3085 to 3621	3084	0.2456271	671.8	673 to 804
3622 to 4095	3621	0.2532714	803.8	805 to 924
4096 to 4524	4095	0.2611331	924.0	925 to 1036
4525 to 4874	4524	0.2685714	1035.9	1037 to 1130
4875 to 5170	4874	0.2770270	1130.0	1131 to 1212
5171 to 5422	5170	0.2858277	1211.9	1213 to 1284
> 5422	5422	0.2959973	1283.9	> 1285

Accurate to ±0.4°C from -100°C to 1250°C.

AD10T2 100-OHM PLATINUM RTD

A 100-ohm platinum RTD (reference DIN 43 760, alpha = 0.00385) must be used with the AD10T2 for the following conversion algorithms to be accurate.

VALUE%	Formula	Temp Range (°C)

< 2111	A = (VALUE% * 0.09474182) - 50
	TEMP = A + (0.000156 * A^2) - (0.0156 * A) - 1.11
		-50 to 150

2111 < 4094	A = (VALUE% - 2111) * 0.1008065 - 100
	TEMP = A + (0.000156 * A^2) + 248.45
		150 to 350

4095 < 6218	A = (VALUE% - 4095) * 0.1082863 - 115
	TEMP = A + (0.00017 * A^2) + 462.76
		350 to 580

> 6218	A = (VALUE% - 6219) * 0.118525 - 13
	TEMP = A + (0.000188 * A^2) + 711.565
		580 to 850

Accurate to ±0.15°C from -50°C to 580°C, to ±0.2°C from 580°C to 850°C.

AD17T TYPE R OR TYPE S THERMCOUPLE

This thermocouple input can be used as either Type R or Type S, depending on the wire it is connected with. The algorithm for both types of thermocouples is the same:

B = VALUE% * 2.43663
TEMP = A0 + (A1 * B) + (A2 * B^2) +(A3 * B^3) + (A4 * B^4)

where A0, A1, A2, A3, and A4 depend on VALUE% and can be derived from the tables below.

When used as a Type R thermocouple:

VALUE%	Coefficients	Temp Range (°C)

< 740	A0 = 0
	A1 = 0.1625144
	A2 = -2.045438 x 10-5
	A3 = 2.540494 x 10-9
	A4 = -1.17679 x 10-13
		-50 to 237

≥ 740	A0 = 46.67453
	A1 = 0.1117991
	A2 = -2.565926 x 10-6
	A3 = 5.347317 x 10-11
	A4 = 0
		237 to 1768

When used as a Type S thermocouple:

VALUE%	Coefficients	Temp Range (°C)

< 479	A0 = 0
	A1 = 0.1641405
	A2 = -2.024176 x 10-5
	A3 = 2.784973 x 10-9
	A4 = -1.41721 x 10-13
		-50 to 167

≥ 479	A0 = 30.1319
	A1 = 0.1215561
	A2 = -2.752449 x 10-6
	A3 = 6.475822 x 10-11
	A4 = 0
		167 to 1768

AD18T TYPE T THERMOCOUPLE

B = (VALUE% * 3.951286) - 5602.92
TEMP = A0 + (A1 * B) + (A2 * B^2) +(A3 * B^3) + (A4 * B^4)

where A0, A1, A2, A3, and A4 depend on VALUE% and can be derived from the following table:

VALUE%	Coefficients	Temp Range (°C)

< 1418	A0 = 0
	A1 = 0.02383709
	A2 = -2.987884 x 10-6
	A3 = -7.194581 x 10-10
	A4 = -1.004194 x 10-13
		-200 to 0

≥ 1418	A0 = 0
	A1 = 0.0256613
	A2 = -6.195487 x 10-7
	A3 = 2.218164 x 10-11
	A4 = -3.55009 x 10-16
		0 to 400

AD19T TYPE E THERMOCOUPLE

B = (VALUE% * (36988.85 / 4095)) - 5236.65
TEMP = A0 + (A1 * B) + (A2 * B^2) +(A3 * B^3) + (A4 * B^4)

where A0, A1, A2, A3, and A4 depend on VALUE% and can be derived from the following table:

VALUE%	Coefficients	Temp Range (°C)

< 367	A0 = 0
	A1 = 0.01572665
	A2 = -1.21021 x 10-6
	A3 = -1.95778 x 10-10
	A4 = -1.66963 x 10-14
		-200 to 34

367 to 3783	A0 = 0
	A1 = 0.01702253
	A2 = -2.209724 x 10-7
	A3 = 5.480931 x 10-12
	A4 = -5.766989 x 10-17
		34 to 400

> 3783	A0 = 19.66945
	A1 = 0.01420774
	A2 = -5.184451 x 10-8
	A3 = 5.636137 x 10-13
	A4 = -1.564634 x 10-18
		400 to 1000

PRODUCT SUPPORT

If you have any questions about this product, contact Opto 22 Product Support Monday through Friday, 8 a.m. to 5 p.m. Pacific Time.

Phone:	800-TEK-OPTO (835-6786) 951-695-3080
Fax:	951-695-3017
E-mail:	support@opto22.com
Opto 22 Web site:	www.opto22.com

When calling for technical support, be prepared to provide the following information about your system to the Product Support engineer:

- Software and version being used
- Controller firmware version
- PC configuration
- A complete description of your hardware and operating systems, including:
  — jumper configuration
  — accessories installed (such as expansion daughter cards)
  — type of power supply
  — types of I/O units installed
  — third-party devices installed (e.g., barcode readers)
- Specific error messages seen

GLOSSARY

AC28

A high-speed adapter card that connects the Pamux bus to IBM PC/AT or compatible computers. The AC28 is compatible with computers that feature a standard 8-MHz ISA bus. Each AC28 can access up to 512 points of I/O. Four AC28s can be installed in one PC, supporting up to 2,048 points of I/O.

AC36

An adapter card that provides a Pamux bus interface for a TTL parallel port. It is compatible with parallel-port devices for MULTIBUS, STD bus, and VME bus products. Each AC36 can access up to 512 points of I/O.

adapter card

A printed circuit board, often installed within a computer, used to transfer data between a bus and a device.

analog

Describes data that can assume a continuous range of values, such as current, voltage, or pressure readings. Opposite of *digital*.

analog point

An input or output with an analog value representing, for example, a value of 0 to 5 volts or a temperature. Pamux analog modules feature 12-bit resolution.

API

Application program interface, a function in a programming library. The Pamux driver provides APIs that allow easy access to the Pamux bus. Microsoft Windows also provides APIs for various purposes, such as accessing serial ports and displaying message boxes.

B4

An addressable digital brain board that can control up to 32 input or output points in distributed I/O applications. Any combination of Pamux B4 brain boards may be linked on a single Pamux bus to control up to 512 points of analog and digital I/O.

B5

An addressable digital brain board that can control up to 16 input or output points in distributed I/O applications. Any combination of Pamux B5 brain boards may be linked on a single Pamux bus to control up to 512 points of analog and digital I/O.

B6

An addressable analog brain board that can control up to 16 input or output points in distributed I/O applications. Any combination of Pamux B6 brain boards may be linked on a single Pamux bus to control up to 512 points of analog and digital I/O. The B6 includes an on-board microprocessor that continually scans all I/O points on the mounting rack, performs necessary conversions, and then updates a dual-port RAM. The host computer transfers data along the Pamux bus by reading from or writing to the dual-port RAM.

bank

A group of eight digital I/O channels. A 16-channel digital I/O mounting rack with a B5 brain board has two banks. If the brain board is at address 20 and the AC28 card is at base I/O address 100, then the two banks are at I/O addresses 120 and 121. Banks and points are used together to access I/O points on a board at a particular address.

base address

The starting I/O address for programmable registers, used as the reference address for all other I/O addresses.

baud rate

The rate of transmission of serial communication data, expressed in bits per second (bps). For example, a device with a baud rate of 9,600 can transmit and/or receive data at 9,600 bits per second.

bit

A single binary digit (0 or 1).

brain board

An interface card that connects an analog or digital I/O mounting rack to a communication bus, such as the Pamux bus.

bus

A single common cable used to connect all devices on a system. The Pamux bus is a 50-pin flat-ribbon cable.

byte

A group of eight bits; an eight-bit binary number. For example, 10011011.

channel

See *point.*

digital

Describes data that can assume only two values: on or off, 1 or 0, true or false. Opposite of *analog*. Also called *discrete.*

digital point

An input or output that is either on or off, 1 or 0, true or false. Also called *discrete point.*

driver

A software program that provides instructions for transferring data between an application program and a peripheral device. The Pamux driver provides an interface that allows a user to program a Pamux system.

DLL

Dynamic link library, a library of executable functions that can be accessed by one or several Windows applications. Libraries may also be provided in static rather than dynamic form.

flat-ribbon cable

See *ribbon cable.*

handle

A number assigned by an operating system or driver to a resource (such as an AC28 card) as a means of identifying it. The operating system or driver may assign sequential numbers to resources to keep track of them.

ICTD

Integrated circuit temperature detector, a probe whose current output is proportional to absolute temperature. ICTDs are highly repeatable and easy to use because they don't require resistance-measuring circuity, high-precision voltage amplifiers, or cold-junction compensation.

I/O

Input/output, the transfer of data to or from a computer system involving communication devices, operator interfaces, and/or data acquisition and control interfaces.

I/O channel

See *point.*

I/O module

A device that provides an interface between signals received from "real world" field devices and the logic signals used in computers and controllers. For example, a thermocouple input module can convert a millivolt signal from a thermocouple into a numeric value that can be interpreted by a host computer. A digital output module can send a signal to turn a field device (such as a motor) on or off.

Most I/O modules feature one point of I/O; an exception is quad pak modules, which feature four points of I/O.

I/O mounting rack

A device into which I/O modules and/or brain boards can be installed.

I/O point

See *point.*

ISA

Industry Standard Architecture, the most common bus architecture on the mother board of DOS-based computers.

link library

A .LIB file used during linking to resolve references to APIs in a DLL. Also called *import library.*

module

See *I/O module*.

mounting rack

See *I/O mounting rack.*

noise

Extraneous electrical signals. Noise can be produced by such external sources as power lines, generators, motors, transformers, and electrical storms. It can also result from internal sources, such as resistors, capacitors, and semiconductors.

Pamux

A high-speed, high-density distributed I/O system that accommodates both digital and analog brain boards and I/O modules. Pamux supports up to 32 stations containing up to 512 I/O points. A Pamux bus can extend up to 500 feet from a host computer or other programming device. Pamux is ideal for low-noise applications, such as robotics and numerical control.

point

A single input/output data access location. An I/O point can accept either input data (read from a field device) or output data (to be transmitted to a field device). Most I/O modules feature one point of I/O; an exception is quad pak modules, which feature four points of I/O. Also called *channel.*

port

A communication connection on a computer or controlling device through which a process gains access to a network or bus.

quad pak module

An Opto 22 I/O module that features four points of discrete I/O in one package.

resolution

The smallest increment of a signal that can be detected by a system. For example, 12-bit resolution describes data accurate to the 12th bit, which implies a possible change of one part in 4,096 (2^{12}) or 0.0244 percent.

ribbon cable

A flat cable in which the conductors are arranged side by side. Also called *flat-ribbon cable.*

RTD

Resistance temperature detector, a metallic probe used to measure temperature based on its coefficient of resistivity.

TERM1

A terminator board for the Pamux bus. The final Pamux brain board on the bus must be terminated. The TERM1 is for use with systems using unshielded flat ribbon cable. These systems are limited to 500 feet.

TERM2

A terminator board for the Pamux bus. The final Pamux brain board on the bus must be terminated. The TERM2 is for use with systems using shielded flat ribbon cable. These systems are limited to 250 feet.

thermocouple

A temperature sensor that includes a junction of two different metals. Temperature can be derived from the voltage produced at the contact point of the metals.

UCA4

A general-purpose adapter card used to connect any TTL device to the Pamux bus. Its purpose is to allow a user to build a custom interface to a Pamux system.

watchdog

A process that monitors a system for activity and registers an error if a set interval has elapsed.

INDEX