



ACCES I/O PRODUCTS INC  
10623 Roselle Street, San Diego, CA 92121  
TEL (619)550-9559 FAX (619)**550-7322**

---

# **MODEL RAD128**

# **USER MANUAL**

## **Notice**

The information in this document is provided for reference only. ACCES does not assume any liability arising out of the application or use of the information or products described herein. This document may contain or reference information and products protected by copyrights or patents and does not convey any license under the patent rights of ACCES, nor the rights of others.

IBM PC, PC/XT, and PC/AT are registered trademarks of the International Business Machines Corporation.

Printed in USA. Copyright 1995 by ACCES I/O Products Inc, 10623 Roselle Street, San Diego, CA 92121. All rights reserved.

## **Warranty**

Prior to shipment, ACCES equipment is thoroughly inspected and tested to applicable specifications. However, should equipment failure occur, ACCES assures its customers that prompt service and support will be available. All equipment originally manufactured by ACCES which is found to be defective will be repaired or replaced subject to the following considerations.

## **Terms and Conditions**

If a unit is suspected of failure, contact ACCES' Customer Service department. Be prepared to give the unit model number, serial number, and a description of the failure symptom(s). We may suggest some simple tests to confirm the failure. We will assign a Return Material Authorization (RMA) number which must appear on the outer label of the return package. All units/components should be properly packed for handling and returned with freight prepaid to the ACCES designated Service Center, and will be returned to the customer's/user's site freight prepaid and invoiced.

## **Coverage**

First Three Years: Returned unit/part will be repaired and/or replaced at ACCES option with no charge for labor or parts not excluded by warranty. Warranty commences with equipment shipment.

Following Years: Throughout your equipment's lifetime, ACCES stands ready to provide on-site or in-plant service at reasonable rates similar to those of other manufacturers in the industry.

## **Equipment Not Manufactured by ACCES**

Equipment provided but not manufactured by ACCES is warranted and will be repaired according to the terms and conditions of the respective equipment manufacturer's warranty.

## **General**

Under this Warranty, liability of ACCES is limited to replacing, repairing or issuing credit (at ACCES discretion) for any products which are proved to be defective during the warranty period. In no case is ACCES liable for consequential or special damage arriving from use or misuse of our product. The customer is responsible for all charges caused by modifications or additions to ACCES equipment not approved in writing by ACCES or, if in ACCES opinion the equipment has been subjected to abnormal use. "Abnormal use" for purposes of this warranty is defined as any use to which the equipment is exposed other than that use specified or intended as evidenced by purchase or sales representation. Other than the above, no other warranty, expressed or implied, shall apply to any and all such equipment furnished or sold by ACCES.

## Table of Contents

<b>Chapter 1: Introduction</b> .....	<b><a href="#">1-1</a></b>
Description .....	<a href="#">1-1</a>
Specifications .....	<a href="#">1-3</a>
<b>Chapter 2: Installation</b> .....	<b><a href="#">2-1</a></b>
CD Installation .....	<a href="#">2-1</a>
3.5-Inch Diskette Installation .....	<a href="#">2-1</a>
Directories Created on the Hard Disk .....	<a href="#">2-2</a>
Calibration .....	<a href="#">2-4</a>
Installation .....	<a href="#">2-4</a>
Input/Output Pin Connections .....	<a href="#">2-5</a>
<b>Chapter 3: Software</b> .....	<b><a href="#">3-1</a></b>
General .....	<a href="#">3-1</a>
Command Structure .....	<a href="#">3-1</a>
Command Functions .....	<a href="#">3-6</a>
Error Codes .....	<a href="#">3-14</a>
<b>Appendix A: Application Considerations</b> .....	<b><a href="#">A-1</a></b>
Introduction .....	<a href="#">A-1</a>
Balanced Differential Signals .....	<a href="#">A-1</a>
RS485 Data Transmission .....	<a href="#">A-3</a>
<b>Notice</b> .....	<b>iii</b>
<b>Warranty</b> .....	<b>iv</b>

## List of Figures

<b>Figure 1-1:</b> RAD128 Block Diagram .....	<a href="#">Page 1-6</a>
<b>Figure A-1:</b> Typical RS485 Two-Wire Multidrop Network .....	<a href="#">Page A-3</a>

## List of Tables

<b>Table 2-1:</b> 50 Pin Connector Assignments (J1 on circuit board) .....	<a href="#">Page 2-5</a>
<b>Table 3-1:</b> RAD128 Command List .....	<a href="#">Page 3-5</a>
<b>Table A-1:</b> Connections Between Two RS422 Devices .....	<a href="#">Page A-1</a>
<b>Table A-2:</b> RS422 Specification Summary .....	<a href="#">Page A-2</a>

# Chapter 1: Introduction

## Features

- Remote Intelligent Module with Analog Inputs and Digital I/O Ports.
- Interfaces via Opto-Isolated RS485 to Host Computer at up to 57.6kbps.
- Eight Single-ended Analog Inputs, 4 Programmable Gains (max +/-10V) on a Channel-by-Channel Basis, 12-Bit Resolution A/D Converter.
- Eight Bits of TTL/CMOS-Compatible Digital I/O Configured on a Bit-by-Bit Basis as either 8 Inputs or 7 Outputs and 1 Input.
- Digital Outputs are Open Collector for External Loads up to 350mA at up to 50V.
- Protective NEMA4 Enclosure for Harsh Atmospheric and Marine Environments.
- Type 8031 Micro-controller with 32K x 8 bits RAM and 32k x 8 bits EEPROM..
- All Programming in Software, No Switches to Set. Jumpers (3) to By-Pass RS485 Opto-Isolators if Single Power Supply Operation is Desired.
- On-board 10MHz Crystal Clock and Three 16-Bit Counter/Timers.

## Description

---

RAD128 is an intelligent, 8-channel, analog-to-digital converter unit that communicates with the host computer via EIA RS485, or Half-Duplex, serial communications protocol. ASCII-based command/response protocol permits communication with virtually any computer system. RAD128 is one of a series of remote intelligent Pods called the "REMOTE ACCES Series". As many as 32 REMOTE ACCES Series Pods (or other RS485 devices) may be connected on a single two or four-wire multidrop RS485 network. RS485 repeaters may be used to extend the number of Pods on a network. Each unit has a unique address. Communication uses a master/slave protocol wherein the Pod talks only if questioned by the computer.

A type 8031 micro-controller (with 32k x 8 bits RAM, 32K bits non-volatile EEPROM, and a watchdog timer circuit) gives RAD128 the capability and versatility expected from a modern distributed control system. RAD128 contains CMOS low-power circuitry, an optically-isolated receiver/transmitter, and power conditioners for local and external isolated power. It can operate at baud rates up to 57.6 Kbaud and distances up to 4000 feet with low-attenuation twisted-pair cabling, such as Belden #9841 or equivalent. Data collected by the Pod can be stored in local RAM and accessed later through the computer's serial port. This facilitates a stand-alone Pod mode of operation.

All programming of RAD128 is in ASCII-based software. ASCII-based programming permits you to write applications in any high-level language that supports ASCII string functions and you can use REMOTE ACCES Series Pods with virtually any computer that has an RS485 port.

The module, or Pod, address is programmable from 00 to FF hex and whatever address is assigned is stored in EEPROM and used as the default address at the next Power-ON. Similarly, the baud rate is programmable for 1200, 2400, 4800, 9600, 14400, 19200, 28800, and 57600. The baud rate is stored in EEPROM and used as default at the next Power-ON.

## Analog Inputs

RAD128 uses a 12-bit successive approximation analog-to-digital converter (A/D). It accepts up to eight single-ended analog inputs with ranges of  $\pm 10\text{V}$ ,  $\pm 5\text{V}$ ,  $0\text{-}10\text{V}$ , and  $0\text{-}5\text{V}$ . All inputs are internally protected up to  $\pm 16.5\text{V}$ . A/D conversions may be initiated either by software command, or by an on-board programmable timer.

The RAD128 provides full compatibility with AIM-16 analog input expansion cards. Each AIM-16 provides capability to connect sixteen differential inputs, allowing as many as 128 differential inputs when you use a RAD128 with a full complement of AIM-16s. The  $\pm 12\text{VDC}$  power required by the AIM-16 must be supplied locally. Alternatively, you can use the AIM-16P-S06, which requires only  $+5\text{VDC}$  of local power. The RAD128 is also compatible with the LVDT-8 and SSH-08 signal conditioning cards, however, the  $+12\text{VDC}$  power for these must be supplied locally.

## Digital I/O

Eight bits of TTL/CMOS-compatible digital I/O, (8 input or 7 outputs and 1 input) are provided. These can be software configured on a bit-by-bit basis as either inputs or outputs. When used as outputs, a maximum of seven bits are available. Each output provides compliance with user-supplied voltages up to  $50\text{ VDC}$  and the maximum current per output bit is  $350\text{mA}$ . When all seven-bits are used as outputs, there is a maximum total current of  $650\text{mA}$ .

There are an additional eight bits which are used strictly for digital output. The setup software has the option of installing operating firmware supporting either the AIM-16 external analog multiplexer or more digital bits (output only). When the RAD128 is used with an AIM-16 Multiplexer card then output bits GN0-GN2 and SEL0-SEL3 are used for gain and channel select. The final output, OP4, is available for general use. Each output has the ability to drive 10 LSTTL loads.

## Counter/Timers

Three 16-bit Counter/Timers are provided on the RAD128. Counter/Timer 0 is enabled by a digital input and may be clocked either by the output of Counter 1 or by an external source up to  $10\text{Mhz}$ . This counter is not committed on the card and its clock, gate, and output lines are available at the I/O connector.

Counter/Timers 1 and 2 are concatenated to form a 32-bit counter/timer. Counter 1 may be enabled by a digital input and is clocked by an on-board, crystal-controlled  $10\text{Mhz}$  oscillator. The output of Counter 1 provides the clock for Counter 2 and is also available at the I/O connector. The output of Counter 2 is connected to input T0 of the microprocessor and is available for custom firmware options.

## Watchdog Timer

The on-board watchdog timer resets the Pod if the micro-controller firmware halts or the regulated VCC drops below  $4.75\text{ VDC}$ . The micro-controller may also be reset by an external manual pushbutton connected to /PBRST.

## Specifications

---

### Serial Communications Interface

- Serial Port: Opto-isolated Matlabs type LTC491 Transmitter/Receiver. Compatible with RS485 specification. Up to 32 drivers and receivers allowed on the line. I/O bus programmable from 00 to FF hex (0 to 255 decimal). Whatever address is assigned is stored in EEPROM and used as default at next Power-On.
- Asynchronous Data Format: 7 data bits, even parity, one stop bit.
- Input Common Mode Voltage: 300V minimum (opto-isolated). If opto-isolators are by-passed: -7V to +12V.
- Receiver Input Sensitivity:  $\pm 200$  mV, differential input.
- Receiver Input Impedance: 12K $\Omega$  minimum.
- Transmitter Output Drive: 60 mA, 100 mA short circuit current capability.
- Serial Data Rates: Programmable for 1200, 2400, 4800, 9600, 14400, 19200, 28800, and 57600 baud. Crystal oscillator provided.

### Analog Inputs

- Channels: Eight, single-ended with common ground.
- Voltage Ranges: Software selectable,  $\pm 5$ V,  $\pm 10$ V, 0-5V, 0-10V.
- Resolution: 12 binary bits.
- Accuracy: 0.025%  $\pm 1$  LSB (est).
- Gain Drift:  $\pm 5$  ppm/  $^{\circ}$ C.
- Input Impedance: 21K $\Omega$  Unipolar, 16K $\Omega$  Bipolar, 10M $\Omega$  with AIM-16.
- Coding: True binary for unipolar inputs and offset binary for bipolar inputs.
- Throughput: 10,000 conversions per second in foreground mode.  
6.67K conversions per second in background mode.  
Baud rate dependant in immediate mode (see programming chapter).
- A/D Type: Successive Approximation.
- Trigger Source: Software command or on-board programmable timer.



## Digital I/O

- Bits 0-6 are configured as input or output on an individual basis *except for bit 7 which is dedicated as input only and is floating.*
- Input Bits 0-7      Logic High:            +2.0V to +50V (5mA max at 50V in)  
                                 Logic Low:                -0.5V to +0.8V at 0.4 mA max.
- Output Bits 0-6      Open Collector        (on J1 pins 12, 14, 16, 31, 29, 27 and 25)  
                                 Logic-Low Output Current: 350 mA maximum per bit  
                                 Total sink current of 650 mA per 7 bit (package limitation)  
                                 Inductive kick suppression diode included in each circuit when a user supplied voltage is connected to the Application Voltage terminal on the connector (APPLV+ J1 pin 38).  
                                 High-Level Output Voltage: Compliant with up to 50VDC from user-supplied voltage. If no user supplied voltage exists, outputs pulled up to +5VDC via 10 k $\Omega$  resistors.

## Digital Outputs

- Eight bits, each with 10 LSTTL load drive capability.
- If you are not using a multiplexer such as the AIM-16, then GN0-GN2 (J1 pins 5, 9 & 11) and SEL0-SEL3 (J1 pins 13, 15, 17 & 19) are available for general purpose use.
- OP4 (J1 pin 7) is always available for general use.

## Programmable Timer

- Type:                    82C54-2 programmable interval timer.
- Counters:             Three 16-bit down counters, two permanently concatenated with 10 MHz clock as programmable timer. One counter is uncommitted.
- Output Drive:        4 mA at 0.4V (10 LSTTL loads).
- Input Gate:            TTL/DTL/CMOS compatible.
- Clock Input Frequency:      DC to 10 MHz.
- Active Count Edge:        Negative edge.
- Minimum Clock Pulse Width: 30 nS high/50 nS low.
- Timer Range:            2.5 MHz to < 1 pulse per hour.

## Environmental

- Operating Temperature Range: 0 °C. to 65 °C. (Optional -40 °C. to +80 °C.).
- Temperature De-rating: Based on the power applied, maximum operating temperature may have to be de-rated because internal power regulators dissipate some heat. For example, when 7.5VDC is applied, the temperature rise inside the enclosure is 7.3°C above the ambient temperature.

### Note

Maximum operating temperature can be determined according to the following equation:

$$V_{I(TJ=120)} < 22.5 - 0.2T_A$$

Where  $T_A$  is the ambient temperature in °C. and  $V_{I(TJ=120)}$  is the voltage at which the integral voltage regulator junction temperature will rise to a temperature of 120 °C.

(Note: The junction temperature is rated to 150 °C. maximum.)

For example, at an ambient temperature of 25 °C., the voltage  $V_I$  can be up to 17.5V.

At an ambient temperature of 100 °F. (37.8 °C.), the voltage  $V_I$  can be up to 14.9V.

- Storage Temperature Range: -20 °C. to +70 °C.
- Humidity: 5% to 95% RH non-condensing.
- Size: NEMA-4 Enclosure 4.53" long x 3.54" wide x 2.17" high.

## Power Required

- Power can be applied from the computer's +12VDC power supply for the opto-isolated section via the serial communication cable and from a local power supply for the rest of the unit. If you do not wish to use power from the computer, a central power supply may be used for the opto-isolated section.
- Local Power: 7.5 to 16 VDC @ 150 mA. (See box below.)
- Opto-Isolated Section: 7.5 to 25 VDC @ 40 mA. (Note: Due to the small amount of current required, voltage drop in long cables is not significant.)

### Note

If the local power supply has an output voltage greater than 16VDC, you can install a Zener diode in series with the supply voltage. The voltage rating of the Zener diode ( $V_Z$ ) should be equal to  $V_I - 16$  where  $V_I$  is the power supply voltage. The power rating of the Zener diode should be  $\geq V_Z \times 0.12$  (watts). Thus, for example, a 24VDC power supply would require using an 8.2V Zener diode with a power rating of  $8.2 \times 0.12 \approx 1$  watt.

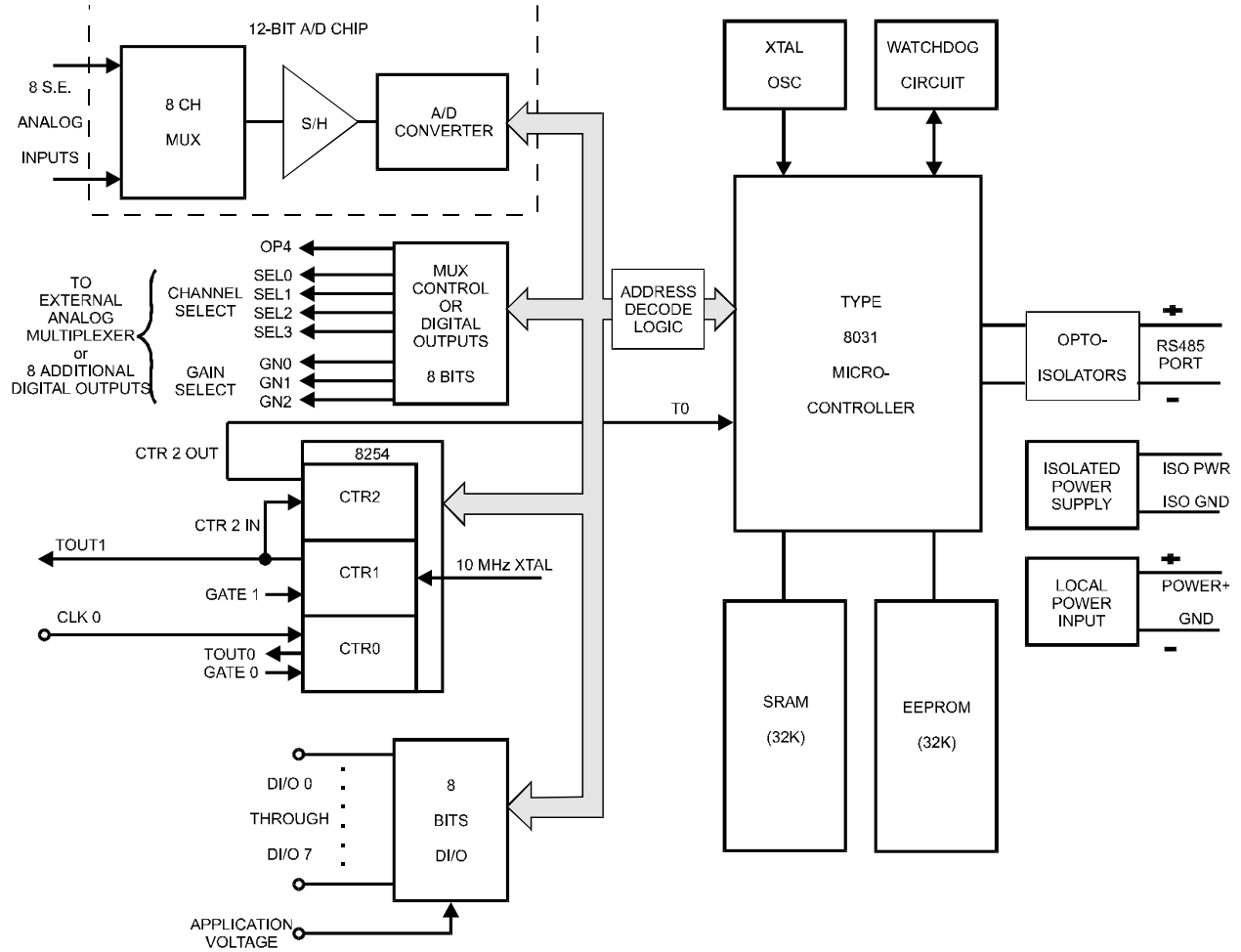


Figure 1-1: RAD128 Block Diagram

# Chapter 2: Installation

The software provided with this card is contained on either one CD or multiple diskettes and must be installed onto your hard disk prior to use. To do this, perform the following steps as appropriate for your software format and operating system. Substitute the appropriate drive letter for your CD-ROM or disk drive where you see d: or a: respectively in the examples below.

## CD Installation

---

### DOS/WIN3.x

- a. Place the CD into your CD-ROM drive.
- b. Type `D:Enter` to change the active drive to the CD-ROM drive.
- c. Type `INSTALLEnter` to run the install program.
- d. Follow the on-screen prompts to install the software for this card.

### WIN95/98/NT

- a. Place the CD into your CD-ROM drive.
- b. The CD should automatically run the install program after 30 seconds. If the install program does not run, click START | RUN and type d:install, click OK or press `Enter`.
- c. Follow the on-screen prompts to install the software for this card.

## 3.5-Inch Diskette Installation

---

As with any software package, you should make backup copies for everyday use and store your original master diskettes in a safe location. The easiest way to make a backup copy is to use the DOS DISKCOPY utility.

In a single-drive system, the command is:

```
D I S K C O P Y  A :  A : Enter
```

You will need to swap disks as requested by the system.

In a two-disk system, the command is:

```
D I S K C O P Y  A :  B : Enter
```

This will copy the contents of the master disk in drive A to the backup disk in drive B.

To copy the files on the master diskette to your hard disk, perform the following steps.

- a. Place the master diskette into a floppy drive.
- b. Change the active drive to the drive that has the diskette installed. For example, if the diskette is in drive A, type `A: Enter`.
- c. Type `INSTALL Enter` and follow the on-screen prompts.

## **Directories Created on the Hard Disk**

---

The installation process will create several directories on your hard disk. If you accept the installation defaults, the following structure will exist.

### **[CARDNAME]**

Root or base directory containing the SETUP.EXE setup program used to help you configure jumpers and calibrate the card.

**DOS\PSAMPLES:** A subdirectory of [CARDNAME] that contains Pascal samples.

**DOS\CSAMPLES:** A subdirectory of [CARDNAME] that contains "C" samples.

**Win32\language:** Subdirectories containing samples for Win95/98 and NT.

### **WinRisc.exe**

A Windows dumb-terminal type communication program designed for RS422/485 operation. Used primarily with Remote Data Acquisition Pods and our RS422/485 serial communication product line. Can be used to say hello to an installed modem.

### **ACCES32**

This directory contains the Windows 95/98/NT driver used to provide access to the hardware registers when writing 32-bit Windows software. Several samples are provided in a variety of languages to demonstrate how to use this driver. The DLL provides four functions (InPortB, OutPortB, InPort, and OutPort) to access the hardware.

This directory also contains the device driver for Windows NT, ACCESNT.SYS. This device driver provides register-level hardware access in Windows NT. Two methods of using the driver are available, through ACCES32.DLL (recommended) and through the DeviceIOControl handles provided by ACCESNT.SYS (slightly faster).

## **SAMPLES**

Samples for using ACCES32.DLL are provided in this directory. Using this DLL not only makes the hardware programming easier (MUCH easier), but also one source file can be used for both Windows 95/98 and WindowsNT. One executable can run under both operating systems and still have full access to the hardware registers. The DLL is used exactly like any other DLL, so it is compatible with any language capable of using 32-bit DLLs. Consult the manuals provided with your language's compiler for information on using DLLs in your specific environment.

## **VBACCES**

This directory contains sixteen-bit DLL drivers for use with VisualBASIC 3.0 and Windows 3.1 only. These drivers provide four functions, similar to the ACCES32.DLL. However, this DLL is only compatible with 16-bit executables. Migration from 16-bit to 32-bit is simplified because of the similarity between VBACCES and ACCES32.

## **PCI**

This directory contains PCI-bus specific programs and information. If you are not using a PCI card, this directory will not be installed.

## **SOURCE**

A utility program is provided with source code you can use to determine allocated resources at run-time from your own programs in DOS.

## **PCIFind.exe**

A utility for DOS and Windows to determine what base addresses and IRQs are allocated to installed PCI cards. This program runs two versions, depending on the operating system. Windows 95/98/NT displays a GUI interface, and modifies the registry. When run from DOS or Windows3.x, a text interface is used. For information about the format of the registry key, consult the card-specific samples provided with the hardware. In Windows NT, NTioPCI.SYS runs each time the computer is booted, thereby refreshing the registry as PCI hardware is added or removed. In Windows 95/98/NT PCIFind.EXE places itself in the boot-sequence of the OS to refresh the registry on each power-up.

This program also provides some COM configuration when used with PCI COM ports. Specifically, it will configure compatible COM cards for IRQ sharing and multiple port issues.

## **WIN32IRQ**

This directory provides a generic interface for IRQ handling in Windows 95/98/NT. Source code is provided for the driver, greatly simplifying the creation of custom drivers for specific needs. Samples are provided to demonstrate the use of the generic driver. Note that the use of IRQs in near-real-time data acquisition programs requires multi-threaded application programming techniques and must be considered an intermediate to advanced programming topic. Delphi, C++ Builder, and Visual C++ samples are provided.

### **Findbase.exe**

DOS utility to determine an available base address for ISA bus , non-Plug-n-Play cards. Run this program once, before the hardware is installed in the computer, to determine an available address to give the card. Once the address has been determined, run the setup program provided with the hardware to see instructions on setting the address switch and various option selections.

### **Poly.exe**

A generic utility to convert a table of data into an nth order polynomial. Useful for calculating linearization polynomial coefficients for thermocouples and other non-linear sensors.

### **Risc.bat**

A batch file demonstrating the command line parameters of RISCTerm.exe.

### **RISCTerm.exe**

A dumb-terminal type communication program designed for RS422/485 operation. Used primarily with Remote Data Acquisition Pods and our RS422/485 serial communication product line. Can be used to say hello to an installed modem. RISCTerm stands for Really Incredibly Simple Communications TERMinal.

## **Calibration**

---

The setup software provided with the RAD128 supports the ability to check calibration and to write correction values into EEPROM so they are available automatically on power-up. Calibration checks need only be performed periodically, not every time power is cycled. The SETUP.EXE software calibration procedure requires the ability to ground an analog channel and provide a voltage source set at 4.5 VDC  $\pm$ 2 mV.

The SAMPLE1 program illustrates the procedure of recalling these values and adjusting the readings. The command descriptions of CAL? and BACKUP=CAL command description provides instruction on adding the calibration procedure to your own software.

## **Installation**

---

The RAD128 enclosure is a sealed, die-cast, aluminum-alloy, NEMA-4 enclosure that is easily mounted. Outside dimensions of the enclosure are: 4.53" long by 3.54" wide by 2.17" high. The cover incorporates a recessed neoprene gasket and the cover is secured to the body by four recessed M-4, stainless steel, captive screws. Two long M-3.5 X 0.236 screws are provided for mounting to the body. Mounting holes and cover-attaching screws are outside the sealed area to prevent ingress of moisture and dust. Four threaded bosses inside the enclosure provide for mounting the printed circuit card assemblies.

There are three jumper locations on the unit and their functions are as follows:

JP2, JP3, and JP4: Normally these jumpers should be in the "ISL" position. If you wish to by-pass the opto-isolators, then you can move these jumpers to the "/ISL" position.

## Input/Output Pin Connections

Electrical connections to the RAD128 are through a watertight gland that seals the wires terminated inside on a 50 pin screw-terminal block listed below. Connector pin assignments for the 50-pin connector are listed below:

Pin	Signal	Pin	Signal
1	N/C	2	N/C
3	CL0 - Clock 0	4	GATE0 - Timer Gate 0
5	GN0 - AIM-16 Gain, Bit 0 (LSB)	6	GATE1 - Timer Gate 1
7	OP4 - Digital Output, Bit 3	8	TOUT1 - Timer Out 1
9	GN1- AIM-16 Gain Bit 1	10	/INT0 - Interrupt 0
11	GN2 - AIM-16 Gain, Bit 2	12	DI/O0 - Digital I/O Bit 0
13	SEL0 - Mux Ch Select, Bit 4 (LSB)	14	DI/O1 - Digital I/O Bit 1
15	SEL1 - Mux Ch Select, Bit 5	16	DI/O2 - Digital I/O Bit 2
17	SEL2 - Mux Ch Select, Bit 6	18	DIGITAL GROUND
19	SEL3 - Mux Ch Select, Bit 7	20	VCC Regulated +5 DC Out
21	DIGITAL GROUND	22	CH07 - Analog Channel 7
23	DI7 - Digital Input Bit 7	24	CH06 - Analog Channel 6
25	DI/O6 - Digital I/O Bit 6	26	CH05 - Analog Channel 5
27	DI/O5 - Digital I/O Bit 5	28	CH04 - Analog Channel 4
29	DI/O4 - Digital I/O Bit 4	30	CH03 - Analog Channel 3
31	DI/O3 - Digital I/O Bit 3	32	CH02 - Analog Channel 2
33	TOUT0 - Counter 0 Output	34	CH01 - Analog Channel 1
35	ANALOG GROUND	36	CH00 - Analog Channel 0
37	ANALOG GROUND	38	APPLV+ Application Voltage
39	PWR+ Local Power	40	POWER GROUND
41	ISOV+ Isolation Voltage	42	ISOGND - ISOLATED GROUND
43	RS485(+) - RS485 Serial Port +	44	RS485(-) - RS485 Serial Port -
45	/PBRST - Push Button Reset	46	/RST - Reset Output
47	ANALOG GROUND	48	ANALOG GROUND
49	ANALOG GROUND	50	ANALOG GROUND

**Table 2-1:** 50 Pin Connector Assignments (J1 on circuit board)



## RAD128 Manual

Terminal markings and their functions are as follows:

**PWR+ and PWR GND:** These terminals are used to apply local power to the Pod from a local power supply. The voltage can be anywhere in the range from 7.5 VDC to 16 VDC. Higher voltage can be used, 24 VDC for example, if an external Zener diode is used to reduce the voltage applied to the RAD128. (See the Specification section of this manual to determine the Zener diode power rating required.)

**ISOV+ and ISOGND:** This is the power connection for the isolator section which may be supplied from the computer's +12VDC supply via a pair of wires on the RS485 network or from a central power supply. This power is independent of "local power". The voltage level can be from 7.5 VDC to 35 VDC. (An on-board voltage regulator regulates the power to +5 VDC.) RAD128 will require only about 7 mA of current and, thus, any loading effects on the computer power will be inconsequential.

### **Note**

If separate power is not available, ISOV+ and ISOGND must be jumpered to the "local power" terminals, which defeats the optical isolation.

**RS485+ and RS485-:** These are the terminals for RS485 communications.

**APPLV+:** Inductive suppression diodes are included in the APPLV circuit. This terminal is for the "application power" or the user provided voltage level to which the DI/O Port outputs are connected when controlling inductive loads. The application power level (APPLV) can be as high as 50 VDC.

To ensure that there is minimum susceptibility to EMI and minimum radiation, it is important that there be a positive chassis ground. Also, proper EMI cabling techniques (cable connect to chassis ground at the aperture, twisted pair wiring, and, in extreme cases, ferrite-level of EMI protection) must be used for input/output wiring.

CE-marked versions of RAD128 meet the requirements of EN50081-1:1992 (Emissions), EN50082-1:1992 (Immunity), and EN60950:1992 (Safety).

# Chapter 3: Software

## General

---

The RAD128 comes with ASCII-based software provided on CD. ASCII programming permits you to write applications in any high-level language that supports ASCII text string functions, allowing the "REMOTE ACCES" series modules to be used with virtually any computer that has an RS485 port.

The communication protocol has two forms: addressed and non-addressed. Non-addressed protocol is used when only one REMOTE ACCES Pod is to be used. Addressed protocol must be used when more than one REMOTE ACCES Pod is to be used. The difference is that an address command is sent to enable the specific Pod. The address command is only sent once during communication between the specific Pod and the host computer. It enables communication with that specific Pod and disables all other REMOTE ACCES devices on the network.

## Command Structure

---

All communication must be 7 data bits, even parity, 1 stop bit. All numbers sent to and received from the Pod are in hexadecimal form. The factory default baud rate is 9600 Baud. The Pod is considered to be in addressed mode any time its Pod address is not 00. The factory default Pod address is 00 (non-addressed mode).

### Addressed Mode

The address select command must be issued before any other command to the addressed Pod. The address command is as follows:

"!xx[CR]" where xx is the Pod address from 01 to FF hex, and [CR] is Carriage Return, ASCII character 13.

The Pod responds with "[CR]". Once the address select command has been issued, all further commands (other than a new address select) will be executed by the selected Pod. *The addressed mode is required when using more than one Pod.* When there's only one Pod connected, no address select command is needed.

You can merely issue commands listed in the following table. Terminology used is as follows:

- a. The single lower case letter 'x' designates any valid hex digit (0-F).
- b. The single lower case letter 'b' designates either a '1' or '0'.
- c. The symbol '±' designates either a '+' or a '-'.
- d. All commands are terminated with [CR], the ASCII character 13.
- e. All commands are not case-sensitive, i.e., upper or lower case may be used.
- f. The symbol '\*' means zero or more valid characters (total msg length < 255 decimal).

**Concepts: Onboard Digital I/O**

The RAD128 has 15 digital outputs, and eight digital inputs. Seven of the outputs are on the same pins at the cable terminus as the eight inputs, and are considered to comprise the DI/O port, or Port 0. The second eight digital outputs are generally dedicated to the control of sub-multiplexer products such as the LVDT-8, AIM-16, and SSH-0x. These second eight outputs are considered the MUX port, or Port 1. Port 1 should be constrained to the Non-mux firmware installable from Setup program.

**Concepts: The Point**

A Point, as used in the RAD128, is a logical channel number. The RAD128 has eight analog inputs. However, using sub-multiplexer boards, each of the inputs might be expanded to 16 separate channels, yielding up to 128 channels total. To determine the Point number for any given channel, place the A/D channel number in the upper nibble of a byte, and the MUX channel number in the lower nibble, as follows:

D7	D6	D5	D4	D3	D2	D1	D0
0	AD2	AD1	AD0	MA3	MA2	MA1	MA

This can be done algorithmically by multiplying the A/D channel by 16 and adding the mux channel.

In C:           pointnum = (adchan << 4) + muxchan;

In Pascal:     pointnum = (adchan shl 4) or muxchan;

**Concepts: The Point List Buffer**

Because the Pod has up to 128 channels of Analog data, and each channel can be acquired at a different gain, and even at a different polarity, some setup is necessary to improve usability. The setup consists of telling the Pod what input range and polarity each channel is going to be acquired at, and which channels are going to be acquired in what order, using the Point List (PL) series of commands. The Point List is a list of the points to be acquired, along with all of the configuration information necessary to select polarity, range and mux gain. All of the commands to set PL entries take a point number (or the keyphrase "ALL"), and a 2-byte (4 hex characters) value describing the configuration for that point (or the keyphrase "DEFAULT"). The format of the 2-byte entry follows:

Polarity				Gain/Range Code				Analog Channel				Mux Channel			
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	BIP/ UNI	5/10	GN2	GN1	GN0	0	AD2	AD1	AD0	MA3	MA2	MA1	MA0

X:                             Don't care bits. State is ignored.

BIP/UNI:                     See following table

5/10:                         See following table:

BIP/UNI	5/10	Input Range (V)
0	0	0 to 5
0	1	0 to 10
1	0	±5
1	1	±10

GN2-GN0: The GN2-GN0 bits of Port 1 (MUX Control). Used to control gain control bits on AIM-16 and other sub-multiplexers. Refer to AIM-16 manual, fig. 3-1, for description of required control bits.

### Note

ALL numbers passed to and from the Pod are in hexadecimal.

Command	Description	Returns
PLnn=xxxx	Set Point List Entry for point number nn to xxxx. See <b>Concepts: The Point List Buffer</b> for more information on the point configuration setup required to use the PL commands.	[CR]
PLnn?	Reads the Point List entry for point number nn	XXXX[CR]
PLALL=xxxx-xxxx	Reserved for future development.	
PLALL?	Reads the Point Configuration for all points (0-127). Data is returned as XXXX XXXX XXXX ... XXXX XXXX[CR] with a total of 128 entries.	See Desc.
PLnn=DEFAULT	Sets point nn to default configuration (±5V, GN2-GN0 all zero) The default point list consists of AD channels 0-7 acquired at ±5V, and all further list entries set to AD channel 0.	[CR]
PLALL=DEFAULT	Sets all points to default configuration (±5V, GN2-GN0 all zero, entries 0-7=A/D channel 0-7, all other = AD channel 0)	[CR]
BACKUP=PL	Stores the point list into the onboard EEPROM, for power-off safekeeping. The point list is restored automatically every time the Pod is reset, or using the PLALL=BACKUP command.	[CR]
PLALL=BACKUP	Restores the EEPROM point list into the current point list. This command is executed automatically upon Pod reset.	[CR]
BACKUP=CAL: xxxx,xxxx	Writes scale and offset calibration constants for Y=MX+B calibration. Used by setup/calibration procedures.	[CR]
CAL?	Returns stored calibration constants. See sample program for example usage.	xxxx,xxxx [CR]
PROGRAM=	<b>Use with caution.</b> Loads new firmware into the Pod's EEPROM. Use ESC (ASCII 27) to abort.	n/a

(Vertical Bar)	Same as PROGRAM=, except used only after a reset to gain access to Pod.	[CR]
BAUD=nnn	Sets Baud rate	=:Baud:0x
POD=xx	Sets the Pod address to xx	=:Pod#xx
S=xxxx	Sets the Sample Rate of the A/D. Valid ranges are 00A2 through FFFF. Value passed is the desired divisor of the rate clock. 00A2 is 6666Hz, or 150us/sample. 039A is 1000Hz, or 1ms/sample. (11,059,200Hz / 12 / timebase = Hz sample rate.	[CR]
S?	Returns the Clock Divisor from EEPROM on the Pod, as above.	xxxx[CR]
!xx	Address Select Command, Selects the Pod at address 'xx' for communication.	[CR]
H*	“Hello” command. H followed by anything returns Pod name, revision, and copyright message.	See Desc.
N	Resend last message. Causes the Pod to re-transmit last message. Should be used upon parity-error, for example.	See Desc.
V	Reads the version number from the firmware.	x.xx[CR]
Mxx	Set bits as Inputs or Outputs. Each bit of xx controls a single bit of the DIO port. One’s set the corresponding port as output, zeroes set the port as input. Writing to a port marked as input will fail. Power-on default is ALL INPUTS.	[CR]
Mx+	Turns DI/O Port bit x to output	[CR]
Mx-	Turns DI/O Port bit x to input	[CR]
O0xx	Writes xx to the DI/O port (Port 0)	[CR]
O1xx	Writes xx to the digital output port (Port 1). Use with Nomux firmware variant.	[CR]
Ox+	Sets output bit x (0-F) to ONE. The DIO port bits are 0-6 (bit 7 is an input only), MUX bits are 8-F.	[CR]
Ox-	Sets output bit x to ZERO.	[CR]
I	This command reads the eight digital inputs (from Port 0)	xx[CR]
In	Single-Bit read from bit 0-7 (the Port Zero bits)	0[CR] or 1[CR]
CMxx[CR]	Writes byte value xx to the counter/timer control byte.	[CR]
CLn,xxxx[CR]	Loads hex value xxxx into counter/timer n.	[CR]

CRn[CR]	Reads the value of the current count in counter/timer n	Cnn=xxxx[CR] ] where nn is counter number 00.01, or 02.
ACnn <sub>1</sub> -nn <sub>2</sub> ,xxxx	Acquires point list entries nn <sub>1</sub> through nn <sub>2</sub> , with a total of xxxx conversions performed. xxxx must be equal to or less than 2710hex (which is 10000 decimal--all numbers must be presented to the Pod in hex). 10000 data samples is the maximum internal storage capacity of the standard Pod. (Optional firmware can increase this capacity. Contact the factory for further information.) Please note that this function does not return any data. Data is stored in internal SRAM storage for later retrieval using the R command.	[CR]
R*	This command reads the last buffer of data acquired using the AC command. The data is returned in CCXXXX CCXXXX ... CCXXXX CCXXXX[CR] format, where CC is the point number that was acquired, and XXXX is the analog conversion data. The total number of return values is identical to the XXXX parameter passed to the most recently issued AC command.	see desc.
Axxxx	Acquires a single channel and returns the data immediately via the serial port, bypassing the point list. This command takes as its sole argument the same 2-byte parameter passed to the PL series of commands. The channel specified in the parameter is acquired using the range, polarity, and mux gain specified. The data is returned immediately, unlike the AC command.	XXXX[CR]
Ann <sub>1</sub> -nn <sub>2</sub> ,xxxx	The A command works very similarly to the AC command. All parameters are the same. However, the A command runs in the foreground, occupying all of the attention of the Pod's micro-controller. This yields higher throughput, but sacrifices flexibility. The Pod acquires data at its fastest possible rate.	Data is returned upon completion as if 'R' had been issued

**Table 3-1:** RAD128 Command List

**Note**

Pod reset occurs upon power-up, programming process, or watchdog time-out.

## Command Functions

---

The following paragraphs give details of the command functions, describe what the commands cause, and give examples. Please note that all commands have an acknowledgement response. You must wait for a response from a command before sending another command.

### Configure Point List

PLnn=xxxx	Sets point list entry for point number nn to xxxx.
PLnn= DEFAULT	Sets Point nn to default configuration
PLALL=DEFAULT	Sets all points to default configuration

Note: Default configuration is  $\pm 5v$ , GN2-GN0 all zero.

### Read Point List

Plnn?	Reads the Point configuration for point number nn.
PLALL?	Reads the Point configuration for all points (0-127)

Note: Data is returned as xxxx xxxx xxxx ... xxxx xxxx[CR] with a total of 128 entries.

### Write Point List to EEPROM

BACKUP=PL	Stores the point list configuration in the onboard EEPROM, for power-off safekeeping.
-----------	---

Note: The point list configuration is automatically restored every time the Pod is reset, or when PLALL=BACKUP command is issued.

### Restore Point List

PLALL=BACKUP	Restores Point List configuration in EEPROM to the current point list.
--------------	--

Note: This command is executed automatically upon Pod reset.

### Write Calibration Parameters

BACKUP=CAL mmmm,bbbb	Write scale and offset calibration values for $Y=mX+b$ calibration in two's-complement hex as two four-digit numbers.
----------------------	---

This function stores the values required to adjust the measurement readings to agree with the last calibration. The SETUP.EXE program will measure and write these calibration parameters. The SAMPLE1 program illustrates using the CAL? command with the results of this function.

## Return Calibration Parameters

CAL?        Recalls the scale and offset calibration constants for  $Y=mX+b$  calibration stored using the BACKUP=CAL command.

This function recalls the stored calibration constants that may be used by your software to adjust the measurement readings to agree with the last calibration. The SAMPLE1 program includes an illustration of the use of this command.

Note:     $Cal\_Value = Original\_Value \times Scale + Offset$

## Set Sample Rate

S=xxxx     Set Sample Rate of the A/D Converter

This function sets the sample rate of the A/D converter. Valid values range from 00A2 to FFFF. The value passed is the desired divisor of the rate clock (11.0592 MHz). The equation to use in calculating the divisor is:

$$\text{Divisor} = [(1/\text{Rate}) - 22\mu\text{Sec}] * [\text{Clock}/12]$$

Examples:

Program the RAD128 for 1K samples per second.

```
SEND:        S0385
RECEIVE:     [CR]
```

Note:    The sample rate configured is stored in EEPROM on the Pod, and will be used as the default (power-on) sample rate. The factory default sample rate (100Hz) can be restored by sending "S0000" to the Pod.

## Read Sample Rate

S?        Returns the clock Divisor from EEPROM on the Pod.

## Configure Bits as Input or Output

Mxx        Configures digital bits as inputs or outputs.  
Mx+        Configures digital bit 'x' as output.  
Mx-        Configures digital bit 'x' as input.

These commands program the digital bits, on a bit-by-bit basis, as input or output. A "zero" in any bit position of the xx control byte designates the corresponding bit to be configured as an input. Conversely, a "one" designates a bit to be configured as an output. (Note: Any bit configured as an output can still be read as an input if the current value output is a "one".)



Examples:

Program even bits as outputs, and odd bits as inputs.

SEND: MAA  
RECEIVE: [CR]

Program bits 0-3 as input, and bits 4-7 as output.

SEND: MF  
RECEIVE: [CR]

### Read Digital Inputs

I Read 8 bits  
In Read bit number n

These commands read the digital input bits from the Pod. All byte responses are sent most-significant nibble first.

Examples:

Read ALL 8 bits.

SEND: I  
RECEIVE: FF[CR]

Read only bit 2.

SEND: I02  
RECEIVE: 1[CR]

### Write Digital Outputs

O0xx Write to all 8 digital output bits. (Port 0)  
Ox± Set bit x hi or low  
O1xx Write to AIM-16 output port. (Port 1)

### Warning

Use of the 'O1xx' command can disrupt data acquisition in process by changing MUX channel and/or gain during acquisition, if MUX firmware is installed.

These commands write outputs to digital bits. Any attempt to write to a bit configured as an input will fail. Writing to a byte or word wherein some bits are input and some are output will cause the output latches to change to the new value, but the bits which are inputs will not output the value until/unless they are placed in output mode.

Single bit commands will return an error (4) if an attempt is made to write to a bit configured as an input.

Writing a "one" (+) to a bit asserts the pull-down for that bit. Writing a "zero" (-) de-asserts the pull-down. Therefore, if the factory default +5V pull-up is installed, writing a one will cause zero volts to be at the connector, and writing a zero will cause +5 volts to be asserted.

Examples:

Write a one to bit 7 (set output to zero volts, assert the pull-down).

```
SEND:      O7+
RECEIVE:   [CR]
```

Write a zero to bit 2 (set output to +5V or user pull-up).

```
SEND:      O2-
or
SEND:      O02-
RECEIVE:   [CR]
```

Write zeros to bits 0-7.

```
SEND:      O00
RECEIVE:   [CR]
```

Write zeros to every odd bit.

```
SEND:      OAA
RECEIVE:   [CR]
```

## Counter/Timer Commands

CMxx[CR]:	Writes the byte value of xx to the counter/timer control byte.
CLn,xxx[CR]:	Loads the hex value of xxx to counter/timer n.
CRn[CR]:	Reads the hex value xxx of the current count in counter nn (00, 01, or 02)

## Read Firmware Revision Number

V: Read the firmware revision number

This command is used to read the version of firmware installed in the Pod. It returns "X.XX[CR]".

Example:

Read the RAD128 version number.

```
SEND:      V
RECEIVE:   1.00[CR]
```

### Note

The "H" command returns the version number along with other information. See "Hello Message" on next page.

### Resend Last Response

n            Resend last response

This command will cause the Pod to return the same thing it just sent. This command works for all responses less than 255 characters in length. Normally this command is used if the host detected a parity or other line fault while receiving data, and needs the data to be sent a second time.

The "n" command may be repeated.

Example:

Assuming the last command was "I", ask Pod to resend last response.

```
SEND:      n
RECEIVE:   FFFFFFF[CR];or whatever the data was
```

### Hello Message

H\*           Hello message

Any string of characters starting with "H" will be interpreted as this command. ("H[CR]" alone is also acceptable.) The return from this command takes the form (without the quotes):

```
"=Pod aa, RAD128 Rev rr Firmware Ver:x.xx ACCES I/O Products, Inc. W/MUX"
```

aa is the Pod address

rr is the hardware revision, such as "B1"

x.xx is the software revision, such as "1.00"

Mux support as "W/MUX" or "NOMUX"

Example:

Read the greeting message.

```
SEND:      Hello?
RECEIVE:   Pod 00, RAD128 Rev B1 Firmware Ver:1.00 ACCES I/O Products,
           Inc.NOMUX[CR]
```

**Configure Baud Rate** (When Shipped by Acces, the Baud Rate Is Set at 9600.)

BAUD=nnn            Program the Pod with a new baud rate

This command sets the Pod to communicate at a new baud rate. The parameter passed, nnn, is slightly unusual. Each n is the same digit from the following table:

Code	Baud Rate
0	1200
1	2400
2	4800
3	9600
4	14400
5	19200
6	28800
7	57600

Therefore, valid values for the command's "nnn" are 000, 111, 222, 333, 444, 555, 666, or 777. The Pod returns a message indicating it will comply. The message is sent in the old baud rate, not the new one. Once the message is transmitted, the Pod changes to the new baud rate. The new baud rate is stored in EEPROM and will be used even after power-reset, until the next "BAUD=nnn" command is issued.

Example:

Set the Pod to 19200 baud.

```
SEND:      BAUD=555
RECEIVE:   Baud:05[CR]
```

Set the Pod to 9600 baud.

```
SEND:      BAUD=333
RECEIVE:   Baud:03[CR]
```

## Configure Pod Address

POD=xx            Program the currently selected Pod to respond at address xx.

This command changes the Pod's address to xx. If the new address is 00, the Pod will be placed into non-addressed mode. If the new address is not 00, the Pod will not respond to further communications until a valid address command is issued. Hex numbers 00-FF are considered valid addresses. The RS485 specification allows only 32 drops on the line, so some addresses may be unused.

The new Pod address is saved in EEPROM and will be used even after power-down until the next "Pod=xx" command is issued. Note that, if the new address is not 00 (i.e., the Pod is configured to be in addressed mode), it is necessary to issue an address command to the Pod at the new address before it will respond.

The Pod returns a message containing the Pod number as confirmation.

Example:

Set the Pod address to 01.

```
SEND:      A=01
RECEIVE:   =:Pod#01[CR]
```

Set the Pod address to F3.

```
SEND:      A=F3
RECEIVE:   =:Pod#F3[CR]
```

Take the Pod out of addressed mode.

```
SEND:      A=00
RECEIVE:   =:Pod#00[CR]
```

## Address Select

!xx            Selects the Pod addressed 'xx'

### Note

When using more than one Pod in a system, each Pod is configured with a unique address. This command must be issued prior to any other commands to that particular Pod. This command needs to be issued only once prior to executing any other commands. Once the address select command has been issued, that Pod will respond to all other commands until a new address select command is issued.

## Enter a New Program

**PROGRAM** This command initiates transfer of a new program to the RAD128.

This command should be used carefully. If you accidentally issue a "PROGRAM=" command, ESC (ASCII 27) will restart the Pod as if power had been reset. This feature is designed to allow ACCES to provide field-upgrades to the RAD128 firmware, and, for advanced users, the opportunity to customize the firmware in the Pod. Documentation relating to use of this command is provided with the CD, or is available separately for a small fee.

| (Vertical Bar) Same as PROGRAM= except it is used only after a reset has occurred. Allows access to Pod to download new firmware.

## Special Acquisition Commands

**ACnn1-nn2,xxxx:** Acquires Point List entries nn1 through nn2, with a total of xxxx conversions performed.

xxxx must be equal to or less than 2710hex (which is 10000 decimal--all numbers must be presented to the Pod in hex). 10000 data samples is the maximum internal storage capacity of the standard Pod. (Optional firmware can increase this capacity. Contact the factory for further information.) Please note that this function does not return any data. Data is stored in the internal SRAM storage for later retrieval using the R command.

**R:** This command reads the last buffer of data acquired using the AC command.

The data is returned in CCXXXX CCXXXX ... CCXXXX CCXXXX[CR] format, where CC is the point number that was acquired, and XXXX is the analog conversion data. The total number of return values is identical to the XXXX parameter passed to the most recently issued AC command.

**Axxxx:** Acquires a single channel and returns the data immediately via the serial port, bypassing the point list.

This command takes as its sole argument the same 2-byte parameter passed to the PL series of commands. The channel specified in the parameter is acquired using the range, polarity, and mux gain specified. The data is returned immediately, unlike the AC command.

**Ann-nn,xxxx** The A command works very similarly to the AC command. All parameters are the same. However, the A command runs in the foreground, occupying all of the attention of the Pod's micro-controller. This yields higher throughput, but sacrifices flexibility. The Pod acquires data at its fastest possible rate.

## Error Codes

---

The following error codes can be returned from the Pod:

- 1: Invalid channel number (too large, or not a number. All channel numbers must be between 00 and 07, in hex).
- 3: Improper Syntax. (Not enough parameters is the usual culprit).
- 4: Channel number is invalid for this task (For example if you try to output to a bit that is set as an input bit, that will cause this error).
- 9: Parity error. (This occurs when some part of the received data contains a parity or framing error).

Additionally, several full-text error codes are returned. All begin with "Error, " and are useful when using a terminal to program the Pod.

Error, Unrecognized Command: {command received}[CR]  
This occurs if the command is not recognized.

Error, Command not fully recognized: {Command received}[CR]  
This occurs if the first letter of the command is valid, but the remaining letters are not.

Error, Address command must be CR terminated[CR]  
This occurs if the address command (!xx[CR]) has extra characters between the Pod number and the [CR].

# Appendix A: Application Considerations

## Introduction

---

Working with RS422 and RS485 devices is not much different from working with standard RS232 serial devices and these two standards overcome deficiencies in the RS232 standard. First, the cable length between two RS232 devices must be short; less than 50 feet at 9600 baud. Second, many RS232 errors are the result of noise induced on the cables. The RS422 standard permits cable lengths up to 4000 feet and, because it operates in the differential mode, it is more immune to induced noise.

Connections between two RS422 devices (with CTS ignored) should be as follows:

Device #1		Device #2	
Signal	Pin No.	Signal	Pin No.
Gnd	7	Gnd	7
TX <sup>+</sup>	24	RX <sup>+</sup>	12
TX <sup>-</sup>	25	RX <sup>-</sup>	13
RX <sup>+</sup>	12	TX <sup>+</sup>	24
RX <sup>-</sup>	13	TX <sup>-</sup>	25

**Table A-1:** Connections Between Two RS422 Devices

A third deficiency of RS232 is that more than two devices cannot share the same cable. This is also true for RS422 but RS485 offers all the benefits of RS422 plus allows up to 32 devices to share the same twisted pairs. An exception to the foregoing is that multiple RS422 devices can share a single cable if only one will talk and the others will all receive.

## Balanced Differential Signals

---

The reason that RS422 and RS485 devices can drive longer lines with more noise immunity than RS232 devices is that a balanced differential drive method is used. In a balanced differential system, the voltage produced by the driver appears across a pair of wires. A balanced line driver will produce a differential voltage from  $\pm 2$  to  $\pm 6$  volts across its output terminals. A balanced line driver can also have an input "enable" signal that connects the driver to its output terminals. If the "enable signal is OFF, the driver is disconnected from the transmission line. This disconnected or disabled condition is usually referred to as the "tristate" condition and represents a high impedance. RS485 drivers must have this control capability. RS422 drivers may have this control but it is not always required.



A balanced differential line receiver senses the voltage state of the transmission line across the two signal input lines. If the differential input voltage is greater than +200 mV, the receiver will provide a specific logic state on its output. If the differential voltage input is less than -200 mV, the receiver will provide the opposite logic state on its output. A maximum operating voltage range is from +6V to -6V allows for voltage attenuation that can occur on long transmission cables.

A maximum common mode voltage rating of  $\pm 7V$  provides good noise immunity from voltages induced on the twisted pair lines. The signal ground line connection is necessary in order to keep the common mode voltage within that range. The circuit may operate without the ground connection but may not be reliable.

Parameter	Conditions	Min.	Max.
Driver Output Voltage (unloaded)		4V	6V
		-4V	-6V
Driver Output Voltage (loaded)	LD and LDGND	2V	
	jumpers in	-2V	
Driver Output Resistance			50W
Driver Output Short-Circuit Current			$\pm 150$ mA
Driver Output Rise Time			10% unit interval
Receiver Sensitivity			$\pm 200$ mV
Receiver Common Mode Voltage Range			$\pm 7V$
Receiver Input Resistance			4KW

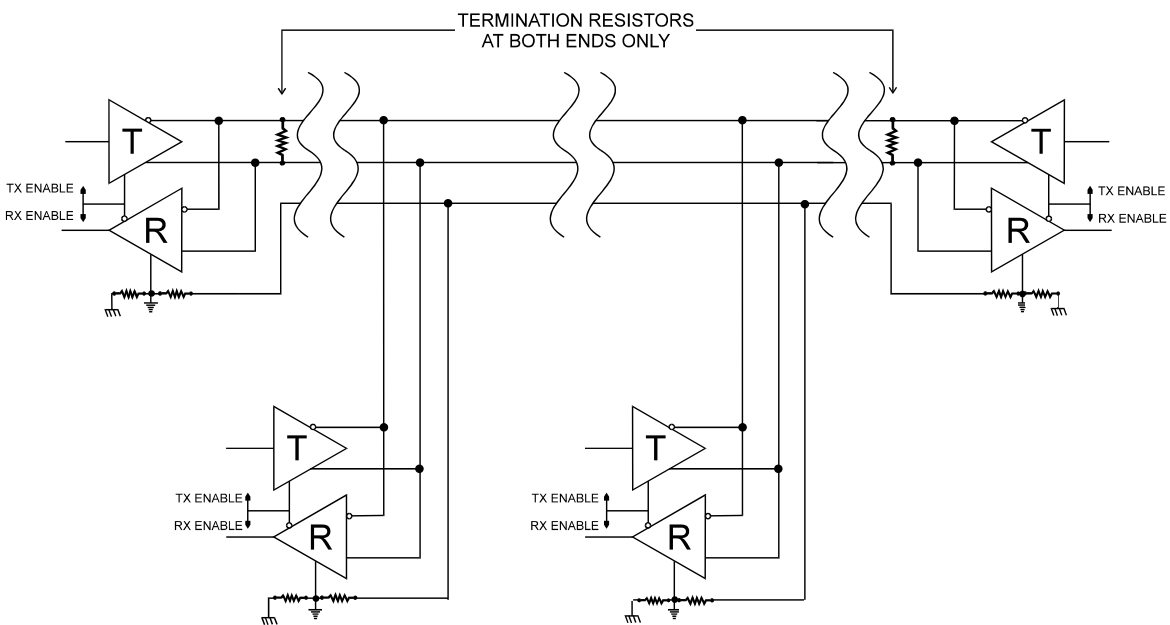
**Table A-2:** RS422 Specification Summary

To prevent signal reflections in the cable and to improve noise rejection in both the RS422 and RS485 mode, the receiver end of the cable should be terminated with a resistance equal to the characteristic impedance of the cable. (An exception to this is the case where the line is driven by an RS422 driver that is never "tri-stated" or disconnected from the line. In this case, the driver provides a low internal impedance that terminates the line at that end.)

## RS485 Data Transmission

The RS485 Standard allows a balanced transmission line to be shared in a party-line mode. As many as 32 driver/receiver pairs can share a two-wire party line network. Many characteristics of the drivers and receivers are the same as in the RS422 Standard. One difference is that the common mode voltage limit is extended and is +12V to -7V. Since any driver can be disconnected (or tri-stated) from the line, it must withstand this common mode voltage range while in the tristate condition.

The following illustration shows a typical multidrop or party line network. Note that the transmission line is terminated on both ends of the line but not at drop points in the middle of the line.



**Figure A-1:** Typical RS485 Two-Wire Multidrop Network



## Customer Comments

If you experience any problems with this manual or just want to give us some feedback, please email us at: [manuals@accessioproducts.com](mailto:manuals@accessioproducts.com). Please detail any errors you find and include your mailing address so that we can send you any manual updates.



10623 Roselle Street, San Diego CA 92121  
Tel. (619)550-9559 FAX (619)550-7322  
[www.accessioproducts.com](http://www.accessioproducts.com)