ACCES I/O PRODUCTS INC
10623 Roselle Street, San Diego, CA 92121
TEL (858)550-9559   FAX (858)550-7322

MODEL RIOD24

USER MANUAL

FILE: MRIOD24.E1

Notice

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

List of Figures

List of Tables

Chapter 1: Introduction

Features

- Opto-Isolated RS485 Serial Interface to Host Computer.
- 24-bit Digital I/O Programmable Bit-byBit, in 8-Bit Byes, or in 24-Bit Words.
- Digital Input Voltages up to 50V.
- Open Collector Digital Outpus for Loads up to 350mA.
- NEMA4 Enclosure for Harsh Atmospheric or Marine Environments.
- Type 8031 Microcontroller with 8K RAM and 8K EEPROM. (32K X 8 optional)
- All Programming in Software, No Switches or Jumpers to Set.
- 16-Bit Digital Input Software Counters.
- Change of State Flag Readable via the Serial Port.
- Digital Outputs May Be Either Level or Pulse.

Description

RIOD24 is an intelligent, 24-bit parallel digital I/O-to-computer interface unit and is packaged in a NEMA4 enclosure for remote installation in harsh environments. Communication with the host computer is via EIA RS485 half-duplex, serial communications protocol. ASCII-based command/response protocol permits communication with virtually any computer system. RIOD24 is one of a series of remote intelligent units called the "REMOTE ACCES" series. As many as 32 REMOTE ACCES Series pods (or other RS485 devices) may be connected on a single two-wire multidrop RS485 network.

A type 8031 microcontroller (with 8K x 8 RAM, 8K x 8 non-volatile EEPROM, and a watchdog timer circuit) gives RIOD24 the capability and versatility expected from a modern distributed control system. To accommodate special programs, the RAM and EEPROM can each be expanded to 32K x 8. The unit 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 at distances up to 5000 feet with low-attenuation twisted-pair cabling.

All programming of RIOD24 is in ASCII-based software and there are no switches or jumpers to set. (One exception to the foregoing is that you have the option of by-passing the optoisolators by re-locating three jumpers.) This permits you to write applications in any high-level language that supports ASCII string functions and you can use REMOTE ACCES series modules with virtually any computer.

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 and is stored in EEPROM and used as default at the next Power-ON.

The time base, used in all time-relevant operations is also software selectable between:

a. Digital input sample rate from 14 Hz to 1 KHz
b. Digital output square wave pulse from 7 Hz to 500 Hz

Digital inputs of up to 50V amplitude may be read individually, or in 8-bit bytes, or in 24-bit word groups. There are also digital input counters on each input. Selectable edges can be counted for up to 65,535 transitions. These counters support Read and Reset commands. Moreover, change-of-state flags can be set on any enabled input bits and can be read via the serial port. This is particularly useful in applications where it's necessary to detect contact closures or openings. This change-of-state detection capability is enabled on a bit-by-bit basis for all bits programmed for input.

Digital outputs may be programmed individually, or in 8-bit bytes, or in 24-bit words. These outputs may be latched, pulsed, or set to free-run for a prescribed period of time. The digital output drivers are open collector circuits that have 350 mA drive capability at a logic "low". The unit can comply with up to 50 VDC (voltage supplied by you) or, if no external voltage is supplied, outputs will be pulled up to 5 VDC.

The built-in watchdog timer resets the pod if, for some unexpected reason, the microcontroller "hangs up". Data collected by the pod can be stored in local RAM and accessed later through the computer's serial port. This facilitates stand-alone operation.

Specifications

Serial Communications Interface

• Serial Port:	Opto-isolated Matlabs type LTC485 Transmitter/Receiver. Compatible with RS485 specification. Up to 32 drivers and receivers allowed on line. Pod I/O bus programmable from 00 to FF hex (0-255 decimal). Whatever address is assigned is stored in EEPROM and used as default at next Power-On.
• Input Common Mode Voltage:	300V minimum (opto-isolated). If opto-isolators are by-passed: -7V to +12V.
• Receiver Input Sensitivity:	±200 mV, differential input.
• Receiver Input Impedance:	12KW minimum.
• Transmitter Output Drive Capability:	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.

Digital Inputs

• Number:	Up to 24. Can be programmed, on a bit-by-bit basis, on an 8-bit byte basis, or on a 24-bit word basis. In this latter case, there would be no capability for digital outputs.
• Sample Rate:	Programmable from 14 Hz to 1 KHz.
• Software Counters:	There are 16-bit software counters on all bits programmed to be inputs. These can be programmed to increment on either rising or falling edges.
• Change of State Detection:	Change-of-state flags can be set on any enabled input bits and can be read via the serial port.
• Logic Input Low:	-0.5V to +0.8V.
• Logic Input High:	+2.0V to +50.0V
• Low-level Input Current:	450 mA maximum.

Digital Outputs

• Number:	Up to 24. Can be programmed, on a bit-by-bit basis, on an 8-bit byte basis, or on a 24-bit word basis. In this latter case, there would be no capability for digital inputs.
• Type:	Outputs can be latched, pulsed, or set to free-run for a prescribed period of time. Pulsed outputs are square wave and programmable from 7 Hz to 500 Hz.
• Logic-Low Output Current:	350 mA maximum. (See box below.) Inductive suppression diode included in each circuit.
• High-Level Output Voltage:	Open Collector, Up to 50VDC compliance. If no user-supplied voltage, outputs are pulled up to 5 VDC via 10K? resistors .
• High-Level Output Leakage Current:	At VCE = 50 V, 100 mA.

- Maximum allowable current per output bit is 350 mA but, for each seven-bit group there is a cumulative maximum total of 650 mA. Output groups are bits 0-6, 7-12, 14-20, and 21-23 (decimal).

Environmental

• Operating Temperature Range:	0 °C. to 65 °C. (Optional -40 °C. to +80 °C.). See box below for temperature de-rating based on power voltage applied.

The power supply voltage level that you use will affect the maximum ambient temperature that can be tolerated. At higher power levels more heat will be generated by integral voltage regulators. (For example, when 7.5 VDC is applied, the temperature rise inside the enclosure is 7.3 °C. above the ambient temperature.) Thus, the maximum allowable ambient temperature may be reduced when power supply voltages greater than 7.5 VDC are used.

The equation to use to determine temperature de-rating is:

$$V_{I(TJ = 120)} < T_A \div 7$$

where T_A is the ambient temperature in °C. and $V_{I(TJ + 120)}$ is the input voltage at which the voltage regulator junction temperature will rise to a temperature of 120 °C. (Note: The maximum junction temperature rating of the voltage regulator used is 150 °C., so limiting to 120 °C. provides a safety margin.)

For example, at an ambient temperature of 25 °C., the voltage V_I can be up to 18.4V. At an ambient temperature of 100 °C., the voltage can be up to 16.6V.

- Storage Temperature Range: -50 °C. to +120 °C.
- Humidity: 5% to 95% non-condensing. Enclosure is designed to meet NEMA4 requirements.
- Size: NEMA4 enclosure 4.53" long by 3.54" wide by 2.17" high.

Power Required

- Power for the opto-isolated section can be applied from the computer's +12 VDC power supply via the serial communication cable. Power for the rest of the pod can be supplied by a local power supply.
- Opto-Isolated Section: 7.5 to 25 VDC @ 40 mA. (Note: Due to the small amount of current required, voltage drop in the communication cable will be inconsequential.)
- Local Power: 7.5 to 16 VDC @ 150 mA. See box below.

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 voltage rating of the zener diode should be ? V_Z x 0.12 watts. Thus, for example, a 24 VDC power supply would require use of an 8.2V zener diode with a power rating of 8.2 x 0.12 ? 1 watt

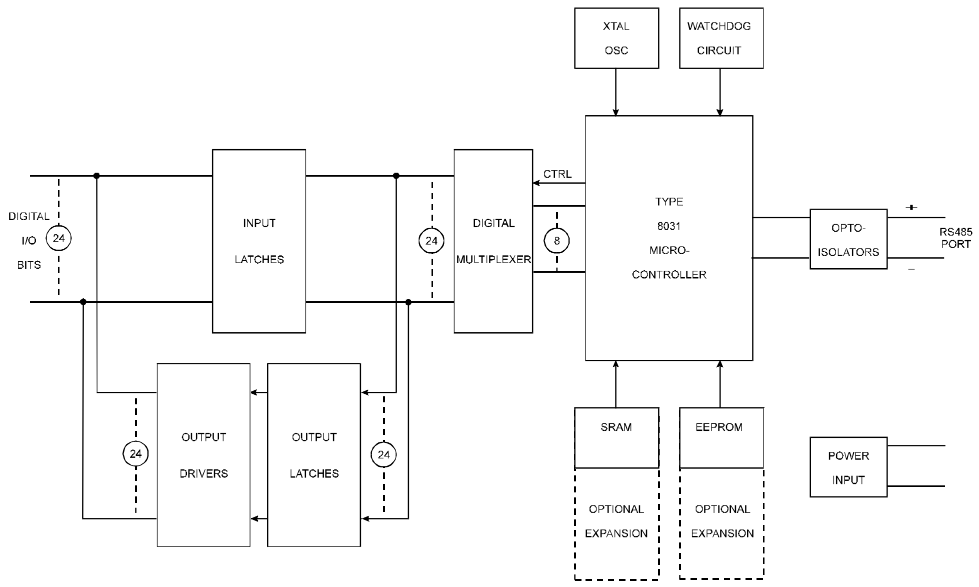

Figure 1-1: Block Diagram

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

1. Place the CD into your CD-ROM drive.
2. Type d:K to change the active drive to the CD-ROM drive.
3. Type installK to run the install program.
4. Follow the on-screen prompts to install the software for this card.

WIN95/98/NT/2000

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 K.
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:

```
diskcopy a: a:K
```

You will need to swap disks as requested by the system.
In a two-disk system, the command is:

```
diskcopy a: b:K
```

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:K.

c. Type installK 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.

Installing the Pod

The RIOD24 enclosure is a sealed, die-cast aluminum-alloy NEMA-4 enclosure that can be 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 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.

Option Selection

There are three jumper locations on the small processor board and their functions are as follows:

JP2, **JP3**, and **JP4**: Normally these jumpers should be in the "ISL" position. If you wish to bypass the opto-isolators (when using only one power supply for the unit), you can move these jumpers to the "/ISL" position. To access the jumpers, ensure power is off to the unit, observe ESD precautions and carefully pull the processor board straight up and off of the main PCB. The jumpers are located on the small board. These jumpers ship in the "ISL" position, so if you're going to use the non-isolated mode, with only one power supply, you need to move all three jumpers to the "/ISL" position. Carefully re-install the processor board onto the main board, ensuring it seats all the way down properly. *Do not force it!*

Electrical connections to the module are made via screw terminals. (See table on the next page.).

Screw Terminal Connections

Digital I/O connection points at the screw terminal connectors are as marked on underside of the printed circuit board TB2. The signal assignment to each of the screw terminals is as follows:

Terminal	Signal	Terminal	Signal
1	Bit 0	20	Bit 19
2	Bit 1	21	Bit 20
3	Bit 2	22	Bit 21
4	Bit 3	23	Bit 22
5	Bit 4	24	Bit 23
6	Bit 5	25	Gnd
7	Bit 6	26	/Int 1
8	Bit 7	27	/T0
9	Bit 8	28	/Int 0
10	Bit 9	29	Local Gnd
11	Bit 10	30	* Local Pwr+
12	Bit 11	31	RS485+
13	Bit 12	32	RS485-
14	Bit 13	33	** Isolator Pwr
15	Bit 14	34	Isolator Gnd
16	Bit 15	35	*** Appl'n Pwr
17	Bit 16	36	Appl'n Gnd
18	Bit 17	37	/PBRST
19	Bit 18		

Table 2-1: RIOD24 Screw Terminal Connections

*"Local" Power is power from a local power supply. The voltage can be anywhere from 7.5 VDC to 16 VDC. Higher local power, 24 VDC for example, can be used if an external zener diode is used to reduce the voltage applied to RIOD24. (See Temperature de-rating comments in the Specification section of this manual under "Power Required".)

**"Isolator" Power is used by the opto-isolator section of RIOD24 and should be independent of "Local Power". Isolator power should be connected between pins 33 and 34. That power can be the computer's +12V supply (via the serial communications cable) or from an isolated local power supply. The power level can be from 7.5 to 35 VDC and the isolator section will require only 7 mA of current. If a separate power supply is not available, then, with loss of some isolation, these isolator power terminals can be connected to the local power pins.

***"Application Power" is the voltage level to which digital inputs and outputs are connected through the loads. Open-collector Darlington amplifiers are used at the outputs. Inductive suppression diodes are included and the application power voltage level 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 RIOD24 meet the requirements of EN50081-1:1992 (Emissions), EN50082-1:1992 (Immunity), and EN60950:1992 (Safety).

Chapter 3: Software

General

You received ASCII-based software on CD for use with RIOD24. ASCII programming permits you to write applications in any high level language that supports ASCII string functions.

The communication protocol has two forms: addressed and non-addressed. Non-addressed protocol can be used when only one RIOD24 is in use. When more than one module (pod) is in use, addressed protocol must be used. The only difference is that an address command is sent to enable the specific pod. The address command is only sent once during communication between the pod and the host computer. It enables communication with that specific pod and disables all other pods on the network.

Command Structure

All communication must be 7 data bits, even parity, 1 stop bit. All numbers sent to or 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 "xxN[CR]" or "xxY[CR]" if an input change of state has occurred on enabled bits since the last "Y" or address command, or with "xxN[CR]" otherwise.

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.

Non-Addressed Mode

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. Wherever xx is used to designate a bit number, only 00-17 are valid.
f. All commands are case insensitive; i.e., upper or lower case letters can be used.
g. The symbol '*' means zero or more valid characters (total msg length <255 decimal).

Command List

Sxxxx	Set a new timebase. 039A<xxxx<=FFFF	CR
SCxxxx	Set a new timebase, and reset all free-run and pulse DOs	CR
MLxx	Configure bits 00-07 as input/output. 0=in, 1=out	CR
MMxx	Configure bits 08-0F as input/output. 0=in, 1=out	CR
MHxx	Configure bits 10-17 as input/output. 0=in, 1=out	CR
I	Read all 24 digital bits.	xxxxxxCR
Ixx	Read a single digital bit. (00<=xx<=17)	bCR
IL	Read digital bits 00-07	xxCR
IM	Read digital bits 08-0F	xxCR
IH	Read digital bits 10-17	xxCR
Oxxxx0xx	Output all 24 digital bits.	CR
Ox±	Output either high or low on bit x	CR
Oxx±	Output either high or low on bit xx	CR
Ox±xx	Output either high or low on bit x for time xx	CR
Oxx±xx	Output either high or low on bit xx for time xx	CR
OLxx	Output xx on bits 00-07	CR
OMxx	Output xx on bits 08-0F	CR
OHxx	Output xx on bits 10-17	CR
Oxx±xx	Output either high or low on bit xx for time xx	CR
Fxx,xx	Set Digital Output xx to free run with period xx	CR
Y	Read digital input COS bit and clear bit	Y or N
TLxx	Set bit 00-07 Mask for COS bit flag, 1=change will set COS	CR
TMxx	Set bit 08-0F Mask for COS bit flag	CR
THxx	Set bit 10-17 Mask for COS bit flag	CR
Dx±	Set digital input active state high or low on bit x	CR
Dxx±	Set digital input active state high or low on bit xx	CR
Cxx	Read digital input xx counter (counts each active pulse)	xxxxCR
or	Read pulse/free-run output xx counter and reload value	xxxxCR
Rxx	Reset digital input counter xx to 0000	CR
Rall	Reset all digital input counters to 0000	CR
V	Read the Firmware version number	x.xxCR
N	Resend last response	varies

H*	Greeting message: copyright, firmware version number	varies
BAUD=xxx	Set new baud rate. Each x is code number for new baud	=:Baud:0x
POD=xx	Set pod address to xx	varies
PROGRAM=	Begin process of uploading custom program to pod	special
D	Download historical storage of digital input data again	varies
FASTDATAL	Acquire bits 0-7 as fast as possible, then display	varies
FASTDATAM	Acquire bits 8-F as fast as possible, then display	varies
FASTDATAH	Acquire bits 10-17 as fast as possible, then display	varies

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 acknowledgment response. You must wait for a response from a command before another command is sent.

Set Time Base

Sxxxx	Set Time Base
Scxxxx	Set Time Base with Output Synchronization

This function sets the pod-global timebase which is used in all time-sensitive operations. Valid values range from 039A to FFFF. Any invalid value will result in the pod-default timebase of 2400 (10ms/100Hz).

039A corresponds to 1KHz, 2400 is 100Hz, and the longest timebase of FFFF corresponds to 14Hz. (11,059,200Hz / 12 / timebase = Hz rate of time base)

The Scxxxx variant sets the timebase as normal, then causes all free-run and pulse outputs to change on the next timebase tick.

Examples:

Program the RIOD24 to a 1msec timebase

```
SEND:        S039A
RECEIVE:     [CR]
```

Program the RIOD24 to a 50msec timebase, and synchronize outputs

```
SEND:        SC4800
RECEIVE:     [CR]
```

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

Application Note

If it is necessary to synchronize free-running outputs, the SCxxxx command will assist in this. First, configure all desired free-run bits as outputs. Then initialize each bit as free-run using the fxx,xx command. Now, by issuing an SCxxxx command, synchronize all the free-run outputs to toggle on the next time-base tick. Now, all free-run outputs with the same time-duration will toggle on the same tick. (20usec skew max per byte).

Configure Bits as Input or Output

Mlxx	Configure bits 0-7 as input/output
MMxx	Configure bits 8-F as input/output
Mhxx	Configure bits 10-17 as input/output

These functions program, on a bit-by-bit basis, the digital bits as input or output, in groups of eight. The MLxx command controls input/output on bits 0-7, MMxx controls bits 8-F, and MHxx controls bits 10-17. A 0 in any bit position of the xx control byte designates the corresponding bit as input. Conversely, a 1 designates a bit to be configured as output.

Examples:

Program even bits as outputs, and odd bits as inputs

```
SEND:       MLAA
RECEIVE:    [CR]
SEND:       MMAA
RECEIVE:    [CR]
SEND:       MHAA
RECEIVE:    [CR]
```

Program bits 0-7 as input, and bits 8-10 as output

```
SEND:       ML00
RECEIVE:    [CR]
SEND:       MMFF
RECEIVE:    [CR]
```

Note: Any bit configured as output can still be read as an input if the current value output is 1.

Read Digital Inputs

I	Read 24 bits
Ixx	Read bit number xx
IL	Read bits 0-7
IM	Read bits 8-F
IH	Read bits 10-17

These commands read the digital input bits from the pod. All byte or word wide responses are sent most-significant nibble first.

Examples:

Read ALL 24 bits.

SEND:	I
RECEIVE:	FFFFFF[CR]

Read only bit 17 (23 decimal, the highest bit on the card)

SEND:	I17
RECEIVE:	1[CR]

Read only bit 2

SEND:	I02
RECEIVE:	1[CR]

Read bits 8-F

SEND:	IM
RECEIVE:	FF[CR]

Write Digital Outputs

Oxxxxxx	Write to all 24 digital output bits
Ox±	Set bit x hi or low
Oxx±	Set bit xx hi or low
Ox±xx	Pulse bit x hi or low for time xx
Oxx±xx	Pulse bit xx hi or low for time xx
Olxx	Write to bits 0-7
Omxx	Write to bits 8-F
Ohxx	Write to bits 10-17
bx±xx	Identical to Ox±xx
bxx±xx	Identical to Oxx±xx

These commands write outputs to digital bits. Any attempt to write to a bit configured as 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 input.

Writing a one to a port asserts the pull-down. Writing a zero de-asserts the pull-down. Therefore, if the factory installed +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. If the factory installed pull-up has been removed, the user supplied pull-up will be asserted.

Pulsing a bit high or low uses the currently configured timebase. The bit will be set to 5V (-) or 0V (+) for a period equal to xx time-ticks, where one time-tick occurs every timebase. So, if the default timebase of 2400 (3916 dec, or 100Hz) is currently selected, and 32 (hex) was specified for a pulse duration, the bit will pulse high or low for 50 counts, or half-a second. (50x10ms=500ms=half-a-second)

Examples:

Write a one to bit 13 (set output to zero volts, assert the pull-down)

```
SEND:       O13+
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:       OL00
RECEIVE:    [CR]
```

Write zeros to every odd bit

```
SEND:       OAAAAAA
RECEIVE:    [CR]
```

Cause bit 7 to pulse to 0Volts for 20ms (assuming S039A was issued)

```
SEND:       O7+14
RECEIVE:    [CR]
```

Write zeros to bits 0-9, 13-17, and ones on all other bits

```
SEND:       O07FC00
RECEIVE:    [CR]
```

Generate a Square Wave Output

fxx,xx	Produce a Free-Running square wave on bit xx with period xx

This function will cause bit xx to change state every xx timebase units, effectively generating a squarewave with period xx, or frequency 1 div xx*2.

Examples

Start a 1Hz squarewave on bit 2. Bit two must be configured as output using ML, such as ML02. Also, timebase is assumed to have been configured to the default (2400, or 10ms), using S2400, or S0000.

```
SEND:       f02,32
RECEIVE:    [CR]
```

Note: It is possible to synchronize the starting edges of any free-running outputs by following the procedure outlined in the application note of the Set Timebase command. (Scxxxx)

Read Change-of-state

Y	Read COS bit.

The pod can set a change-of-state flag for any input that has been configured to do so. This command will read then reset that bit. Therefore, this command will always return "N[CR]" unless the T command has first been used to enable change-of-state detect for any given bit.

If a change-of-state has been detected since the last "Y" command (see note), the pod will return "Y[CR]" otherwise "N[CR]" will be returned.

Examples:

Read COS bit

```
SEND:       Y
RECEIVE:    N[CR]
```

Note: The address command for any given pod will also return "Y" or "N" and clear the Change-of-state flag in the pod.

Enable Change-of-State Detection

Tlxx	Set COS mask for bits 0-7
Tmxx	Set COS mask for bits 8-F
Thxx	Set COS mask for bits 10-17

These commands configure the bit-by-bit mask to enable change-of-state to set the COS flag on the pod for readback by the "Y" or address commands. If a one is set for a particular bit, that bit will set the COS flag if/when the bit changes state. A zero will disable change-of-state detection.

Examples:

Allow bit 13 and only bit 13 to set the COS flag

SEND:	TL00
RECEIVE:	[CR]
SEND:	TM00
RECEIVE:	[CR]
SEND:	TH08
RECEIVE:	[CR]

Allow a change of state on ANY bit to set the COS flag

SEND:	TLFF
RECEIVE:	[CR]
SEND:	TMFF
RECEIVE:	[CR]
SEND:	THFF
RECEIVE:	[CR]

Note: The COS Flag is read via either the "Y" command or a valid address command. The COS Flag is reset to FALSE by either command.

Selecting Which Edge Will Increment Counter

dx±	Set Digital input active state on bit x
dxx±	Set Digital input active state on bit xx

These commands allow you to set whether a rising or falling edge will increment the digital input counter; i.e., if all bits are set to rising edge, the digital input counter for any given bit will increment each time a rising edge is detected. "+" is rising edge, "-" is falling edge.

Examples:

Set bit 1 to rising edge active

SEND:	D1+
or	
SEND:	D01+
RECEIVE:	[CR]

Set bit 17 to falling edge active

SEND:	D17-
RECEIVE:	[CR]

Note: The digital input counters are read with the "cxx" command, and reset with the "rxx" command.

Read Digital Input Counter & Read Time Left on Pulse or Free Run Output

cxx	Read digital input counter xx
cxx	Read pulse or free-run status and reload counters

This command performs two duties, depending on whether the bit is configured as input or output. If the bit is configured as an input, this command will read how many times bit xx has changed to its active state (as configured with dx± or dxx±) since the last reset command (rxx).

If this bit is configured as an output, this command will indicate how much time (in time-base units) remains in a pulse-output or free-run-output before the pulse terminates. If the output is configured as free-run, it also returns what period is programmed into the output counter.

Input counters are configured as 16-bit counters. Counter content is provided most significant bit first.

Output return values are divided into two eight-bit counters. The first byte of the output counters is the time-remaining before the output pulse expires, the second byte is the originally-programmed period of free-run outputs. The second byte is zero for pulse outputs.

Examples:

Read digital input counter for bit #1

SEND:	C01
RECEIVE:	0213[CR];assuming 213hex edges since last reset

Read pulse output counter for bit #F

SEND:	C0F
RECEIVE:	1F00;1F is number of timebase units remaining before pulse expires, 00 indicates pulse, not free-run.

Read free-run output counter for bit 17

SEND:	c17
RECEIVE:	045F;04 is number of timebase units remaining before pulse expires, 5F indicates that the duration of each period is 5F units.

Note: It is possible to cause pulse and free-run outputs to prematurely expire or change state by using the SCxxxx variant of the Set TimeBase command. Also, it is possible to terminate pulse and free-run outputs without toggling the outputs by issuing a counter reset command (rxx) for each output to be terminated.

Reset Counter & Turn off Pulse or Free-Run Output

rxx	Reset digital input counter xx
rxx	Turn-off digital output pulse or free-run xx

This command is normally used to reset a digital input counter to zero. It can also be used to stop digital output pulses (Oxx±xx) or free-run outputs (fxx,xx)

Examples:

Reset digital input counter for digital input number 3

```
SEND:       r03
RECEIVE:    [CR]
```

Stop free-run digital output on output number 14

```
SEND:       r14
RECEIVE:    [CR]
```

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 RIOD24 version number

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

Note: The "H" command returns the version number along with other information.

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 character 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 that the last command was "I", ask pod to resend last response

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

Note: This command may not be used for the FASTDATA-L, -M, or -H commands, as they exceed the 255 character limit. Use the "D" command to perform the same task for these three commands.

Hello Message

H*	Hello message

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

"=Pod aa, RIOD-24 Rev rr Firmware Ver:x.xx ACCES I/O Products, Inc."

aa	is the pod address
rr	is the hardware revision, such as "B1"
x.xx	is the software revision, such as "1.00"

Example:

Read the greeting message

SEND:	Hello?
RECEIVE:	=Pod 00, RIOD-24 Rev B1 Firmware Ver:1.00 ACCES I/O Products, Inc.[cr]

Setting a New Baud Rate

BAUD=xxx Program the pod with a new baud rate

This command sets the pod to communicate at a new baud rate. The parameter passed, xxx, is slightly unusual. Each x 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 xxx 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 a new "BAUD=xxx" 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]
```

Programming Pod Address

A=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 many addresses will be unused.

The new pod address is saved in EEPROM and will be used even after power-down until the next "A=xx" command is issued. Note that, it 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]
```

Read and Store Digital Input Data

FASTDATAL	Read digital bits 0-7 as fast as possible
FASTDATAM	Read digital bits 8-F as fast as possible
FASTDATAH	Read digital bits 10-17 as fast as possible

These commands read the respective byte of digital input data and store it in SRAM at the fastest possible rate: 21 microseconds between samples. The commands will store as much data as the pod can hold: RAM size-1KByte. Typically, this is 7Kbytes of data, however, a 32k RAM version is optionally available, which would provide 31Kbytes of data storage.
Once the data has been stored, it is dumped to the serial port. The data is formatted into 3-byte chunks, followed by a space: xxxxxx xxx...etc. There are no carriage returns until the last byte has been sent.

All normal pod activities (parsing commands, receiving commands, pulse output countdowns, free-run generation, COS detect, etc.) STOP until the serial data is done transmitting. NOTHING else works until the data has been dumped.

Re-send Data

D	This will dump the last stored historical data to the serial port.

Data can be resent by issuing a "D" command. This will dump the last stored historical data to the serial port and can be used, for example, if line noise or similar problems are suspected.

This command should only be used after FASTDATAL, FASTDATAM, or FASTDATAH have been issued, because random data fills the buffer until one of these commands acquire data.

The format of the data is identical to the FASTDATAx commands. See the previous description of the FASTDATAx command for more information about the format and length of returned data.

Example:

Resend the data buffer

```
SEND:       D
RECEIVE:    xxxxxx xxxxxx xxxxxx ... xxxxxx for size of buffer.
```

Entering a New Program

PROGRAM=	This command initiates the transfer of a new program to the RIOD24.

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 RIOD24 firmware, and, for advanced users, the opportunity to customize the firmware in the pod. Documentation relating to the use of this command is provided with the CD, or is available separately for a small fee.

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 17, in hex. (0-24 decimal))
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, this error code will occur)
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 5000 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^+	24	RX^+	12
TX^-	25	RX^-	13
RX^+	12	TX^+	24
RX^-	13	TX^-	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 ±2 to ±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 ±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			±150 mA
Driver Output Rise Time			10% unit interval
Receiver Sensitivity			±200 mV
Receiver Common Mode Voltage Range			±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 "tristated" 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 tristated) 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.

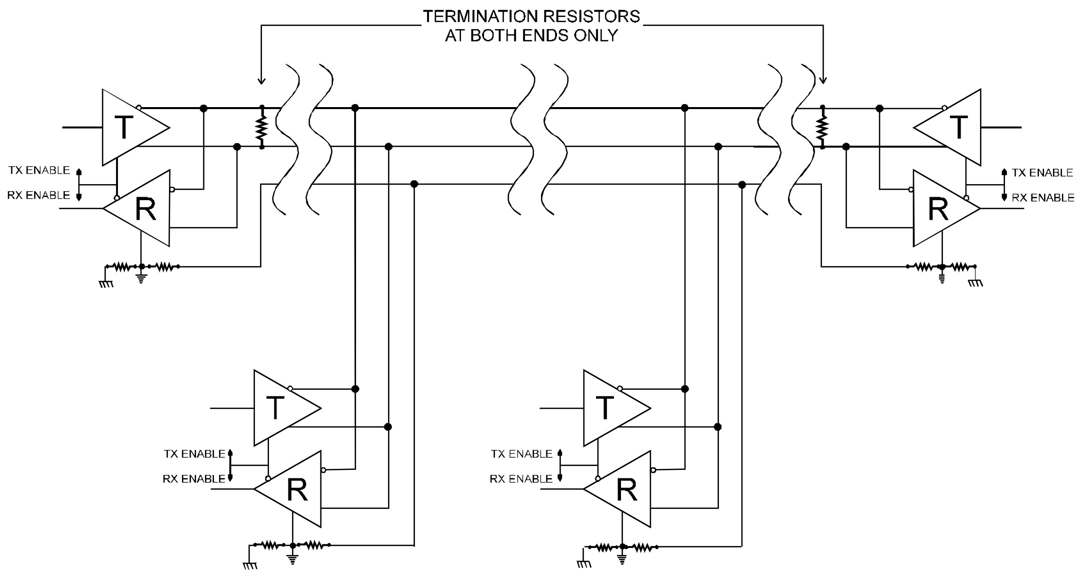

Figure A-1: Typical RS485 Two-Wire Multidrop Network

Customer Comments

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

ACCES
I/O PRODUCTS, INC.
10623 Roselle Street, San Diego CA 92121
Tel. (858)550-9559 FAX (858)550-7322
www.accesioproducts.com