

NOTICES

The information in this document is provided for reference only. Portwell I/O PRODUCTS INC 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 Portwell, nor the rights of others.

Printed in USA. Copyright All rights reserved.

TABLE OF CONTENTS

| | |
|---------------------------------------------------|-----|
| INSTALLATION | 1-1 |
| INSTALLING THE SOFTWARE | 1-1 |
| INSTALLING THE POD | 1-4 |
| PIN CONNECTIONS | 1-3 |
| FUNCTIONAL DESCRIPTION | 2-1 |
| FEATURES | 2-1 |
| DESCRIPTION | 2-1 |
| BLOCK DIAGRAM | 2-2 |
| SOFTWARE | 3-1 |
| GENERAL | 3-1 |
| Command Structure | 3-1 |
| Addressed Mode | 3-1 |
| Non-Addressed Mode | 3-2 |
| Command List | 3-2 |
| COMMAND FUNCTIONS | 3-2 |
| Set Time Base | 3-2 |
| Read Digital Inputs | 3-3 |
| Read Change-of-State | 3-3 |
| Enable Change-of-State Detection | 3-4 |
| Selecting Which Edge Will Increment Counter | 3-5 |
| Read Digital Input Counter | 3-5 |
| Reset Counter | 3-5 |
| Read Firmware Revision Number | 3-6 |
| Resend Last Response | 3-6 |
| Hello Message | 3-6 |
| Setting a New Baud Rate | 3-7 |
| Programming Pod Address | 3-8 |
| Entering a New Program | 3-8 |
| ERROR CODES: | 3-9 |
| SPECIFICATIONS | 4-1 |
| SERIAL COMMUNICATIONS INTERFACE | 4-1 |
| DIGITAL INPUTS | 4-1 |
| ENVIRONMENTAL | 4-1 |
| POWER REQUIRED | 4-2 |
| WARRANTY | 5-1 |
| APPLICATION CONSIDERATIONS | A-1 |

INSTALLATION

INSTALLING THE SOFTWARE

You have received with your product a CD that contains all the software you need to use your card. The CD is compatible with any type of Windows or DOS system.

To install the software required for your card:

1. Insert CD in your CD ROM - If the install program does not start within 30 seconds, run "install.exe" from the root directory of the CD.
2. Click the **Install Software to Hard Disk** button.
3. Select the product you wish to install from the list shown.
4. Click **Next**.
5. The CD creates a directory with a default name; if you want to change it, click **Change** and select the path you prefer.
6. We advise you to also install the Tools Package at least once per system.
7. Click **Quick Install** to run the install process or click **Detailed Install** if you want more information on the files installed.
8. Click **Finish**.
9. Click **Exit install program** when finished.

You now have two types of files on your hard disk:

1. Software, including samples in C, Pascal, QuickBasic and a setup program, specifically for your card.
2. Software to help you use Portwell cards under a variety of environments:

Setup.exe Setup program

Findbase.exe DOS utility to determine an available base address for ISA bus , non-PnP 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 n^{th} 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 Portwell 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

In the *ACCES32* directory:

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 NT. This device driver provides register-level hardware access from Windows NT, normally called through ACCES32.DLL. Two methods of using the driver are provided, the ACCES32.DLL (recommended) and the DeviceIOControl handles direct to the SYS file (slightly faster)

ACCES95 and ACCESNT

These two drivers are mentioned for users migrating from older versions of Portwell Tools. The functionality of ACCES95 and ACCESNT has been combined into ACCES32.DLL, which is described up .

In order to modify your software to use the new ACCES32.DLL, change the file you link to from ACCES95 or ACCESNT to ACCES32. No other changes are necessary.

If you want to avoid recompiling software that was written for ACCES95 or ACCESNT, just rename ACCES32.DLL to the appropriate name (95 or NT).

In the *BSAMPLES* directory:

A Quickbasic sample.

In the *CSAMPLES* directory:

Samples in C.

In the *PCI* directory:

This directory contains PCI-bus specific programs and information. If you are not using an Portwell PCI card, you can ignore or delete this directory.

In the *PSAMPLES* directory:

This directory contains samples in Pascal

In the VBACCES

directory: 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.

In the WIN32IRQ

directory: you have utility software to handle IRQs from any card under Win95/98 or NT

1. Drivers.src with 3 subdirectories

a. DLL 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.

b. SYS The samples in this directory are provided ONLY for WindowsNT. The DeviceIOControl based interaction with the register-level driver is only available in NT. If your code is written to use this method, it will not work with Windows 95 or Windows 98.

The SYS file is the actual workhorse behind hardware access in WindowsNT. It utilizes the DeviceIOControl API function for interaction with user code. Samples are provided demonstrating this API call, but it is strongly recommended that the DLL interface be used. The DLL described above encapsulates the SYS file and performs the DeviceIOControl calls at a small penalty in speed. (A call through the DLL interface)

c. VXD source fo the driver

2. Samples: Samples in VisualC, Delphi and C++ Builder

INSTALLING THE POD

The RDI-54 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.

A screw terminal assembly is provided for electrical connections inside the pod. You can make these connections by assembling a cable of whatever length you need. At the other end you can either assemble a 62-pin connector to mate with a header on your termination panel (for ease of connect/disconnect) or whatever termination method best fits your application.

If you desire, Portwell can provide a custom cable fabricated according to your specifications.

(Continued on next page)

PIN CONNECTIONS

Connections to RDI-54 are made at a screw terminal assembly located at the top of the module. Screw terminal numbers silk-screened on that card and the corresponding signals are as follows:

| Term. # | Signal | Term.# | Signal |
|---------|-----------|--------|------------------|
| 1 | Bit 5 *** | 32 | Bit 53 |
| 2 | Bit 4 | 33 | Bit 52 |
| 3 | Bit 3 | 34 | Bit 6 |
| 4 | Bit 2 | 35 | Bit 7 |
| 5 | Bit 33 | 36 | Bit 1 |
| 6 | Bit 34 | 37 | Bit 0 |
| 7 | Bit 35 | 38 | Bit 24 |
| 8 | Bit 39 | 39 | Bit 25 |
| 9 | Bit 38 | 40 | Bit 26 |
| 10 | Bit 8 | 41 | Bit 27 |
| 11 | Bit 9 | 42 | Bit 31 |
| 12 | Bit 10 | 43 | Bit 30 |
| 13 | Bit 11 | 44 | Bit 29 |
| 14 | Bit 13 | 45 | Bit 28 |
| 15 | Bit 12 | 46 | Bit 32 |
| 16 | Bit 14 | 47 | Bit 18 |
| 17 | Bit 15 | 48 | Bit 19 |
| 18 | Bit 37 | 49 | Bit 20 |
| 19 | Bit 36 | 50 | Bit 21 |
| 20 | Bit 40 | 51 | Bit 22 |
| 21 | Bit 41 | 52 | Bit 23 |
| 22 | Bit 42 | 53 | Reset |
| 23 | Bit 43 | 54 | RS485+ |
| 24 | Bit 47 | 55 | RS485- |
| 25 | Bit 46 | 56 | /INT0 |
| 26 | Bit 45 | 57 | Local Pwr * |
| 27 | Bit 44 | 58 | Local Pwr Gnd |
| 28 | Bit 16 | 59 | Isolator Pwr ** |
| 29 | Bit 17 | 60 | Isolator Pwr Gnd |
| 30 | Bit 51 | 61 | Bit 50 |
| 31 | Bit 48 | 62 | Bit 49 |

Notes:

- * "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 RDI-54. (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 RDI-54 and should be independent of “Local Power”. Isolator power should be connected between terminals 59 and 60. 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 terminals. Regardless of isolation mode selected via the ISO/ISO jumpers, power must be applied to these pins for serial communication to function.

- *** Digital Inputs are protected by series resistors. There are no onboard pull-up or pull-down resistors on the inputs. An unconnected input will be a floating logic level and its state can not be guaranteed.

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 on input/output wiring.

CE-marked versions of RDI-54 meet the requirements of EN50081-1:1992 (Emissions), EN50082-1:1992 (Immunity), and EN60950:1992 (Safety).

FUNCTIONAL DESCRIPTION

FEATURES

- * Opto-Isolated RS-485 Serial Interface to Host Computer.
- * 54 Digital Inputs
- * Digital Input Voltages up to 50V.
- * 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.
- * 8-Bit Digital Input Software Counters.
- * Change of State Flag Readable via the Serial Port.

DESCRIPTION

RDI-54 is an intelligent interface unit that connects up to 54 parallel digital inputs to a computer. It is packaged in a NEMA4 enclosure for remote installation in harsh environments. Communication with the host computer is via EIA RS-485 half-duplex, serial communications protocol. ASCII-based command/response protocol permits communication with virtually any computer system. RDI-54 is one of a series of remote intelligent units called the "REMOTE Portwell" series. As many as 31 REMOTE Portwell Series pods (or other RS-485 devices) may be connected to the computer on a single two-wire multidrop RS-485 network.

A type 8031 microcontroller (with 8Kx8 RAM, 8Kx8 non-volatile EEPROM, and a watchdog timer circuit) gives RDI-54 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 RDI-54 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.) Use of ASCII-based software permits you to write applications in any high-level language that supports ASCII string functions and you can use REMOTE Portwell series modules with virtually any computer.

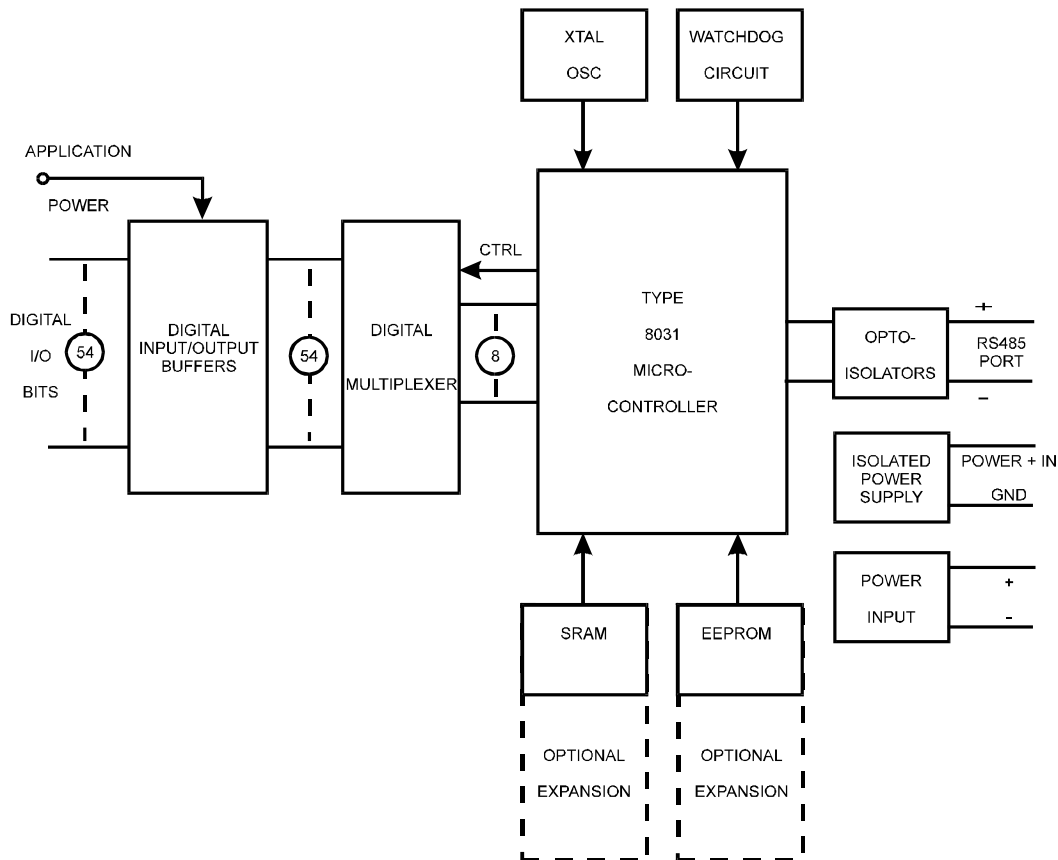
The module, or pod, address is programmable from 00 to FF hex and whatever address is assigned is stored in EEPROM and used as the default address at the next Power-ON. Similarly, the baud rate is programmable for 1200, 2400, 4800, 9600, 14400, 19200, 28800, and 57600 and is stored in EEPROM and used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

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

provide digital-input sample rates from 14 Hz to 1 KHz

Digital inputs of up to 50V amplitude may be read individually, or in 8-bit bytes. There are also digital input counters on each input. Selectable edges can be counted for up to 255 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 input bits.

BLOCK DIAGRAM



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 a stand-alone pod mode of operation.

SOFTWARE

GENERAL

You received ASCII-based software on CD for use with RDI-54. 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 RDI-54 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:

- The single lower case letter 'x' designates any valid hex digit (0-F).
- The single lower case letter 'b' designates either a '1' or '0'.
- The single lower case letter 'p' designates eight-bit port.

- d. The symbol '±' designates either a '+' or a '-'.
- e. All commands are terminated with CR, the ASCII character #13.
- f. Wherever xx is used to designate a bit number, only 00-35 are valid.
- g. All commands are case insensitive; i.e., can be either upper case or lower case
- h. The symbol '*' means zero or more valid characters (total msg length <255 decimal)

Command List

| | | |
|----------|-------------------------------------------------------------|-----------|
| Sxxxx | Set a new timebase. 039A<xxxx<=FFFF | CR |
| I | Read all digital bits. | xxxxxxCR |
| Ixx | Read a single digital bit. (00<=xx<=35) | bCR |
| Ip | Read digital bits (p*8 through p*8+7) 0<=p<=6 | xxCR |
| Y | Read digital input COS bit and clear bit | Y or N |
| Tpxx | Set bit (p*8) - (p*8+7) Mask for COS bit flag, 0=ignore COS | 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) | xxCR |
| Rxx | Reset digital input counter xx to 00 | CR |
| Rall | Reset all digital input counters | 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 |

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 another command is sent.

Set Time Base

Sxxxx Set Time Base

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)

Examples:

Program the RDI-54 to a 1msec timebase

```
SEND:          S039A
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.

Read Digital Inputs

```
I      Read 54 bits
lxx    Read bit number xx
lp     Read bits (p*8) through (p*8+7)
```

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 54 bits.

```
SEND:          I
RECEIVE:       FFFFFFFFFFFFFFFF[CR]
```

Read only bit 35 (53 decimal, the highest bit on the card)

```
SEND:          I35
RECEIVE:       1[CR]
```

Read only bit 2

```
SEND:          I02
RECEIVE:       1[CR]
```

Read bits 8-F

```
SEND:          I1
RECEIVE:       FF[CR]
```

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.

Example:

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

Tp_{xx} Set COS mask for bits (p*8) through (p*8+7)

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:          T000
RECEIVE:      [CR]
SEND:          T100
RECEIVE:      [CR]
SEND:          T208
RECEIVE:      [CR]
SEND:          T300
RECEIVE:      [CR]
SEND:          T400
RECEIVE:      [CR]
SEND:          T500
RECEIVE:      [CR]
SEND:          T600
RECEIVE:      [CR]
```

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

```
SEND:          T0FF
RECEIVE:      [CR]
SEND:          T1FF
RECEIVE:      [CR]
SEND:          T2FF
RECEIVE:      [CR]
etc...
```

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 35 to falling edge active

SEND: D35-

RECEIVE: [CR]

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

Read Digital Input Counter

| | |
|-----|-------------------------------|
| cxx | Read digital input counter xx |
|-----|-------------------------------|

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). Input counters are configured as 8-bit counters. Counter content is provided most significant bit first.

Example:

Read digital input counter for bit #1

SEND: C01

RECEIVE: 13[CR] ;assuming 13hex edges since last reset

Reset Counters

| | |
|------|----------------------------------|
| rxx | Reset digital input counter xx |
| rall | Reset all digital input counters |

These commands are used to reset digital input counters to zero.

Example:

Reset digital input counter for digital input number 3

SEND: r03

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 RDI-54 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 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 that the last command was "l", ask pod to resend last response

```
SEND:          n
RECEIVE:  FFFFFFFFFFFFFFFF[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, RIOD-24 Rev rr Firmware Ver:x.xx Portwell 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, RDI-54 Rev B1 Firmware Ver:1.00
Portwell 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 |
|------|-------|
| 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

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 many addresses will 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]
```

Entering a New Program

PROGRAM= This command initiates the transfer of a new program to the RDI-54.

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 Portwell to provide field-upgrades to the RDI-54 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 upgrade 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 35, in hex. (0-54 decimal))
- 3: Improper Syntax. (Not enough parameters is the usual culprit)
- 4: Channel number is invalid for this task
- 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].

SPECIFICATIONS

SERIAL COMMUNICATIONS INTERFACE

Serial Port: Opto-isolated Matlabs type LTC485 Transmitter/Receiver. Compatible with RS-485 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 bypassed: -7V to +12V.

Receiver Input Sensitivity: ± 200 mV, differential input.

Receiver Input Impedance: 12K Ω 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 54. Can be read on a bit-by-bit basis, or an 8-bit byte basis, or all 54 at once..

Sample Rate: Programmable from 14 Hz to 1 KHz.

Software Counters: There are 8-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 μ A maximum.

ENVIRONMENTAL

Operating Temperature Range: 0 $^{\circ}$ TO 65 $^{\circ}$ C (Optional -40 $^{\circ}$ to +80 $^{\circ}$ 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:

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

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

For example, at an ambient temperature of 25°, the voltage V_I can be up to 18.4V. At an ambient temperature of 100°, 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 @ 100 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 $V_1 - 16$ where V_1 is the power supply voltage. The voltage rating of the zener diode should be $\leq V_z \times 0.12$ watts. Thus, for example, a 24 VDC power supply would require use of an 8.2V zener diode with a power rating of $8.2 \times 0.12 \leq 1$ watt

WARRANTY

Prior to shipment, Portwell products are thoroughly inspected and tested to applicable specifications. However, should equipment failure occur, Portwell assures its customers that prompt service and support will be available. All equipment originally manufactured by Portwell 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 Portwell' Customer Service department. Be prepared to give the 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, freight prepaid, to the Portwell 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 Portwell option with no charge for labor or parts not excluded by warranty. Warranty commences with equipment shipment.

Following Years: Throughout your equipment's lifetime, Portwell 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 Portwell

Equipment provided but not manufactured by Portwell 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 Portwell is limited to replacing, repairing or issuing credit (at Portwell discretion) for any products which are proved to be defective during the warranty period. In no case is Portwell liable for consequential or special damage arising from use or misuse of our product. The customer is responsible for all charges caused by modifications or additions to Portwell equipment not approved in writing by Portwell or, if in Portwell 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 sold or furnished by Portwell.

APPENDIX A

APPLICATION CONSIDERATIONS

INTRODUCTION

Working with RS-422 and RS-485 devices is not much different from working with standard RS-232 serial devices and these two standards overcome deficiencies in the RS-232 standard. First, the cable length between two RS-232 devices must be short; less than 50 feet at 9600 baud. Second, many RS-232 errors are the result of noise induced on the cables. The RS-422 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 RS-422 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 |

A third deficiency of RS-232 is that more than two devices cannot share the same cable. This is also true for RS-422 *but RS-485 offers all the benefits of RS-422 plus allows up to 32 devices to share the same twisted pairs.* An exception to the foregoing is that multiple RS-422 devices can share a single cable if only one will talk and the others will all receive.

BALANCED DIFFERENTIAL SIGNALS

The reason that RS-422 and RS-485 devices can drive longer lines with more noise immunity than RS-232 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. RS-485 drivers must have this control capability. RS-422 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.

RS-422 SPECIFICATION SUMMARY

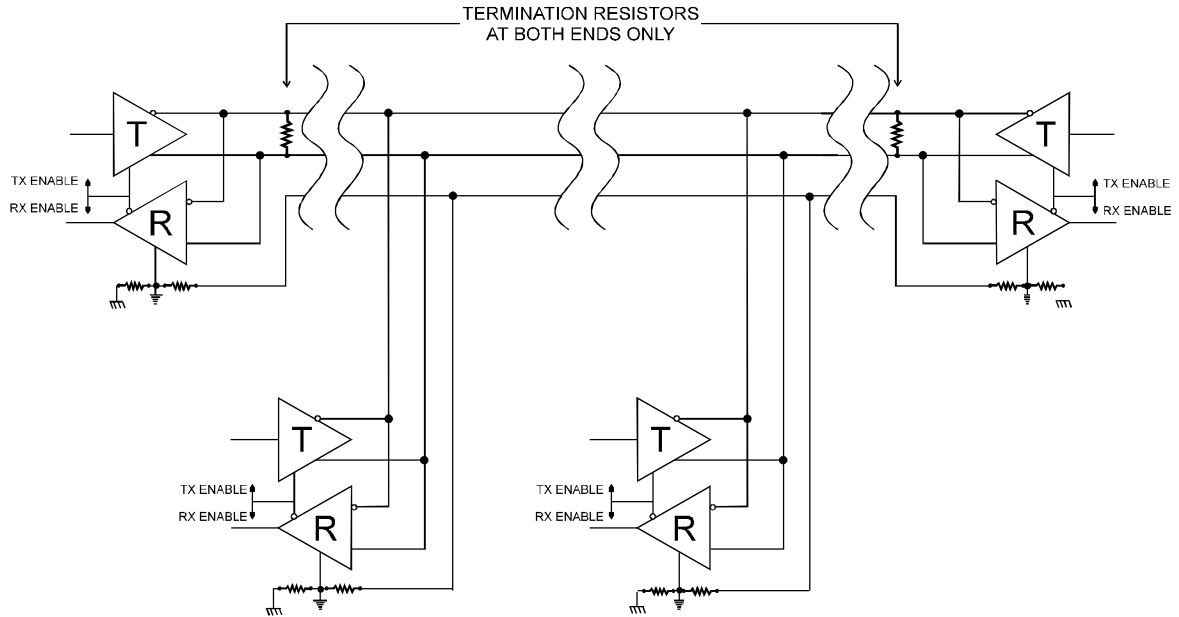
| Parameter | Conditions | Min. | Max. |
|-------------------------------------|-------------------------|-----------|-------------------|
| Driver Output Voltage (unloaded) | | 4V -4V | 6V -6V |
| Driver Output Voltage (loaded) | LD and LDGND jumpers in | 2V -2V | |
| Driver Output Resistance | | |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

To prevent signal reflections in the cable and to improve noise rejection in both the RS-422 and RS-485 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 RS-422 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.)

RS-485 DATA TRANSMISSION

The RS-485 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 RS-422 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.



Typical RS-485 Two-Wire Multidrop Network