Digi®

Connectware™

NetSilicon

A Digi International Company

NET+RealPort Developer's Guide

Making DEVICE NETWORKING *easy*™

NET+RealPort Developer's Guide

Part number/version: 90000657_B
Release date: September 2004
www.digi.com

Contents

Using This Guide

Review this section for basic information about the guide you are using, as well as for general support and contact information.

About this guide

This guide provides information about installing NET+RealPort software and devices. In addition, this guide describes how to monitor port activity, validate that your serial ports are set up and working as expected, and collect data for debugging.

Who should read this guide

This guide is for developers who are:

- Adding NET+RealPort COM port redirection to their NET+Works application testing
- Testing and supporting client-side applications that communicate with a device based on NET+Works with NET+RealPort

To complete the tasks described in this guide, you must:

- Be familiar with installing and configuring network software and development board systems.
- Have sufficient system or user privileges to do these tasks.

What's in this guide

This table shows where you can find specific information in this guide:

To read about	See
An overview of NET + RealPort; installation instructions	Chapter 1, "Getting Started"
Using Portmon	Chapter 2, "Monitoring Port Activity"
Using rptest	Chapter 3, "Testing"
Using rpdump	Chapter 4, "Debugging"
The functions of the RealPort API	Chapter 5, "RealPort API

Conventions used in this guide

This table describes the typographic conventions used in this guide:

This convention	Is used for
italic type	Emphasis, new terms, variables, and document titles.
Select **menu → menu option or options.**	Menu selections. The first word is the menu name; the words that follow are menu selections.
`monospaced type`	File names, path names, and code examples.

Customer support

To get help with a question or technical problem with this product, or to make comments and recommendations about our products or documentation, use the contact information in the table shown next:

For	Contact information
Technical support	Telephone: 1 877 912-3444/ 1 952 912-3456 Fax: 1 952 912-4960
Documentation	techpubs@digi.com
Digi home page	www.digi.com/support/eservice/eservicelogin.jsp
Online problem reporting	www.digi.com/problemreporting.jsp

Getting Started

CHAPTER 1

This chapter provides an overview of NET+RealPort. This chapter also provides installation information and instructions.

Overview

NET+RealPort extends the NET+Works development environment by providing a complete set of software components and APIs that enable you to seamlessly integrate patented Digi RealPort technology in embedded applications.

The RealPort software leverages the widely-used TCP/IP network infrastructure to provide a virtual connection to serial devices. The drivers, which are installed directly on the network client systems, allow applications to talk to devices across a network as though the devices were directly attached to the system.

The design of the RealPort software allows multiple connections to multiple ports over a single TCP/IP connection. Because NET+RealPort does not require dedicated TCP/IP connections for each serial port, network overhead and processor load are dramatically reduced. RealPort also supports other leading features, as well as connection recovery.

The RealPort toolkit includes:

- Binary libraries for NET+OS
- Certified drivers for all major operating systems (including Microsoft WHQL certification on Windows 2000, Windows XP, and Windows Server 2003)
- A testing suite
- Two sample applications:
  - `narealport`
  - `narealport_ep`
- Complete documentation

RealPort server module

The RealPort server under NET+OS is invoked by a user application after the TCP/IP stack is initialized. Run only one instantiation of the server on a system at a time.

Serial ports are dynamically registered for use with the server and are available to client application only after they are registered.

RealPort server API

The RealPort server API is encapsulated in a library and provides these functions:

- Invokes the RealPort server daemon, which accepts these parameters:
  - The maximum number of concurrent clients
  - The TCP port on which to accept connections
  - The NET+OS kernel timeslice and priority base
- Registers a serial port to use with RealPort, which enables the port to be opened

For details on the RealPort server API, see the online help.

RealPort daemon and RealPortStartServer function

The RealPort daemon, which accepts connections and spawns threads to process connections, exists as a thread that is started by a call from a user application, from the `RealPortStartServer` call.

The call to `RealPortStartServer` spawns all threads and handles all initialization tasks, including:

- Verifying configuration
- Initializing the RealPort server
- Spawning helper threads

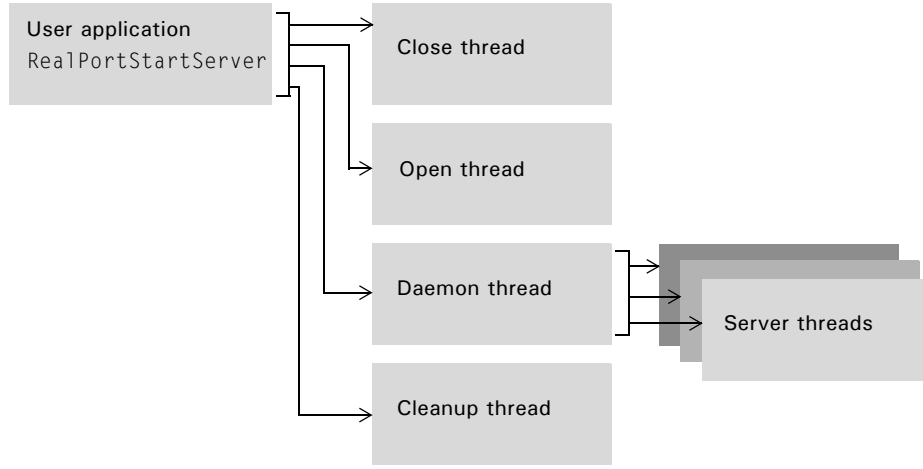
Architecture

The user application thread invokes four threads through a call to `RealPortStartServer`:

- The close thread
- The open thread
- The daemon thread
- The cleanup thread

The daemon thread then spawns server threads that process incoming connections. The cleanup thread handles thread cleanup for the server threads.

This diagram illustrates the call to `RealPortStartServer` and the threads:



Resource availability

The RealPort server can share ports with other processes on the system. If, however, a port is already open, the RealPort server cannot open it. The RealPort server supports a multiple-access model in which clients have exclusive access on a "first come, first served" basis.

RealPort dynamically allocates threads and thread stacks, buffers, and other resources. Memory is allocated proportionally to the number of client connections and the number of configured concurrent connections. If memory is not available when the server is invoked, the RealPort server cannot run. In addition, if memory is not available for an incoming connection, the connection is not processed.

The RealPort server daemon requires a listener socket and a TCP port to listen on. Sockets are allocated for each incoming client connection and closed when the connection ends. If no socket file descriptors are available, client connections fail. If another listener is on the same TCP port, or if no socket descriptors are available, server instantiation fails.

Tuning

Using the RealPortStartServer API, you can tune the RealPort server to your system. Three parameters control the way in which the server interacts with the system:

- **Priority base.** The threads that are associated with the RealPort server have priority equal to the base or +/- 1 of the base. Adjust this value to reflect the RealPort's priority relative to other processors on your system. When the RealPort server is running as a standalone application, a priority of 10 is acceptable
- **Timeslice.** Timeslice is measured in ticks. As a standalone application, the RealPort server operates best when the number of ticks is equal to approximately 10 ms.

  To determine the number of ticks per second, see the value of `BSP_TICKS_PER_SECOND` in the `bsp.h` file.

  When `BSP_TICKS_PER_SECOND` is 1000, ten ticks is an acceptable value; when `BSP_TICKS_PER_SECOND` is 100, 1 tick is an acceptable value.
- Maximum number of concurrent connections. Set this parameter to the number of client users. RealPort clients make persistent connections to servers, and their requests are queued – even if no ports are open.

  If, for example, four clients are sharing a device with one serial port, set the number of concurrent connections to 4 (clients), not 1 (port).

C++ considerations

Because the RealPort library was written in C++, C++ startup initialization is required for proper functioning.

Be sure that you set `INCLUDE_CXX_STARTUP` to 1 in your application `makefile`.

BSP considerations

RealPort expects registered drivers to implement a POSIX termios API.

- Define `BSP_SERIAL_API_TERMIOS`, and set it to `1` in the platform `bsp.h` file.
- Set `BSP_SERIAL_API_LEGACY` to `0`, or undefined.
- Define `BSP_SERIAL_PORT_API` as `BSP_SERIAL_API_TERMIOS`.
- Configure `BSP_SERIAL_PORT_N` as `BSP_SERIAL_UART_DRIVER` (asychronous serial port).

Be aware that if you make any changes to the `bsp.h` file, you *must* recompile the BSP.

Preparing to install RealPort

This section describes what you need to do or be aware of before you start the installation.

Before you install the RealPort software, make sure that:

- You can log into Windows as either `administrator` or a user that is a member of the administrator's group.
- You are running a compatible version of NET+OS (6.0C).
- Your device is powered on and connected to the network.

In addition, you need to get this information:

- The IP address of the device
- The model name of the device
- The TCP port to use
- Whether your network or host computer has firewalls or proxies installed, which can keep your device from being discovered as the RealPort Setup wizard searches the network for RealPort devices.

If your device is not listed on the **Select Device** page of the wizard, you may need to do either of these steps:

- If it is feasible to do so, disable the firewall and/or proxy before you install RealPort.
- Install the device manually by selecting **Device not listed** on the **Select Device** page and then providing the necessary information.

Installing a RealPort device and the RealPort software

▶ **To install a RealPort device and RealPort software:**

1. Place the RealPort CD ROM in your system's CD drive.
2. Run `Setup.exe`., the RealPort Setup wizard.
3. Respond to the wizard prompts by making selections and providing information as needed.

   The wizard searches the network for RealPort devices and displays the devices it finds on the **Select Device** page.

Monitoring Port Activity

CHAPTER 2

This chapter describes how to use Portmon application to monitor your system's port activity.

Overview

Portmon is a Microsoft Windows utility that you use to monitor and display your system's serial and parallel port activity.

You use Portmon as you develop device drivers and user mode serial applications. You also can use Portmon when you begin to experience a problem where none existed before. You can then analyze the data in the trace to identify where the problem is.

For example, suppose you attach a device to COM1, and everything works as expected. Then you attach your device to a Digi device, and problems occur. In such a case, you would want to create two Portmon traces:

- One of COM1, which shows what communication looks like when everything is operating correctly
- One of the Digi port

By looking at the differences between the traces, you determine why the device fails and COM1 works. You can either view the data on your screen or save it to a file for later analysis.

A freeware utility, Portmon is available for download from this address:
`www.sysinternals.com`

Portmon tutorial

This section provides a brief tutorial about Portmon in which you will:

- Connect Portmon to your local machine.
- Configure Portmon. You can configure Portmon in any of several ways; in this tutorial, you will use the configuration that will capture the data that is most useful to you.
- Create a Portmon trace and run a sample application.

To learn more about Portmon, see the Portmon online help.

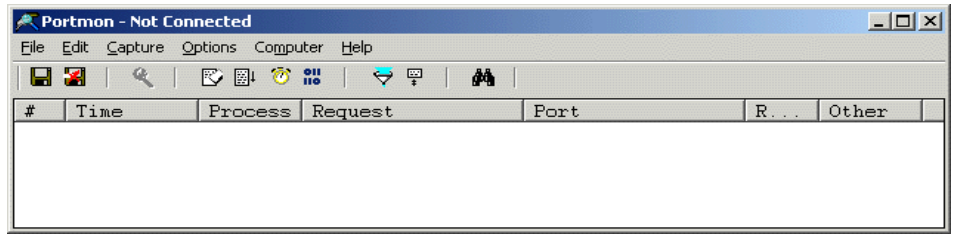
Connecting Portmon to your local machine

▶ **To connect PortMon to your local machine:**

1 Start Portmon.

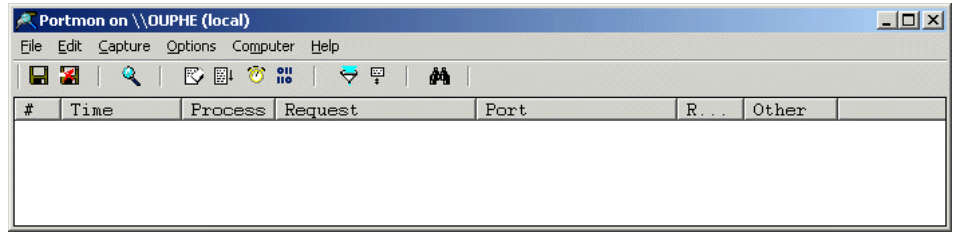
The Portmon window opens, displaying this message:

```
Portmon - Not Connected
```



2 To connect Portmon, select **Computer → Connect Local**.

The Portmon window now displays the name of the PC to which it is connected:



3 Keep the Portmon window open and go on to the next section.

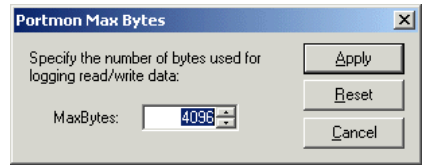
Configuring Portmon

Using the next procedure, you'll get results that provide more useful information than those you would get from other configurations.

▶ **To configure Portmon:**

1 Select **Edit** → **Max Output Bytes**.

The **Portmon Max Bytes** dialog box opens:



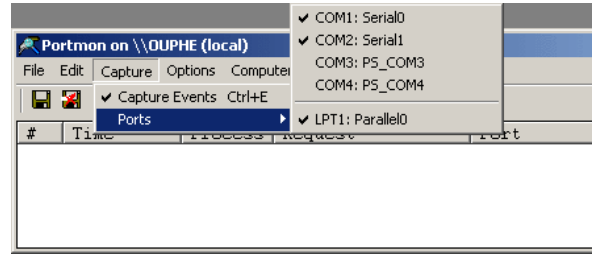
2 Set the size of the data of each logged read/write to **4069**, and then click **Apply**.

3 From the **Options** menu, do these steps:

- Check **Show Time.**
- Check **Show Hex.**
- Uncheck **Clock Time**.

Be aware the **Hide Toolbar**, **Auto Scroll**, and **Always On Top** do not affect the data that is recorded to the log file.

4 To specify which ports to monitor, select **Capture** → **Ports**, and then click the ports you want, as shown here:



You can select *only* ports that are not being used by another application.

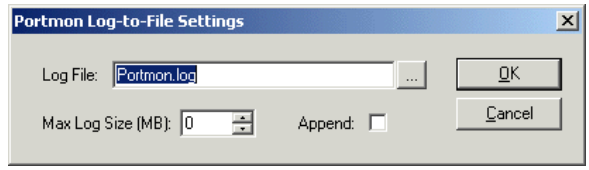
5 Go on to the next section.

Creating a Portmon trace and running a test application

▶ **To create a Portmon trace:**

1 Select **File → Log to File**.

The **Log-to-File Settings** dialog box opens:

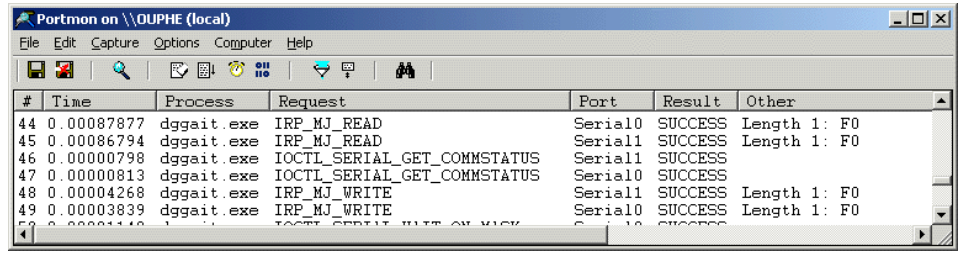


2 Do these steps:

- Leave **Max Log Size (MB)** set to `0` (which leaves the log file size unrestricted).
- To specify the file in which to log the data, click ... (to the right of the **Log File** input box), navigate to the location, select the file, and click Save.
- Leave **Append** unchecked.
- Click **OK**.

3 To create some data, either run your test application or exercise the serial ports.

Portmon automatically displays the serial port activity as it happens and logs it to the file, as shown here:



4 When you finish testing, turn of logging by selecting **File → Log to File**.

At this point, you can open the log file using a text editor and examine the data.

Testing

CHAPTER 3

This chapter describes how to use the `rptest` application.

Overview

To verify that the ports are set up correctly and working as you expected, use the `rptest` application. This application provides options for specifying the ports to test, whether to view live data or save it to file, port settings, and port execution.

Starting rptest

You run `rptest` from the command line.

▶ **To run rptest:**

1. Open a command window by selecting **Start → Run**.
2. Enter the command for your Windows platform:
   - For Windows 2000/ XP/ SO3 systems, enter:
     `cmd`
     and press Enter.
   - For Windows 98/Me/NT systems, enter:
     `command`
     and press Enter.
3. Change to the directory in which `rptest` is located:
   `cd directoryname`
   and press Enter.

rptest options

The next sections describe the `rptest` options, which you use to specify:

- The ports to test and whether the ports are receive-only or transmit-only
- Display options
- Port setting
- Port execution behavior

Getting rptest usage help

To get `rptest` help, enter this command, and press Enter:

```
rptest /?
```

You see information that looks similar to this example. The current settings are displayed; in this case, all the options are set to their default settings:

```
Digi RealPort Test
Copyright 1996-2004 Digi International Inc.
Version 2.6.32.0

Designating I/O Channels        Command-line Arg.
----------------------- -----------------
Transmit/Receive ports          -bxp #[-#][,#[-#]]
Receive-only ports              -rxp #[-#][,#[-#]]
Transmit-only ports             -txp #[-#][,#[-#]]

Changing DisplayOptions         Commands         Settings
----------------------- -------- --------
Display live tput (0=no, 1=yes) -tpu # 1

\Adjusting Port Settings        Commands         Settings
---------------------- -------- --------
Bits per second (baud rate)     -bps #           9600
Data bits (5=5, 6=6, 7=7, 8=8)  -dbs #           8
Parity (0=None, 1=Odd, 2=Even   -par #           0
3=Mark, 4=Space)
Stop bits (0=1, 1=1.5, 2=2)     -sbs #           0
DTR\DSR flow control (0=disable,  -dtr #         0
1=enable, 2=handshake)
RTS\CTS flow control (0=disable,  -rts #         0
1=enable, 2=handshake, 3=toggle)
Xon\Xoff (0=disable, 1=enable)  -xon #           0
ReadIntervalTimeout (millisecs.)  -rit #         0
ReadTotalTimeoutMultiplier (ms.)  -rtm #         0
ReadTotalTimeoutConstant (ms.)  -rtc #           15600
WriteTotalTimeoutMultiplier (ms.) -wtm #         0
WriteTotalTimeoutConstant (ms.) -wtc #           15600

Adjusting Port Execution Behavior Commands       Settings
------------------------------- -------- --------
Buffer size (>=0)               -bss #           1152
Number of buffers(-1=infinite)  -nob #           10

To abort while testing, depress the key sequence, <ctrl>-c.

E.g., the following command will test COM1 and COM2
at 115200 bits per second using hardware flow control:

"rptest -bxp 1-2 -bps 115200 -rts 2"
```

Specifying I/O channels

The only `rptest` options you *must* use are those that specify:

- The ports to test
- Whether the ports you are testing are receive-only or transmit-only

You can specify individual ports, ranges of ports, or a combination of individual ports and ranges.

Option	Description
`-bxp #[-#][,#[-#]]`	Specifies the serial ports to test
`-rxp #[-#][,#[-#]]`	Specifies ports that are receive-only
`-txp #[-#][,#[-#]]`	Specifies ports that are transmit-only

The `rptest` application issues the same number of reads and writes for all the ports you are testing.

You specify this value in the `-nob` option, and you define the size of the buffer with the `-bss` option. (For information about `-nob` and `-bss`, see "Adjusting port execution behavior," later in this chapter.)

Changing display options

The `-tpu` option specifies whether to display:

- The number of bytes transmitted
- The number of bytes received
- The calculated throughput

Enabling `-tpu` is useful if you are viewing the live data on your screen.

If, however, you are saving the data to a file, you may want to disable `-tpu`. Each time the throughput calculation is updated, a new line of information is added to the output file; as a result, the output contains extraneous data. In such a case, disabling `-tpu` makes the output file more concise, and it increases readability.

Option	Description
-tpu #	Display live tput: ■ 0 = no ■ 1 = yes

Specifying port settings

Use the options in the next to specify port settings. The settings apply to all ports you are testing.

Option	Description
-bps #	The number of bits per second (baud rate)
-dbs #	Data bits: ■ 5 = 5 ■ 6 = 6 ■ 7 = 7 ■ 8 = 8
-par #	Parity: ■ 0 = none ■ 1 = odd ■ 2 = even ■ 3 = mark ■ 4 = space
-sbs #	Stop bits: ■ 0 = 1 ■ 1 = 1.5 ■ 2 = 2
-dtr #	DTR/DSR flow control: ■ 0 = disable ■ 1 = enable ■ 2 = handshake To enable handshaking, set either this option, the -rts option, or both to 2.

Option	Description
`-rts #`	RTS/CTS flow control: ■ `0` = disable ■ `1` = enable ■ `2` = handshake ■ `3` = toggle To enable handshaking, set either this option, the `-dtr` option, or both to `2`.
`-xon #`	Software flow control: ■ `0` = disable ■ `1` = enable

Adjusting port execution behavior

Using the options in the next table, you specify the size of the buffer and the number of reads and writes that will be performed:

Option	Description
`-nob #`	Number of read and writes
`-bss #`	Buffer size

Example

This `rptest` example requires a connection between COM1 and COM2 with a null modem cable:

Example	Description
`rptest -bxp 1-2 -bps 115200-rts2`	Sends data between ports COM1 and COM2 at 115200 bps. The test sends the default number and size of buffers – 10 buffers of 1152 bytes from COM1 and COM2 – and attempts to read the same from COM1 and COM2.

Debugging

CHAPTER 4

This chapter describes how to use the `rpdump` application.

Overview

`rpdump` is an application that you use for debugging. The `rpdump` application intercepts and filters RealPort commands and data that pass between a host computer and a PortServer, and then records or displays the data.

You can either view real-time RealPort data or save the data to a file and review it later. Using filter options, you specify which data to record or display.

To intercept real-time data, you must install `rpdump` into the TCP connection between the RealPort daemon and the PortServer. To do so, you start `rpdump` and provide the node name of the port you want to monitor. Then you start the RealPort daemon and provide the node name of the system on which you are running `rpdump`. The daemon connects to `rpdump`, and `rpdump` connects to the server.

rpdump options

The rpdump application provides many options; included among them are options for:

- Getting help
- Selecting mode
- Specifying packet type
- Selecting packets based on timestamp information
- Enabling and disabling the display of data on your screen and specifying a file to write the data to

Selected packets are formatted and printed to `stdout` unless you select either -s or -o. In these cases, packets are written to the output file that you specify.

You can select packets by:

- Packet type (`-p`)
- Number (`-n`)
- Time (either `-D` or `-T`)
- Port number (`-P`)

This table describes the rpdump options, arranged alphabetically:

Option	Description
-?	Displays a summary of rpdump commands on your screen and then exits.
-1	Captures one session and stops. The rpdump application typically loops to capture multiple sessions.
-a file	Disables printing and appends all new data to *file* for later viewing with *-i* or *-I*.
-B	Prints input file byte offsets before each line.
-b	Inserts a blank line between packets to improve readability.
-c	When used with *-o*, enables printing and writes all data to the output file. (The *-p* options affect only printing.)
-d	Simulates the delays you would see on a slow speed link.
-D date:date	Selects a specified packet timestamp *date* range in [[[*MM*]*DD*]*hh*]*mm* format. For example: ■ *-D 1600:30* specifies 4 PM to 4:30 PM each day. ■ *-D 06010000:302359* selects June every year. To select multiple ranges, specify this option multiple times.
-e string	Reports this *string* in error messages in place of the program name. This option is useful when you are running multiple copies that refer to different RealPort servers.
-f	Specifies fast mode. Packet data is not interpreted and can be either collected or copied to an output file with minimal overhead.
-H hdrSize	Specifies the assumed size of a TCP header when rpdump calculates delays for a slow WAN link. The default size is 46 bytes
-h	Prints this page and exits.
-I file	View recorded RealPort data from the file that you previously created with the *-a* or *-o* option. The application waits for additional data to appear when it reaches the end of the file.
-i file	Views recorded RealPort data from the file that you previously created with either the *-a* or *-o* option. All filtering and display options remain available.

Option	Description
-l port	Sets the TCP listen port. The default setting is port 771.
-m	Monitors progress in capture mode by regularly updating the current packet number on the screen below the command line.
-n	Prints packet numbers with each packet.
-N num[:num]	Selects specified packet numbers to be displayed. To select multiple ranges, specify this option multiple times.
-o file	Disables printing and writes all selected packets to *file* for later viewing with either the *-i* or *-I* option.
-q	Specifies whether to turn on data collection. The default is off. ■ To turn on data collection, send the SIGTRAP program (signal 5). ■ To turn off data collection, send SIGABRT (signal 6). Signals work only with UNIX and Cygwin versions.
-P port[:port]	Selects a single port or a range of ports. To select multiple ranges, specify this option multiple times
-p options	Specifies the type of packets to print or copy to the output file. You can specify any combination of these options to select: ■ *a* All *-p* options ■ *c* All port commands and responses, except synchronization packets ■ *d* Data packet summaries (no contents) ■ *e* Event packets ■ *f* Window (flow) packets ■ *g* Global packets; that is, those not associated with a particular port ■ *i* Interesting options: cdefgs ■ *k* KME and debug packets ■ *m* Module select packets ■ *n* Synchronization packets ■ *r* The contents of all read data packets sent from server to client ■ *s* Status and error messages produced during data capture ■ *w* The contents of all write data packets sent from client to server

Option	Description
-r rfile	In normal mode, writes all input `tty` data that is sent by the server and that meets the port and packet constraints to *rfile* in raw form. In fast mode, the entire server packet stream is written in raw form to *rfile*.
-s count,size	Writes output to *count* files in round-robin fashion, so that as each file grows to *size* (in Kbytes), the next file in sequence is truncated and then filled. Files are named *file00*, *file01* and so on, using the `file` argument from either of these options: ■ *-o* – Truncates all files and begins output with *file00* ■ *-a* – Preserves all existing files and appends data first to the newest file
-S speed	Sets the line *speed* in bits/second used to compute delays seen on a WAN network. Can be suffixed by *k* to enter kbits/second. The default speed is 56K.
-T seconds	Selects the last *seconds* of data in the input file. This option requires an accurate OS file modification time.
-T low:high	Selects all input file packets with a relative timestamp between *low* and *high* seconds. To select multiple ranges, specify this option multiple times.
-t options	Specifies that packet capture is shown as specified by one or more of these characters: ■ *a* Absolute date and time ■ *d* Delta time since last packet ■ *r* Relative time since capture began ■ *s* An approximation of the time delay seen across a slow WAN link
-V	Prints the `rpdump` version number and exits.
-v	Suppresses packet verification. You might find this option useful for reading: ■ A damaged packet file. ■ A packet file that contains only a subset of the original packets.

Option	Description
-w wfile	In normal mode, writes all input `tty` data that is sent by the server and that meets the port and packet constraints to *wfile* in raw form. In fast mode, the entire client packet stream is written in raw form to *wfile*.
-x	Displays all selected packets in hexadecimal format. If *-p* does not appear on the command line, the application assumes *-pi*.

Examples

This table provides some `rdump` examples:

Example	Description
`rpdump portserver.dgii.com`	1 Listens for an incoming PortServer connection on TCP port 771, the default port number. 2 The program connects to port 771 on `portserver.dgii.com` and begins to display data passed in both directions through the program.
`rpdump -1 4000 -m -0 /tmp/file/fpd portserver.dgii.com`	1 Listens for the daemon connection on the local file system TCP port 4000. 2 Writes all data received to `/tmp/realportfile`. Packet numbers are displayed on the screen as data is captured.

Example	Description
`rpdump -I /tmp/file.rpd -T60| less`	1 Reads in the RealPort data file, skipping all but the last 60 seconds of the file data. 2 At the end of the file, does not exit, and waits for more data to appear.
`rpdump -I /tmp/file.rpd -P 3,5 -P 7 -p cef -t r`	Reads in the RealPort data file and displays control, event, and flow control information for ports 3, 4, 5, and 7. Each packet is displayed relative to the time that the first packet found in the file.

Exit status codes

This table lists exit status codes and the conditions or error types that cause them:

Exit status code	Condition
0	Normal completion
1	Fatal runtime errors; generates a message to `stderr`
2	Parameter errors; generates a message to `stderr`

RealPort API

CHAPTER 5

This chapter describes the functions and return codes of the RealPort API.

RealPortRegister

The RealPortRegister function makes a serial port available for use by the RealPort server.

Devices are enumerated to clients in the order in which they are registered. The device name is as it appears in the DDI device table.

Declaration

```
int RealPortRegister (char * device);
```

Arguments

Name	Description
device	The name of the device listed in the DDI device table to register

Return values

Value	Description
REALPORT_SUCCESS	Successful registration
REALPORT_FAILURE_DEVICE_INVALID	Invalid or already registered device
REALPORT_FAILURE_ALREADY_STARTED	RealPort server already started
REALPORT_FAILURE_SYSTEM	System resources unavailable

Version information

Available since version 6.0.

RealPortStartServer

This function instantiates the RealPort server using the configuration values given in the REALPORT_SERVER_CONFIG_TYPE argument.

Devices are enumerated to clients in the order in which they are registered.

Declaration

```
int RealPortStartServer (REALPORT_SERVER_CONFIG_TYPE * rsct);
```

Arguments

Name	Description
rsct	Server configuration structure

Return values

Value	Description
REALPORT_SUCCESS	Successful instantiation
REALPORT_FAILURE_DEVICE_INVALID	Invalid or already registered device
REALPORT_FAILURE_PARAMETER_INVALID	Invalid argument
REALPORT_FAILURE_ALREADY_STARTED	RealPort server already started
REALPORT_FAILURE_SYSTEM	System resources unavailable

Version Information

Available since version 6.0.

REALPORT_SERVER_CONFIG_TYPE

This structure defines the configuration parameters needed for RealPort.

Declaration

```
typedef struct {
    unsigned long tcp_port;
    unsigned long max_concurrent;
    unsigned long priority;
    unsigned long timeslice'
} REALPORT_SERVER_CONFIG_TYPE
```

Members

Name	Description
tcp_port	Listener TCP port of the RealPort server.
max_concurrent	Maximum number of concurrent clients.
priority	Numerical priority base for RealPort threads. A valid priority value ranges from 0 to 31, where a value of 0 represents the highest priority.
timeslice	Number of time-ticks that RealPort threads are allowed to execute without checking to see if there are any other threads of the same priority ready to execute. A valid timeslice value is in the range of 1 through 0xFFFFFFFF.

Version information

Available since version 6.0.

RealPort return codes

These completion codes are returned by the RealPort API.

Members

Value	Description
REALPORT_SUCCESS	The operation was successful; no error.
REALPORT_FAILURE_DEVICE_INVALID	The device passed to the API call has already been registered or is not a valid DDI device.
REALPORT_FAILURE_PARAMETER_INVALID	An invalid argument was passed to the API call.
REALPORT_FAILURE_ALREADY_STARTED	A RealPort server is already running.
REALPORT_FAILURE_SYSTEM	An attempt to allocate system resources failed.

Version information

Available since version 6.0.

Index

N

O

P

R

S

T

U

W

Connectware™