ember

Application Note
5049
23 September 2010
120-5049-000G

EZSP-UART Host Interfacing Guide

For the EM2XX and EM35x Product Families

This document describes procedures for initial tests of a host connected to an Ember ZigBee processor using EZSP-UART. It assumes that you have already read the *UART Gateway Protocol Reference Guide* (120-3010-000). You should have a basic understanding of the UART Gateway protocol, as well as the signals needed by the UART interface.

This document applies to EmberZNet 4.1 or later.

Contents

Ember Corporation
47 Farnsworth Street
Boston, MA 02210
+1 (617) 951-0200
www.ember.com

Software Overview

Ember designed the EZSP-UART host software to be easy to adapt to most host gateway systems. You can build and run the software on a PC running either Linux or Windows (with Cygwin installed). This lets you develop and test your application on a PC, and then move it to a different product host processor with few changes.

A Network Co-Processor (NCP) runs the EmberZNet ZigBee stack and is controlled by the host processor through EZSP-UART commands. (EZSP stands for EmberZNet Serial Protocol.) The NCP may be a chip in either the Ember EM2XX or EM35x product family.

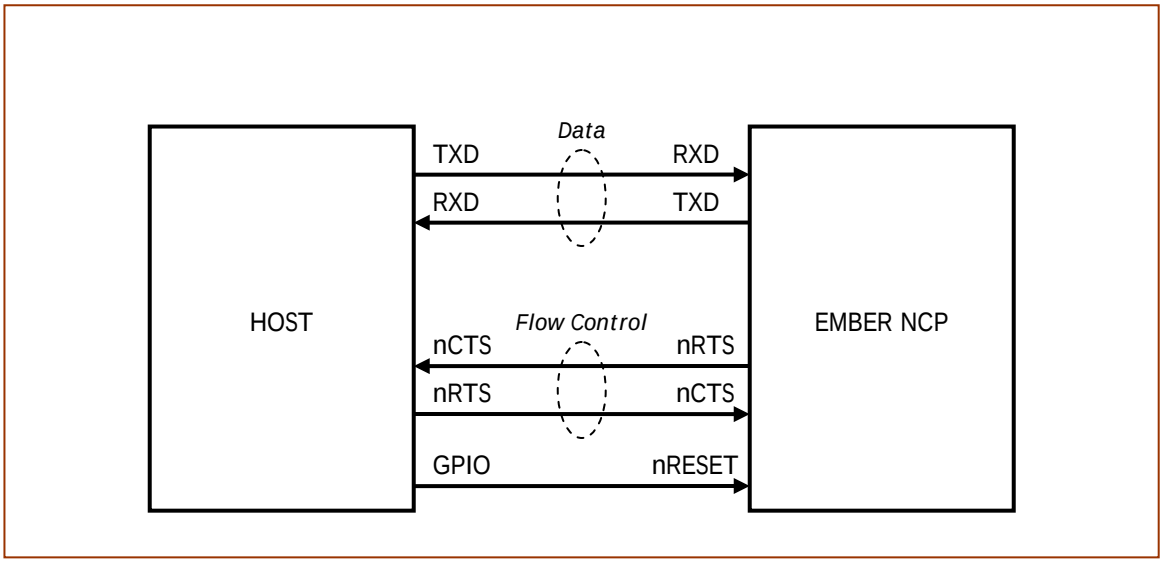
EZSP-UART host software does not interface directly to the operating system, but goes through an abstraction layer contained in the C source files ash-host-io.c (serial port I/O), ash-host-ui.c (console I/O), and system-timer.c (time of day). In most cases, these will be the only files that you will need to edit for a particular host system. To allow the same host software to run on both Windows and Linux, operating system calls use the POSIX API that is supported by Cygwin on Windows, and natively by Linux.

Three test programs help with EZSP-UART host development: uart-test-1, uart-test-2, and uart-test-3. These test programs are much simpler than a full-fledged gateway sample application and thus can more efficiently pinpoint EZSP-UART interface problems. They are controlled by command line options when invoked and interactive console input during execution. Since the purpose of these tests is to validate proper operation of the serial interface between the host and the NCP, they do not enable any wireless functions.

Interface Signals

Figure 1 illustrates the signals the EZSP-UART hardware interface uses.

Figure 1. NCP UART hardware interface signals



- Serial data: TXD and RXD
  The ASH protocol sends data in both directions, so both TXD and RXD signals are

required. An external pull-up resistor should be connected to TXD to avoid data glitches while the NCP is resetting.

- Flow control: nRTS and nCTS (optional)
  ASH can use either XON/XOFF or RTS/CTS flow control, but will usually have the best performance using RTS/CTS. In RTS/CTS flow control, nRTS enables transmission from the host to the NCP, and nCTS enables NCP transmissions to the host.

  When RTS/CTS flow control is used, network gateways will almost always need to support nRTS, but usually the host will not need to throttle transmission by the NCP. In that case, nCTS may be left unconnected since it has an internal pull-down and will be continuously asserted.

- Reset control: nRESET
  The host must be able to reset the NCP to run the ASH protocol, and the best way to do this is using a host output connected to nRESET. If this is not feasible, the host can send a special ASH frame that requests the NCP to reboot, but this method is less reliable than asserting nRESET.

RTS/CTS flow control is required for operation at 115,200 bps. (Other baud rates may be selected by a configuration value programmed into flash during manufacturing.) If RTS/CTS flow control cannot be supported by the host hardware, XON/XOFF may be used at 57,600 bps.

Note: The EM260 and EM35x Breakout Boards make RTS and CTS available with RXD and TXD. However, the EM250 Breakout Board does not make RTS and CTS immediately available with RXD and TXD. To use EZSP-UART with RTS/CTS on the EM250 Breakout Board, add four wires to bring the RTS/CTS flow control signals to the external DB-9 serial connector:

1. Connect U6 pin 7 (TR2_OUT) to J17 pin 8 (RTS).
2. Connect U6 pin 8 (RX2_IN) to J17 pin 7 (CTS).
3. Connect U6 pin 9 (RX2_OUT) to GPIO11 in J6 (nCTS).
4. Connect U6 pin 10 (TR2_IN) to GPIO12 in J6 (nRTS).

J17 is the DB-9 serial connector, U6 is the TTL-RS232 transceiver (LTC1386CS) above J17, and J6 is the GPIO scratchpad area.

Configure the jumpers on J5, the Extended Debug Connector to the left of the scratchpad area, so its signals do not conflict with those of the DB-9 serial connector. Remove the jumpers between the J5 pins 21-22, 23-24, 25-26 and 27-28.

It is very desirable for the host to able to reset the NCP via the nRESET signal. The connection from the host to nRESET can be a GPIO output, or if supplied by its serial port, the DTR signal. The EM260 Breakout Board and the EM35x EZSP Host Breakout Board permit either kind of reset connection where the onboard USB interface supplies DTR (inverted). If using the TTL interface or the other Breakout Boards, then a direct connection to nRESET must be used.

Note: A PC UART or a USB serial adapter uses RS-232 EIA voltage levels that must be converted to NCP 3.3V logic levels.

If connecting to the EM260 Breakout Board TTL interface, a RS-232/TTL level converter such as the B & B Electronics model 232LPTTL-33 is required. In this case DTR cannot be used to drive the EM260 nRESET input.

The EM250 and EM35x Breakout Boards have an RS-232/TTL level converter so an external converter is not required.

RTS/CTS Flow Control

The NCP's nRTS output holds off host transmission to prevent data loss due to buffer overflow. The NCP stores received serial data using DMA into two 32-byte buffers. When a buffer is filled, it deasserts nRTS and then reasserts nRTS as soon as there is an empty buffer that can receive more data.

To avoid receive overflow on the NCP, the host processor must not send more than 32 bytes after its CTS input is deasserted. The command line option `-o 0` may improve host flow control performance if it is does not stop sending quickly enough.

If needed, the host can restrain the NCP from sending to it by deasserting the NCP's nCTS input. The NCP will stop sending immediately, except to finish sending a byte already in progress.

XON/XOFF Flow Control

The NCP implements a one-way version of XON/XOFF in which the NCP tells the host to stop sending it data by sending an XOFF; when it can accept serial data again, it sends an XON. The host does not have the same capability, and any XON or XOFF bytes sent by the host are ignored by the NCP.

The NCP sends an XOFF when its receive buffer fills up to a threshold, and sends an XON when the buffer drains down to a second lower threshold. If it does not receive any data from the host for a few seconds, it will send up to 3 more XONs. It also sends an XON after it is reset. These additional XONs prevent EZSP-UART from hanging if an XON is lost or corrupted by noise, or if another byte sent by the NCP is corrupted into an XOFF.

Serial Port Selection

EZSP-UART serial port names follow Linux usage. A traditional UART serial port is typically named dev/ttyS*n* (where *n* is the unit number), and a USB virtual serial port is usually named /dev/ttyUSB*n*. Since there are several port naming conventions, the port name and number are stored as a string in the ASH host configuration.

Linux serial port numbers start at 0, and those greater than 9 may be expressed as two-digit decimal numbers or as upper- or lower-case letters. Thus, /dev/ttyS10 on one system might be /dev/ttySa and /dev/ttySA on another.

Because Windows serial port numbering starts at 1, the options `-p COMn` or `–p n` under Cygwin are converted to the device name /dev/ttyS*n-1*.

The following examples show how `–p` values are converted to actual device names under Windows and Linux:

Windows

`-p 1`	/dev/ttyS0
`-p COM10`	/dev/ttyS9

Linux

`-p 1`	/dev/ttyS1
`-p ttyUSB10`	/dev/ttyUSB10

Note: When using USB serial adapters under Windows, the registry may accumulate a large number of reserved serial port numbers. To reuse those port numbers, run regedit to delete them from HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\ Ports\.

EZSP-UART is a binary protocol and must operate the host serial port in "raw mode" to bypass input echoing, interrupt characters, and the like. Linux support for serial ports may differ between distributions, versions, and devices, so you may need to modify the function `ashInternalSerialnit()` in ash-host-io.c to achieve true raw mode operation.

Linux differs from Windows because serial port access usually requires root privileges, so you must set permissions properly to permit opening the port.

Command Line Options

All of the test programs accept the same command line options. In most cases the `-p` option is the only one that will be needed, but several other options may be useful.

`-n (0,1)`
This option specifies the type of NCP to which the host is connected.

The choices are as follows:

`-n 0` specifies an EM2XX or EM35x using RTS/CTS at 115,200 bps

`-n 1` specifies an EM2XX or EM35x using XON/XOFF at 57,600 bps

`-p <port>`
This option is usually required to specify the host serial port. The port can be given as a string (e.g., ttyUSB0) or as a number that is translated into "/dev/ttySn". If this option is omitted, the default serial port "/dev/ttyS0" (COM1 in Windows) is used.

`-r {d, c, r}`
This option tells EZSP-UART how to reset the NCP. There are three choices: `d` for DTR, `c` for custom, and `r` for a soft reset using the RST frame.

- The `d` choice is usable with the EM260 Breakout Board USB interface, EM35x EZSP Host Breakout Board USB interface, or any configuration in which the host serial port can provide a logic-level DTR output to the NCP. DTR must be inverted to provide the proper polarity at nRESET. Note that this choice should not be used with other EM2XX or EM35x Breakout Boards unless DTR has been manually wired up to the NCP.

- The `c` choice applies when the host has a GPIO output connected to nRESET. To use this reset method, the placeholder function `ashResetCustom()` in ash-host-io.c must be edited to control the proper GPIO output.
- The `r` choice sends a RST frame to the NCP that asks it to reset itself. Although useful for development, this method will fail if the NCP is incapable of receiving or responding to UART inputs. The `r` choice is the default reset method.

`-t {0,1, …, 15}`

This option controls outputting host diagnostic trace information to stderr. The value of this option specifies four bit flags that enable various trace outputs.

- Bit 3 enables EZSP frame tracing. A trace line is output whenever EZSP sends a command, receives a response, or adds or removes data from the host receive queue. The trace data identifies the EZSP action and its associated frame ID.
- Bit 2 enables event tracing, such as connecting, disconnecting, and various errors. These events are similar to those sent by the NCP that can be captured and viewed using InSight Desktop. (All tracing is disabled by default, but event tracing is enabled in uart-test-2.)
- Bit 1 enables verbose frame tracing. Every frame sent or received by the host is output, along with various internal ASH variables.
- Bit 0 enables brief frame tracing. Every frame sent or received by the host is output. (Note that Bit 1 takes precedence over Bit 0.)

For example, `-t 4` enables only host event tracing, similar to the events output by the NCP that can be captured using InSight Desktop.

`-o {0,1}`

`o 0` has the host write to its serial port one byte at a time, instead of a frame at a time. In some cases this can improve how quickly a host can stop sending to the NCP in response to flow control, although it may slightly decrease throughput.

`-o 1` selects the default frame at a time serial output by the host.

`-h`

This displays the command line options accepted by `ashProcessCommandOptions()`. Ember recommends that only the previously listed options should be used with uart-test-1, uart-test-2, or uart-test-3. The following is the output of the `–h` option.

```
$ ./uart-test-2  -h
Usage: uart-test-2 {ncp type} {options}
  ncp type:
    -n 0,1,2          0=EM2xx/EM3xx @ 115200 bps, RTS/CTS
                      1=EM2xx/EM3xx @ 57600 bps, XON/XOFF
                      (if present must be the first option)
  options:
    -b <baud rate>    9600, 19200, 38400, 57600, 115200, etc.
    -f r,x            flow control: r=RST/CTS, x=XON/XOFF
    -h                display usage information
    -i 0,1            enable/disable input buffering
    -o 0,1            enable/disable output buffering
    -p <port>         serial port name or number (eg, COM1, ttyS0, or 1)
```

```
-r d,r,c          ncp reset method: d=DTR, r=RST frame, c=custom
-s 1,2            stop bits
-t <trace flags>  trace B0=frames, B1=verbose frames, B2=events, B3=EZSP
-v <offset>       select backchannel server port offset, 4900 by default
-x 0,1            enable/disable data randomization
```

uart-test-1

uart-test-1 should be the first program you use to test the host-to-NCP UART connection. It performs the following four steps:

1. Opens the serial port.
2. Asks the user to reset the NCP.
3. Listens for input from the NCP.
4. Sends data to the NCP and waits for a response.

Step 1: Open the serial port

This step confirms that a serial port of that name exists, can be accessed by the program, and can support a baud rate of 115,200 bps. If this step fails, try running uart-test-1 again with host event trace enabled using the command line option `-t 4` for further diagnostic information.

Some of the error messages (as shown in Windows) for this step, and possible causes, are:

`Serial port /dev/ttyS0 open failed: Permission denied`
Possible cause: the serial port is in use by another program.

`Serial port /dev/ttyS0 open failed: No such file or directory`
Possible cause: the serial port doesn't exist—check the device name and port number.

`Could not set baud rate to 115200 bps`
Possible cause: the serial port cannot operate at 115,200 bps.

Note: Opening a serial port or setting its attributes may fail in Windows if a program using the same port exited abnormally, such as being terminated by CTRL+C. If host event tracing is enabled, an error message like the following may be output:

```
7 [main] uart-test-1 5276 fhandler_serial::open: couldn't set
initial state for /dev/ttyS0, Win32 error 995
```

Step 2: Ask the user to reset the NCP

The program asks the user to manually reset the NCP—on a Breakout Board this is done by pressing the RESET pushbutton—and then press ENTER. This is necessary to verify

that the host can receive data from the NCP without relying on data transmission in the other direction.

Step 3: Listen for input from the NCP

This verifies that the NCP and its serial data connection to the host are working correctly.

When the NCP is reset, it takes approximately one or two seconds to reboot and then sends a 7-byte RSTACK (reset acknowledge) frame to the host:

0x1A, 0xC1, 0x02, 0x02, 0x9B, 0x7B, 0x7E

The fourth byte, 0x02, specifies the reason for the NCP reset: power-on. Note that the EM2XX cannot distinguish between resets due to power-on and one due to asserting nRESET: both return "power-on" as the reset reason.

The host will wait up to 10 seconds for the RSTACK frame before this step fails. Some possible causes for a failure include:

- The wrong host serial port was opened
- The NCP is not powered up or is being held in reset
- The connection from the NCP's TXD to the host's RXD is not correct
- A level converter is needed between the NCP's TXD and the host's RXD
- The NCP is inhibited from sending because its nCTS input is held high
- The host and NCP baud rates are not equal (`-b` option was used)
- The host and the NCP are running different versions of the ASH protocol.

Note: uart-test-1 disables RTS/CTS flow control on the host. On some hosts this may cause the host's RTS output to be deasserted. If this occurs, the NCP's nCTS input must be disconnected for this test or else it will not be able to transmit. (EZSP-UART enables an internal pulldown resistor on nCTS.)

To disconnect the NCP's nCTS input from the host:

1. On the EM260 Breakout Board, remove jumper J18.
2. On the EM250 Breakout Board, temporarily disconnect the wire between J6 pin 13 and U6 pin 9.
3. On the EM35x Breakout Board, remove jumper J25.
4. On the EM35x EZSP Host Breakout Board, remove jumper J27.

Step 4: Send data to the NCP and wait for a response

This step verifies that the host can transmit data to the NCP.

The host sends a 5-byte RST (reset) frame to the NCP:

0x1A, 0xC0, 0x38, 0xBC, 0x7E

When the NCP receives the RST, it reboots and after about 1.4 seconds sends a RSTACK frame to the host:

0x1A, 0xC1, 0x02, 0x0B, 0x0A, 0x52, 0x7E

The reset cause byte, 0x0B, indicates that the reset was due to a software reboot.

If this step fails, these are some possible causes:

- The connection from the host's TXD to the NCP's RXD is not correct
- A level converter is missing between the host's TXD and the NCP's RXD

Note: These RST and RSTACK frames are unusual in having a Cancel Byte (0x1A) sent before them. This tells the receiver to ignore any superfluous bytes input when the serial ports were being initialized, level shifter was powering-on, and so forth.

uart-test-2

The uart-test-2 test verifies that EZSP can be initialized, validates the EZSP version, and then exercises the serial link using some simple EZSP commands. It is a prerequisite that uart-test-1 can execute successfully. Since this test's purpose is to exercise only serial communication, it does not use any wireless functions.

uart-test-2 performs the following steps:

1. Opens the serial port and resets the NCP.
2. Checks that the NCP EZSP version is compatible.
3. Sends multiple ezspEcho commands to the NCP.

Step 1: Open the serial port and reset the NCP

This step uses the reset method specified on the command line or by sending a RST frame by default. This may fail for the following reasons:

- The selected reset method is not able to reset the NCP.
- Either the host or the NCP is inhibited from sending due to incorrect RTS/CTS flow control inputs.
- Under Windows, the previous user of the port exited abnormally (by pressing CTRL+C, for example).

Step 2: Verify the EZSP version

The host sends an ezspVersion command, and verifies that the NCP EZSP version returned is compatible. If not, the expected NCP version and what was read back is displayed.

Step 3: Send multiple ezspEcho commands

In this step any of several variations on the ezspEcho test may be run. The user is prompted to enter the number of the test to run; pressing ENTER with no test number displays a list of the tests available.

```
Enter test number (1 to 11), s for statistics or q to quit:
Test     Repeat  Data Length   Timer
Number   Count   Min    Max    Period
                               (msec)
  1        10     10     10      0
  2        50    100    100      0
  3       200      1    100      0
  4       200      1    100    100
  5       200      1    100     40
  6       200      1    100     25
  7         0     10     10      0
  8         0      1    100      0
  9         0      1    100    100
 10         0      1    100     40
 11         0      1    100     25
(repeat count 0 means run forever)
(timer period 0 disables timer callbacks)
```

The tests differ in how many ezspEcho commands are sent and the minimum and maximum length of their data payloads. When the minimum and maximum are not equal, the payload length increments starting at the minimum up to the maximum, and then restarts at the minimum again. To terminate a test with a zero repeat count and which therefore can run forever, press Enter.

Some tests have the NCP send timer callbacks at regular intervals. These callbacks will arrive interspersed with ezspEcho responses. A simple timer callback handler counts the number of callbacks received.

Selecting longer ezspEcho payloads and more repetitions may cause the NCP to ask the host (using the selected flow control method) to hold off transmitting. It is possible, though not very likely, that problems with flow control could lead to communication errors during these tests. When the ezspEcho test finishes, the elapsed time is output, as well as the number of timer callbacks received if these were enabled. The following example was produced on a Windows PC connected to an EM260 Breakout Board via the USB input.

```
Enter test number (1 to 8), s for statistics or q to quit: 6
Test 6
Sending 200 ezspEcho commands with 1 to 100 bytes of data,
with timer callbacks every 25 milliseconds... succeeded.
Received 352 timer callbacks.
Elapsed time: 8.808 seconds.
```

Host event tracing is enabled during these tests. Normally no errors or events should be output, but the following are some that might occur during a test:

```
0.062 Rec'd frame: CRC error
3.250 Timer expired waiting for ACK
```

uart-test-3

uart-test-3 verifies that the NCP is able to use flow control, either RTS/CTS or XON/XOFF, to hold off transmission from the host. Because flow control is only required

when the NCP is heavily loaded with network activity, in normal use it is difficult to determine whether it is working properly. Since EZSP-UART automatically recovers from occasional errors such as buffer overflows, it may not be apparent when flow control is not working correctly, but if it isn't, the result will be lower throughput when the network is busiest.

Before performing uart-test-3, be sure to verify that uart-test-1 and uart-test-2 both execute successfully.

uart-test-3 opens the host serial port, resets the NCP, connects using EZSP, and verifies that the host and NCP versions of EZSP and ASH are compatible. Then it pauses and waits for the user to press Enter before performing the flow control test.

The host starts the test by sending an ezspDelayTest command to the NCP, which makes it delay before reading the next command. Then the host sends an ezspEcho command that is much larger than the NCP receive buffer. Because the NCP delays reading out of the buffer, it will overflow unless flow control is able to make the host pause transmitting.

After the ezspEcho command is complete, the host reads NCP error counters to see whether the NCP receive buffer overflowed. It also verifies that the host did not need to retransmit the ezspEcho command. If either occurred, flow control did not work properly and the test has failed.

The following examples show the output of uart-test-3 when it succeeds and when it fails.

```
Opening serial port and initializing EZSP... succeeded.
Checking EZSP version... succeeded.

Press ENTER to test RTS/CTS flow control:

Succeeded.
```

```
Opening serial port and initializing EZSP... succeeded.
Checking EZSP version... succeeded.

Press ENTER to test XON/XOFF flow control:

Failed.
```

EZSP Fatal Errors

If one of the UART tests encounters a fatal EZSP error, it prints out the error before it exits. The following errors are those most likely to be seen when running uart test programs.

- Host error: EZSP_ASH_ERROR_VERSION

  The ASH version read from the NCP is not compatible with the host ASH version.

- Host error: EZSP_ASH_ERROR_TIMEOUTS

The NCP repeatedly failed to acknowledge a command sent from the host within the time. The number of consecutive failures is a host configuration parameter, normally set to 3. This error can be caused by a high error rate on the serial line in either direction, including complete disconnection, or if the NCP loses power, is stuck in reset, or is not operating for any other reason.

- Host error: EZSP_ASH_ERROR_RESET_FAIL

  The host did not receive an RSTACK from the NCP within the time allowed after it attempted to reset it. This can be due to mismatched host and NCP baud rates (or other serial port parameter), use of the wrong reset method, or a disconnected or very noisy serial connection.

- Host error: EZSP_ASH_ERROR_NCP_RESET

  The host received RSTACK from the NCP when it did not reset it. This could be due to a power loss or brownout on the NCP, or glitches on the NCP nRESET signal.

- Host error: EZSP_ASH_ERROR_SERIAL_INIT

  The host was not able to initialize its serial port to the requested characteristics. This could be due to specifying the wrong serial port number, the baud rate not being supported by the port, or numerous other problems.

- Host error: EZSP_ASH_ERROR_XON_XOFF

  The host ASH layer received an XON or XOFF byte from the serial port driver when XON/XOFF is in use as the flow control protocol. This can happen if the serial port was not properly set up to use XON/XOFF.

- EZSP error: EZSP_ERROR_OVERFLOW

  The NCP had more callbacks waiting to be sent to the host than it could store in its memory, so one or more had to be discarded. This error should only occur in uart-test-2 for tests that include timer callbacks. This error could result from a low baud rate, a noisy serial connection, or if the host holds off the NCP too much using RTS/CTS flow control.

- EZSP error: EZSP_ERROR_NO_RESPONSE

  The host did not receive the response to an EZESP command within the allowed time. This error can result if the host is randomizing data and the NCP is not, or vice versa.

Statistics and Errors

Typing **s** after running one of the tests in uart-test-2 displays various performance and error statistics. The following example shows the statistics from running test 6 on a Windows PC connected to the EM260 TTL serial port.

```
Enter test number (1 to 11), s for statistics or q to quit: s

Host Counts        Received Transmitted
Total bytes           13664       13586
DATA bytes            11752       10900
I/O blocks              392         500

Total frames            413         613
DATA frames             413         200
ACK frames                0         413
```

```
NAK frames                0           0
Retry frames              0           0
Cancelled                 0           0
nRdy frames               0           0

Host Receive Errors
CRC errors                0
Comm errors               0
Length < minimum          0
Length > maximum          0
Bad controls              0
Bad lengths               0
Out of buffers            0
Retry dupes               0
Out of sequence           0
ACK timeouts              0

NCP Counts
Overflow errors           0
Overrun errors            0
Framing errors            0
```

Most of this data represents the host's view of the serial link, except for the NCP counts that are read using the ezspReadAndClearCounters command. These counts record serial port errors that occurred on the NCP.

This data can be helpful in diagnosing problems with the serial connection between the host and the NCP. Some connection problems and the statistics that they could affect are listed next.

- Host CTS to NCP nRTS flow control is not working
  - Received: NAK frames, Retry frames
  - Transmitted: Retry frames
  - ACK timeouts
  - NCP receive overflow errors
- NCP nCTS to host RTS flow control is not working
  - Received: Retry frames
  - Transmitted: NAK frames, Retry frames
  - CRC errors
  - ACK timeouts
- Noise on the serial line (both directions)
  - Received: NAK frames, Retry frames
  - Transmitted: NAK frames, Retry frames
  - CRC errors
  - ACK timeouts
  - NCP framing errors
- Host baud rate is not accurate
  - Received: NAK frames, Retry frames
  - Transmitted: NAK frames, Retry frames

- CRC errors
- ACK timeouts
- NCP framing errors

NCP Debug Events

The NCP outputs various debug events to the InSight Adapter that can be captured and viewed using InSight Desktop.

ASH Started: The NCP was reset. This is only output if the reset was not due to an error.

ASH Disconnected: A serious error caused the NCP to enter the error state. The NCP will send the error code to the host. Tests will print the value of this error code if they receive it from the NCP.

ASH Timed out waiting for ACK: An ACK for an NCP data frame was not received within the timeout period. Either a frame sent from the NCP to the host was lost, or the host's ACK of the frame was lost. The NCP will retry sending the frame.

ASH Rec'd frame with CRC error: The NCP received a frame with an invalid CRC. The frame may have been corrupted by noise, or data may have been lost due to receive buffer overflow.

ASH Rec'd frame out of sequence: The NCP received a frame that was out of sequence. This is usually because a previous data frame from the host was lost due to noise or because the NCP receive buffer overflowed.

After Reading This Document

If you have questions or require assistance with the procedures described in this document, contact Ember Customer Support. The Ember Customer Support portal provides a wide array of hardware and software documentation such as FAQ's, reference designs, user guides, application notes, and the latest software available to download. To obtain support on all Ember products and to gain access to the Ember Customer Support portal, visit http://www.ember.com/support_index.html.

Ember Corporation
47 Farnsworth Street
Boston, MA 02210
+1 (617) 951-0200
www.ember.com