

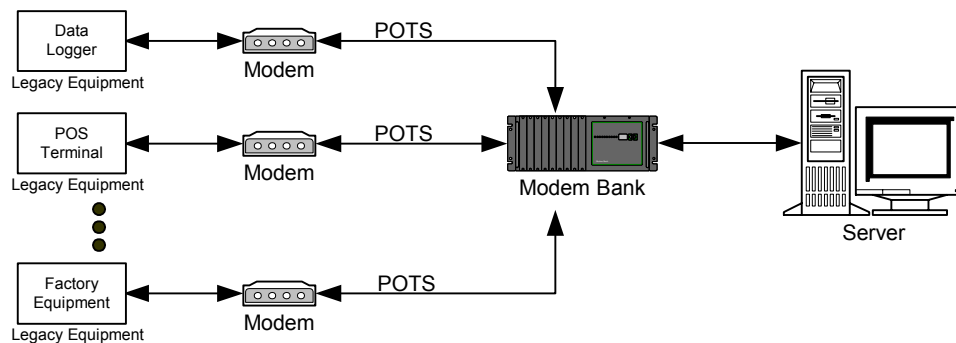
INTRODUCTION

For many applications, the greatest obstacle to adapt to the latest technology is incompatibility with existing hardware. While it is technically feasible, the associated costs are usually prohibitive. As the networking trend continues to grow, there exist devices that do not understand the Internet Protocol (IP) — the underlying technology for the World Wide Web (WWW). With its onboard TCP/IP network stack, the Tiny InterNet Interfaces (TINI[®]) platform can add network capabilities to legacy equipment almost on-the-fly. This application note presents a software virtual modem implemented on a TINIm390 Verification Module. All references to “device” hereafter refer to entities/equipment that are designed to communicate with a modem and do not have any Ethernet networking capability.

SYSTEM OVERVIEW

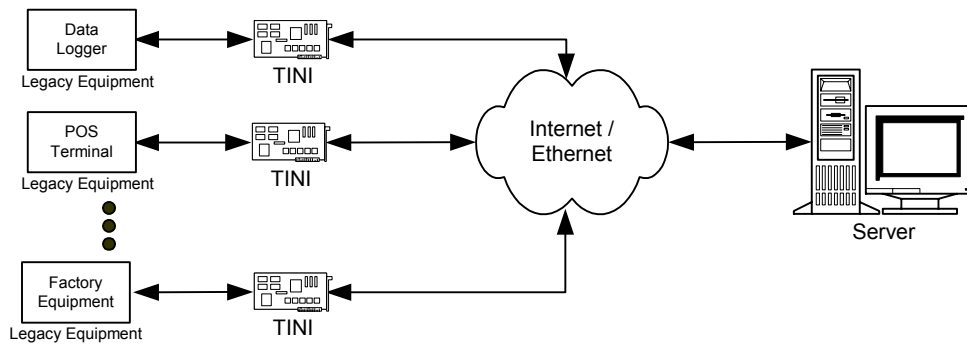
Figure 1 shows the typical setup of connecting the device to a server via a modem. For such devices, there is usually one phone line associated with each modem on the client premises. On the server side, a modem bank is installed to collect data from the modems and transfer the data to the server.

DEVICES COMMUNICATING TO A SERVER VIA MODEM Figure 1



With TINI in place of the modem, the device can now make use of network infrastructure. Data is transferred directly via Ethernet to the server. Note the elimination of POTS and modem bank between the legacy device and the server as depicted in devices communicated to server via TINI in Figure 2. This makes the device more network-centric and allows for easy integration and interaction with other network components.

DEVICES COMMUNICATED TO SERVER VIA TINI Figure 2



HARDWARE

The following components are used to construct the virtual modem.

E20 Sockets board (Rev C¹)

TINIm390 (Rev D)

The virtual modem makes use of RTS and CTS signals. These signals are located at J3 on the E20 socket board. Earlier revisions do not supply these control signals. A null-modem cable is used to connect J3 (DTE) to the device. For more information on hardware setup, please read `readme.txt` included with the source code from <ftp://ftp.dalsemi.com/pub/tini/appnotes/AN196/VirtualModem.zip>.

¹ Socket board and module schematic are available from <http://www.maxim-ic.com/products/tini/chipsetrefdesign.cfm>

TINI SOFTWARE

`VirtualModem` is the super class for all the virtual modem classes. It defines modem specific data structures such as registers, command syntax, flow control, etc. `VirtualModem` provides the following public methods:

```
public int ring()
```

The `ring` method signals an incoming connection request, e.g. an incoming socket connection attempt. Depending on its auto-answer state, the modem will automatically answer the call. Otherwise, it will just write "RING" (or the equivalent numeric code).

```
public void reset()
```

This method simulates a hardware reset of the `VirtualModem` by loading the default register settings and aborting all data connections.

```
public void notifyDTR(boolean state)
```

This method is invoked when the state of DTR changes. The `state` variable indicates whether or not DTR is asserted.

```
public void processInput()
```

The `processInput` method handles input data from the inbound stream.

The following abstract methods are defined for subclasses to override:

```
abstract int answer()
```

This method is called when the command interpreter reads the ATA command. The subclass of this command could accept an incoming socket connection. This method is also responsible for updating the `dataIn` and `dataOut` streams.

```
abstract int dial(String destination)
```

The `dial` method is invoked when the command interpreter reads the ATD command. The subclass of this command could connect a socket to a remote computer. `dial` is required to update `dataIn` and `dataOut`.

```
abstract void setDCD(boolean state)
```

This method asserts or deasserts DCD as specified by `state`.

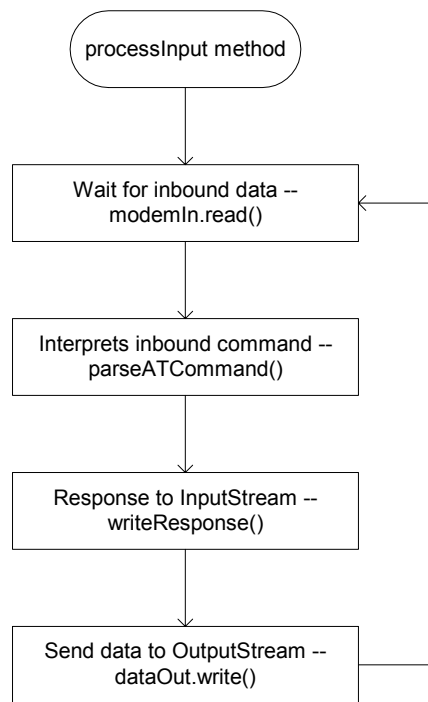
```
abstract void setFlowcontrol(int mode)
```

This method sets the flow control usage as specified by `mode`.

```
abstract void setReceiveTimeout(int ms)
```

The `setReceiveTimeout` method sets the timeout value (specified by `ms`) on the modem input stream.

All network and serial I/O is managed in the `processInput` method. The program flow of the `processInput` method is shown in Figure 3.

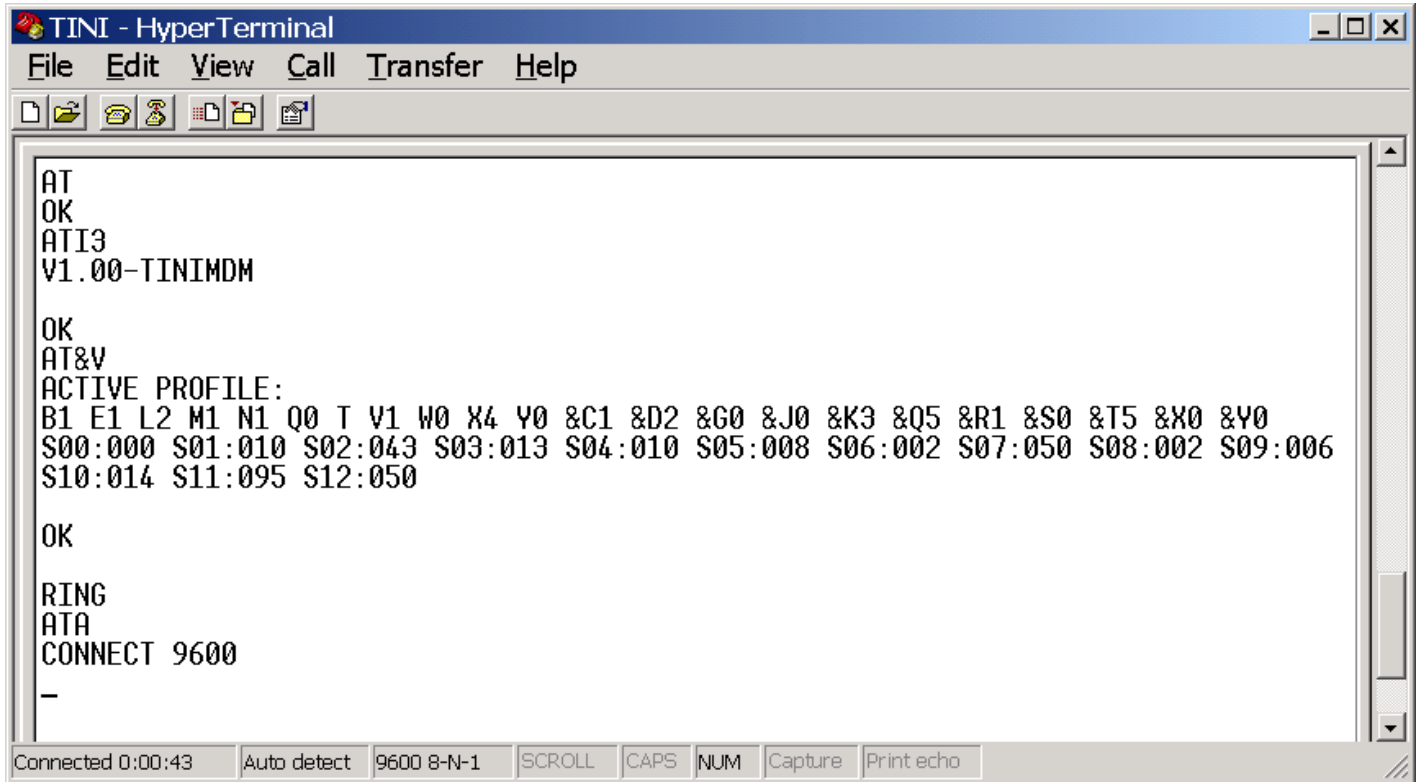
VIRTUALMODEM PROCESSINPUT METHOD FLOW Figure 3

The TCPSerialVirtualModem class provides the following constructor:

```
public TCPSerialVirtualModem(SerialPort serial, int speed, int tcpport)
```

The `serial` parameter specifies the serial port to which this virtual modem is connected. The `speed` parameter indicates the speed of the serial communication in bits per second and the `tcpport` parameter sets the TCP port on which the virtual modem awaits inbound connections.

The TCPSerialVirtualModem converts a serial input stream to a TCP/IP output stream and vice versa. Figure 4 shows a sample session. To provide a common test case, a TINI with Ethernet access is connected to the serial port of a PC. As can be seen from the display, the systems behavior is identical to that of a modem.

SAMPLE COMMUNICATION SESSION Figure 4

The image shows a HyperTerminal window titled "TINI - HyperTerminal". The window has a menu bar with "File", "Edit", "View", "Call", "Transfer", and "Help". Below the menu bar is a toolbar with icons for file operations. The main text area contains the following text:

```
AT
OK
ATI3
V1.00-TINIMDM

OK
AT&V
ACTIVE PROFILE:
B1 E1 L2 M1 N1 Q0 T V1 W0 X4 Y0 &C1 &D2 &G0 &J0 &K3 &O5 &R1 &S0 &T5 &X0 &Y0
S00:000 S01:010 S02:043 S03:013 S04:010 S05:008 S06:002 S07:050 S08:002 S09:006
S10:014 S11:095 S12:050

OK

RING
ATA
CONNECT 9600
-
```

At the bottom of the window, there is a status bar with the following information: "Connected 0:00:43", "Auto detect", "9600 8-N-1", "SCROLL", "CAPS", "NUM", "Capture", and "Print echo".

CONCLUSION

Using the Ethernet-ready TINI platform, TCP/IP capability can easily be added to a legacy device that is designed to communicate with a modem. From the system integration point of view, no changes are needed on the client premises. Instead of listening to a modem bank, the server now listens on a TCP/IP port. With minimal changes, equipment can be online with the other network components and can make use network infrastructure.

MAXIM INTEGRATED PRODUCTS / DALLAS SEMICONDUCTOR CONTACT INFORMATION

Company Addresses:

Maxim Integrated Products, Inc

120 San Gabriel Drive
Sunnyvale, CA 94086
Tel: 408-737-7600
Fax: 408-737-7194

Dallas Semiconductor

4401 S. Beltwood Parkway
Dallas, TX 75244
Tel: 972-371-4448
Fax: 972-371-4799

Product Literature / Samples Requests:

(800) 998-8800
(408) 737-7600

World Wide Web Site:

www.maxim-ic.com

Product Information:

<http://www.maxim-ic.com/MaximProducts/products.htm>

Ordering Information:

<http://www.maxim-ic.com/BuyMaxim/Sales.htm>

FTP Site:

<ftp://ftp.dalsemi.com>