This paper describes a local area network (LAN) used to link PCs to other serial devices using an RS-232-C interface. The hardware and software elements of the network controller are explained in terms of the actual implementation at Atmel Corporation. The LAN design reflects a strategy to provide a transparent switching capability using only RS-232-C data transmission.

**INTRODUCTION**

Anyone who has worked with serial devices such as PCs and printers can appreciate the inherent difficulties of solving the simple problem of trying to share a printer with two or three PCs using an RS-232-C interface. Once swapping cables has become tiresome, switch boxes are a fairly common remedy. Yet the use of a manual switch presents reliability problems and offers no means to control the flow of data.

Our aim at Atmel is to support several PCs connecting to several serial devices. Other than providing the capability to switch the connections between PCs and serial devices, the network controller must determine whether an existing serial device is busy before allowing the connection to be made. The definition of busy in our environment is the continuous flow of data, in either direction, between PC and serial device. With these simple network requirements in mind, we attempted to find an existing local area network (LAN) system we could use.

We found no LAN system that was suited to directly controlling devices with an RS-232-C interface, which led us to design a LAN of our own. It was only by drastically limiting the conception we had of what a network system should do that we were able to make the LAN discussed in this paper come about.

**LAN SYSTEM REQUIREMENTS**

The overall system requirements for our LAN were dictated primarily by the type of hardware we use at our plant and its use in both an engineering and production capacity. As shown in Figure 1, we had six IBM-compatible PCs linked through a PC network to a shared disk, and 40 semiconductor memory test devices. The test devices use an RS-232-C interface, at baud rates up to 19.2K, as a means to load and debug test programs with interactive commands. In production mode, the test devices are controlled by device handlers through a parallel interface, and are at that time capable of running independently of the serial interface. A communication line was established through a telephone patch panel by connecting a cable from a PC slot to a test device slot. We retained this innovative use of a patch panel to serve as a backup system while we checked out the new LAN.

The LAN we developed replaced the telephone patch panel shown in Figure 1 with a serial network controller (SNC). Figure 2 shows the corresponding LAN topology, where PCs are shown circled since we do not support PC-to-PC communication. The SNC controls the selection of test devices (TD) by PCs as well as the routing of data between PCs and test devices.

At the time we designed the local area network, we had plans to move to a new building and came up with the following requirements to meet our anticipated growth:

- Control RS-232-C transmission of ASCII data at baud rates up to 19.2K
- Support simultaneous communication between up to 16 PCs and 100 test devices, controllable from any PC
- Include a mechanism to prevent connection from
To support the 16 PCs and 100 testers we decided we would need in our manufacturing area, the resulting hardware would require 58 DUARTs. We could easily fit eight 28-pin DUARTs on a single PC board, and use a 50-pin connector to wire to 16 external lines, which was the resulting PC board configuration we used. This allows for systems supporting as few as 16 communication lines. Because of the limitation imposed by using 8-bit control codes, a maximum of nine PC boards, or 143 communication lines, can be supported: 16 PC lines plus 127 serial devices.

We designed the PC boards using the XR88C681 and DS1488/89 drivers/receivers. Other logic on the board is used to decode the board number and base address for memory mapping to control the DUARTs. Both board number and base address are jumper selectable. We limited the logic on each board to allow for one to eight boards in the network controller. This supports a maximum of 128 communication channels. The PC can be configured to support up to 512K of memory with a full network system of 64 DUARTs.

**LAN SOFTWARE IMPLEMENTATION**

The software elements of the system reflect an approach to minimize network management requirements. Essentially, the network must provide a way for any PC to connect with any of the test devices, provided that test device is not in use by another PC. Once a specific configuration of PCs and test devices and their baud rates are defined, the remaining network management criterion is determining whether a communication line to a specified test device is in use.

<table>
<thead>
<tr>
<th>PC Busy Flag</th>
<th>Test Device (TD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>0</td>
<td>PC 1 requests connection to TD100</td>
</tr>
<tr>
<td>0</td>
<td>SNC returns busy code, clears busy</td>
</tr>
<tr>
<td>100</td>
<td>PC 1 again requests connection to TD100</td>
</tr>
<tr>
<td>0</td>
<td>SNC returns connect code and connects</td>
</tr>
</tbody>
</table>

A busy flag is set during the SNC polling loop any time data is transmitted on an established communication line. A request for a test device in use by another PC causes the busy flag for the PC connected to the requested test device to be cleared after it is examined. If no data transfer has occurred either way on the line by the time of the second request, the test device is disconnected from the current PC and attached to the requesting PC. It is thus left to the application program running on the PC to determine how long a time interval is adequate before
making a second request for a busy test device.

For example, suppose PC 1 in Figure 3 requests connection to test device 100. The busy flag of PC 2 is set to 0 (not busy) and the SNC returns a busy code to PC 1. If PC 1 makes the same request again, and the busy flag is still 0, the test device is then assigned to PC 1, and PC 2 is disconnected from that test device.

For transmission of 7-bit ASCII data, the SNC acts as a transparent medium to route data between a PC and its connected test device. When a PC sends an 8-bit code with the MSB set, the SNC interprets this as a network request:

- 128 disconnect from test device
- 129+ values through 240 request connection to test device 1 to 112

After receipt of a network request, the SNC returns an 8-bit code to the requesting PC to indicate the status of the request:

- 1+ values through 112 indicate test device to which connection is made
- 200 invalid test device requested
- 201+ values through 216 indicate test device in use by PC 1 to 16

LAN RESPONSE TIME

By limiting the functionality of the SNC to selection of a test device, we achieved a minimal overhead on the processing of incoming data. Benchmarks on the polling loop running on an 80286 8 MHz processor show the processing of incoming data from 16 PCs and 16 test devices to take 220 microseconds. This equates to 6.8 microseconds per character. For 32 communication lines to run at 19.2K baud, a response time of 16.2 microseconds per character is required.

The processing of a request by a PC takes 20 microseconds, so this function, which is invoked only rarely, has no effect on the system throughput. The baud rates and channel assignments must be defined in a file before the network controller is started. This removes any versatility in channel assignments that would be required. A short diagnostic is run by each channel whenever the network software is started. We also developed a diagnostic program to run against new boards before their use in a system.

As we envision all the things we might get our network to do for us, we rest easily knowing that we have a solid system capable of meeting our basic needs.

The LAN we built was able to meet very closely all the system requirements listed above. The overall cost was just over $10,000, and we were unable to implement connection between a PC and serial device not running at the same baud rates (this was not a specific but desired requirement). Although we have had no failure of the network hardware in the six months it has been active, we are confident that as long as no more than two communication boards go out at one time, we can get the network back on line in less than one hour. We use six PC boards in the current system, supporting 12 PCs and 74 test devices. When it becomes necessary, we can use an expansion chassis to add two boards to the system, supporting an additional 32 test devices.

The cornerstone in the success of our LAN is limiting and automation of network management requirements. Starting with the network requirements, we included only those functions essential to our particular environment. The hardware was based on an IBM-compatible PC, which affords flexibility in system configuration at a low cost. With our specific definition of busy, the network controller responds automatically and without ambiguity to connection requests from PCs. The use of a polling loop ties the system response time to the speed of the processor, but at the same time eliminates system overhead from interrupt processing. The baud rates and channel assignments to PCs or test devices must be defined in a file before the network controller is started. This removes any versatility in channel assignments, but eliminates the associated management that would be required. A short diagnostic is run on each channel whenever the network software is started. We also developed a diagnostic program to run against new boards before their use in a system.

As we envision all the things we might get our network to do for us, we rest easily knowing that we have a solid system capable of meeting our basic needs.