9-Bit Protocol Tests

This is a copy of an actual report that was generated for one of our clients. They wanted to know if a 9-bit serial protocol could be managed using just the serial port drivers native to Windows. Their name and the details of the serial packets were omitted to protect them and their proprietary information.

I ran some tests for your 9-bit protocol on Windows 95C, 98, NT 4.0, 2K, and XP platforms.

The transmitted packet looked like this:

```
10  02  60  00  02  30  01  00  00  10  03  00  93
```

and the received packet was:

```
00  60  02  31  01  00  06  60  04  0C  1F  12  03  10  03  01  3E
```

So everything looks like it should. The one thing that is the most important is sending the 9-bit protocol by manipulating the parity setting in a Windows program. In the transmitted packet, the third byte has to be sent with MARK parity – all other bytes in the packet are sent with SPACE parity. This is how the packet is sent:

- The first two bytes are sent with SPACE parity.
- When the call to transmit the two bytes returns, the program delays for 2292 usecs (2 byte times at 9600 baud with 11 bits – start bit , 8 data bits, parity bit, stop bit).
- The parity is changed to MARK parity.
- The third byte is sent with MARK parity.
- When the call to transmit the byte returns, the program delays for 1146 usecs (1 byte time at 9600 baud with 11 bits).
- The parity is changed back to SPACE parity.
- The rest of the packet is sent with SPACE parity.

Just as a note: I'm a little concerned with using time delays to determine when a byte has been transmitted. This technique could fail if the call to transmit the data returns and, for some reason, the data doesn't get transmitted right away. I've never seen this happen – but it could.

In the following figures (Figure 827-1 to 827-10), the top trace (in blue) is the packet being transmitted and the bottom trace (in red) is the RTS signal that I use to show when the parity is set to MARK parity (RTS low) and when the parity is set back to SPACE parity (RTS high). Note that sometimes you will see an extra "spike" in the RTS trace (high-low-high). This is because of the way the UART hardware reacts when its settings are changed and this peculiarity has to be accounted for in software.

These are the packets sent in "debug" mode. About the only thing that can be said is that debug mode takes a long time to change the parity settings. But the figures do show distinctly the separate transmits of the packet data. Figures 827-1 through 827-5 also show that as you

graduate from the older Windows 95 to the newer Windows XP that the native Windows code that manages the serial port parameters evidently becomes more efficient.

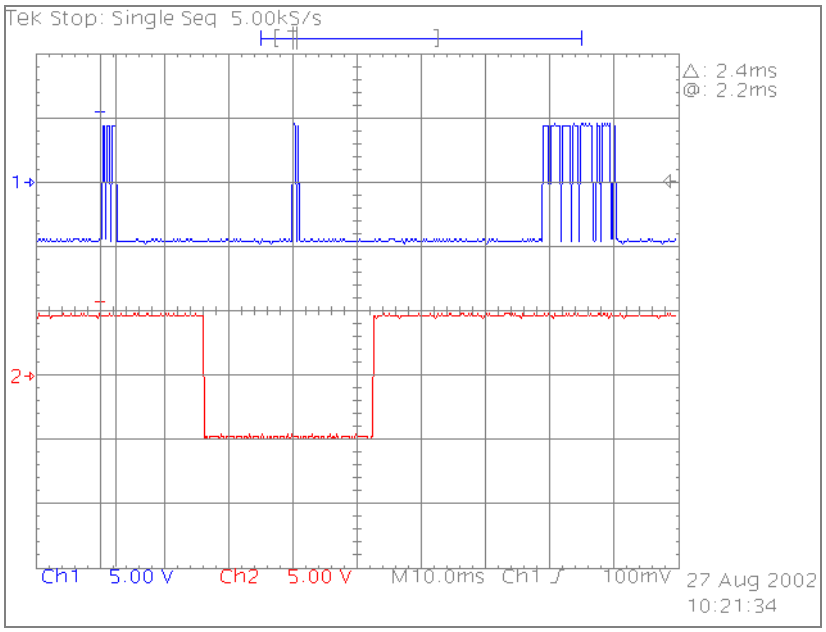

Figure 827-1: Win95C 9-Bit Packet Debug

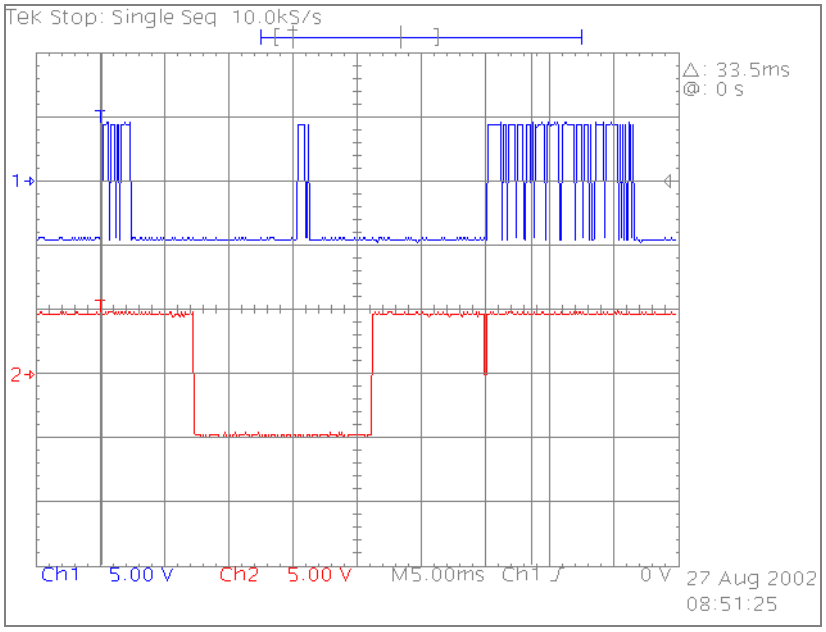

Figure 827-2: Win98 9-Bit Packet Debug

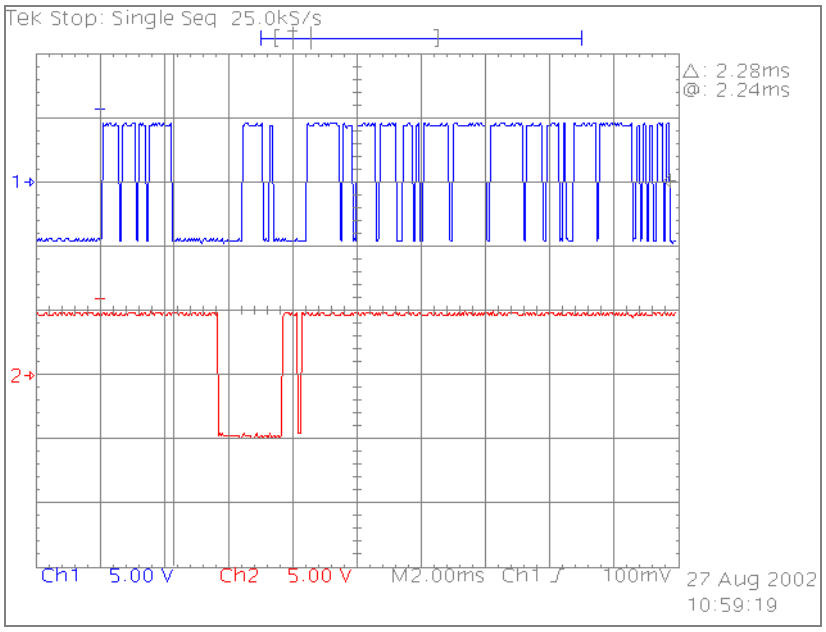

Figure 827-3: WinNT 9-Bit Packet Debug

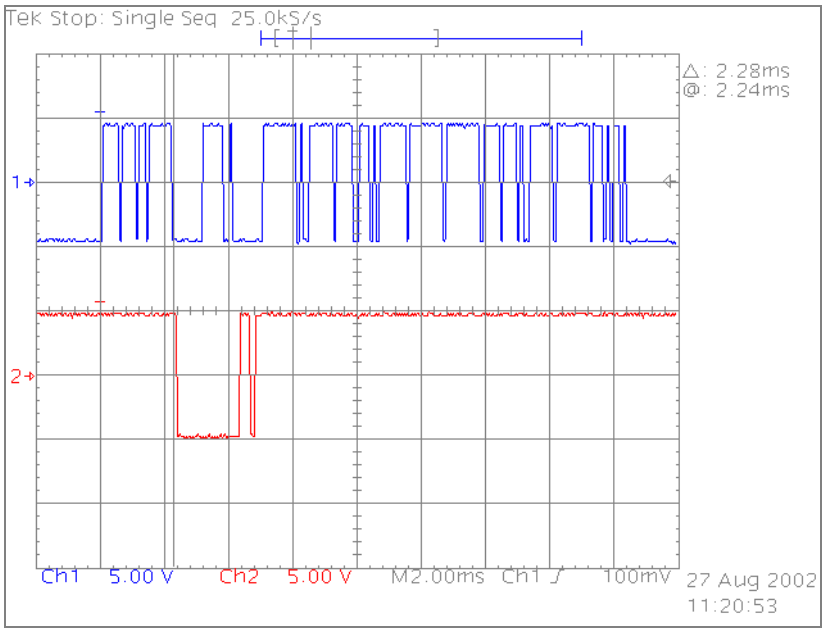

Figure 827-4: Win2K 9-Bit Packet Debug

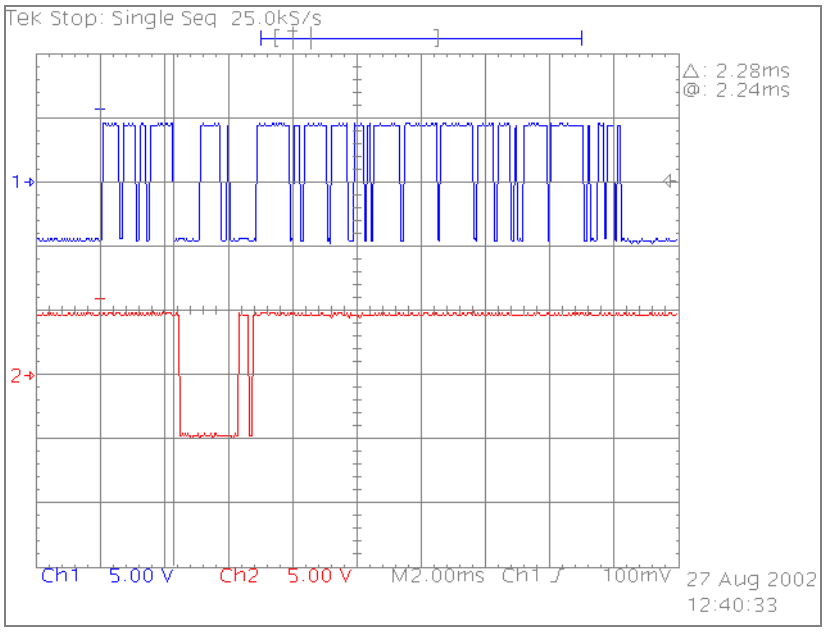

Figure 827-5: WinXP 9-Bit Packet Debug

These are the same packets sent in “release” mode. It can be seen that the native Windows code to handle the serial port parameters is much more efficient in release mode than in debug mode – but this is completely expected and normal. There is less distinction between Windows 95 and Windows XP although you can still see slight differences.

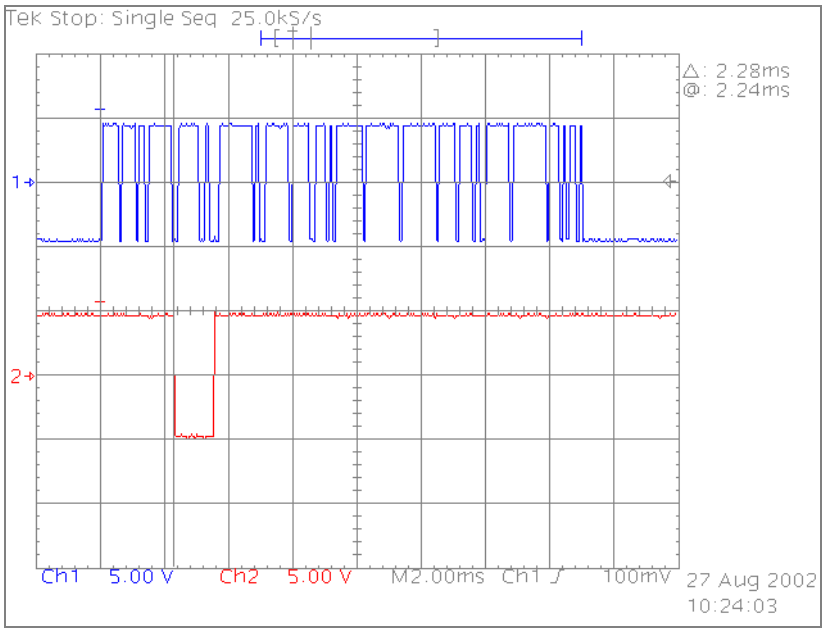

Figure 827-6: Win95C 9-Bit Packet Release

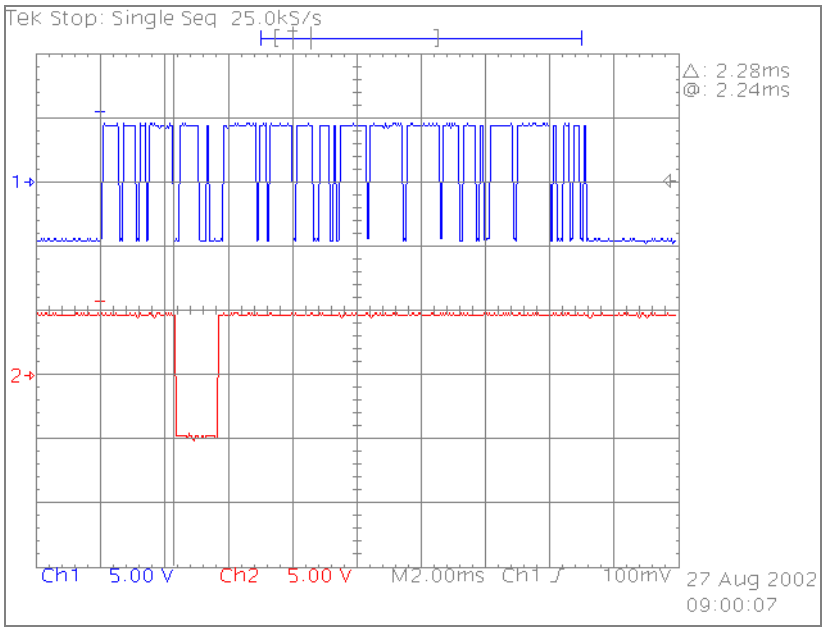

Figure 827-7: Win98 9-Bit Packet Release

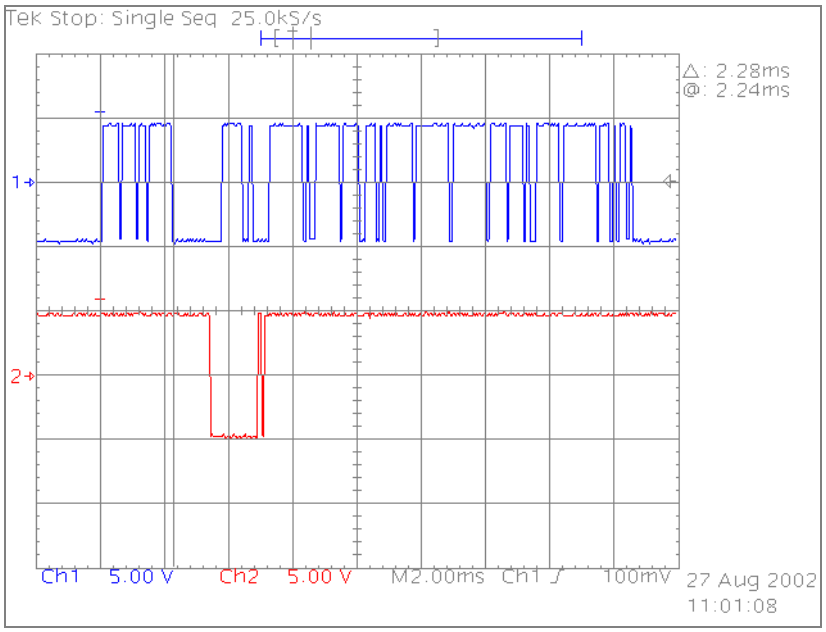

Figure 827-8: WinNT 9-Bit Packet Release

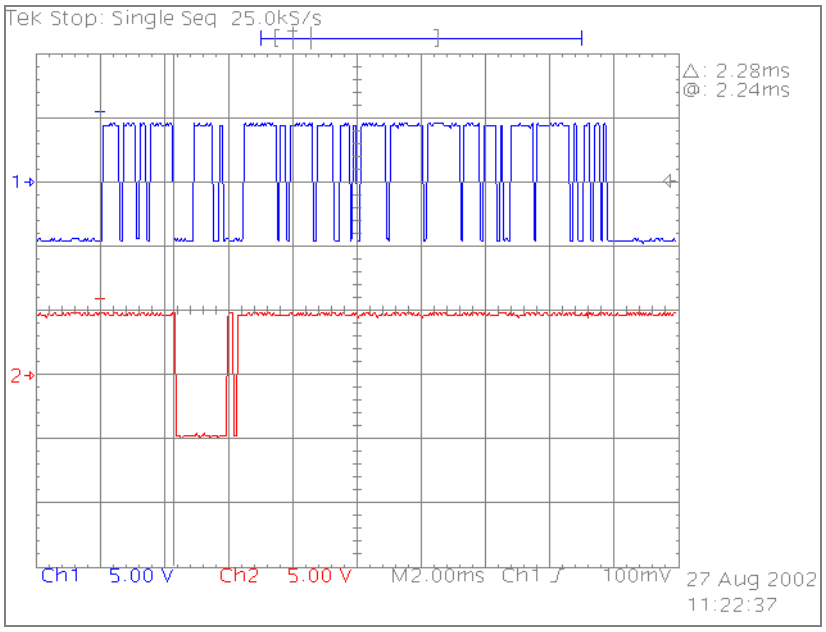

Figure 827-9: Win2K 9-Bit Packet Release

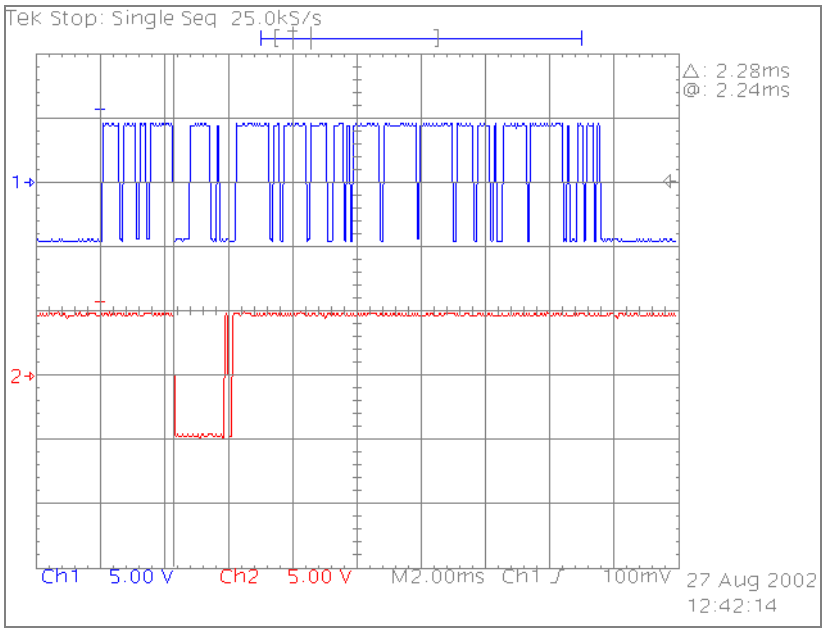

Figure 827-10: WinXP 9-Bit Packet Release

All-in-all the packet transmission looks OK. But I'll keep thinking about it to see if I can come up with a better way to determine when the data has been transmitted. The biggest problem is that there is no way to ask the UART if the data has been transmitted. Maybe I can tie the transmit signal to the CTS signal and use CTS to determine when the transmission has finished – but that would require a special serial cable and we want to avoid that if at all possible.