RS232/422/485 Serial Communications

Users Manual

(SERIAL)

Version 4.2

May 7, 2005

This software is provided as-is.
There are no warranties, expressed or implied.

MarshallSoft Computing, Inc.
Post Office Box 4543
Huntsville AL 35815 USA

Voice : 1-256-881-4630
FAX : 1-256-880-0925
Email : info@marshallsoft.com
Web : www.marshallsoft.com

TABLE OF CONTENTS

1 The UART

The heart of serial communications is the "Universal Asynchronous Receiver Transmitter", or UART for short. The UART is responsible for controlling the computer's RS-232/422/485 port.

1.1 Emulated UARTs

Some computers, particularly laptops and Handheld / Pocket PC's, do not contain real UART chips. Instead, the UART's functionality is emulated as part of some larger scale chip. The emulated UART chips do not always behave identically to a real UART. For example, emulated UART's may only support specific baud rates.

1.2 UART Types

Real UARTs can be broken down into two classes - buffered and unbuffered. The 8250 and 16450 are unbuffered, while the 16550, 16650, and 16750 are buffered.

1.2.1 National 8250

The National 8250 was the original UART used in the IBM PC and compatibles, and are only suitable up to 9600 baud on slower DOS (before the IBM/AT) machines. The 8250A is somewhat faster than the 8250, but should also be limited to slower DOS machines.

1.2.2 National 16450

The National 16450 was designed to work with the IBM PC/AT (16-bit data bus) and faster machines. Faster than the 8250, it still has a one-byte buffer. The 16450 chip is suitable for many DOS applications and some Windows applications up to about 38400 baud.

1.2.3 National 16550

The National 16550 features 16-byte transmit side and receive side FIFO buffers. The interrupt trigger level (on the receive side) can be set at 1, 4, 8 or 14 bytes. The TX FIFO trigger is fixed at 16 bytes.

The FIFOs on the original 16550 UARTs did not work. The 16550A soon followed the 16550 and is the "standard" UART for Windows machines.

The 16550A is recommended as the minimum chip on any new serial board purchase.

1.2.4 StarTech 16650

The StarTech 16650 features 32-byte FIFOs and on-chip flow control, and can be run at up to 460800 baud. It is also pin for pin compatible with the 16550 UART.

1.2.5 TI 16750

The Texas Instruments 16750 features 64-byte FIFOs and on-chip flow control and can be run at up to 921600 baud, but is not pin for pin compatible with the 16550 UART.

1.3 UART Operation

The purpose of the UART is:

(1) To convert bytes from the CPU (Central Processing Unit) into a serial format by adding the necessary start, stop, and parity bits to each byte before transmission, and to then transmit each bit at the correct baud rate.

The first bit is always the start bit, followed by 5 to 8 data bits, (optionally) followed by the parity bit, followed by the stop bit or bits.

(2) To convert the incoming stream (at a specified baud rate) of serial bits into bytes by removing the start, stop, and parity bit before being made available to the CPU.

The UART is capable of operating in one of two modes, 'polled' and 'interrupt driven'. The serial communications functions in the BIOS use the polled method. In this approach, the CPU is typically in a loop asking the UART over and over again if it has a byte ready. If a byte is ready, the polling code returns the byte. But, if the next byte comes in before the polling code is executing again, then that byte is lost.

In the interrupt driven approach, when a byte is received by the UART, an 'Interrupt Service Routine' (ISR) is executed immediately, suspending temporarily whatever is currently executing. The ISR then moves the byte from the UART to a buffer so that the application program can later read it.

The 16550 can be programmed so that a receive (RX) interrupt is not triggered until 4 (or 8 or 14) bytes have been received, while the 16650 can be triggered at up to 30 bytes, and the 16750 can be triggered at up to 56 bytes. This can significantly reduce the CPU processing time, since 14 (or 30 or 56) bytes can be moved at once.

Transmitted bytes are queued up awaiting transmission. When a byte is moved from the UART transmitter holding register to the UART transmitter shift register, an interrupt is generated and the next byte is taken from the transmitter buffer by the ISR and written to the UART holding register.

Up to 16 bytes can be written at once to the transmitter FIFO buffer while processing one transmitter interrupt if an 16550 UART is used, while the 16650 can write up to 32 bytes at once, and the 16750 can write up to 64 bytes at once.

1.4 RS-232 Signals

RS-232 is the name of the serial data interface standard used to connect computers to modems.

A summary of the serial port pins and their function follows. For more detailed information, refer to one of the many books dealing with RS-232 interfacing.

1.4.1 Signal Ground Pin 7 (DB25), Pin 5 (DB9)

The SG line is used as the common signal ground, and must always be connected.

1.4.2 Transmit Data Pin 2 (DB25), Pin 3 (DB9)

The TX line is used to carry data from the computer to the serial device.

1.4.3 Receive Data Pin 3 (DB25), Pin 2 (DB9)

The RX line is used to carry data from the serial device to the computer.

1.4.4 Data Terminal Ready Pin 20 (DB25), Pin 4 (DB9)

The DTR line is used by the computer to signal the serial device that it is ready. DTR should be set high when talking to a modem.

1.4.5 Data Set Ready Pin 6 (DB25), Pin 6 (DB9)

The DSR line is used by the serial device to signal the computer that it is ready.

1.4.6 Request to Send Pin 4 (DB25), Pin 7 (DB9)

The RTS line is used to 'turn the line around' in half duplex modems, and for hardware flow control in most modems that require flow control. RTS is controlled by the computer and read by the serial device (modem).

1.4.7 Clear to Send Pin 5 (DB25), Pin 8 (DB9)

The CTS line is used to 'turn the line around' in half duplex modems, and for hardware flow control in most modems that require flow control. CTS is controlled by the serial device (modem) and read by the computer.

1.4.8 Data Carrier Detect Pin 8 (DB25), Pin 1 (DB9)

The DCD line is used by the modem to signal the computer that a data carrier signal is present.

1.4.9 Ring Indicator Pin 22 (DB25), Pin 9 (DB9)

The RI line is asserted when a 'ring' occurs.

1.5 UART Registers

Data sheets can be obtained from the following sources on the Internet.

[UART]	[URL]
16550	www.national.com and www.exar.com
16650	www.exar.com
16750	www.ti.com
16850	www.exar.com
16950	www.oxsemi.com

These UARTs consists of 8 register ports as follows:

[REGISTER]	[DESCRIPTION]
Reg 0 R/W	Receiver (read) / Transmitter (write)
Reg 1 R/W	Interrupt Enable Register (IER)
Reg 2 R	Interrupt Identification Register (IIR)
Reg 2 W	FIFO Control Register (FCR : 16550/650/750)
Reg 2 R/W	Enhanced Features Register (EFR : 16650 Only)
Reg 3 R/W	Line Control Register (LCR)
Reg 4 R/W	Modem Control Register (MCR)
Reg 5 R/W	Line Status Register (LSR)
Reg 6 R/W	Modem Status Register (MSR)
Reg 7 R/W	Scratch register. Not used.

For the standard PC ports, the UART registers are based at 3F8h (COM1), 2F8h (COM2), 3E8h (COM3), and 2E8h (COM4). COM1 and COM3 share interrupt request line IRQ4 while COM2 and COM4 share request line IRQ3. This means that COM1 and COM3 can't be used concurrently. Similarly for COM2 and COM4.

Four sources of interrupts are possible. (1) receiver error or BREAK, (2) receiver data ready, (3) ready to transmit, and (4) RS-232 input. Additional sources can be generated by the 16650 (see 16650 spec sheet).

These four sources of interrupts are summarized as follows:

[Source of Interrupt]	[Action Required to Clear]
Receiver error or BREAK.	Read Line Status register.
Receiver data.	Read data from data register.
Transmitter Buffer Empty.	Write to data register or read IID reg.
RS-232 input.	Read Modem Status register.

Serial ports are configured as either Data Communications Equipment (DCE) or Data Terminal Equipment (DTE). Modems are always configured as DCE, while serial printers are (almost) always configured as DTE. Most serial (computer) ports are designed to talk to modems and are therefore configured as DTE. Serial (computer) ports designed to talk to serial printers (rare today) are configured as DCE. Most other serial device are configured as DCE.

A normal RS-232 cable is used to connect two serial ports with opposite configuration (DTE & DCE), while a null modem cable is used to connect two serial ports of the same configuration (both DTE or both DCE).

1.6 Register Summary

1.6.1 REG 0 : Data Register

Reading from the data register fetches the next input byte, once it is ready. Writing to the data register transmits the byte written to it over the serial line.

1.6.2 REG 1 : Interrupt Enable Register (IER)

The Interrupt Enable register enables each of four types of interrupts when the appropriate bit is set to a one.

```
[BIT]     [DESCRIPTION]
bit 3  :  Enable interrupt on RS-232 input.
bit 2  :  Enable interrupt on receiver error or break.
bit 1  :  Enable interrupt on transmitter buffer empty (TBE).
bit 0  :  Enable interrupt on received data (RxRDY).
```

1.6.3 REG 2 : Interrupt Identification Register (IID)

Reading the Interrupt Identification (read only) register once an interrupt has occurred identifies the interrupt as follows:

[Bit 2]	[Bit 1]	[Bit 0]	[Priority]	[Interrupt]
0	0	1	none	none
1	1	0	0 (high)	Serialization or break.
1	0	0	1	Received data.
0	1	0	2	Transmitter Buffer Empty.
0	0	0	3 (low)	RS-232 Input.

1.6.4 REG 2 : Interrupt Identification Register (IID)

In the 16550, 16650, and 16750, REG 2 (write only) is also the FIFO control register. Writing bits 6 & 7 will set the RX FIFO trigger level (number of bytes received before an interrupt is generated).

[Bit 7]	[Bit 6]	[16550 Trigger]	[16650 Trigger]	[16750 Trigger]
0	0	1 byte	8 bytes	1 byte
0	1	4 bytes	16 bytes	16 bytes
1	0	8 bytes	24 bytes	32 bytes
1	1	14 bytes	28 bytes	56 bytes

The TX FIFO level can also be set in the 16650 by setting bits 4 & 5. The 64-byte FIFO mode on the 16750 can be enabled by setting bit 5 in the FCR. See the 16650 & 16750 data sheets for more details.

1.6.5 REG 2 : Enhanced Feature Register (EFR) [16650 ONLY]

The EFR can only be accessed after writing a BF to the LCR, after which the advanced features on the 16650 are enabled by setting bit 4 of the EFR. For more details, see the 16650 data sheet.

1.6.6 REG 3 : Line Control Register (LCR)

RS-232 line parameters are selected by writing to this register.

[BIT]		[DESCRIPTION]
bit 7	:	DLAB = 0
bit 6	:	BREAK on(1), off(0).
bits 5-3	:	Parity None(000),ODD(001),EVEN(011),MARK(101),SPACE(111)
bit 2	:	One stop bit(0), two stop bits(1).
bits 1-0	:	Data bits = 5 (00), 6(01), 7(10), 8(11).

[PARITY]		[DESCRIPTION]
Odd	:	The parity bit is 1 if the sum of the data bits is odd.
Even	:	The parity bit is 1 id the sum of the data bits is even.
None	:	There is no parity bit.
Mark	:	The parity bit is always set to 0.
Space	:	The parity bit is always set to 1.

When the Divisor Latch Access Bit (DLAB) is 1, registers 0 and 1 become the LS and MS bytes of the Baud Rate Divisor registers.

The baud rate is computed as (115200 / BaudRateDivisor). Thus, common baud rates correspond to divisors as follows:

[BAUD]	[DIVISOR]	[BAUD]	[DIVISOR]	[BAUD]	[DIVISOR]
300	0180	4800	0018	38400	0003
1200	0060	9600	000C	57600	0002
2400	0030	19200	0006	115200	0001

NOTES:
(1) Must write BF hex to LCR before EFR [16650 ONLY] can be accessed (see 16650 data sheet).
(2) Must set DLAB = 1 (80 hex) before 64 byte FIFO bit can be accessed (see 16750 data sheet).
(3) The 3 parity bits in the UART are named "Parity Enable" (bit 3), "Parity Select" (bit 4), and "Stick Parity" or "Sticky Bit" (bit 5).

1.6.7 REG 4 : Modem Control Register (MCR)

RTS, DTR, loopback testing, and General Purpose Outputs #1 and #2 are controlled by the Modem Control register as follows:

```
[BIT]     [DESCRIPTION]
bit 7  :  Clock select. X1 (if 0), X4 (if 1). [16750 ONLY]
bit 6  :  IR enable [16650 ONLY]
bit 5  :  Interrupt type select [16650 ONLY] or
          Flow control enable [16750 ONLY].
bit 4  :  Enable local loopback.
bit 3  :  Enable GP02. Necessary for UART interrupts.
bit 2  :  Enable GP01.
bit 1  :  Set / clear RTS.
bit 0  :  Set / clear DTR.
```

1.6.8 REG 5 : Line Status Register (LSR)

Reading the Line Status register provides status information as follows (1 for TRUE, 0 for FALSE) :

```
[BIT]     [DESCRIPTION]
bit 7  :  FIFO data error [16650 & 16750 ONLY].
bit 6  :  Transmitter Empty (TXE).
bit 5  :  Transmitter Buffer Empty (TBE).
bit 4  :  BREAK detect.
bit 3  :  Framing error.
bit 2  :  Parity error.
bit 1  :  Overrun error.
bit 0  :  Data Ready.
```

1.6.9 REG 6 : Modem Status Register (MSR)

Reading the Modem Status register provides the following status information (1 for TRUE, 0 for FALSE) :

```
[BIT]     [DESCRIPTION]
bit 7  :  DCD status.
bit 6  :  RI status.
bit 5  :  DSR status.
bit 4  :  CTS status.
bit 3  :  Delta DCD status.
bit 2  :  Delta RI status.
bit 1  :  Delta DSR status.
bit 0  :  Delta CTS status.
```

The delta bits (bits 0 through 3) are set whenever one of the status bits (bits 4 through 7) changes (from 0 to 1 or from 1 to 0) since the last time that the Modem Status register was read. Reading the Modem Status register clears the delta bits.

1.6.10 REG 7 : Scratch Register

There is no function associated with register 7. It does not exist in early versions of the 8250.

2 Modems

A modem is used to extend the distance over which you may communicate. Without a modem, your RS-232 cable is limited to a maximum of approximately 50 feet. But with a modem, you can communicate literally around the world.

2.1 Modem Standards

Two modems can communicate over a telephone line only if they are both using the same signaling frequencies and modulation, which are determined by the modem standards used. Modem standards can be divided into three sets: (1) speed, (2) data compression used, and (3) error control.

The Bell standards (103 & 212A) are those of AT&T. The CCITT (The International Consultative Committee for Telephone and Telegraph) standards are designated as 'V. '.

2.1.1 Speed

- Bell 103 : 300 baud
- Bell 212A : 1200 baud
- V.21 : 300 baud
- V.22bis : 1200 & 2400 baud
- V.32 : 4800 & 9600 baud
- V.32bis : 4800, 7200, 9600, 12000, and 14400 baud
- V.34 : V.32bis plus 16800, 19200, 21600, 24000, 26400, and 28800 baud.
- V.34bis : V.34 plus 31200 and 33600 baud.
- USR X2 : US Robotics 56KB standard (33.6 KB uploads).
- K56flex : Rockwell's 56KB standard (33.6 KB uploads).
- V.90 : The new 56K standard.

2.1.2 Data Compression

- MNP 5 : Microcom Networking Protocol (proprietary).
- V.42bis : International data compression standard.

2.1.3 Error Control

- MNP 2,3,4 : Three level error correction (public domain).
- V.42 : International error correction standard.

2.2 Modem AT Command Set

The first AT command set was developed by Hayes, and allowed the programmer to communicate directly with the modem in "command mode" using the "AT command set". When a modem is first powered up, it enters "command mode", and will respond to the various AT commands that it recognizes. Once connected to another modem ("online" or "connected" mode), AT commands are not recognized.

Other modem manufacturers soon followed Hayes, typically making their command set upwardly compatible with the original Hayes command set. However, different modem manufacturers choose different AT commands for the same functionality, such as setting flow control. This lead to the present condition, in which each modem manufacturer has it's own command set with a core set of commands common to all modems.

The User's Guide for your modem should contain a list of the AT commands that it uses. If not, you should be able to get this information from the manufacturer. Most modem manufacturers have this information on their Internet site.

2.3 Flow Control

With modems using data compression, the modem to modem connection will run at various speeds depending on the quality of the line. The computer to modem connection will be at a fixed baud rate. Therefore, a protocol (flow control) is necessary to synchronize the data flow between a modem and the computer to which it is connected. Refer to your modem manual for information on flow control protocols supported.

Two flow control protocols are used by most all modems which require flow control. Software flow control is called 'XON/XOFF' (other software flow control character pairs are defined but operate the same as XON/XOFF) and hardware flow control is called 'RTS/CTS'. Most modems which require flow control enable hardware flow control by default.

In XON/XOFF (software) flow control, the computer suspends transmitting data if it receives a XOFF character (13 hex) from the modem, and continues transmitting when it receives a XON character (11 hex). Similarly, the computer can signal the modem not to send any more data by transmitting a XOFF to it, and can tell the modem to continue transmission be sending a XON.

In RTS/CTS (hardware) flow control, the RTS line is used by the computer to signal the modem , while the CTS line is used by the modem to signal the computer. The RTS line is set OFF by the computer to tell the modem to suspend transmission, and set to ON to tell the modem to continue transmission. The CTS line is set to OFF by the modem to tell the computer to stop transmitting, and set to ON to tell the computer to continue transmitting.

Given the choice, always choose hardware flow control over software flow control so that all data transmission is transparent. If hardware flow control is not the default (which it almost always is), you should modify your modem initialization string to turn hardware flow control on.

2.4 Modem Initialization

If an application uses a modem (as opposed to using a null modem cable), then it should always send an initialization string to the modem. Communication programs such as PROCOMM and TELIX always send such a string automatically as soon as they start up.

The particular initialization string depends on the make of your modem. For most modems, the following string (followed by a carriage return) should work:

```
AT E1 S7=60 S11=60 V1 X1 Q0 S0=0
```

Recall that the modem must be in command mode in order to send AT commands. To force command mode, send the character string "+-+" (without the quotes), preceded by 1 second of silence, and followed by one second of silence.

Refer to your Modem User's Guide for a full discussion of these commands. A brief description is as follows:

```
[COMMAND]     [DESCRIPTION]
   AT      :  Modem attention command.
   E1      :  Modem will echo what you send to it.
   S7=60   :  Wait 60 seconds for carrier and/or dial tone.
   S11=60  :  Use 60 milliseconds for tone dialing duration & spacing.
   V1      :  Display result code as words (not numbers).
   X1      :  Use the extended result message (CONNECT XXXX) set.
   Q0      :  Modem displays result codes.
   S0=0    :  Do not answer RING.
```

If the application will answer incoming calls, set the S0 register to the ring on which to automatically answer.

Most modems can be set to the 'factory default' by transmitting

```
AT&F
```

2.5 Modem CONNECT

When dialing a modem (using the ATDT command) or when answering an incoming call, modems will typically respond with a "CONNECT xxxxx" string, where the "xxxxx" is the connect baud rate. Some modems, notably many WinModems, require that you change the UART baud rate (using SioBaud) to match the "xxxxx" value.

2.6 More Modem Documentation

There is a lot of information regarding modems on the Internet, including a listing of the AT command set for most modems.

An excellent book covering a wealth of modem information is the Programmer's Technical Reference: Data and Fax Communications by Robert L. Hummel (Ziff-Davis Press).

3 RS422 and RS485

RS422 and RS485 use the same UARTs as RS232. However, both RS422 and RS485 are based on balanced differential signals, as opposed to RS232 which uses unbalanced signal levels. In other word, RS422 uses the difference in voltage levels between 2 wires whereas RS232 uses the voltage level of a single wire with respect to a common signal ground.

RS422 and RS485 both require a pair of wires for every signal. RS422 is only usable in point to point systems.

RS485 has tri-state capability (its driver can be disabled) and can support up to 32 receivers (multidrop) on the same line.

RS485 can also be wired as "2-wire" in which the same pair of wires are used for both transmitting and receiving. Typically, RTS is set before transmitting and dropped after the last bit of the last byte is sent.

RS422 and RS485 may both require (for long run lengths) termination resistors and/or biasing resistors, which is beyond the scope of this discussion.

RS422 an RS485 are typically used in industrial settings where long run lengths (to 4000 feet) are necessary. For more information, refer to one of the many technical references on the Internet such as:

- `http://www.arcelect.com/485info.htm`
- `http://www.bb-elec.com/techlibr.html`
- `http://www.kksystems.com/serdesc1.html`
- `http://www.sealevel.com/tech.html`

4 Other Serial Devices

To be sure, the modem is the most common serial device. But there are many other serial devices such as digitizing tablets, scanners, digital cameras, numerical control machines, card readers, panel displays, etc.

Some serial devices (such as modems) use hardware (RTS/CTS) flow control, but DTR/DSR flow control and software (XON/XOFF) flow control are also common.

If you are writing a program to communicate with a serial device, keep in mind the following:

(1) Always set DTR and RTS. Many serial devices "play dead" if DTR is not set.

(2) You may need to add a small time delay (0.25 sec) between transmitted characters. This can be reduced or eliminated once everything is working.

(3) Make sure that your receive buffer is sufficiently large. You want to avoid buffer overflow.

`[END]`