XModem Protocol with CRC

Introduction

The Xmodem protocol was created years ago as a simple means of having two computers talk to each other. With its half-duplex mode of operation, 128- byte packets, ACK/NACK responses and CRC data checking, the Xmodem protocol has found its way into many applications. In fact most communication packages found on the PC today have a Xmodem protocol available to the user.

Theory of Operation

Xmodem is a half-duplex communication protocol. The receiver, after receiving a packet, will either acknowledge (ACK) or not acknowledge (NAK) the packet. The CRC extension to the original protocol uses a more robust 16-bit CRC to validate the data block and is used here. Xmodem can be considered to be receiver driven. That is, the receiver sends an initial character “C” to the sender indicating that it’s ready to receive data in CRC mode. The sender then sends a 133-byte packet, the receiver validates it and responds with an ACK or a NAK at which time the sender will either send the next packet or re-send the last packet. This process is continued until an EOT is received at the receiver side and is properly ACKed to the sender. After the initial handshake the receiver controls the flow of data through ACKing and NAKing the sender.

Table 1. XmodemCRC Packet Format

Byte 1	Byte 2	Byte 3	Bytes 4-131	Bytes 132-133
Start of Header	Packet Number	(Packet Number)	Packet Data	16-bit CRC

Definitions

The following defines are used for protocol flow control.

Symbol	Description	Value
SOH	Start of Header	0x01
EOT	End of Transmission	0x04
ACK	Acknowledge	0x06
NAK	Not Acknowledge	0x15
ETB	End of Transmission Block (Return to Amulet OS mode)	0x17
CAN	Cancel (Force receiver to start sending C's)	0x18
C	ASCII “C”	0x43

Byte 1 of the XmodemCRC packet can only have a value of SOH, EOT, CAN or ETB anything else is an error. Bytes 2 and 3 form a packet number with checksum, add the two bytes together and they should always equal 0xff. Please note that the packet number starts out at 1 and rolls over to 0 if there are more than 255 packets to be received. Bytes 4 - 131 form the data packet and can be anything. Bytes 132 and 133 form the 16-bit CRC. The high byte of the CRC is located in byte 132. The CRC is calculated only on the data packet bytes (4 - 131) .

Synchronization

The receiver starts by sending an ASCII “C” (0x43) character to the sender indicating it wishes to use the CRC method of block validating. After sending the initial “C” the receiver waits for either a 3 second time out or until

a buffer full flag is set. If the receiver is timed out then another “C” is sent to the sender and the 3 second time out starts again. This process continues until the receiver receives a complete 133-byte packet.

Receiver Considerations

This protocol NAKs the following conditions: 1. Framing error on any byte 2. Overrun error on any byte 3. Duplicate packet 4. CRC error 5. Receiver timed out (didn't receive packet within 1 second) On any NAK, the sender will re-transmit the last packet. Items 1 and 2 should be considered serious hardware failures. Verify that sender and receiver are using the samebaud rate, start bits and stop bits. Item 3 is usually the sender getting an ACK garbled and re-transmitting the packet. Item 4 is found in noisy environments. And the last issue should be self-correcting after the receiver NAKs the sender.

Sender						Receiver
					<----	“C”
						Times Out after 3 Seconds
					<----	“C”
SOH	0x01	0xFE	Data	CRC	---->	Packet OK
					<----	ACK
SOH	0x02	0xFD	Data	CRC	---->	(Line Hit during Data Transmission)
					<----	NACK
SOH	0x02	0xFD	Data	CRC	---->	Packet OK
					<----	ACK
SOH	0x03	0xFC	Data	CRC	---->	Packet OK
(ACK Gets Garbled)					<----	ACK
					<----	ACK
SOH	0x04	0xFB	Data	CRC	---->	(UART Framing Error on Any Byte)
					<----	NACK
SOH	0x04	0xFB	Data	CRC	---->	Packet OK
					<----	ACK
SOH	0x05	0xFA	Data	CRC	---->	(UART Overrun Error on Any Byte)
					<----	NACK
SOH	0x05	0xFA	Data	CRC	---->	Packet OK
					<----	ACK
EOT					---->	Packet OK
					<----	ACK
ETB					---->	Finished
Finished					<----	ACK

Sample crc calculation code

```
int calcrc(char *ptr, int count)
{
    int  crc;
    char i;

    crc = 0;
    while (--count >= 0)
    {
        crc = crc ^ (int) *ptr++ << 8;
        i = 8;
        do
        {
            if (crc & 0x8000)
                crc = crc << 1 ^ 0x1021;
            else
```

```
                crc = crc << 1;
            } while(--i);
        }
        return (crc);
    }
```

Back to Welcome - Contact Amulet - Amulet Home