AVR360: XmodemCRC Receive Utility for the AVR

Product Description

Application Note

Features

- **Programmable Baud Rate**
- **Half Duplex**
- **128 Byte Data Packets**
- **CRC Data Verification**
- **Framing Error Detection**
- **Overrun Detection**
- **Less than 1k Bytes of Code Space**
- **C High Level Language Code**

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 (NACK) the packet. The original Xmodem protocol used a standard checksum method to verify the 128-byte data 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 NACK 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 ACK'ed 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

Rev. xxxxA–02/99

Definitions

The following defines are used for protocol flow control.

Table 2. Protocol Flow Control

Symbol	Description	Value
SOH	Start of Header	0x01
EOT	End of Transmission	0x04
ACK	Acknowledge	0x06
NAK	Not Acknowledge	0x15
C	Ascii 'C'	0x43

Byte 1 of the XmodemCRC packet can only have a value of SOH or EOT, 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.

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 NACKs 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 same baud 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 NAK's the sender.

DataFlow Diagram

The data flow diagram below simulates a 5-packet file being sent.

Table 3. XmodemCRC DataFlow with Errors

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
SOH	0x03	0xFC	data	CRC	---->	Duplicate packet
					<----	nack
SOH	0x04	0xFB	data	CRC	---->	(UART framing error on any byte)

Table 3. XmodemCRC DataFlow with Errors (Continued)

Sender						Receiver
					<----	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 gets garbled)					<----	ack
EOT					---->	Packet OK
Finished					<----	ack

Modifications to receive protocol

Users may wish to count how many 'C's were sent during synchronization and after 'n' number of tries abort the receive attempt.

For embedded applications it's not mandatory to have a 128-byte packet. You could have 64, 32 or even a 16-byte packet. The sender of course would have to comprehend this. For those users that may want to migrate to Atmel's MegaAVR series there is a version of Xmodem that uses a 1k-byte packet. Or you can use an external SRAM with an AT90S4414 or an AT90S8515 to allow the increase in packet size.

If users do not wish to use the CRC method of data verification, simply replace sending a 'C' for synchronization with a NAK instead. The sender will then send only the simple checksum of the data packet. Of course, the buffer size decreases by 1 and you may get data errors. This modification would allow communication with equipment that supports only the checksum method of data verification.

Software

Routines were compiled using IAR's C Compiler Version 1.30a with the Size optimization set to nine. Software was tested using ProComm, DynaComm, WinComm, and Hyperterminal at baud rates up to 115.2 kbps. The receiver expects 8 start bits, 1 stop bit, and no parity bits.

The STK200 starter kit is used as a test platform with minor, optional, modifications. A baud rate friendly crystal was used for this code. Replace the 4.0 MHz crystal on the STK200 starter kit with a 7.3728 MHz crystal for proper operation. If users wish to use the default crystal then modify the init routine to properly set up the uart baud rate register UBRR. Wait loops in the sendc and the recv_wait routines would also need modification.

To verify proper operation of this code, the STK200 jumpers should be set for RS-232 operation and PORTD bit 2 should be connected to the switches on the starter kit. Refer to the STK200 user manual for jumper locations and definitions. Connect a 9 pin serial cable from a PC to the starter kit, turn on power and use pushbutton 2 as a start of reception signal. Use an ICEPRO emulator, an AT90S4414-8PC, or an AT90S8515-8PC to execute the code.

Table 4. Routines

Name	Size in Bytes	Function
calcrc	66	Calculates 16 bit CRC
init	30	Low level hardware initialization
main	30	Main
purge	36	Reads uart data register for 1 second

Table 4. Routines (Continued)

Name	Size in Bytes	Function
receive	66	main receive routine
recv_wait	40	waits until buffer full flag is set or 1 second timeout
respond	44	sends an ack or a nak to the sender
sendc	88	sends an ascii 'C' character to the sender until the buffer full flag is set
timer1	28	timer1 interrupt
uart	102	uart receive interrupt
validate_packet	166	validates senders packet

Total memory space consumed is 942 bytes of CODE memory and 187 bytes of DATA memory.

Pseudo-Code

purge.c

```
initialize timer1 counter for a 1 second delay read uart for 1 second
```

receive.c

```
send a 'C' character to sender until receive buffer is full validate received packet send an ack or a nak to
sender
    if packet was bad then wait for new  good packet
while not end of transmission
    wait for buffer to fill
    validate the packet
    send an ack or a nak to sender
```

recv_wait.c

```
initialize timer1 counter for a 1 second delay wait till buffer is full or timeout
```

respond.c

```
clear error flags
    if packet was good a duplicate packet or end of transmission then
send an ack
    else
purge senders uart transmit buffer
    send a nack
```

sendc.c

```
initialize timer1 counter for a 3 second delay
    clear error flags
while buffer is not full
    send 'C' character to sender, signaling CRC mode
enable timer counter
    wait for buffer full or timeout
if timed out clear error flags
    restart timer
```

uart.c

```
    check uart for framing or overrun errors
        read byte from uart
    verify first byte in receive buffer is valid
        if buffer is full set buffer full flag
```

validate_packet.c

```
    if not timed out then
        if no uart framing or overrun errors then
        if first character in buffer is SOH then
            if second character in buffer is the next packet number
    then
            if second character in buffer plus the third character in buffer = 0xff
    then
            compute CRC on packet data
            if CRC ok
    then
        increment packet number
        packet = good
    else
        packet = bad
    else
        bad packet number checksum
    else
        duplicate packet number
    else
        if first character in buffer is EOT then
        end of transmission
    else
        at least 1 byte had a framing or overrun error, packet is bad
    else
        timed-out without receiving all characters, packet is bad
```

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel U.K., Ltd.
Coliseum Business Centre
Riverside Way
Camberley, Surrey GU15 3YL
England
TEL (44) 1276-686677
FAX (44) 1276-686697

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road
Tsimshatsui East
Kowloon, Hong Kong
TEL (852) 27219778
FAX (852) 27221369

Japan
Atmel Japan K.K.
Tonetsu Shinkawa Bldg., 9F
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4 42 53 60 00
FAX (33) 4 42 53 60 01

Fax-on-Demand
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

xxxxA–02/99/xM