Xmodem-1K implementation

HMS800 C Compiler for the HMS800 series MCU

DESCRIPTION

This application note describes how to implement Xmodem-1K with HMS800 C Compiler for HMS800 series MCU. This is helpful to understand Xmodem-1K implementation with HMS800 C Compiler.

Xmodem is one of the most widely used file transfer protocols. The original Xmodem protocol uses 128-byte packets and a simple "checksum" method for error detection. A later enhancement, Xmodem-CRC, uses a more secure Cyclic Redundancy Check (CRC) method for error detection. Xmodem protocol always attempts to use CRC first. If the sender does not acknowledge the requests for CRC, the receiver shifts to the checksum mode and continues its request for transmission.

Introduction

XMODEM is a half duplex, 8-bit protocol which transfers files by sending a portion of the file in a data structure called a packet and then waiting for a response to that packet. If the response is positive (indicating successful transmission), the next packet is sent. If the response is negative, the packet is resent.

The original XMODEM protocol transferred 128-byte data blocks with a 1-byte checksum for error detection. The layout of the XMODEM packet is:

1. SOH(Start Of Header) : 1byte ‘0x01’
2. Packet Number : 1byte
3. 1's Complement of Packet Number : 1byte
4. Data :128 bytes
5. Arithmetic checksum : 1byte

All fields except the data field are one byte. The data field contains 128 bytes of either ASCII or binary data that is usually obtained from the file being transferred.

Figure 1 shows a simplified progression of a typical transmission. The receiver initiates the transfer by asking the sender to transmit the first data packet. For the original XMODEM, this start character is a NAK (0x15). Once a packet has been sent, the sender waits for the receiver to respond to the packet. If the receiver responds with an ACK(0x06), indicating that the packet was received successfully, the sender transmits the next packet. A NAK indicates the receiver did not receive the packet successfully and sender resends the packet.

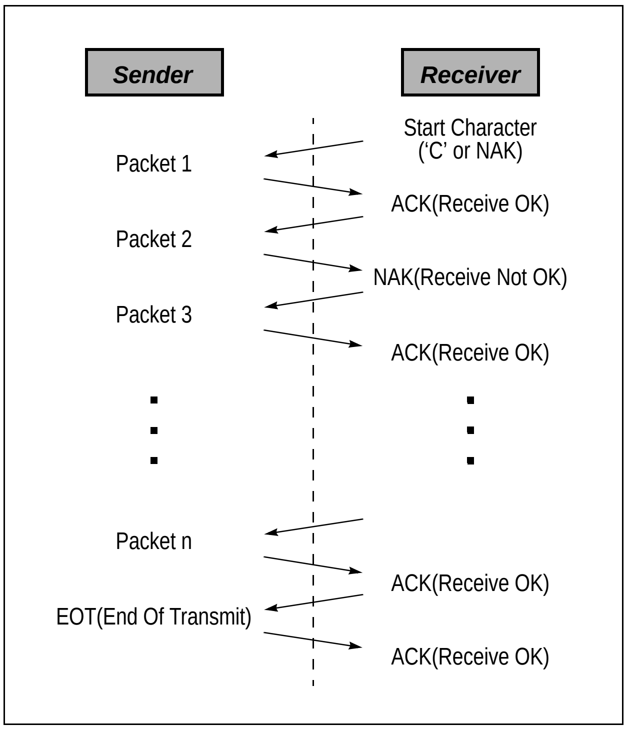

Figure 1. Xmodem file transfer

After transferring all the packets, the sender signals end-of-transmission with an EOT(0x04) and waits for an ACK. The transmission is complete once the sender receives an ACK.

Xmodem-1K

Xmodem-1K is essentially Xmodem CRC with 1K (1024 byte) packets. On some systems and bulletin boards it may also be referred to as Ymodem. Differencies of original Xmodem are structure of a packet and character of start(ASCII ‘C’ → ‘G’). The layout of the Xmodem-1K packet is :

1. STX(Start Of Header) : 1byte ‘0x02’
2. Packet Number : 1byte
3. 1's Complement of Packet Number : 1byte
4. Data :1024 bytes
5. CRC-CCITT or Checksum : 2byte or 1byte

An STX(02) replaces the SOH(01) at the beginning of the transmitted block to notify the receiver of the longer packet length. The transmitted packet contains 1024 bytes of data. The receiver should be able to accept any mixture of 128 and 1024 bytes packets. The packet number is incremented by one for each packet regardless of the packet length. The sender must not change between 128 and 1024 bytes packet lengths if it has not received a valid ACK for the current packet.

Failure to observe this restriction allows certain transmission errors to pass undetected. If 1024 bytes packets are being used, it is possible for a file to "grow" up to the next multiple of 1024 bytes.This does not waste disk space if the allocation individuality is 1024 or greater. With YMODEM batch transmission, the file length transmitted in the file name packet allows he receiver to discard the padding, preserving the exact file length and contents. CRC-CCITT should be used with the Xmodem-1K to preserve data integrity over 1024 bytes packets may be used with batch file transmission or with single file transmission. If the sender transmit the NAK at first, the fifth field of a packet should be checksum with 1 byte.

Implementation

Basically it need to be capable of UART communication and possess sufficient size of RAM more than one packet. This application note will provide generalized functions for users to implement Xmodem-1K. This functions are concentrated on implementation of the state-machine for Xmodem-1K. Especilly the functions for transmission and receiving will be described.

Descriptions of program :

A) Structure of the program

1. Definition statements
- Definition of the macros and events.
2. XmodemR function
- Implement of the receiving state-machine.
3. XmodemT function
- Implement of the transmission state-machine.

B) State transition diagram of the XmodemR

Refer to figure 2. The XmodemR service routine start to the “CRCgo” state, and then send ASCII ‘C’. If the sender do not respond or exceed to retry 5 times at the “STXread” state, the routine goes to the “CHKSUMgo” state sending the NAK character. Once if the service routine goes to the “STXread” state, it always must check whether the “TIMEOUT” state. This is evaluated to the “ucElapse” variable with 10 second limitation. All the flowcontrol is composed of “switch” statement and function call, omitted data processing.

The implemented service routine is in appendix A.

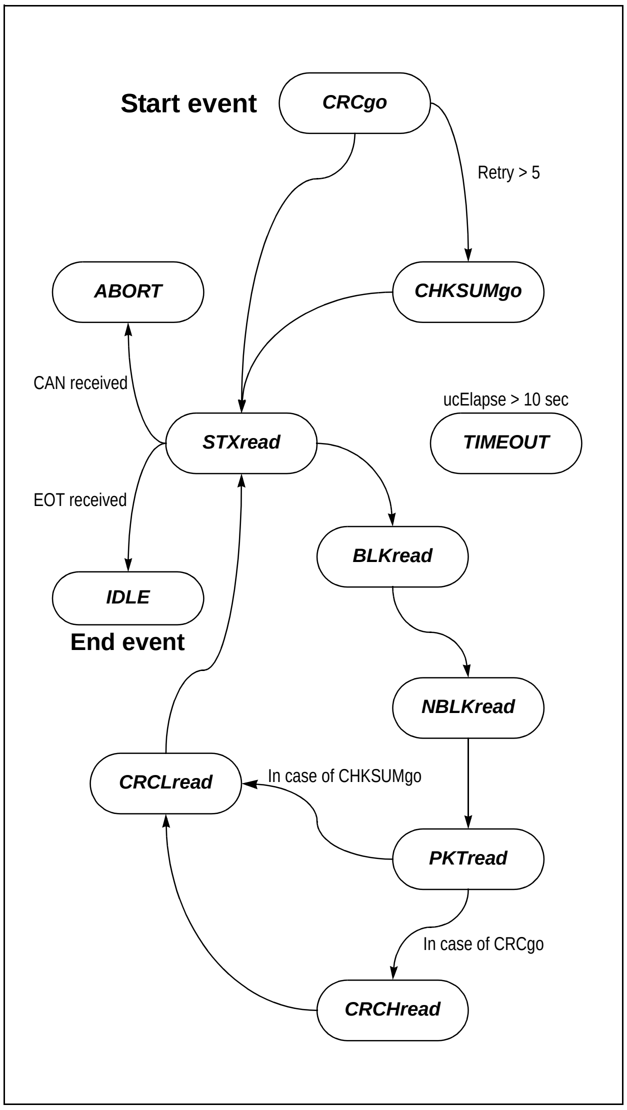

Figure 2. State transition diagram for the XmodemR

C) State transition diagram of the XmodemT

Refer to Figure 3. The XmodemT service routine start to the “STANDBY” state, and then goes to the “START” state in order to initialize transmission of a file formerly received. The next state is the “ACKWAIT” to be ready to transmit a packet. If the receiver do not respond or exceed to retry more 5 times at the “ACKWAIT” state, the routine goes to the “TIMEOUT”state. The “ABORT” state is to abort transmission with the CAN character at the “ACK-WAIT” state. The “RESEND” state from the “ACK-WAIT” state goes to resend a packet receiving NAK character. The “IDLE” state indicates to finish transmission of packets. The “NEXT” state indicates to transmit next packet.

The implemented service routine is in appendix A.

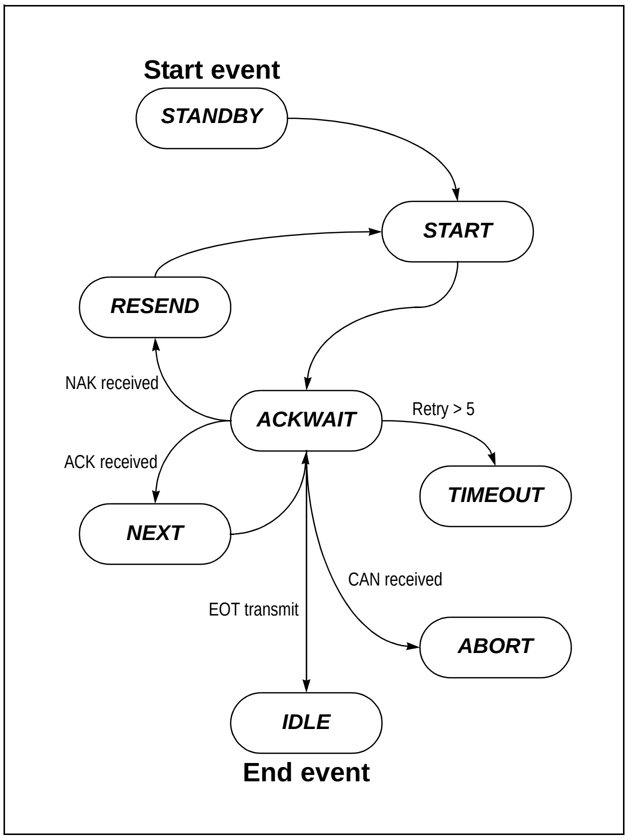

Figure 3. State transition diagram for the XmodemT

Programming

The receive and transmit functions which are programmed to implement Xmodem-1K are supposed to use any C program. Therefore an user need to modify scripts to adapt user's program. The unimplemented functions are related to CRC and other packet processes are not handled here.

Author: *Nicholas Oh*
MCU application team
e-mail: jesu.oh@hynix.com

Appendix A: Functions.

```
/*******************************************************************************************
 Title : Xmodem-1K implementation
 Programming date : 2004.6.1.
 Present filename : Xmodem.c
*******************************************************************************************
 Program description
   CRC-CCITT(16 bit), 1024 bytes block, 9600 Baud-rate, No parity, 1 stop
   Test : Windows XP HyperTerminal
*******************************************************************************************/
// Communication control characters
#define SOH       0x01
#define STX       0x02
#define ETX       0x03
#define EOT       0x04
#define ENQ       0x05
#define ACK       0x06
#define NAK       0x15
#define ETB       0x17
#define CAN       0x18
#define EOF       0x1A

// Xmodem state machine, states
#define IDLE      0
#define STANDBY   1
#define START     2
#define NEXT      3
#define RESEND    4
#define ACKWAIT   5
#define ABORT     6
#define TIMEOUT   7
#define CRCgo     8
#define CHKSUMgo  9
#define SOHread   10
#define BLKread   11
#define NBLKread  12
#define PKTread   13
#define CRCHread  14
#define CRCLread  15

// Type redefine.
typedef unsigned char  uchar;
typedef unsigned long  ushort;
typedef unsigned char  BYTE;
typedef unsigned short WORD;

#define PAGE1  __attribute__((section("PAGE1"))) // HMS800 C compiler derivative for RAM page.
#define INT9   __attribute__((interrupt("9")))   // HMS800 C compiler derivative for UART Tx interrupt.
#define INT10  __attribute__((interrupt("10")))  // HMS800 C compiler derivative for UART Rx interrupt.

// Bit-accessable structure statement of 1byte.
union BitAccess
{
                uchar  ToByte;
                struct P8Bit
                {
                   char pb0:1 ; // 1
                   char pb1:1 ; // 2
                   char pb2:1 ; // 3
                   char pb3:1 ; // 4
                   char pb4:1 ; // 5
                   char pb5:1 ; // 6
                char pb6:1 ; // 7
                char pb7:1 ; // 8
             }  bitn;
};

volatile union BitAccess XModemFlags;
uchar ucState;             // State-machine event value.
uchar ucRetries;           // Retry count.
ushort usElapsedTime;      // Total elapsed time.
uchar ucBufferIndex;       // Index for the buffer addressing.
uchar ucStartOfHeader;     // The start of header character
uchar ucBlockNo;           // Storage for block number
uchar ucNotBlockNo;        // Complement of block number
uchar ucRXRstorage;        // Stored data of the UART Rx.
uchar ucXbuffer[1026]  PAGE1;   // First half of the 1026 bytes packet buffer

#define SendEndFlag     XModemFlags.bitn.pb0  // Set:None blocks to send.
#define CrcFlag         XModemFlags.bitn.pb1  // Set:CRC mode, Clear:Checksum mode.
#define ReadStbyFlag    XModemFlags.bitn.pb2  // Set:Ready to read a character.
#define TxNBusyFlag     XModemFlags.bitn.pb3
#define RxNBusyFlag     XModemFlags.bitn.pb4

// Proto-type functions.
char XModemR(void);
void Receive(void);
uchar XModemT(void);
void Transmit(void);
void SendCharacter(uchar);
uchar ReadCharacter(void);

// Functions used ,but unimplemented
/* Delay(),SendPacket(),AssemblePacket(),
   ErrorCheckForPacket(),ReadyToSendNextPacket() */

/*****************************************************************************************
 Function title : Initialize the XModem State Machine to begin a transfer.
 Input arguments : None.
 Return data : IDLE state for state display.
 Description : A file receive with the Xmodem-1K.
*****************************************************************************************/
char XModemR(void)
{
             XModemFlags.ToByte = 0;
             ucRetries = 10;                     // Number of total retries : 10.
             CrcFlag = 1;                        // CRC mode setup.
             ucState = CRCgo;                    // To start Xmodem Receive.
             while ( ucRetries > 0 )             // Escape if retries > 10.
             {
                usElapsedTime = 0;
                Receive();                       // Sends the 'C' or NAK.
                while ( ++usElapsedTime < 10000 ) // Escape if elapsed time > 10 sec.
                {
                   if (ReadStbyFlag)
                   {
                      ReadStbyFlag = 0;
                      ucState = SOHread;         // Start to the "SOHread" state
                      ucRetries = 10;            // 10 retry.
                      Receive();
                      while ( !( (ucState==IDLE)||(ucState==ABORT)||(ucState==TIMEOUT) ) )
                      {
                         if (ReadStbyFlag)
                         {
                                ReadStbyFlag = 0;     // Read standby flag initialize.
                                Receive();
                             }
                             if ( ucRetries == 0 )
                             {
                                ucState = TIMEOUT;    // Retry exceed.
                             }
                          }
                          return ucState;
                       }
                       Delay();
                    }
                    if(  --ucRetries == 5 )           // If "ucRetries <= 5", Checksum mode set.
                    {
                       CrcFlag = 0;
                       ucState = CHKSUMgo;            // Checksum mode setup.
                    }
                 }
                 ucState = IDLE;
                 return ucState;
}
/*******************************************************************************************
 Function title : XModem Receive.
 Input arguments : None.
 Return data : None.
 Description : State machine implementation for Xmodem receiving.
*******************************************************************************************/
void Receive(void)
{
                 uchar uc;
                 ucRXRstorage = ReadCharacter();
                 switch(ucState)
                 {
                    case PKTread:                      // Packet read state.
                       ucXbuffer[ucBufferIndex++] = ReadCharacter();
                       if ( ucBufferIndex == 1024 )
                       {
                          if (!CrcFlag)
                             ucState = CRCLread;       // Last byte checksum.
                          else
                             ucState = CRCHread;       // Last 2 bytes CRC-CCITT.
                       }
                       return;
                       break;
                    case SOHread:
                       if ( ucRXRstorage == STX )      // Xmodem-1K start of header.
                       {
                          ucStartOfHeader = ucRXRstorage;
                          ucState = BLKread;
                          return;
                       }
                       else if ( ucRXRstorage == CAN ) ucState = ABORT;
                                                       // Host cancel ?
                       else if ( ucRXRstorage == EOT ) // Packet receiving end ?
                       {
                          SendCharacter(ACK);
                          ucState = IDLE;              // Receive completion.
                          return;
                       }
                       ucRetries--;                    // Retry count decrement.
                       SendCharacter(NAK);
                       return;
```

```
                break;
             case BLKread:
                ucBlockNo = ucRXRstorage;       // Block number read.
                ucState = NBLKread;
                return;
                break;
             case NBLKread:
                ucNotBlockNo = ucRXRstorage;    // Complementary block number read.
                ucState = PKTread;
                return;
                break;
             case CRCHread:
                ucRxCrcHighByte = ucRXRstorage; // CRC higher byte read.
                ucState = CRCLread;
                return;
                break;
             case CRCLread:
          ucRxCrcLowByte = ucRXRstorage;  // CRC lower byte read.
          switch(ErrorCheckForPacket())
          {
             case OK:
                ucRetries = 5;
                ucState = SOHread;        // Get next packet
                ucBufferIndex = 0;
                SendCharacter(ACK);       // ACK the packet
                break;

             case BAD_BLKNUM:
                ucRetries--;
                ucState = SOHread;
                SendCharacter(NAK);       // The last packet resend request.
                break;

             case BAD_CRC:
                ucState = SOHread;
                ucRetries--;
                ucBufferIndex = 0;
                SendCharacter(NAK);       // The last packet resend request.
                break;
          }
          return;
                break;
             case CRCgo:
                ucBufferIndex = 0;
                SendCharacter('C');             // CRC mode response.
                return;
                break;
             case CHKSUMgo:
                ucBufferIndex = 0;
                SendCharacter(NAK);             // Checksum mode response.
                return;
                break;
             default: break;
          }
}

/*****************************************************************************************
 Function title : Xmodem transmission service routine.
 Input arguments : None.
 Return data : Present state value of the state-machine.
 Description : UART occurrence flag initialize.
*****************************************************************************************/
```

```
uchar XModemT(void)
{
                SendEndFlag = 0;
                usElapsedTime = 0;        // Elapsed time initialize.
                ucState = STANDBY;        // Start to the "STANDBY" state.
                ucOwnCodeHighAdr = 0x40;
                ucOwnCodeLowAdr = 0;
                while ( !( (ucState==IDLE)||(ucState==ABORT)||(ucState==TIMEOUT) ) )
                {
                   if (ReadStbyFlag)      // 1 byte receive from host ?
                   {
                      ReadStbyFlag = 0;   // Initialize flag for next receive.
                      Transmit();         // State-machine process.
                   }
                   else
                   {
                      Delay();            // Delay for elapsed timing : 200msec.
                      if ( usElapsedTime++ == 200 ) ucState = TIMEOUT; // Force exit if timed out
                   }
                }
                return ucState;
}

/******************************************************************************************
 Function title : State-machine implementation for the transmit.
 Input arguments : None.
 Return data : None.
 Description : State-machine implementation.
******************************************************************************************/
void Transmit(void)
{
                uchar uc;
                switch(ucState)
                {
                   case IDLE:                          // IDLE state -> Function escape.
                      break;
                   case STANDBY:                       // If the state is "STANDBY"
                      ucRetries = 10;                  // Initialize retry.
                      ucRXRstorage = ReadCharacter();
                      if ( ucRXRstorage == NAK )
                      {
                         CrcFlag = 0;                  // Checksum mode status.
                         ucBlockNo = 1;                // First block.
                         AssemblePacket();             // Build the packet.
                         ucState = START;
                      }
                      if ( ucRXRstorage == 'C' )
                      {
                         CrcFlag = 1;                  // Crc mode status.
                         ucBlockNo = 1;                // First block.
                         AssemblePacket();             // Building packet.
                         ucState = START;              // Goto START state.
                      }
                      break;
                   case ACKWAIT:                       // The "ACKWAIT" state ?
                      ucRXRstorage = ReadCharacter();
                      if ( ucRXRstorage == ACK )
                      {
                          if (SendEndFlag)             // Transmission end ?
                          {
                             SendCharacter(EOT);            // EOT character send.
                             ucState = IDLE;           // To the system "IDLE" state.
                          }
                          else                        // Next packet transmission.
                          {
                             ReadyToSendNextPacket();  // Function ready to send next packet.
                             ucState = NEXT;
                          }
                      }
                      if ( ucRXRstorage == NAK )
                      {
                         if ( --ucRetries != 0 )
                            ucState = RESEND;          // Packet resend.
                         else
                            ucState = TIMEOUT;         // Abort if retrys exceeded.
                      }
                      if ( ucRXRstorage == CAN ) ucState = ABORT; // Abort if CAN received
                      break;
                   case START:
                   case RESEND:
                      SendPacket();                   // Send the packet
                      usElapsedTime = 0;              // Reset timeout
                      ucState = ACKWAIT;              // Wait for an ACK or NAK
                      break;
                      break;
                   case NEXT:                         // If we are to send the next packet
                      ucBlockNo++;                    // Next block;
                      AssemblePacket();               // Build the packet
                      SendPacket();                   // Send the packet
                      usElapsedTime = 0;              // Reset timeout
                      ucState = ACKWAIT;              // Wait for an ACK or NAK
                      break;
                   default: break;
                }
}

/*****************************************************************************************
 Function title : Initialize UART interrupts.
 Input arguments : None.
 Return data : None.
 Description : UART occurrence flag initialize.
*****************************************************************************************/
void INT9 UART_TX_INT(void)
{
          TxNBusyFlag = 1;       // UART Tx occurrence.
}

void INT10 UART_RX_INT(void)
{
          RxNBusyFlag = 1;       // UART Rx occurrence.
}

/*****************************************************************************************
 Function title : Send a character with UART interrupts.
 Input arguments : Writing data to transmit register.
 Return data : None.
 Description : Send a character with UART Tx.
*****************************************************************************************/
void SendCharacter(uchar uc)
{
          TxNBusyFlag = 0;
                TXR = uc;               // Load the TXR.
                while(!TxNBusyFlag) {;}
}
```

```
/*******************************************************************************************
 Function title : Receive a character with UART interrupts.
 Input arguments : None.
 Return data : Received register of the HMS81C0048.
 Description : Receive a character with UART Rx.
*******************************************************************************************/
uchar ReadCharacter(void)
{
                RxNBusyFlag = 0;
                while(!RxNBusyFlag) {;}
                RxCharReady = 1;
                return RXR;            // Download the RXR.
}
```