Xmodem function source code

without html formatting: source code for xmodem.c

```
/*
 * Copyright 2001-2010 Georges Menie (www.menie.org)
 * All rights reserved.
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 *     * Redistributions of source code must retain the above copyright
 *       notice, this list of conditions and the following disclaimer.
 *     * Redistributions in binary form must reproduce the above copyright
 *       notice, this list of conditions and the following disclaimer in the
 *       documentation and/or other materials provided with the distribution.
 *     * Neither the name of the University of California, Berkeley nor the
 *       names of its contributors may be used to endorse or promote products
 *       derived from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND ANY
 * EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE REGENTS AND CONTRIBUTORS BE LIABLE FOR ANY
 * DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 * LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
 * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
 */

/* this code needs standard functions memcpy() and memset()
   and input/output functions _inbyte() and _outbyte().

   the prototypes of the input/output functions are:
     int _inbyte(unsigned short timeout); // msec timeout
     void _outbyte(int c);

 */

#include "crc16.h"

#define SOH  0x01
#define STX  0x02
#define EOT  0x04
#define ACK  0x06
#define NAK  0x15
#define CAN  0x18
#define CTRLZ 0x1A

#define DLY_1S 1000
#define MAXRETRANS 25

static int check(int crc, const unsigned char *buf, int sz)
{
        if (crc) {
                unsigned short crc = crc16_ccitt(buf, sz);
                unsigned short tcrc = (buf[sz]<<8)+buf[sz+1];
                if (crc == tcrc)
```

```
                        return 1;
        }
        else {
                int i;
                unsigned char cks = 0;
                for (i = 0; i < sz; ++i) {
                        cks += buf[i];
                }
                if (cks == buf[sz])
                return 1;
        }

        return 0;
}

static void flushinput(void)
{
        while (_inbyte(((DLY_1S)*3)>>1) >= 0)
                ;
}

int xmodemReceive(unsigned char *dest, int destsz)
{
        unsigned char xbuff[1030]; /* 1024 for XModem 1k + 3 head chars + 2 crc + nul */
        unsigned char *p;
        int bufsz, crc = 0;
        unsigned char trychar = 'C';
        unsigned char packetno = 1;
        int i, c, len = 0;
        int retry, retrans = MAXRETRANS;

        for(;;) {
                for( retry = 0; retry < 16; ++retry) {
                        if (trychar) _outbyte(trychar);
                        if ((c = _inbyte((DLY_1S)<<1)) >= 0) {
                                switch (c) {
                                case SOH:
                                        bufsz = 128;
                                        goto start_recv;
                                case STX:
                                        bufsz = 1024;
                                        goto start_recv;
                                case EOT:
                                        flushinput();
                                        _outbyte(ACK);
                                        return len; /* normal end */
                                case CAN:
                                        if ((c = _inbyte(DLY_1S)) == CAN) {
                                                flushinput();
                                                _outbyte(ACK);
                                                return -1; /* canceled by remote */
                                        }
                                        break;
                                default:
                                        break;
                                }
                        }
                }
                if (trychar == 'C') { trychar = NAK; continue; }
                flushinput();
                _outbyte(CAN);
                _outbyte(CAN);
                _outbyte(CAN);
                return -2; /* sync error */

        start_recv:
                if (trychar == 'C') crc = 1;
                trychar = 0;
                p = xbuff;
                *p++ = c;
                for (i = 0;  i < (bufsz+(crc?1:0)+3); ++i) {
                        if ((c = _inbyte(DLY_1S)) < 0) goto reject;
                        *p++ = c;
                }

                if (xbuff[1] == (unsigned char)(~xbuff[2]) &&
                        (xbuff[1] == packetno || xbuff[1] == (unsigned char)packetno-1) &&
                        check(crc, &xbuff[3], bufsz)) {
                        if (xbuff[1] == packetno)       {
                                register int count = destsz - len;
                                if (count > bufsz) count = bufsz;
                                if (count > 0) {
                                        memcpy (&dest[len], &xbuff[3], count);
                                        len += count;
                                }
                                ++packetno;
                                retrans = MAXRETRANS+1;
                        }
                        if (--retrans <= 0) {
                                flushinput();
                                _outbyte(CAN);
                                _outbyte(CAN);
                                _outbyte(CAN);
                                return -3; /* too many retry error */
                        }
                        _outbyte(ACK);
                        continue;
                }
        reject:
                flushinput();
                _outbyte(NAK);
        }
}

int xmodemTransmit(unsigned char *src, int srcsz)
{
        unsigned char xbuff[1030]; /* 1024 for XModem 1k + 3 head chars + 2 crc + nul */
        int bufsz, crc = -1;
        unsigned char packetno = 1;
        int i, c, len = 0;
        int retry;

        for(;;) {
                for( retry = 0; retry < 16; ++retry) {
                        if ((c = _inbyte((DLY_1S)<<1)) >= 0) {
                                switch (c) {
                                case 'C':
                                        crc = 1;
                                        goto start_trans;
                                case NAK:
                                        crc = 0;
                                        goto start_trans;
                                case CAN:
                                        if ((c = _inbyte(DLY_1S)) == CAN) {
                                                _outbyte(ACK);
                                                flushinput();
                                                return -1; /* canceled by remote */
                                        }
                                        break;
                                default:
                                        break;
                                }
                        }
                }
                _outbyte(CAN);
                _outbyte(CAN);
                _outbyte(CAN);
                flushinput();
                return -2; /* no sync */

                for(;;) {
                start_trans:
                        xbuff[0] = SOH; bufsz = 128;
                        xbuff[1] = packetno;
                        xbuff[2] = ~packetno;
                        c = srcsz - len;
                        if (c > bufsz) c = bufsz;
                        if (c >= 0) {
                                memset (&xbuff[3], 0, bufsz);
                                if (c == 0) {
                                        xbuff[3] = CTRLZ;
                                }
                                else {
                                        memcpy (&xbuff[3], &src[len], c);
                                        if (c < bufsz) xbuff[3+c] = CTRLZ;
                                }
                                if (crc) {
                                        unsigned short ccrc = crc16_ccitt(&xbuff[3], bufsz);
                                        xbuff[bufsz+3] = (ccrc>>8) & 0xFF;
                                        xbuff[bufsz+4] = ccrc & 0xFF;
                                }
                                else {
                                        unsigned char ccks = 0;
                                        for (i = 3; i < bufsz+3; ++i) {
                                                ccks += xbuff[i];
                                        }
                                        xbuff[bufsz+3] = ccks;
                                }
                                for (retry = 0; retry < MAXRETRANS; ++retry) {
                                        for (i = 0; i < bufsz+4+(crc?1:0); ++i) {
                                                _outbyte(xbuff[i]);
                                        }
                                        if ((c = _inbyte(DLY_1S)) >= 0 ) {
                                                switch (c) {
                                                case ACK:
                                                        ++packetno;
                                                        len += bufsz;
                                                        goto start_trans;
                                                case CAN:
                                                        if ((c = _inbyte(DLY_1S)) == CAN) {
                                                                _outbyte(ACK);
                                                                flushinput();
                                                                return -1; /* canceled by remote */
                                                        }
                                                        break;
                                                case NAK:
```

```
                                                default:
                                                        break;
                                                }
                                        }
                                }
                                _outbyte(CAN);
                                _outbyte(CAN);
                                _outbyte(CAN);
                                flushinput();
                                return -4; /* xmit error */
                        }
                        else {
                                for (retry = 0; retry < 10; ++retry) {
                                        _outbyte(EOT);
                                        if ((c = _inbyte((DLY_1S)<<1)) == ACK) break;
                                }
                                flushinput();
                                return (c == ACK)?len:-5;
                        }
                }
        }
}

#ifdef TEST_XMODEM_RECEIVE
int main(void)
{
        int st;

        printf ("Send data using the xmodem protocol from your terminal emulator now...\n");
        /* the following should be changed for your environment:
           0x30000 is the download address,
           65536 is the maximum size to be written at this address
         */
        st = xmodemReceive((char *)0x30000, 65536);
        if (st < 0) {
                printf ("Xmodem receive error: status: %d\n", st);
        }
        else  {
                printf ("Xmodem successfully received %d bytes\n", st);
        }

        return 0;
}
#endif
#ifdef TEST_XMODEM_SEND
int main(void)
{
        int st;

        printf ("Prepare your terminal emulator to receive data now...\n");
        /* the following should be changed for your environment:
           0x30000 is the download address,
           12000 is the maximum size to be send from this address
         */
        st = xmodemTransmit((char *)0x30000, 12000);
        if (st < 0) {
                printf ("Xmodem transmit error: status: %d\n", st);
        }
        else  {
                printf ("Xmodem successfully transmitted %d bytes\n", st);
        }
        return 0;
}
#endif
```