

```

/*****
/*
/*          S E R T R A N S . C
/*
**-----**
/* Task      : File transfer through serial port
/*            with error correction
/*
**-----**
/* Author     : Michael Tischer / Bruno Jennrich
/* Developed on : 04/08/1994
/* Last update  : 04/07/1995
/*
**-----**
/* COMPILER   : Borland C++ 3.1, Microsoft Visual C++ 1.5
/*****
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <process.h>

#include "serutil.h"
#include "sertrans.h"

/*- Document the following lines within project -----*/
#include "serutil.c"
#include "win.c"
#include "irqutil.c"
#include "args.c"

/*****
/* CalcCRC : Calculate cyclic redundancy checksum
/*
**-----**
/* Input   : uCRC - previous checksum
/*           bData - byte to be incorporated
/* Output  : New checksum including passed byte
/*
**-----**
/* Info   : In the first call to this function the "previous"
/*           checksum must be set to 0xFFFF.
/*
**-----**
UINT CalcCRC( UINT uCRC, BYTE bData )
{ static INT j;
  for( j = 0; j < 8; j++ )
  {
    uCRC <= 1;
    if( bData & 0x0080 ) uCRC ^= CCITT_POLYNOM;
    bData <= 1;
  }
  return uCRC;
}

/*****
/* Packet_Build : Initialize structure of type PACKET for sending
/*
**-----**
/* Input   : pPacket - address of packet being initialized
/*           bType    - user type of packet
/*           pData    - address of packet data
/*           uDataSize - number of valid bytes
/*                   in packet
/*           uNr      - packet number (consecutive)
/*
**-----**
VOID Packet_Build( PPACKET pPacket,
                  BYTE      bType,
                  PBYTE     pData,
                  UINT       uDataSize,
                  UINT       uNr )
{
  UINT i;
  pPacket->bType = bType;
  pPacket->uDataSize = uDataSize;
  pPacket->uNr = uNr;
  for( i = 0; i != uDataSize; i++ )
    pPacket->bData[ i ] = pData[ i ];
}

/*****
/* trans_WriteUINT : Send single UINT via serial port
/*
**-----**
/* Input   : iSerPort - base port of interface
/*           uData     - UINT being sent
/* Output  : Error status
/*
**-----**

```

```

/**-----**/
/* Info : This function first sends the LO-, then the HI-          */
/*        BYTE of a UINT. Warning - no handshake!                  */
/******/
INT trans_WriteUINT( INT iSerPort, UINT uData )
{
    if( ser_WriteByte( iSerPort,
                       LOBYTE( uData ),
                       0x8000,
                       0,
                       0 ) == SER_SUCCESS )
        if( ser_WriteByte( iSerPort,
                           HIBYTE( uData ),
                           0x8000,
                           0,
                           0 ) == SER_SUCCESS )
            return TRANS_SUCCESS;
    return TRANS_TRANSERROR;
}

/******/
/* trans_ReadUINT : Read a single UINT from port                    */
/**-----**/
/* Input  : iSerPort - base port of interface                      */
/*          pData   - address of UINT accepting                    */
/*          received value.                                         */
/* Output : Error status                                           */
/******/
INT trans_ReadUINT( INT iSerPort, PUINT pData )
{
    BYTE bErrL, bErrH;
    if( ser_ReadByte( iSerPort, &bErrL, 0x8000, 0, 0 ) == SER_SUCCESS )
        if( ser_ReadByte( iSerPort, &bErrH, 0x8000, 0, 0 ) == SER_SUCCESS )
        {
            *pData = MAKEWORD( bErrH, bErrL );
            return TRANS_SUCCESS;
        }
    return TRANS_TRANSERROR;
}

/******/
/* trans_WriteNak : Send negative acknowledge via port              */
/**-----**/
/* Input  : iSerPort - base port of interface                      */
/*          iCode    - extended error code                          */
/* Output : Error status                                           */
/******/
INT trans_SendNak( INT iSerPort, UINT iCode )
{
    if( ser_WriteByte( iSerPort, ASCII_NAK, 0x8000,0,0 ) == SER_SUCCESS )
        return trans_WriteUINT( iSerPort, iCode );
    return TRANS_TRANSERROR;
}

/******/
/* trans_WriteAck : Send (positive) acknowledge via port            */
/**-----**/
/* Input  : iSerPort - base port of interface                      */
/* Output : Error status                                           */
/******/
INT trans_SendAck( INT iSerPort )
{
    if( ser_WriteByte( iSerPort, ASCII_ACK, 0x8000,0,0 ) == SER_SUCCESS )
        return TRANS_SUCCESS;
    return TRANS_TRANSERROR;
}

/******/
/* trans_SendPacket : Send entire data packet via port             */
/**-----**/
/* Input  : iSerPort - base port of interface being used          */
/*          pPacket   - address of data                            */
/*          iPacketSize - data quantity                            */
/* Output : Error status                                           */
/******/
INT trans_SendPacket( INT iSerPort, PBYTE pPacket, UINT iPacketSize )
{

```



```

        uAction = SEND_GETREPLY;
    else return TRANS_TRANSERROR;
    break;
    default: uTimeOut--; break;
}
}
}
return TRANS_TIMEOUT;
}

/*****
/* trans_ReceivePacket :Receive data packet from port */
/*-----*/
/* Input   : iSerPort    - base port of interface being used */
/*          pPacket      - address of receive buffer */
/*          iPacketSize  - buffer size */
/* Output  : Error status */
*****/
INT trans_ReceivePacket( INT iSerPort, PBYTE pPacket, UINT iPacketSize )
{
    UINT uCRC, uAction, uTimeOut, uActByte;
    BYTE bData, bDataOk, bCrcLo;          /* First byte of CRC code */
    INT iAckError;

    uTimeOut = 0xFFFF;
    uAction = RECEIVE_SIGNAL;

    uCRC = 0xFFFF;                        /* Access variables at least once */
    uActByte = 0;

    while( uTimeOut )
    {
        bDataOk = FALSE;                  /* No data available */
        if( ser_IsDataAvailable( iSerPort ) )
            if( ser_ReadByte( iSerPort, &bData, 1, 0, 0 ) != SER_SUCCESS )
                return TRANS_TRANSERROR;
            else bDataOk = TRUE;           /* Or is there? */

        if( bDataOk )
        {
            switch( uAction )
            {
                case RECEIVE_RECEIVE:
                    uCRC = CalcCRC( uCRC, bData );
                    pPacket[ uActByte++ ] = bData;
                    if( uActByte == iPacketSize ) uAction = RECEIVE_GETCRCLO;
                    uTimeOut = 0xFFFF;
                    break;

                case RECEIVE_GETCRCLO:
                    bCrcLo = bData;
                    uAction = RECEIVE_GETCRCHI;
                    break;

                case RECEIVE_GETCRCHI:
                    if( uCRC == MAKEWORD( bData, bCrcLo ) )
                        uAction = RECEIVE_PUTACK;
                    else
                    {
                        iAckError = TRANS_CRC;
                        uAction = RECEIVE_PUTNAK;
                    }
                    break;
            }
        }
        else
            if( ser_IsWritingPossible( iSerPort ) )
            {
                switch( uAction )
                {
                    case RECEIVE_SIGNAL:
                        if( ser_WriteByte( iSerPort,
                                           ASCII_SYN,
                                           1,
                                           0,
                                           0 ) == SER_SUCCESS )
                        {

```

```

        uCRC = 0xFFFF;
        uActByte = 0;
        uAction = RECEIVE_RECEIVE;
    }
    else return TRANS_TRANSERROR;
break;
case RECEIVE_PUTACK:
    if( ser_WriteByte( iSerPort,
                        ASCII_ACK,
                        1,
                        0,
                        0 ) == SER_SUCCESS )
        return TRANS_SUCCESS;
    else return TRANS_TRANSERROR;

case RECEIVE_PUTNAK:
    if( ser_WriteByte( iSerPort,
                        ASCII_NAK,
                        1,
                        0,
                        0 ) == SER_SUCCESS )
        uAction = RECEIVE_PUTNAKLO;
    else return TRANS_TRANSERROR;
break;
case RECEIVE_PUTNAKLO:
    if( ser_WriteByte( iSerPort,
                        LOBYTE( iAckError ),
                        1,
                        0,
                        0 ) == SER_SUCCESS )
        uAction = RECEIVE_PUTNAKHI;
    else return TRANS_TRANSERROR;
break;
case RECEIVE_PUTNAKHI:
    if( ser_WriteByte( iSerPort,
                        HIBYTE( iAckError ),
                        1,
                        0,
                        0 ) == SER_SUCCESS )
        return iAckError;
    else return TRANS_TRANSERROR;
default: uTimeOut--;
    }
}
}
return TRANS_TIMEOUT;
}

/*****
/* trans_ReceiveFile : Receive entire file via port */
/*-----*/
/* Input  : iSerPort - base port of interface being used */
/* Output : Error status */
*****/
INT trans_ReceiveFile( INT iSerPort )
{
    PACKET Packet;
    INT iTimeOut;
    FILE *fHandle = NULL;
    UINT uNr = 0;

    iTimeOut = 10; /* Wait a bit longer for first packet */
    while( iTimeOut )
    {
        INT err;
        err = trans_ReceivePacket( iSerPort, /* Receive packet */
                                   ( PBYTE ) &Packet,
                                   sizeof( Packet ) );

        if( err == TRANS_SUCCESS )
        {
            iTimeOut = 4; /* Shorten timeout */
            if( uNr != Packet.uNr )
            {
                if( fHandle ) fclose( fHandle );
                trans_SendNak( iSerPort, ( UINT )PACKET_SEQUENCE );
                return PACKET_SEQUENCE;
            }
        }
        else

```



```

    } while( ( err == TRANS_TIMEOUT ) && ( iCnt ) );

    if( err != TRANS_SUCCESS )
    {
        fclose( fHandle );
        return err;
    }

    do
    {
        iSize = fread( Packet.bData, 1, PACKET_DATASIZE, fHandle );
        if( iSize )
        {
            Packet_Build( &Packet,
                          PACKET_FILEDATA,
                          Packet.bData,
                          iSize,
                          uNr++ );

            iCnt = 10;
            printf( "\r%d", uNr );
            do
            {
                iCnt--;
                err = trans_SendPacket( iSerPort,
                                       ( PBYTE )&Packet,
                                       sizeof( PACKET ) );
            } while ( ( err == TRANS_TIMEOUT ) && ( iCnt ) );
            if( err != TRANS_SUCCESS )
            {
                fclose( fHandle );
                return err;
            }
        }
    } while( iSize == PACKET_DATASIZE );

    fclose( fHandle );
    Packet_Build( &Packet,
                  PACKET_FILECLOSE,
                  ( PBYTE )&pName[iLen],
                  strlen( &pName[iLen] ),
                  uNr++ );

    iCnt = 10;
    printf( "\r%d", uNr );
    do
    {
        iCnt--;
        err = trans_SendPacket( iSerPort,
                               ( PBYTE )&Packet,
                               sizeof( PACKET ) );
    } while( ( err == TRANS_TIMEOUT ) && ( iCnt ) );
    return err;
}
return TRANS_NOFILE;
}

/*****
/* trans_PError : Output error message
**-----*/
/* Input : iError - error that occurred
*****/
VOID trans_PError( INT iError )
{
    switch( iError )
    {
        case TRANS_SUCCESS:
            printf( "Transmission OK!\n" ); break;
        case TRANS_TIMEOUT:
            printf( "ERROR: Byte timeout" ); break;
        case TRANS_TRANSERROR:
            printf( "ERROR: General error\n" ); break;
        case TRANS_PROTOCOL:
            printf( "ERROR: Ack or NAck missing!\n" ); break;
        case TRANS_CRC:
            printf( "ERROR: CRC error!" ); break;
        case TRANS_NOFILE:

```

```

        printf("ERROR: Cannot open file!\n");      break;
    case PACKET_SEQUENCE:
        printf("ERROR: Invalid packet sequence!\n"); break;
    case PACKET_UNKNOWN:
        printf("ERROR: Unknown packet ID!\n");      break;
    case PACKET_TIMEOUT:
        printf("ERROR: Packet timeout\n" );        break;
}
}

/*****
*/ M A I N   P R O G R A M
*/
*****/

VOID main(INT argc, PCHAR argv[] )
{
    INT iSerPort, iCom;
    LONG lBaud;
    PCHAR pFileName;

    if( FindString( argv, "?", argc ) )
    {
        printf("Call:\n" );
        printf("SERTRANS File [-COM:comport] [-BAUD:baudrate]\n");
        printf("COM port = 1 or 2 (Default: 1 )\n");
        printf("Baud rate = 50 - 115200 (Default: 9600)\n");
        exit(0);
    }

    if( GetArg( argc, argv, "-COM:", _int, &iCom, 1 ) )
    {
        if( iCom == 1 )
            iSerPort = SER_COM1; /* Only COM1 and COM2 are "standardized" */
        else
            if( iCom == 2 )
                iSerPort = SER_COM2; /* Only COM1 and COM2 are "standardized" */
            else
            {
                printf("Unsupported COM port!\n");
                exit(0);
            }
    }
    else
        iSerPort = SER_COM1; /* Only COM1 and COM2 are "standardized" */

    if( !GetArg( argc, argv, "-BAUD:", _long, &lBaud, 1 ) )
        lBaud = 9600L; /* Maximum baud rate for UART 8450A */
    if( lBaud > SER_MAXBAUD )
    {
        printf("Baud rate too high!\n");
        printf("Maximum: %ld Bd\n", SER_MAXBAUD );
    }

    if( !ser_Init( iSerPort, lBaud, /* Maximum for UART 8450A */
                  SER_LCR_8BITS | SER_LCR_1STOPBIT | SER_LCR_NOPARITY ) )
    {
        printf("No port!\n");
        exit( 0 );
    }

    if( GetNArg( argc, argv, "-", &pFileName, 1 ) )
    {
        printf("Sending\n" );
        trans_PError( trans_SendFile( iSerPort, pFileName ) );
    }
    else
    {
        printf("Receiving\n");
        trans_PError( trans_ReceiveFile( iSerPort ) );
    }
}

```