```c
#ifndef SLIP_H
#define SLIP_H
/** \file slip.h*/
/** Implements SLIP(Seriel Line Internet Protocol) transfer over serial communication(RS-232).
 *          For info: http://tools.ietf.org/html/rfc1055
 *
 *          Notes:
 *          <ul>
 *              <li> Can send a maximum of DATA_MAX characters at a time
 *          </ul>
 *  \author NSRD
 *  \date 17/05 -11
 */
#include <ezV24/ezV24.h> // Used for serial connection

// Global constants
#define DATA_MAX 1050

#ifdef __cplusplus
    extern "C" {
#endif

// Functions
/** \brief Makes connection over "/dev/ttyS1".
*   \pre Another serial connection is not open.(Remember to close after using this connection)
*   \post A connection is made(Returns '1' if open else '0') with the settings for port and baud rate(Good idea to use ezV24 enum for values).
*/
int SLIPConnect(char* port, int baudRate);

/** \brief Closes connection.
*   \pre A serial connection is open.
*   \post The connection is closed.
*/
void SLIPClose();

/** \brief Sends a package.
*   \pre A connection is open. Data contains minimum 'n' number of bytes. 'n' is less than or equal to DATA_MAX.
*   \post 'n' number of bytes from 'data' is sent.
*/
void sendPackage(char* data, size_t n);

/** \brief Receives a package.
*   \pre A connection is open. Data can contain atleast DATA_MAX.
*   \post 'data' now contains the data from the package and the number of bytes received is returned.(0 if error stream or timeout)
*/
size_t receivePackage(char data[]);

#ifdef __cplusplus
    }
#endif

#endif

```

```c
// File: slip.c
/*
    Description: Implements 'slip.h' functions for serial communication.
    Author: NSRD
    Date: 24/05 -11
*/
#include "slip.h"

// Global defines
#define FRAME_CHAR 'A' // Start and end of data
#define FRAME_CHAR_SUB1 'B' // Substitute character 1 in case of FRAME_CHAR occuring in data
#define FRAME_CHAR_SUB2 'C' // Substitute character 2 in case of FRAME_CHAR occuring in data
#define SUB_SUB 'D' // If FRAME_CHAR_SUB1 occurs in data

// Global variables
v24_port_t* currentConnection = 0; // Used to keep track of connection


// Implementation of the functions in "slip.h"
int SLIPConnect(char* port, int baudRate)
{
    // Tmp variables
    int params = 0;

    // Open
    currentConnection = v24OpenPort(port, V24_STANDARD);
    if(currentConnection == 0) // Error occured
        return(0);

    // Settings
    /*
     *  Baud rate: 9600
     *  Databit: 8 bit
     *  Parity bit generation: Disabled
     */
    params = v24SetParameters(currentConnection, baudRate, V24_8BIT, V24_NONE);
    if(params!=V24_E_OK) // Error occured
    {
        v24ClosePort(currentConnection);
        return(0);
    }
    v24SetTimeouts (currentConnection, 50); // Timeout(So doesn´t waits forever in reading loop)(Sat to 5 seconds)

    return(1);
}

void SLIPClose()
{
    v24ClosePort(currentConnection);
}

void sendPackage(char* data, size_t n)
{
    // Check size
    if(n > DATA_MAX)
        return;

    // Start of package
    v24Putc(currentConnection, FRAME_CHAR);

    // Send data
    while(n) // For each byte
    {
        // Case of FRAME_CHAR or FRAME_CHAR_SUB1 in data(Else just send)
        switch(*data)
        {
            case FRAME_CHAR:
                v24Putc(currentConnection, FRAME_CHAR_SUB1);
                v24Putc(currentConnection, FRAME_CHAR_SUB2);
            break;
            case FRAME_CHAR_SUB1:
                v24Putc(currentConnection, FRAME_CHAR_SUB1);
                v24Putc(currentConnection, SUB_SUB);
            break;
            default: // Just send
                v24Putc(currentConnection, *data);
            break;
        }
        // Next byte
        data++;
        n--;
    }

    // End of package
    v24Putc(currentConnection, FRAME_CHAR);
}

size_t receivePackage(char data[])
{
    // Tmp variables
    int currentChar = 0; // Current char read from buffer('int' because checking for errors[-1])
    int specialChar = 0; // In case of byte stuffing('int' because checking for errors[-1])
    size_t index = 0; // Index for changing 'data'
    size_t read = 0; // How many bytes read


    // Get data
    while(1) // Until end of 'frame' encountered
    {
        // Read one char
        currentChar = v24Getc (currentConnection);

        // Check for error
        if(currentChar == -1)
            return(0);

        // Check if special char or just normal
        if((char)currentChar == FRAME_CHAR) // Case if beginning of package or end of package
        {
            if(read > 0) // Have read something so should be end of package
                return(read);
        }
```

```
113             else if((char)currentChar == FRAME_CHAR_SUB1)
114             {
115                 specialChar = v24Getc (currentConnection); // Get next char
116                 switch((char)specialChar)
117                 {
118                     case FRAME_CHAR_SUB2:
119                         data[index] = FRAME_CHAR;
120                     break;
121                     case SUB_SUB:
122                         data[index] = FRAME_CHAR_SUB1;
123                     break;
124                     default: // Error occured(But still send, then upper-layer should decide what to do)
125                         data[index] = (char)specialChar;
126                         return(0);
127                     break;
128                 }
129                 read++;
130                 index++;
131             }
132             else // Normal
133             {
134                 data[index] = (char)currentChar;
135                 read++;
136                 index++;
137             }
138         }
139     }
140
```

```c
#ifndef TRANSPORTLAG_H
#define TRANSPORTLAG_H
// File: transportlag.h
/*
    Description: Transport layer functions for serial communication.
    Author: NSRD
    Date: 24/05 -11
*/
#include "slip.h"

#define MAX_DATA_LENGTH = 1024; // Max amount of data able to send(If trying to send larger amount then it only sends this amount of bytes)

void serialSend(char * data , int size/* size = der skal sendes */);
int serialReceive(char * data);
void serialConnect(char* port, int baudRate);
void serialClose();

#endif

```

```cpp
// File: transportlag.cpp
/*
    Description: Transportlag for serial file sharing.
    Author: NSRD
    Date: 24/05 -11
*/

#include "CRC-16.h"
#include "transportlag.h"
#include <cstdio> // printf function.


void serialSend(char * data , int size) // size = der skal sendes
{
    // Size checking
    if(size > MAX_DATA_LENGTH)
        size = MAX_DATA_LENGTH;

    // Initiate used variables etc.
    static char seq = 0;
    char crchigh;
    char crclow;
    char array[1028];
    char buffer[1028];

    // Calculate the CRC16
    crcCalc(data, size, crchigh, crclow);

    // Define each part of transportlayer into an array.
    array[0] = crchigh;
    array[1] = crclow;
    array[2] = 0;   // Data to be sent
    array[3] = seq;

    // Fill up the rest of the array(1024) with the data to be sent.
    for(int i = 4; i < size+4; i++)
    {
        array[i] = data[i-4];
    }

    do   // This has to be done once, no matter what.
    {
    sendPackage(array, size+4); // Send crc+data+seq+databit (everything)

    // Print sent data.
    printf("Data: %s\n", array + 4);

    while(!receivePackage(buffer)) // Wait till we get something back from other end.
        {}

    }while(!buffer[2] == 1 && buffer[3] == seq); // Keep doing before mentioned till we get awknowledge and the seq are the same as we sent.

    seq = !seq; // Keep changing seq after each succesfull delivered package, to make sure that the old/new isnt alike this in the checks. (to prevent packet loss)
}

int serialReceive(char * data)
{
    // Initiate variables
    char buffer[1028];
    char crchigh;
    char crclow;
    int read = 0;

    // Keep recieving data till CRC is correct on own/sent calculated CRC.
    do
    {
        while(!(read = receivePackage(buffer))) // Wait to recieve something
            {}

        // Print out recieved data.
        printf("Data: %s\n", buffer+4);

        // Calculate CRC16 from the recieved info
        crcCalc(buffer+4, read-4, crchigh, crclow);

        if(crchigh == buffer[0] && crclow == buffer[1]) // Hvis pakke er rigtig
        {
            char ack[] = {0,0,1,buffer[3]};     // Make array with ack with seq number.
            sendPackage(ack, 4);                // Send the ack package back to sender.
            break;                              // Break out of do while(1)
        }
        else // Fejl i pakke
        {
            char wrong[] =  {0,0,0,buffer[3]};  // If recieved package is wrong, make new package with no acknowledge, but with recieved seq
            sendPackage(wrong , 4);             // Send the wrong package back to sender.
        }
    }while(1);

    for(int i = 4; i < read; i++) // Copy recieved to 'data'
        data[i-4] = buffer[i];

    return(read-4);  // Return only data bytes recieved.
}

void serialConnect(char* port, int baudRate)
{
   SLIPConnect(port, baudRate); // Connect through slip with certain port/boudrate.
}

void serialClose()
{
    SLIPClose(); // Close slip connection
}

```

```
#ifndef CRC_16_H
#define CRC_16_H
// File: CRC-16.h

void crcCalc(char *buf, int len,char &crc1, char &crc2);

#endif

```

```
// Projekt        DKT1 Øvelser
//
// Fil            CRC-16.cpp
// Forfatter      David Schwaderer
//                ændret af Erik Gross Jensen
//
// Beskrivelse    Udregner CRC16 på et array af char's .

// Version:
// The logic for this method of calculating the CRC 16 bit polynomial is taken
// from an article by David Schwaderer in the April 1985 issue of PC Tech
// Journal.
//                1.1 04022002 tilpasset DKT1.
//
//
//

#include "CRC-16.h"


//************************************************************
//Konstruere en CRC lookup tabel
//************************************************************
  int     crctab[] = /* CRC lookup table */
{
 0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0xC241,
 0xC601, 0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481, 0x0440,
 0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0E40,
 0x0A00, 0xCAC1, 0xCB81, 0x0B40, 0xC901, 0x09C0, 0x0880, 0xC841,
 0xD801, 0x18C0, 0x1980, 0xD941, 0x1B00, 0xDBC1, 0xDA81, 0x1A40,
 0x1E00, 0xDEC1, 0xDF81, 0x1F40, 0xDD01, 0x1DC0, 0x1C80, 0xDC41,
 0x1400, 0xD4C1, 0xD581, 0x1540, 0xD701, 0x17C0, 0x1680, 0xD641,
 0xD201, 0x12C0, 0x1380, 0xD341, 0x1100, 0xD1C1, 0xD081, 0x1040,
 0xF001, 0x30C0, 0x3180, 0xF141, 0x3300, 0xF3C1, 0xF281, 0x3240,
 0x3600, 0xF6C1, 0xF781, 0x3740, 0xF501, 0x35C0, 0x3480, 0xF441,
 0x3C00, 0xFCC1, 0xFD81, 0x3D40, 0xFF01, 0x3FC0, 0x3E80, 0xFE41,
 0xFA01, 0x3AC0, 0x3B80, 0xFB41, 0x3900, 0xF9C1, 0xF881, 0x3840,
 0x2800, 0xE8C1, 0xE981, 0x2940, 0xEB01, 0x2BC0, 0x2A80, 0xEA41,
 0xEE01, 0x2EC0, 0x2F80, 0xEF41, 0x2D00, 0xEDC1, 0xEC81, 0x2C40,
 0xE401, 0x24C0, 0x2580, 0xE541, 0x2700, 0xE7C1, 0xE681, 0x2640,
 0x2200, 0xE2C1, 0xE381, 0x2340, 0xE101, 0x21C0, 0x2080, 0xE041,
 0xA001, 0x60C0, 0x6180, 0xA141, 0x6300, 0xA3C1, 0xA281, 0x6240,
 0x6600, 0xA6C1, 0xA781, 0x6740, 0xA501, 0x65C0, 0x6480, 0xA441,
 0x6C00, 0xACC1, 0xAD81, 0x6D40, 0xAF01, 0x6FC0, 0x6E80, 0xAE41,
 0xAA01, 0x6AC0, 0x6B80, 0xAB41, 0x6900, 0xA9C1, 0xA881, 0x6840,
 0x7800, 0xB8C1, 0xB981, 0x7940, 0xBB01, 0x7BC0, 0x7A80, 0xBA41,
 0xBE01, 0x7EC0, 0x7F80, 0xBF41, 0x7D00, 0xBDC1, 0xBC81, 0x7C40,
 0xB401, 0x74C0, 0x7580, 0xB541, 0x7700, 0xB7C1, 0xB681, 0x7640,
 0x7200, 0xB2C1, 0xB381, 0x7340, 0xB101, 0x71C0, 0x7080, 0xB041,
 0x5000, 0x90C1, 0x9181, 0x5140, 0x9301, 0x53C0, 0x5280, 0x9241,
 0x9601, 0x56C0, 0x5780, 0x9741, 0x5500, 0x95C1, 0x9481, 0x5440,
 0x9C01, 0x5CC0, 0x5D80, 0x9D41, 0x5F00, 0x9FC1, 0x9E81, 0x5E40,
 0x5A00, 0x9AC1, 0x9B81, 0x5B40, 0x9901, 0x59C0, 0x5880, 0x9841,
 0x8801, 0x48C0, 0x4980, 0x8941, 0x4B00, 0x8BC1, 0x8A81, 0x4A40,
 0x4E00, 0x8EC1, 0x8F81, 0x4F40, 0x8D01, 0x4DC0, 0x4C80, 0x8C41,
 0x4400, 0x84C1, 0x8581, 0x4540, 0x8701, 0x47C0, 0x4680, 0x8641,
 0x8201, 0x42C0, 0x4380, 0x8341, 0x4100, 0x81C1, 0x8081, 0x4040
};


//************************************************************************
void crcCalc(char *buf, int len,char &crc1, char &crc2)
// Udregner CRC16 på en given buffer
// Input :
// *buf : Pointer til en char buffer som der skal udregnes cheksum på
// len  : Antal chars i buf
// Output:
// crc1 : Her retuneres mest betydende byte af den udregnede crc
// crc2 : Her retuneres mindst betydende byte af den udregnede crc
//************************************************************************
{
    int i,crc=0;

    for (i=0; i<len; i++)
        crc = ((crc >> 8) & 0xff) ^ crctab[(crc ^ *buf++) & 0xff];

    //opdel 16 bit crc i 2 bytes
    crc1 = crc / 256;
    crc2 = crc % 256;

}
```

```cpp
// File: client.cpp
/*
    Description: Client for serial file sharing.
    Author: NSRD
    Date: 24/05 -11
*/

#include <stdio.h>  // file function etc.
#include <stdlib.h> //atoi function.
#include <string.h> // memset function.

#include "transportlag.h"

#define MAXRCVLEN 1024

int main(int argc, char *argv[])
{
    //-------------------------------------------------------------------
    // Init variabler og stringe.
    char* filnavn = argv[1];
    char buffer[MAXRCVLEN + 1];

    int filelenght;
    //-------------------------------------------------------------------
    // Connect to USB serial.
    serialConnect("/dev/ttyUSB0", V24_B9600); // USB port and 9600 baud rate
    //-------------------------------------------------------------------
    // Send requested filename, then recieve filelenght
    printf("Client started:\n");

    serialSend(filnavn, strlen(filnavn)+1); // Send filename.

    serialReceive(buffer); // Recieve filelenght in bytes as string

    filelenght = atoi(buffer); /* Convert string with bytes to integer */

    if(filelenght == 0) /* If file is not known by server */
    {
        printf("No such file, exiting.\n");
        exit(1); /* Close terminal */
    }

    //-------------------------------------------------------------------------------
    // Open new file/existing with specified filename and write in binary.

    FILE *fp;
    fp = fopen(filnavn, "wb");

    //-------------------------------------------------------------------------------
    // Print and serialSend file.

    printf("Filesize received %s.\n", buffer); // Print out the filelenght

    memset(&buffer, 0, sizeof(buffer)); /* Set memory to 0's */

    while(filelenght > MAXRCVLEN) /* Recieve and write to file while bytes are 1024 and above */
    {
        serialReceive(buffer); // Recieve more data

        fwrite(buffer, 1, MAXRCVLEN ,fp); /* Write to file */

        memset(&buffer, 0, sizeof(buffer));

        filelenght = filelenght - MAXRCVLEN; /* Calculate remaining bytes */

    }

    // Manage recieving and writing to file when bytes is 1024 and below (Last part of file).
    serialReceive(buffer);
    fwrite(buffer, 1, filelenght, fp);

    memset(&buffer, 0, sizeof(buffer));

    //-----------------------------------------------------------
    // End program.

    printf("End of Program\n");

    fclose(fp); /* Close file */

    serialClose(); // Close serial connection

   return EXIT_SUCCESS;
}
```

```
#ifndef SERVER_H
#define SERVER_H
// File: Server.h
/*
    Description: Server class for file transfer over serial connection.
    Author: NSRD
    Date: 17/05 -11
*/
#include "transportlag.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFFERSIZE 1024

class Server
{
    public:

    Server(char* port, int baudRate); // Note: Baud rate values can be found in 'ezV24' enum.
    ~Server();
    void waitForClient();

    private:

    char msgBuffer[BUFFERSIZE+1];
    long getFileLength(FILE* file);
    void sendFile(FILE* file, long fileSize);
};
#endif

```

```cpp
// File: Server.cpp
/*
    Description: Server class for file transfer over serial connection.
    Author: NSRD
    Date: 17/05 -11
*/

#include "Server.h"

Server::Server(char* port, int baudRate)
{
    serialConnect(port, baudRate); // Connect to serial with defined port and baudrate.
}
Server::~Server()
{
    serialClose(); // Close serial connection.
    printf("Server closed\n");
}

void Server::waitForClient()
{

    printf("Waiting for client.\n");

    serialReceive(msgBuffer); // Recieve filename from client.

    FILE* stream;

    if(stream = fopen(msgBuffer, "r"))      // If opening a file with the name recieved is succesfull continue
    {
        long lengthOfFile = getFileLength(stream);      // Get file length of opened file in bytes.

        printf("User requested: %s with a size of %ld bytes\n", msgBuffer, lengthOfFile); // Print out filename and lenght of file.

        sprintf(msgBuffer, "%ld", lengthOfFile); // Convert lenght of file to string.

        serialSend(msgBuffer, strlen(msgBuffer)+1); // Send lenght of file


        sendFile(stream, lengthOfFile);     // Use function to send file, giving pointer and lenght of the file to send.

        fclose(stream);  // Close file.
    }
    else // If file couldnt be found, print out message and send to client 0 filelenght (its not found).
    {
        printf("Client tried to get: %s\n", msgBuffer);
        serialSend("0", 2);
    }

    printf("Client connection closed\n");
}

// -Helper functions
long Server::getFileLength(FILE* file)
{
    fseek(file, 0L, SEEK_END); // Search for end of file.
    long length = ftell(file); // Returns the current value of the file-position indicator for the stream measured in bytes from the beginning of the file.
    rewind(file); // Back to start of file

    return(length); // Return the lenght.
}
void Server::sendFile(FILE* file, long fileSize)
{
    puts("Sending file");
    size_t readFromFile = 0;

    // Start reading
    while(1)
    {
        readFromFile  = fread(msgBuffer, 1, BUFFERSIZE, file);

        // serialSend info
        serialSend(msgBuffer, readFromFile);

        if(feof(file)) // If everything has been read, break out of while loop.
            break;
    }
}

```

```cpp
#include "Server.h" // The class
// File: ServerMain.cpp
/*
    Description: Main for using the Server class.
    Author: NSRD
    Date: 17/05 -11
*/
using namespace std;

int main(int argc, char *argv[])
{
    Server myServer("/dev/ttyUSB0", V24_B9600); // VMWare port and baud rate of 9600
    while(1)
    {
        myServer.waitForClient();
    }

    return(0);
}


```