

```
#####
PC-Link communication protocol V2.1
Tomas Pribyl

1.10.2001

#####
```

## UART parameters (Serial PC-Link)

115200bps, 8 data bits, 1 stopbit, no parity bit; (8N1)  
Handshaking: CTS/RTS

## Packet format

*Header* consists of these 3 bytes.

----- *HEADER* ----- mandatory  
\$00 - Packet\_Type  
\$01 - Packet\_Length  
\$02 - Header\_CRC

*Data Block* in packet consists of binary data followed by *Data\_CRC* byte. If packet does not contain any data, *Packet\_Length* in *Header* must be set to 0. *Data Block* can be 255 bytes long followed by 1 byte *CRC8* calculated from all bytes in *Data Block*.

----- *DATA BLOCK* ----- additional data only if PacketLength>0  
\$00  
| - DATA  
\$xx  
- DATA\_CRC

## Packet\_Type

\$00 - packet *RAW\_DATA*, binary data transferred  
\$01 - cmd packet *SENDFILE* (open file for reading and send it to master)  
\$02 - cmd packet *GETFILE* (open file for writing and receive if from master)  
\$03 - cmd packet *COMMAND* (make command)

\$80 - status packet *Data\_OK* (continue next packet)  
\$81 - status packet *EOF* (end of file)  
\$82 - status packet *COMMUNICATION ERROR* (repeat last packet)  
\$83 - status packet *FILE OPEN ERROR*  
\$84 - status packet *DELETE ERROR*  
\$85 - status packet *RENAME ERROR*  
\$86 - status packet *READ ERROR*  
\$87 - status packet *WRITE ERROR*  
\$88 - status packet *TIMEOUT ERROR* (no response, terminate communication)  
\$89 - status packet *EOT* (end of transmission)  
\$ff - status packet *UNKNOWN ERROR*

\$10 - *ACTIVATE* code (wake up server program and test presence)

\$00 - Packet *RAW\_DATA*

Used for binary data transfer. This packet consists of *Header* and *Data Block*. Number of bytes transferred in *Data Block* must be greater or equal 1 and less or equal 255 bytes. Data in *Data Block* are checked by *CRC8*.

\$01 - cmd Packet *SENDFILE*

Used for opening file for reading. *Data block* in packet must contain filename of opened file in ASCII. No terminating character required. Data in *Data Block* are checked by *CRC8*.

\$02 - cmd Packet *GETFILE*

Used for opening file for writing. *Data block* in packet must contain filename in ASCII of file to be written. No terminating character required. Data in *Data Block* are checked by CRC8.

\$03 - cmd Packet *COMMAND*

Used for sending commands to PC. *Data block* in packet must contain command in ASCII. No terminating character required. Data in *Data Block* are checked by CRC8.

Possible commands:

s:filename - delete file of directory  
c:disk: - change disc  
m:filename - make directory  
r:newname=oldname - rename file or directory

\$80 - status Packet *Data\_OK*

Send as acknowledge if received Packet was OK. *Data block* must not contain any data.

\$81 - status Packet *EOF* (end of file)

This packet indicates that whole file was transferred. *Data block* must not contain any data. After this packet is communication terminated and server program waits for commands.

\$82 - status Packet *Communication Error*

Packet is send as request for repeating last packet if data in *Data block* or Header were corrupted.

\$83 - status Packet *File Open Error*

Packet is send as result after *SENDFILE*, *GETFILE* or *COMMAND* packet if file was not open successfully. After this packet is communication terminated and server program waits for commands.

\$84 - status Packet *Delete Error*

Packet is send as result after *COMMAND* packet with scratch string 'S:filename' and this file or directory was not deleted. After this packet is communication terminated and server program waits for commands.

\$85 - status Packet *Rename Error*

Packet is send as result after *COMMAND* packet with rename string 'R:newname=oldname' and this file or directory was not renamed. After this packet is communication terminated and server program waits for commands.

\$86 - status Packet *Read Error*

Packet is send as result after *SENDFILE* packet if read error in server occurs. After this packet is communication terminated and server program waits for commands.

\$87 - status Packet *Write Error*

Packet is send as result after *GETFILE* packet if write error in server occurs. After this packet is communication terminated and server program waits for commands.

\$88 - status Packet *TimeOut Error*

Packet is send to server if C64 detects any TimeOut in communication. This packet terminates communication and server program will wait for next commands.

\$89 - status Packet *EOT* (end of transmission)

---

Packet is send to server if C64 wants to close the communication channel. This packet terminates communication and server program will wait for next commands.

\$ff - status Packet *Unknown Error*

---

Packet is send to C64 after any unknown error in server program. This packet terminates communication and server program will wait for next commands.

\$10 - *Activation code*

---

If server is OK and ready for commands every received bytes are inverted (EOR\$FF) and send back to C64 to indicate presence. Only *activation code* will remove server from this mode.

*Packet\_Length*

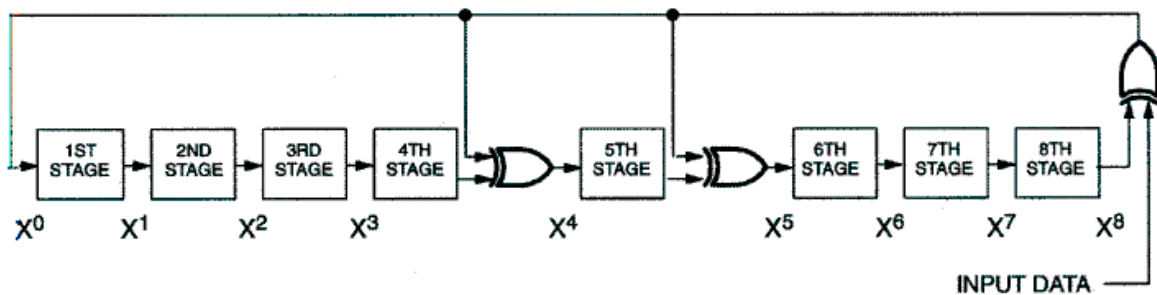
---

Number of data bytes contained in *Data Block* of packet. Can be 0 (no data in *Data Block*) or 1-255.

*Header\_CRC*

---

Polynomial expression:  $CRC8 = X^8 + X^5 + X^4 + 1$



To check *Header* consistence make:

$Packet\_Type (CRC8) \text{ } Packet\_Length (CRC8) \text{ } Header\_CRC = \$00$

*DATA\_CRC*

---

Polynomial expression is the same like *HEADER\_CRC*.

To check *Data Block* consistence make:

$Data0 (CRC8) \dots Data(n) (CRC8) \text{ } Data\_CRC = \$00$

## Communication Scheme

---

---

### SEND FILE/DIRECTORY FROM SLAVE TO MASTER

---

---

STATE	Master C64		Slave PC
1	SEND <i>Activate</i> code	->	GET <i>Activate</i> code *)
2	GET <i>Echo</i>	<-	SEND <i>Echo</i> of received code
3	SEND Packet (Cmd <i>SENDFILE</i> +name)	->	GET Packet (Cmd+Filename)
4	GET Packet (Status)	<-	SEND Packet (Status OK/ERROR)
5	SEND Packet (Cmd <i>NEXT/REPEAT</i> )	->	GET Packet (Cmd <i>NEXT/REPEAT</i> ) *)
6	GET Packet (Data), Goto State5	<-	SEND Packet (Data), Goto State5

### SEND FILE FROM MASTER TO SLAVE

---

---

STATE	Master C64		Slave PC
1	SEND <i>Activate</i> command	->	GET <i>Activate</i> command *)
2	GET <i>Echo</i>	<-	SEND <i>Echo</i> of received command
3	SEND Packet (GETFILE+name)	->	GET Packet (Cmd)
4	GET Packet (Status)	<-	SEND Packet (Status OK/ERROR)
5	SEND Packet (Data)	->	GET Packet (Data) *)
6	GET Packet (Status), Goto State5	<-	SEND Packet (Status

NEXT/REPEAT), Goto State5

### DELETE F/D, RENAME F/D, MAKEDIR, CHANGE DIR, CHANGE DISC

---

---

STATE	Master C64		Slave PC
1	SEND <i>Activate</i> command	->	GET <i>Activate</i> command *)
2	GET <i>Echo</i>	<-	SEND <i>Echo</i> of received command
3	SEND Packet (Cmd+Filename)	->	GET Packet (Cmd+Filename) + make command
4	GET Packet (Status OK/ERROR)	<-	SEND Packet (Status OK/ERROR)

\*) Timeout isn't tested. State is reset by *ACTIVATE* command.

Standard timeout in C64 and PC is 1 second.