From Norbert Unterberg’s Parallel-to-Sharp “SharpLink” manual

Manual: [Sharp to PC Link Manual (tripod.com)](https://edgar-pue.tripod.com/sharp/files/bigpc/sharplink.html)

Code and program: https://edgar-pue.tripod.com/sharp/files/bigpc/sharp.zip

**5.2 The Transmission Format**

The format with which the Sharp writes its data on magnetic tape is (unfortunately) not entirely trivial. Since this section is expected to be a little longer (and also interesting), I'll break it down a little more finely. Let's go:

**5.2.1 In the beginning was the bit - or "Of zeros and ones"**

Like every other digital calculator, the Sharp only knows zeros and ones (usually simply called bits). Heard everything before. This type of representation is definitely appropriate for storage, but it is extremely unsuitable for magnetic tapes (i.e. music cassettes ).

That's why the Sharp makes sounds out of the bits, because cassette recorders can handle that quite well. Each one becomes a tone of 4000 Hz, and each zero becomes a tone of 2000 Hz. He also stipulates that each bit lasts exactly 2 ms. After a little calculation you get the following Result (tables always look more professional):

              Bit¦ the Sharp sends  
              ---+----------------------  
               0 ¦ 4 cycles at 2000 Hz  
               1 ¦ 8 cycles at 4000 Hz

**5.2.2 From Bit to Byte - or "Nibble, change"**

Since the Sharp, as is well known, combines eight of these bits into one byte, but the magnetic tape can only record one frequency at a time (if you prefer simple software and secure transmission), it shovels the bits serially in single file over the line. However, he uses a different format than is generally used. It splits the byte in the middle and sends each nibble individually, each with a start and stop bit(s).

However, he does not seem to agree which order is best, as it changes from time to time. But more on that later. First of all, I have broken down the two possible transmission formats. I called them "H-Type" and "D-Type" because the former is mostly found in the header, the latter almost exclusively in data blocks.

Suppose the byte to be transmitted is `hgfedcba', where a is bit 0 and h is bit 7.Then the Sharp produces the following bit mash:

**H-Type: Untwisted**

0abcd10efgh11  
 with 0--------------- 1 start bit  
 -abcd----------- 4 data bits (lower nibble)  
 -----1---------- 1 stop bit  
 ------0--------- another start bit  
 -------efgh----- 4 data bits (upper nibble)  
 -----------11 2 to 5 stop bits

**D-Type: low and high nibble swapped**

0efgh10abcd11...  
 with 0--------------- 1 start bit  
 -efgh----------- 4 data bits (upper nibble)  
 -----1---------- 1 stop bit  
 ------0--------- another start bit  
 -------abcd----- 4 data bits (lower nibble)  
 -----------11 2 to 5 stop bits

Where which type with how many stop bits appears, I will indicate in the next chapters.

**5.2.3 The Header - or "Head Through the Tape"**

Before the actual transfer, the Sharp sends a small block of data through the line in which it is encrypted, what type of file it is, what the file is called, etc. This so-called header has the following structure:  
  
              Byte¦ Type ¦ Function  
              ----+--------+-------------------------  
               00 ¦ D-type ¦ file type  
               01 ¦ Name ¦ file name  
               0A ¦ Name ¦ Password, if available

A `Name' looks like this:

              Byte¦ Type ¦ Function  
              ----+-------+-------------------------------- ----------  
               00 ¦ H-type ¦ file name, 7th letter  
               01 ¦ H-type ¦ file name, 6th letter  
               ¦ ... ¦   
               05 ¦ H-type ¦ file name, 2nd letter  
               06 ¦ H-type ¦ file name, 1st letter  
               07 ¦ D-type ¦ identifier for the end of the name (always 5F)  
               08 ¦ D-type ¦ checksum over bytes 00 to 07

The file name is always stretched or shortened to seven characters. If the name is shorter, it is padded with zeros (the ASCII character NUL, not "0"). This also applies if no name is specified, it then consists of seven zeros. The password field only exists if it is noted in the file type, otherwise it is omitted. The following file types are known to me so far

             Type ¦ Meaning  
             -----+-------------+---------------------------  
               70 ¦ Basic program  
               71 ¦ Basic program with password  
               74 ¦ Data file  
               76 ¦ binary file

Incidentally, in the header, the Sharp sends 5, in words: "five", stop bits after each byte.

**5.2.4 Blocks and Checksums**

The Sharp forms an 8-bit checksum over each data block as follows: The checksum is set to 0 at the beginning of the block. Each byte that is to be included in the checksum is broken down into a low nibble and a high nibble. Then first the high nibble, then the low nibble is added to the checksum. If the checksum overflows during the first addition (that of the high nibble), it is increased by one. A possible overflow when adding the low nibble is ignored. In assembler, this can be put into commands like this:

checksum DB 0 ; this is where the checksum ends up  
          ; ...  
          Check PROC  
          ; The byte to be `checked' is in AL  
                      mov ah, al ; save a copy  
                      mov cl, 4  
                      shr ah, cl ; AH contains the high nibble  
                      and al, 0Fh ; and AL the low nibble  
                      add checksum, ah ; add high, overflow in carry  
                      adc checksum, al ; and low nibble with carry  
                      ret  
          Check ENDP

It should be noted that the checksum is formed using the "real" data. This plays a role in `D-type' bytes (see above) where the Sharp twists the nibbles. The checksum is only calculated after the byte has been "de-noded" (or before it is rotated if you are sending the data yourself). With the checksums in the header, you have to make sure that the first byte (the file type) is not checked as well. Only the checksum is formed using the name and the password (if available), whereby it is of course reset to zero before the latter. The password field in the header is considered an independent "data" block.

The structure of the actual data after the header has its pitfalls. I only know the structure of a Basic program to some extent, I can't give any more detailed information about the other data types. So to the basic program. Incidentally, all bytes in a basic program that come after the header are of the "D type", including the checksums. In addition, the Sharp reduces the number of stop bits in the data to two. In general, a checksum is inserted after every 120 (decimal!) data bytes of the Basic program. A basic block is 120 bytes long. After each block, the checksum is reset to zero. At the end of the program there are 2 FF bytes, the first of which actually belongs to the program. After the two FF-FFs there is a final checksum containing all bytes of this last block, up to and including the first FF byte - the last FF is not included. Since I had some difficulties with these rules when checking the checksums at the end of a program, I want to formulate the process differently: First, the entire program must be loaded. Then the data blocks are checked "normally" in a loop (checksum every 120 bytes) until exactly 3 bytes are left. Take the first one (it has to be an FF) and include it in the checksum. Then take the second one (it must also be FF) and ignore it. Then take the third and last one and compare it to the calculated checksum. They must match, otherwise there is something wrong with your program. You (unfortunately) have to make this effort, otherwise you will get problems if the last data block is about 120 bytes long (119 or 120 bytes).