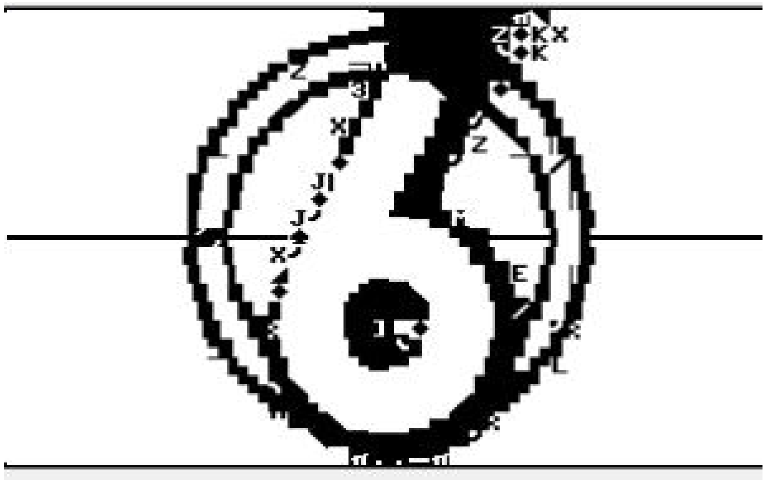
CBM PETSCII video converter

low resolution video using PETSCII based images



By Jan Derogee

Preface

The VIC-20, C64 and all those other “fancy modern machines” have the ability to display high resolution graphics. Either by modifying the character set or by a real high res mode. The older Commodore computers, like the PET series don't have this feature and can only use the build in character set to display modest graphics.

However is is possible to convert high resolution images into images entirely made up from characters. This is called ASCII-art or to be more precise PETSCII art considering Commodore uses PETSCII and not ASCII. The quality is not always nice to look at but you'll be amazed in how interesting conversion could look like. Even more interesting when animated. However, the results look at their best when viewed from a distance of approx. 3 meters when using a 15”screen. Because from a distance the characters blend together and the errors of the image are less noticeable.

One huge benefit from PETSCII over highres is simply that it requires less data. And when combined with compression acceptable speeds can be achieved over a limited bandwidth channel. Depending on the compression rate (directly coupled to the absolute differences between each video frame) frame rates of 10fps could be achieved. Therefore this concept is very suited for use with the Cassiopei, as this device uses the cassetteport in combination with a special data transfer protocol to achieve 2000bytes/sec. Meaning that uncompressed 25x40 PETSCII video could be transferred with 2 fps. However, that would more look like a fast slide show then a video, but it will demonstrate the use of compression. This document describes the conversion tool, the dataformat and the concept of delta compression.

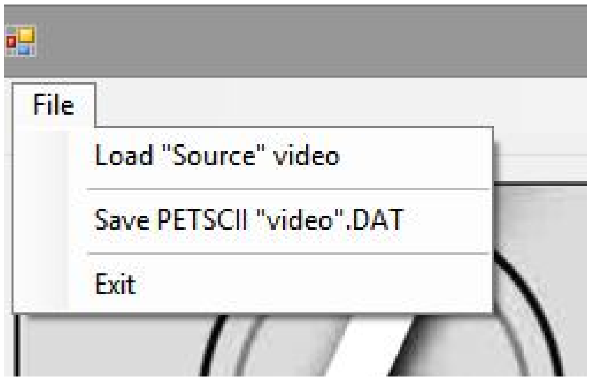
There is only one slight problem… PETSCII video files are completely silent, the do not contain any sound.

Contents

The conversion tool

A simple but effective conversion tool has been created to automate the process of converting videos into PETSCII animation files. Below is a screenshot of this tool.

Load source video

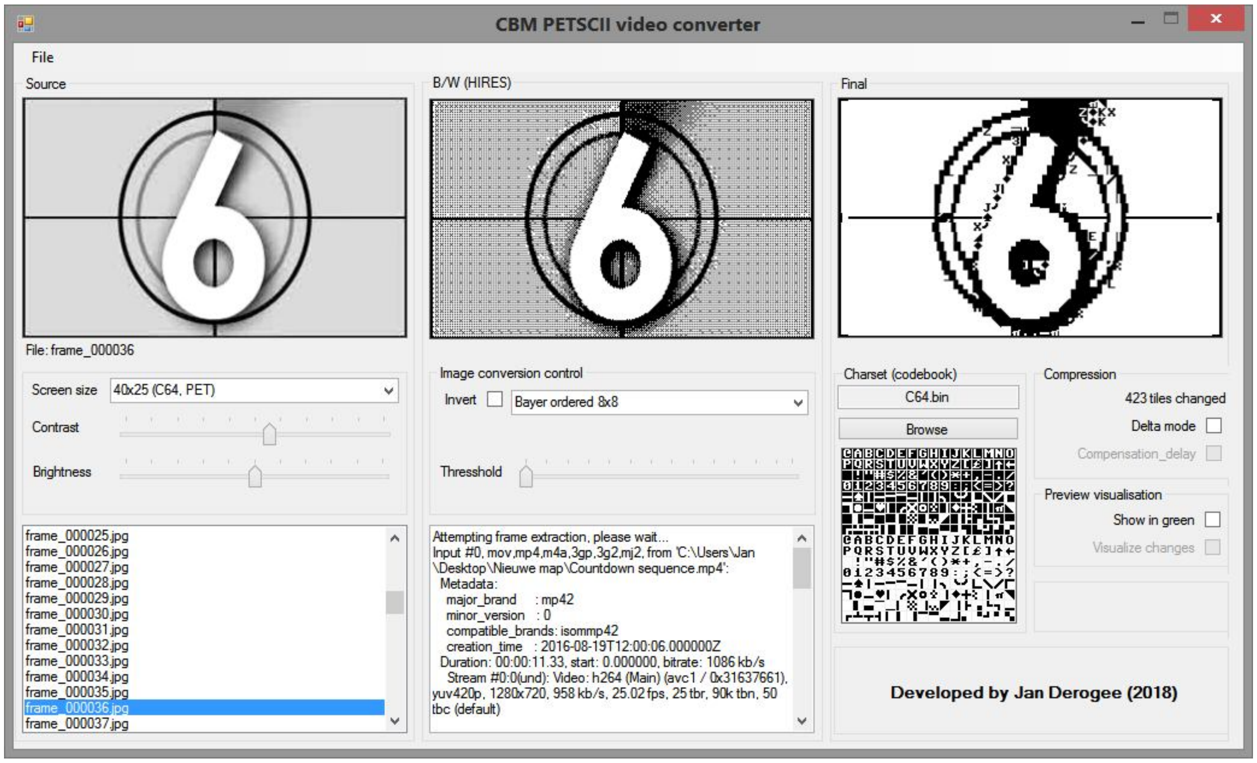


In order to get started the user needs to load a video file. On top of the screen there is the File menu, there are 3 menu items. “Load source video” is the item required to select the video file that needs to be converted. After clicking on this menu item a file browser will appear. Directly after selection of a file the user is prompted to acknowledge the removal of the previously generated files. Because the program will convert the video file into separate files. For a video of 100 frames, 100 (temporary) images files will be created. Therefore to prevent the computer's harddisk from being filled with too many temporary image files the user is prompted to acknowledge this removal.

The conversion of the video file is done using ffmpeg, this is a very common and powerful video conversion tool (*https://www.ffmpeg.org*). This tool extracts the individual frames from the video and does the size rescaling. All files generated are stored in the “my documents” folder on the computer. These files are to be considered temporary as they are not the final product of the program.

Overview and functions

The program basically consists of 3 parts: Source, B/W (intermediate) and Final.



File: frame_000036

Screen size 40x25 (C64, PET)
Contrast
Brightness

Image conversion control
Invert ☐ Bayer ordered 8x8
Threshold

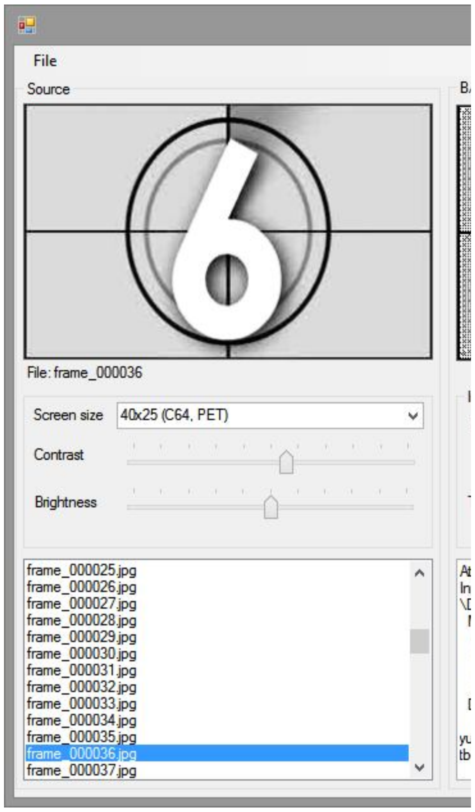
Charset (codebook) C64.bin

Compression
423 tiles changed
Delta mode ☐
Compensation_delay ☐

Preview visualisation
Show in green ☐
Visualize changes ☐

```
frame_000025.jpg
frame_000026.jpg
frame_000027.jpg
frame_000028.jpg
frame_000029.jpg
frame_000030.jpg
frame_000031.jpg
frame_000032.jpg
frame_000033.jpg
frame_000034.jpg
frame_000035.jpg
frame_000036.jpg
frame_000037.jpg
```

```
Attempting frame extraction, please wait...
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'C:\Users\Jan
\Desktop\Nieuwe map\Countdown sequence.mp4':
  Metadata:
    major_brand     : mp42
    minor_version   : 0
    compatible_brands: isommp42
    creation_time   : 2016-08-19T12:00:06.000000Z
  Duration: 00:00:11.33, start: 0.000000, bitrate: 1086 kb/s
    Stream #0:0(und): Video: h264 (Main) (avc1 / 0x31637661),
yuv420p, 1280x720, 958 kb/s, 25.02 fps, 25 tbr, 90k tbn, 50
tbc (default)
```



Screen size: 40x25 (C64, PET)

```
frame_000025.jpg
frame_000026.jpg
frame_000027.jpg
frame_000028.jpg
frame_000029.jpg
frame_000030.jpg
frame_000031.jpg
frame_000032.jpg
frame_000033.jpg
frame_000034.jpg
frame_000035.jpg
frame_000036.jpg
frame_000037.jpg
```

The left part contains everything related to the source image. The image shown here is the current frame of the video file currently processed.

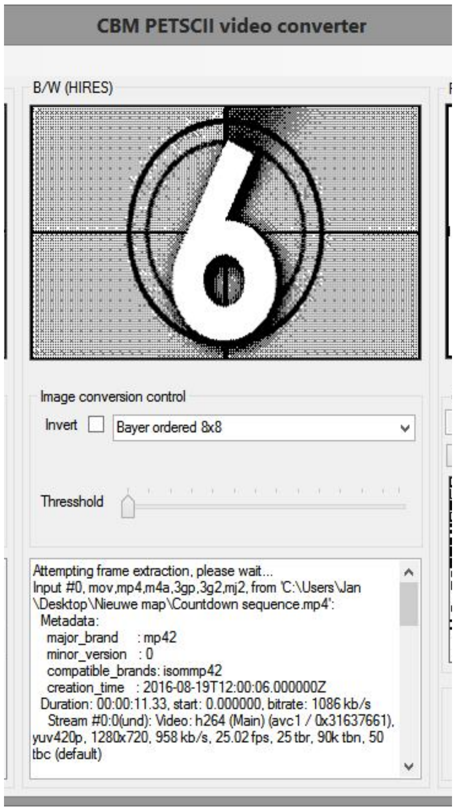
Screen size: select width x height (computer model).

Contrast: set the contrast of the source picture.

Brightness: set the brightness of the source picture.

There is a listbox that holds the individual frames of the videofile. Use the scrollbar and click on the desired frame to be processed. When this listbox has the “focus” of the user, the user can also scroll through this list using the arrow keys on the keyboard. This is very practical as is allows to manually “play” the animation. This way the user can get a feeling of what the end result may look like.

Unfortunately, computation of the frames does require a lot of processing power meaning that this will not runs as fast/smooth on slower computers. Meaning that scrolling through the files with the cursors keys on slower machines is not recommended.



Bayer ordered 8x8

```
Attempting frame extraction, please wait...
Input #0, mov,mp4,m4a,3gp,3g2,mj2, from 'C:\Users\Jan
\Desktop\Nieuwe map\Countdown sequence.mp4':
  Metadata:
    major_brand     : mp42
    minor_version   : 0
    compatible_brands: isommp42
    creation_time   : 2016-08-19T12:00:06.000000Z
  Duration: 00:00:11.33, start: 0.000000, bitrate: 1086 kb/s
    Stream #0:0(und): Video: h264 (Main) (avc1 / 0x31637661),
yuv420p, 1280x720, 958 kb/s, 25.02 fps, 25 tbr, 90k tbn, 50
tbc (default)
```

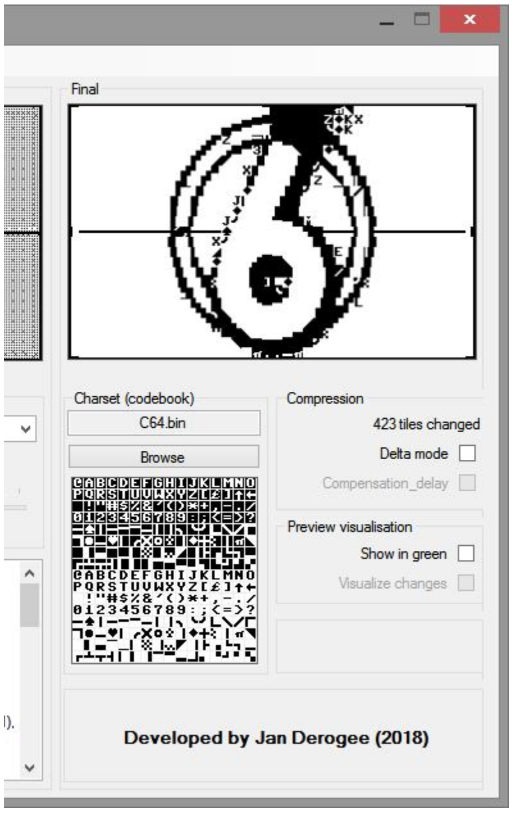
The middle section of the program, shows what the image looks like after is has been converted to B/W.

The drop down box allows the user to select dithering to make the result look smoother, although Floyd and Steinberg produces the best results for the B/W preview screen, it doesn't mean it will give the best result in the final result. This is mostly because Floyd and Steinberg produces a rather “random” pattern that is hard to match with the characters in the character set. When using Bayer, the pattern is very consistent and the changes are that it will result in similar characters being used, creating a much “cleaner” end result. In all cases, it is best to play with these settings.

When a form of dithering is selected, the threshold can/must be adjusted to tune the result.

The text field shows the output from ffmpeg, so if a file fails to convert, the reason for that should be found here.

The right section of the program shows the final result, this is how it will look like on your CBM computer. However this is very much depending on the available character set, therefore a different character set can be selected.



Default the C64's character set is shown as this is the most common one, but it is best to use the character set of the correct computer model. This because not all PETSCII character set were equal. For example the character set for a PET is a little “thinner”. Something that caused problems on VIC-20's and C64's that were connected on a TV-set. The used character set is shown in a small preview window.

There are several check-boxes:
Delta mode :check box for delta compression.

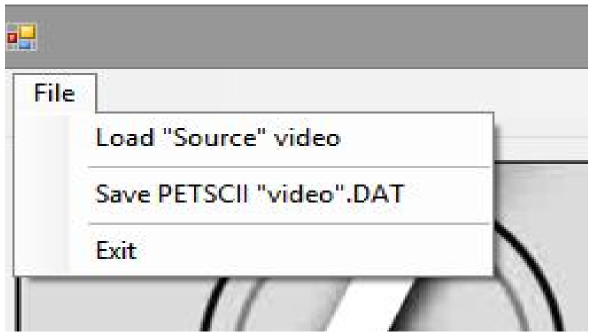
Compression delay: check to adds a compensation delay.
A small delay may be required to prevent the animation from being shown to quickly if compression is high. The delay value is automatically calculated but never perfect.

Show green: check to turn the preview green.
As this is the native color for PET computers.

Visualize changes: check to show the difference between the previous and the current frame. A practical and interesting way to get an idea of the effect of OR need for compression.

Save to PETSCII video

When done and satisfied with the settings you want to generate the animation file.



Go to the File menu and click on “Save PETSCII “video”.DAT”. A requester will appear and asks for the name/destination of the output file.

When the file is generated you require the PETSCII video player on your CBM computer to playnback the file.

Conversion from pixels to characters

In order to convert an image of a certain size from pixels to a collection of characters some calculations are required. First of all the image needs to be resized to the same size as it will be in characters. Knowing that a character is a block of 8x8 pixels we know that a screen of 25x40 characters times 8x8 (25*8 x 40* 8) is 320x200 pixels.

The computer makes a map of groups of 8x8 pixels. These groups of pixels are called tiles, after converting the image into tiles it is a matter of comparing the tiles with the available characters from the character set. You could say that the character that fits the tile the best is the character that will be used on that location. But in practice it is not a matter of which characters fits best, it more like which characters has the least number of differences (errors) for a certain tile.

This is actually a form of vector quantization. Although the codebook that will be used is in fact a fixed codebook, as we cannot change the character set.

In order to get the best results, the user can always play with contrast and brightness of the images before converting them to PETSCII. Also converting an image to gray scale can be done in various ways. For example simple B/W (very hard and not to be preferred), Bayer to Floyd and Steinberg dithering can be used to prepare the image for final conversion into PETSCII.

The file that is being generated is in a file format named .DAT, this is just a general name as .DAT is used by the Cassiopei for any kind of file. The main reason to call it .DAT is simply to distinguish it from .PRG and .TAP.

Compression

Converting pixels into characters is in fact a form of compression, although a very lossy one. Because we change from 8 bytes (per 8x8 pixel block) to a single byte. That's a compression factor of 8. A great achievement but a huge cost, because lot's of information is lost, though sometimes this isn't a problem at all. But compression can go even further, by sending only the data of the frame that is being changed compared to the previous frame. There is no need to re-send that huge piece of white background again, or if the face didn't move, we re-send it. So by sending only the pieces of the frame that changed huge amounts of data can be saved. Therefore only sending the differences or changes (we call these changes “delta”) between the current frame and the next frame we are doing delta-compression.

There is a small side note, because we need to identify where the changes in the next frame need to be made. So we need to send some form of a coordinate and this is extra data that need to be send. In other words “overhead”. Therefore there are a few situations where the uncompressed image is smaller then the compressed image but these situations should/could be detected by the conversion software and therefore prevented.

Dataformat

Below the definition of the .DAT file format for PETSCII video files

Header	
Function	**Description**
Magic word (12 bytes)	The hexadecimal values: 50, 45, 54, 53, 43, 49, 49, 56, 49, 44, 45, 4F Represent the following text: PETSCIIVIDEO
Version (1 byte)	The version of the definition of this file format The file format as described in this document is version: 1 This version is PETSCII only and therefore does not support hires graphics.
Number of images (2 bytes)	High-byte of the number of images stored in this file
	Low-byte of the number of images stored in this file
Image size X (tiles) (1 byte)	Width of the images. Could be any value, but most likely are: 22, 40, 80 Note: to keep the player optimized it only supports native video sizes of the computer it is being played on
Image size Y (tiles) (1 byte)	Height of the images. Could be any value, but most likely are: 23, 25 Note: to keep the player optimized it only supports native video sizes of the computer it is being played on
Reserved (1byte)	<undefined>
Reserved (1byte)	<undefined>
Reserved (1byte)	<undefined>
Reserved (1byte)	<undefined>
Reserved (1byte)	<undefined>
Data	
Mode (1 byte)	Bit 7: 0 = normal mode, 1 = delta mode Bit 6: 0 = no codebook data, 1 codebook data (this field is for High-res VQ images on systems with a definable charset) Bit 5: - undefined - Bit 4: - undefined - Bit 3: - undefined - Bit 2: - undefined - Bit 1: - undefined - Bit 0: - undefined -
image data (multiple bytes)	**Normal mode (bit 7 of mode byte is 0) :** Image data, data size is fixed to X*Y bytes
	Delta mode (bit 7 of mode byte is 1) : position of changed tile (this could be 0 if it was the tile in the top-left of the frame) data of changed tile position of changed tile (this is never 0, if it is then it is to indicate the end of the frame) data of changed tile position of changed tile (this is never 0, if it is then it is to indicate the end of the frame) data of changed tile position of changed tile (this is never 0, if it is then it is to indicate the end of the frame) data of changed tile etc. (until all data has been send, indicated by the positon=0) delay value (allows additional delay to slow down the playback for higly compressed frames) Be aware that the position value is an incremental value. Meaning that the first position is 0 + the position value, the second position is the previous position + position value and so on. If a position is required that is outside the range of a position value then additional tile information that are within the addressable range must be send even though a tile did not change at that location. For instance if the last tile of a 40x25 tile image has to be changed,then we need to send 3 additional tiles at the locations, 255, 512, 765. Keeping the position value limited to a single byte (instead of an X and Y byte) the data can be decoded even faster and less data has to be stored. More details further on in this document.

Normal mode explained

The normal mode is nothing more then a stream of all the bytes within the frame, for example a 25x40 screen uses 1000 bytes, all these bytes will be written to the file. This way the CBM that will process this file can directly copy the data to the screen locations 0-1023.

Delta mode explained

The delta mode only transfers the changes between the old frame and the new frame. Because not every tile is send but only the changed tiles, we need to indicate the location of the tile. We could send the X and Y coordinate of the tile, this requires 2 bytes, plus the data of the tile itself. Meaning 3 bytes. This could be reduced even more, by not sending the absolute position but a relative position. The relative position is the position of the previous changed tile related to the position of the next changed tile. So now we have enough of 1 byte to indicate a range of 256 positions. If we need to address a changed tile that is outside this range then we need to send a tile pretending to be changed.

This method is not efficient for tiny amounts of changes (that are far apart) in a frame, but it is highly unlikely that only 3 tiles change in a screen. The max overhead in a screen of 25x40 would be 25x40/255 is 3 pretending to be changed tiles, so that would be 6 bytes. So in most cases the method of sending a relative position reduces the amount of transmitted data.

```
Old frame                                  New frame (Delta=old frame - new frame= 3 tiles)
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXX-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXX-XXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX-XX
```

```
Information that is transferred
XXX-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   pos=0+3, value "-"                         (send: 3,-)
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX-XXXXXXXXXXXXXXXXXXXXXXXXXXX   pos=3+209, value "-"                       (send: 209,-)
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   pos=3+209+255, value "X"                   (send:255, X)
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   although nothing has changed on this location, we must
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   pretend something has changed in order to increment the
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   position counter (because our delta position is only
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   8-bits wide, allowing only a delta address of max. 255)
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX   pos=3+209+255+255, value "X"               (send:255, X)
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX-XX   pos=3+209+255+255+255, value "X"           (send:255, X)
                                           pos=3+209+255+255+255+255, value "-"       (send:255, X)
```