AN-05 (LP)

September 23, 2005

HOW TO PRINT RLE COMPRESSED GRAPHICS IN LINE PRINTER MODE

Overview

The O'Neil printers have several graphics modes that allow graphics to either be stored in the printer's Flash and printed repeatedly upon demand (useful for logos or icons), or sent to the printer and printed one time only (useful for signatures or transient images). Graphics sent to be printed one time only can be sent in their full bitmap form, or they can be compressed using a Run Length Encoding scheme (RLE). This document describes how to encode a graphic image using RLE.

Graphic images are formed of line after line of individual dots, like tiny slices across the image. For the O'Neil printers, each of those lines is .005" tall, and each dot is .005" wide. A single byte then can represent an area only .005 inches tall and .040 inches wide (8 dots). It takes approximately 5000 bytes to represent a single square inch of graphic data. But most images contain large white or black areas. For example, a signature is mainly white space. Since this white space data is the same byte repeated, over and over, large horizontal areas across one dotline can be compressed into only two bytes – one byte contains the graphics information and the second byte contains the number of times that byte is repeated (called Run Length Encoding or RLE). A similar technique can compress multiple horizontal white lines into only two bytes – one of which is a command to advance paper and the other is the number of lines to advance. Using these techniques, the total data is usually reduced to ½ or less of its original size (many signatures can come down to 1/3 to ¼ of their original size). RLE does not compress well when the graphic is very "busy". So the O'Neil implementation allows the application to determine on a DOTLINE BY DOTLINE basis whether the line is best sent using compression or not.

Compressing the image using RLE

All RLE images begin with the two-character escape sequence ESC B (0x1B, 0x42) and end with the two-character escape sequence ESC E (0x1B, 0x45). If a series of dotlines are all whitespace, the vertical white space can be compressed by sending a two-character sequence consisting of an 'A' (0x41) followed by a single byte containing the number of blank dotlines to advance. Each non white-space dotline must represent enough bytes for the total width of the printer (4" is 832 bits or 104 bytes, 3" is 576 bits or 72 bytes, and 2" are 384 bits or 48 bytes). The first bit printed is the MSbit of the first byte received. Each dotline is preceded by a 'U' (0x55) if that dotline is sent Uncompressed, or a 'G' (0x47) if that dotline is sent Compressed. If the dotline is compressed, then that dotline will consist of a series of byte pairs after the G. The first byte in the pair is the graphics byte; the second byte is the number of times to repeat that byte.

The following example will show how a simple, contrived image can be compressed. We will use a mythical printhead that is 20 bytes wide and a graphics image 10 dotlines tall (200 bytes total before compression. Each pair below represents one byte in Hex of our graphic:

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 0F 80 00 00 00 00 FF FF D2 00 00 00 00 00 00 00
00 00 00 0F FF FF C2 00 00 78 45 45 D2 D2 D2 F9 F9 00 00 00
00 0F F8 00 0E E0 00 00 FF FF 01 E0 FF D2 00 88 73 FC C7 00
00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

We begin the compression with the first two bytes of ESC B (telling the printer we are using compressed graphics):

1B 42

The first three dotlines in our example are all white space. So this can be compressed into two bytes, 'A' or 0x41 for Advance and 03 for 3 dotlines:

41 03

The next line contains graphics information, so we will need to use either compressed graphics or non-compressed graphics. We will try compressed first:
Four 00's will compress to the byte pair 00 04 (repeat 00 four times)
One 0F stands alone but still requires the byte pair 0F 01
Similarly, 80 requires 80 01, the four 00's compress to 00 04, the two FF's are represented by FF 02, the D2 requires D2 01, then we have 7 00's which compress to 00 07, making the entire string 00 04 0F 01 80 01 00 04 FF 02 D2 01 00 07. Note that if you add up all of the byte counts (4+1+1+4+2+1+7), it totals 20, the width of the printhead. We represent this as compressed graphics by preceding it with a 'G' (0x47):
47 00 04 0F 01 80 01 00 04 FF 02 D2 01 00 07

Similarly, the next line could be represented as compressed:
47 00 03 0F 01 FF 02 C2 01 00 02 78 01 45 02 D2 03 F9 02 00 03

or non-compressed (preceded with a 'U' (0x55) using the same number of bytes.
55 00 00 00 0F FF FF C2 00 00 78 45 45 D2 D2 D2 F9 F9 00 00 00

Since the next line is "busier" than the preceding line, using RLE will probably make it longer (and we can see that it does):
00 01 0F 01 F8 01 00 01 0E 01 E0 01 00 02 FF 02 01 01 E0 01 FF 01 D2 01… etc.,

so it is best represented as uncompressed:
55 00 0F F8 00 0E E0 00 00 FF FF 01 E0 FF D2 00 88 73 FC C7 00

The next two lines can each best be represented using a compressed line (note that 0x13 is 19 decimal):

47 00 01 FF 13
47 00 01 FF 13

And, the last two lines again can be compressed to only two bytes:

41 02

This is followed by the end code (printer returns to normal ASCII):

1B 45

And finally, we have the final compressed graphic ready to send to our mythical 20 byte wide printer (the example is arranged to show each line on a separate line, but this is really one continuous string of 75 bytes):

```
1B 42
41 03
47 00 04 0F 01 80 01 00 04 FF 02 D2 01 00 07
47 00 03 0F 01 FF 02 C2 01 00 02 78 01 45 02 D2 03 F9 02 00 03
55 00 0F F8 00 0E E0 00 00 FF FF 01 E0 FF D2 00 88 73 FC C7 00
47 00 01 FF 13
47 00 01 FF 13
41 02
1B 45
```

Using RLE we have compressed a 200 byte file into only 75 bytes making it only 37.5% of its original size!

THE ALGORITHM:

The input to this algorithm must be a bit mapped graphic of the same width (N BYTES) as the printhead used for printing.

1. Start the RLE compressed graphic with ESC B (0x1B, 0x42)
2. Scan the next dotline of N bytes, replacing each byte by a byte pair where the 1st byte is the byte in the original graphic and the second byte is the number of times that is repeated. Call the length of this sequence of bytes M
3. **IF** the line was all 00's [result of (2) is 00 N] and the previous line was not all zeros, then add 'A' (0x41) followed by a count ***C*** of 01
   ELSE IF the line was all 00's [result of 2 is 00 N] and the previous line was all zeros, then increment the count ***C*** by 1
   ELSE IF M <= (N) then add 'G' (0x47) followed by the NEW sequence of pairs of bytes to the compressed graphic string (M bytes)
   ELSE add 'U' (0x55) followed by the ORIGINAL bytes (N bytes)
4. Repeat 2 and 3 until all dotlines have been processed
5. End the RLE compressed graphic with ESC E (0x1B, 0x45)