Network Working Group
Request for Comments: 387
NIC: 11359
Categories: D.6, F
Obsoletes:
References: RFC #292

Karl C. Kelley
Jaacov Meir

8/10/72

SOME EXPERIENCES IN IMPLEMENTING NETWORK GRAPHICS PROTOCOL LEVEL 0

We are in the process of implementing NGP-0 at several hosts. For the time being, we are forced to consider the remote host as the "last intelligent machine". We are attempting to translate NGP-0 to a machine dependent code for the Computek display. The remote hosts are CCN, UCSD, and soon RANDCSG. More comments about that work will be made in subsequent RFC's. The concern of this RFC is twofold:

1. Clarify the coordinate number system.

2. Puzzle over how to do TEXTR string without either:

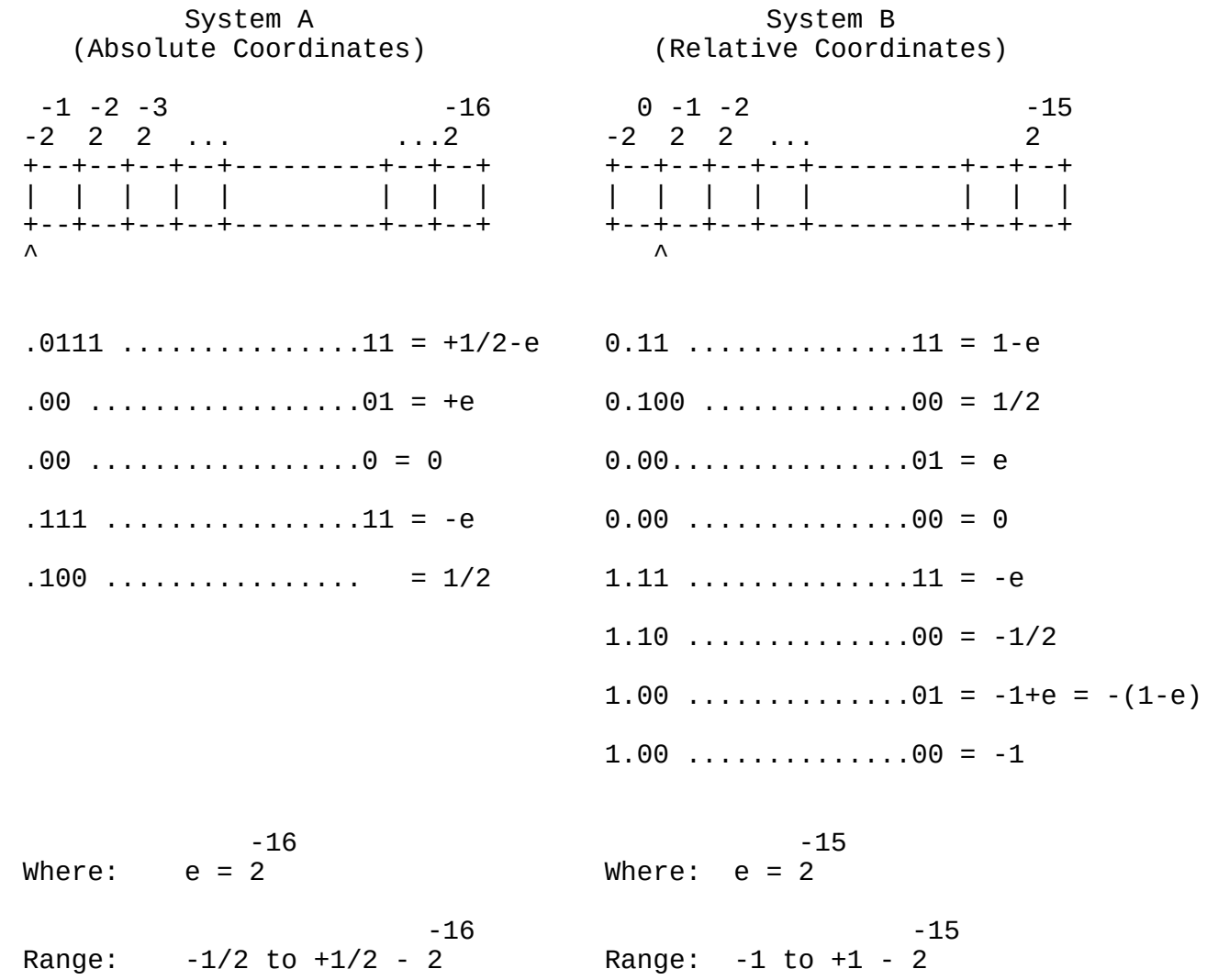
    a. Reading current position and saving it while the text string is being output, or

    b. Monitoring the beam position for each NGP command and saving this information somewhere.

An appendix to this RFC will outline the conversion from the NGP coordinate system to the floating point arithmetic on the PDP-1O.

The Coordinate Data

The document for NGP-0 (RFC 292) does not say specifically that the format of coordinate data is the same whether the command is in absolute or relative mode. The only thing stated is that they are in two's complement notation with the leftmost bit being the sign bit. It is possible to use two different 2's complement schemes:



I submit that one could interpret the requirement for absolute coordinate data to be in the range -1/2 to +1/2 - e as requiring that two different number systems should be used. Thinking along those lines, System A has the advantage that you never get handed a number out of range, which saves some checking worries. It also has one whole bit more of precision.

I further submit that having two systems to contend with merely clouds the issue and requires extra coding. It makes more sense just to stick with System B above. Among the advantages in its use are:

1. The single system can handle both absolute and relative coordinates.

2. If an absolute coordinate exceeds range, simply forcing the sign bit on causes a nice wrap-around.

3. The representation is the same as the mantissa for floating point numbers on most machines. Notice, however, that mantissas of normalized floating point numbers are not in the range for absolute coordinates. The program will have to shift the mantissa until exponent is 0.

It may be that few of us interpreted the NGP document to mean two number systems were needed. If that is the case, so much the better. In any case, until shaken from the position by the overwhelming force of contrary logic, we will, in all of our implementations, use System B above for both absolute and relative coordinates.

The TEXTR Command

The last paragraph on page 4 of RFC 292 says, "...a command be included only if its output is a function solely of the current command and the "beam position" current at the start of the command. In other words, the interpreter for level 0 need have no internal storage for 'modes' or pushdown stacks."

In the case of the Computek display, most of the NGP commands correspond to capabilities of the device. The lone exception is the TEXTR command. There are two ways to know what beam position to return to after the string is displayed. One way is to read the cursor position from the display just before doing the string output. This is no good because it requires reading from the device (which we can't do until input protocols are implemented). Also, on this device, the cursor position is accurate only to within 4 scope points.

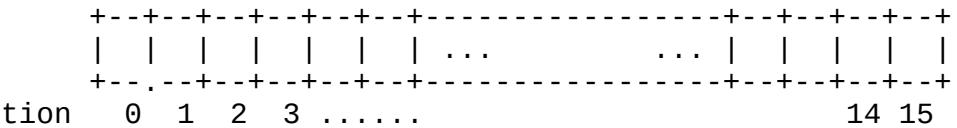
The second way to know what beam position to return to is to monitor all motions of the beam in software. Thus our implementations of NGP-0 to Computek translations will employ a software X register and Y register. On absolute commands, the registers will be set to the coordinates for that command. On relative commands, the coordinate data will be added to the registers. At the beginning and end of picture, these registers will be set to 0.

The TEXTR command will also cause these software beam registers to be changed. That is, the X register will be incremented for each character of the string to correspond to what is happening in the display itself.

APPENDIX

NGP-0 to PDP-10 Floating Point

The NGP-0 looks at all data numbers (X and Y parameters) as a fraction number in the following format (16 bits per number).



with the binary point assumed between bits 0 and 1. Bit 0 is the sign bit and all negative numbers are represented as their two's complement. The PDP-10 machine code representation of fractions in floating point (mantissa part) is very similar to the above (with one exception--the number -1), so the transformation could be obtained simply by two operations, move and substitute.

Data (X,Y) Conversion

NGP (extreme points) (16 bits)		Floating Point (PDP-10) (36 bits) exp mantissa	
1/2	0.1000 0	0 10000000	10 0
-1/2	1.1000 0	1 01111111	10 0
-1	1.00 0	1 01111101	10 . . . . . .0 Special case
1-e	0.11 1	0 10000000	1111 1

Translation from NGP into floating point for PDP-1O:

1. Move sign bit (leftmost one) to sign bit.

2. Move fraction part (15 bits) to mantissa part (left justified; fill with zero's to right).

3. Fill in exponent part (8 bits) according to:

   a. If positive number exp = 10000000 = (80) hex

   b. If negative number exp = 01111111 = (7F) hex

   c. Exception _in_only_ one number

      -1 in NGP (negative sign and fraction all zero's)

      (1) mantissa becomes same as -1/2

      (2) exponent becomes the one's complement of (82) hex = (7D) hex

The methods of conversion will remain the same regardless of the number of bits (up to 24) that are used for the NGP fraction.