Conversion of NGP-0 Coordinates to Device
----------------------------------------

Specific Coordinates
---------------------

Conversion of NGP-0 coordinates to floating point PDP-10 coordinates was discussed in RFC #387. In general, however, it is undesirable to convert NGP coordinates to floating point coordinates because real devices require integer addressing. To this end, a means is described to convert NGP coordinates to integer coordinates in the range zero to M, where M is the maximum address of the device screen on a machine using 2’s complement arithmetic. It would not, however, be difficult to modify this algorithm to operate on machines using one’s complement or sign-magnitude arithmetic.

First consider the NGP coordinate format:

```
+---+-----------+
 |   | n        |
 +---+-----------+
    s ^ FRACTION
   i
g   n
```

Where the sign occupies the most significant bit of the coordinate followed by bits of numerical information (initial implementation of NGP requires N=15). Negative numbers are represented by 2’s complement. Conversion to device coordinates is accomplished by:

\[
D = S \times f + S
\]

Where \(D\) => integer device coordinate
\(S\) => scaling factor (typically \(M/2\))
\(f\) => NGP fractional coordinate

Let us rewrite this as:

\[
D = S \times (2 \times f)/2 + S
\]
Now factor $S$ into two terms:

$$S = Q \times 2^n$$

Where $Q$ is an odd integer and $I$ is an integer.

When:

$$D = Q \times 2^n \times (2^n f)/2 + S$$

The factor $(2^n f)$ is represented in 2’s complement form simply by extending the sign bit of $f$ into the upper portion of the computer word. If $Q = 1$ (as it would be with many devices), it can be ignored. If $Q > 1$, we may console ourselves that an integer multiply is faster on most machines than a floating point multiply. In fact, on a PDP-10, this multiply can usually be performed with no access to memory since $Q$ is usually small.

We are now left with the $2^{I-n}$ factor. This can be accomplished with an arithmetic shift left by $(I-n)$ or an arithmetic shift right by $(n-I)$ as is appropriate. The offset factor, $S$, may now be added using an integer add.

The procedure for converting NGP coordinates to integer device coordinates is then:

1. move coordinate to a register and extend sign
2. integer multiply by $Q$ (if necessary)
3. arithmetic shift left by $(I-n)$
4. integer add $S$

This procedure would generally be much faster than:

1. move coordinate to register and extend sign
2. float fractional coordinate
3. floating point multiply
4. floating point add
5. conversion to fixed point

[ This RFC was put into machine readable form for entry ]
[ into the online RFC archives by BBN Corp. under the ]
[ direction of Alex McKenzie. 1/97 ]