Scope compatibility for symbolic files

TS symbolic files are written in 7-bit, 8 character/word symbolic code, bit justified and with multiple blank encoding. Scope programs expect to see a totally different format: 6-bit, 10 character/word card images. Therefore, the Scope interface must provide the necessary conversion. This turns out to cost about 9 μs/character when implemented by the following program:

X6 = word being assembled

B6 = shift count for assembly

X5 = word being disassembled

B5 = counter for disassembly

B7 = word address for store

B4 = word address for load

X4 = 177B

B3 = end of input buffer

B2 = end of output buffer

B1 = 1

X1 = temp

```
        *getnew input word
5   L5  SB5   8
2 5     SB4   B4+B1
5       SA5   B4
        * main loop for every character, unpack
5   L2  LX5   8
2 5     BX1   X5*X4
        * convert
12      SA1   X1+CONVTAB
5       NG    X1,L6                 negative means special (ignore or multiple blank)
2       * pack
6       LX1   B6 X1
5       BX6   X6+X1
        * to next output char
5       EQ    B6,0,L1
3
5       SB6   B6-6
        * to next input char
5   L3  SB5   B5-1
5       NE    B5,0,L2
        * need new input word
5       NG    X5,L4                 jump on end of line, flagged by 1 in bit 3
0
13      NE    B4,B3,L5
        * end of input buffer
```

```
* store output word

10 L1    SA6  B7
5        SB6  60-6
5        SX6  0
5        SB7  B7+A0
13       LE   B7, B2, L3
```

$$T_{min} = 8.1 + \frac{1}{10} \cdot 4.0 + \frac{1}{8} \cdot 3.4$$

$$= 8.9 \ \mu s \quad \text{or} \quad 1 \text{ second} / 13K \text{ full cards}$$

Probably a similar conversion will be required for output, unless the output is destined solely for printing and we want to provide a special routine for this purpose.

Of course, nearly all of this cost can be eliminated by imbedding the conversion into the program reading the input, especially if it is keyed enough to unpack each line into a letter/word buffer.