$OBJECT
$SOURCE

These verbs are peculiar to the BASIC-2C implementation and so are accepted grammatically but perform no operation at run time. This duplicates the behaviour of the BASIC-2C non-interpretative runtime.

BASIC-2C, like Wang's BASIC-2, is a line oriented interpreter with all the information about an individual program line contained in that line. This keeps the program structure simple but results in poor load and resolve performance and a runtime overhead.

On the other hand **KCML** is a compiled language with global structures such as symbol tables stored with the program lines in a program file giving dramatic improvements in load and resolve times for large programs. However as a consequence it is not possible to construct a program file by concatenating program lines nor is it possible to dissect out individual lines.

The main purpose of $OBJECT was to allow programs to write other programs which might then be overlaid on the current program. This was possible, though laborious and required detailed knowledge of the structure of program files. Under **KCML** the same job can be accomplished with PRINT statements and the $COMPILE verb without any knowledge of program structure e.g.

```
SELECT PRINT "/tmp/madeup"
REPEAT
     LINPUT "Enter line no and statement", line$
     PRINT line$
UNTIL line$ == " "
SELECT PRINT /005
```

This program can then be directly loaded:

```
LOAD "/tmp/madeup"
```

KCML will compile the program as it is loading it so it will execute at full speed. To put the program on a platter and to make the LOAD faster the program can be first compiled e.g.

```
SELECT #1"/tmp"
$COMPILE T#1,"madeup" TO T#0,
```

1. Immediately after the crash get the customer to type in:

SELECT PASSWORD ! /usr/lib/kcml/checknum

this command will immediately respond with a check string. Tim will then need to run the *checknum* utility on a SCO Unix machine using approximately the same KCML version. This utility will prompt for the string returned by the SELECT PASSWORD ! command. and the password used to scramble the program. After a couple of seconds it will return a number which should be entered by the cus
execute a LIST L

- This is quite complicated but should work.

If he cannot get this to work then perhaps he can recreate this himself. It is impossible for me to find out what the problem is without knowing at least what statement is generating the error.

2. Any further information on this one would be useful. I don't understand what you mean when you say that the variables were once in a global partition. Where has this system come from, has it always been in KCML?

By the way I shipped 10 sets of manuals, more will follow when we get them.

Steve