)

The state variable PROC indicates whether this delimiter is an expression bracket or a procedure call bracket.

If it is an expression bracket, we first check that we are in fact within an expression. This kills "a + (b)". The translation of the preceding expression is completed using TAKE and UNSTAK, after which the top of the stack should be the delimiter (. This is unstacked and discarded, having reset PROC to the stacked value.

If it is a procedure call bracket, ACTOP is called to complete the translation of the preceding parameter. PROCPO is saved in I after the top-of-stack test, and the stacked state variables restored. A test is now made to see whether the count of parameters in the call is equal to that in the declaration; if not, it may be that this procedure is itself a formal parameter. If so, the specifier in question has not told us how many parameters are required, and we must fill the count in from this call. However, we must first check that this count has not been filled in by some previous use, if so it must fail as in the last line of the following example.

```
real procedure JIM (a,b); integer a ; real
                                  procedure b;

    begin integer q,r,s;

        a : = b (q,r,s);
        a : = b (q,r);
```

The test on f [I] determines what sort of instruction to compile for this procedure call. Finally, if E is set to statement level a check is made that the next delimiter is an "end-of-statement" delimiter, otherwise in the expression case TYPBOX must be set.

ERRORS

FAIL 81;	misused) other than in expression.
FAIL 82;	unmatched closing round bracket.
FAIL 5;	illegal parameter list.
FAIL 51;	incorrect number of parameters.
FAIL 111;	incorrect number of parameters in use of formal procedure.
FAIL 84;	wrong delimiter after procedure statement.