15. BLOCK PREFIXING

The compiler generated coding for a prefixed block is:

1. A call on the begin prefixed block (BPB) subroutine.

2. In-line coding to evaluate the parameters and store them into the block instance.

3. A call on the end prefixed block parameters (EPBPAR) subroutine to indicate end of the parameter evaluation.

4. In-line coding for declarations within the block.

5. A call on the begin prefixed block return (BPBR) subroutine to indicate the end of the declarations within the block.

6. In-line coding for statements within the block.

7. A call on the end prefixed block (EPB) subroutine to indicate the end of the prefixed block.

A prefixed block is assumed to have the exit from the block indicated in the prototype.

The static link from this driver is to the block B statically enclosing the prefixed block P.

The reactivation point (pex,drex) for a prefixed block is initially none. It is set by a resume statement or a call on the store collapse.

The procedures in the formal description associated with a prefixed block are:
BPB begin prefixed block
BPBR begin prefixed block return
EPB end prefixed block
EPBPAR end prefixed block parameters

procedure BPB (x); ref (prototype) x;
    begin ref (driver) a,y,z;
        comment begin prefixed block;
        z := new driver (new object(x),CD,none,none,none,true,
                         x.level);
        z.rp := z.ob := z.pb := true;
        z.obj.MDP := z;
        a := CD;
        while not a.rp do a := a.drex;
        a.pex := none;
        a.drex := z;
        if x.nrp ≠ 0 then
            begin
                y := new driver(CD.obj,CD.drp,none,z,none,false,
                                 CD.level);
                y.con := CD.con;
                y.cdrp := CD.cdrp;
                CD := y;
                go to exit
            end else CD := z;
        DISPLAY[x.level] := CD.obj;
        DDISPLAY[x.level] := CD;
        go to x.prefix[0].declare
    end BPB;

procedure BPBR;
    go to CD.obj_PP.prefix[0]. statements;

procedure EPB;
    go to CD.obj_PP.prefix[CD.obj_PP.lev-1].inretur;
procedure EPBPAR;
    begin ref (driver) y;
        comment end prefixed block parameters;
        y := CD;
        CD := CD.drex;
        deletenotice (y);
        DISPLAY[CD.level] := CD.obj;
        DDISPLAY[CD.level] := CD;
        goto CD.obj.PP.prefix[0].declare
    end EPBPAR;