16. SEQUENCING STATEMENTS

16.1 Detach

```
ref (object) procedure detach;
    begin ref (driver) x,y; ref (program) out;
       x :- CD;
       if x.rp then
       begin
          while not x.pb do
          begin x :- x.drp;
                while not x.rp do x :- x.drex;
          end;
          CD.pex :- exit;
          CD.drex :- CD;
          y :- x;
          x :- x.drp;
          while not x.rp do x :- x.drex;
          x.drex :- y;
          x.pex :- none;
          while y.pex == none do y :- y.drex;
          out :- y.pex;
          CD :- y.drex;
       end else
       begin
          out :- CD.pex;
          y :- CD.drex;
          CD.pex :- exit;
          CD.drex :- CD;
          CD.rp := true;
          detach :- CD.obj;
          restore (CD.acs);
          CD.acs :- none;
          CD :- y;
       end;
       update display;
       go to out
end detach;
```

If x is a prefixed block, the statement detach is a dummy statement. If x is an attached object, x is given (pexit, CD) as reactivation point and we return to the exit of x. The function value is in this case x.

If x is a detached object, the associated prefixed block is located. The detach statement is equivalent to a resume of this prefixed block.

16.2 Resume

```
procedure resume (x);  ref (object) x;
   begin
      ref (driver) y,z;
      Boolean b;
      if x =/= none then
      begin
         z :- x.MDP;
         if z == none then error ("resume",1);
         if not z.rp then error ("resume",2);
         y :- CD;
         while not y.rp then y :- y.drex;
         y.drex :- CD;
         y.pex :- exit;
         CD :- z;
         while CD.pex == none do CD :- CD.drex;
         exit :- CD.pex;
         CD :- CD.drex;
         y :- z;
   L:    b := y.pb;
         y :- y.drp;
         while not y.rp do y :- y.drex;
         if not b then go to L;
         y.drex :- z;
         update display;
         go to exit
   end
   else error ("resume",3)
end resume
```

Possible errors:

1. x is terminated.

2. x is not detached.

3. x is none.

Starting on current driver and following dynamic links (drex), locate the first detached object or prefixed block.

There must be at least one, since the program, by definition, is a prefixed block. (Cfr. Common Base definition, section 8).

The reactivation point is inserted in the driver of this object.

The driver of x becomes the current driver.

Starting on the driver of x, follow downward links (drex) until an object with pex not equal to none is found. This brings us down to the current detached object. This may step through more than one quasi-parallel system. Current driver may be changed during this prosess.

Note:
Pex of the master driver for an operating detached object must be none if there exists an inner quasi-parallel system for this object. Drex must point to the master driver of the active object of this quasi-parallel system.

The reactivation point pex,drex of the driver referenced by CD is fetched to (exit,CD). This brings us down to the innermost block of the operating object.

Locate the nearest detached object in the outer quasi-parallel system and modify drex in the master driver of this object to indicate that x is now the operating object of the inner quasi-parallel system.

Display is updated.

The object referenced by CD is then entered.

16.3 Call

The procedure "call" is not a part of the Common Base, but is a natural part of a SIMULA 67 Common Base implementation. "call" is formally a procedure with one unqualified parameter.

Let the actual parameter of a call on attach be a reference to a detached object y, which is a component of a quasi-parallel system s. Since y cannot be referenced from outside s, there must be a component x of s which is operating. The call on "call" has the following effects:

1. The OSC of s and LSC of x enters y at the current position of its LSC.

2. y becomes attached to x.

Informally a call on "call" means that the reactivation point of y is replaced by the exit to the call on attach and the control enters y at the position of its reactivation point (which may be in some inner block or even in some inner quasi-parallel system).

```
procedure call (x);
   ref (object) x;
   begin ref (driver) a,y,z; ref (program) next;
      if x =/= none then
         begin z :- x.MDP;
            if z == none then error ("call",1);
            if not z.rp then error ("call",2);
            y :- z;
            while y.pex == none do y :- y.drex;
            next :- y.pex;
            z.pex :- exit;
            a :- y.drex;
            z.drex :- CD;
            z.rp := false;
            CD :- a;
            update display;
            go to next
         end else error ("call",3);
      end call;
```

Note: This definition of call is tentative, since the problem is currently being studied by a Technical Committee under the SIMULA Standards Group.