RC INFORMATION	class EXT.	repl.		ident EAH810401	
	X RC 4000	X RC 6000	X RC 8000	RC 3600	page 1/2

subj. Handling of ISQ-files in a Coroutine System.

1. The problem.

A dangerous pitfall exists when the ISQ-system is used in a coroutine system with implicit passivate on the isq-zone.

A runtime error in an activity is often trapped (by activate or by a trap label in the monitor block or an outer block) with the intention of closing the files before leaving the program. You will then probably call the procedure setreadi to remove a possible update mark from the ISQ-file.

If, however, the alarm occurred while another activity was implicit passivated during an ISQ-operation, this operation is never finished.

Suppose the unfinished operation was a call of getreci. This procedure begins by saving the keyfields of the wanted record, after which the corresponding block is read into the zone, and at this point the activity is passivated.

Whenever setreadi is called on a file in update mode, the contents of the saved keyfields are restored into current record in order to prevent the user from destroying the keyfields. But current record is still the previous one because the last call of getreci hasn't been finished. So the keyfields of the wanted record are inserted into the old current record causing an erroneous key sequence in the file.

RC INFORMATION	class EXT.	repl.		ident EAH810401	
	X RC 4000	X RC 6000	X RC 8000	RC 3600	page 2/2

subj. Handling of ISO-files in a Coroutine System.

2. Remedy.

After a run-time alarm in a coroutine system you should always allow the implicit passivated activities to finish their started area transports.

The following procedure will do the job when called in the monitor block, provided that the coroutines are passivated by something else than area transports.

```
external
procedure finis_trans;
begin
  comment    the procedure finishes area transports in implicit
             passivated activities;
  integer array ia(1:12), messbuf(1:3), proc_descr(1:1);
  integer max_act, act, res;

  max_act:= system (12, 0, ia);

  for act:= 1 step 1 until max_act do
  begin
    repeat
      system (12<*act.descr*>, act, ia);
      res:= ia(8);
      if res = 2 then  <*implicit passivated*>
      begin
        system (5<*core move*>, ia(1), messbuf);
        if abs messbuf(3) > 100 then  <*not pending answer*>
          system (5, abs messbuf(3), proc_descr)
        else
          proc_descr(1):= 4;
        res:= if proc_descr(1) = 4  <*kind=area*>
              then activate(act) extract 24
              else 0;
      end res = 2;
    until res <> 2;
  end for act;
end finish_area_transports;

end;
```