Proposal

I. OBSERVABLE STATES OF A SUBPROCESS

These are the possible values of the call stack entry for the subprocess, when it is not the current running subprocess. This description does not include F-return count or class code information/

A. About to execute instruction word at address P.

B. In the middle of a complicated XJ instruction at address P.

Should occur only on explicit or implicit subprocess call. Implicit subprocess call's occur due to an error return or F-returns with multilevel operations.

C. Finishing up an XJ instruction at address P.

The body of the action has been completed, a normal return has occurred, but no P counter offset computation has yet occurred, two cases where this can occur are:

i. When the P-counter offset with place P-counter outside the field length or negative.
ii. On a get event or luug ECS action when an interrupt is recognized after an event has been received.

II. A CATALOGE OF VARIOUS CASES

A. ECS action (other than get event or hang) *luug*

State B cannot occur unless the ECS action returns with F return in a multilevel operation or error returns, (in both cases causes immediate implicit subprocess call.)

B. Get event or hang

State B can only occur in the event of an error return.

If an interrupt is recognized before an event is received, state A occurs iwth P equal to the address of the XJ instruction. If an interrupt is recognized after an event B received then the stack entry will be in state C with P equal to the address of the XJ instruction.

C. Subprocess call, explicit or implicit
(except error due to P-counter offset negative or outside field length)

The stack entry will be in state B with P equal to the address of the XJ instruction. <u>Any interrupts will not be recognized until some other subprocess is actually running</u>.

D. Interrupts

When an interrupt is recognized the interrupted subprocess entry in the call stack will be set as follows:

i. If recognized between instruction words, the stack state will become A with P equal to the address of the next instructions to execute.

        ii. If recognized during get event or hang and before an event is received, the stack state will become A with P equal to the address of the get event or hang XJ instruction.

        iii. If recognized during get event or hang and after an event has been received, the stack state will become C with P equal to the address of the get event or hang XJ instruction.

        iv. The above 3 cases are the only cases in which an interrupt can be recognized by a subprocess.

E. Errors

When an error occurs in a subprocess the stack entry will be set as follows:

    i. Arithemetic or scope type error.
    The stack entry will be placed in state A

    ii. Error return from ECS action or explicit or implicit subprocess call.

        The stack entry will be placed in state B with P equal to the address of the XJ instruction.

    iii. P counter off set would make P counter negative or outside field length.

        The stack entry will be placed in state C with P equal to the address of the XJ instruction.

    iv. There is no XJ instruction at the address expected for eomputation of a P counter offset.

        The stack entry will be placed in state C with P equal to the address of the XJ instruction.

    v. <u>The above 4 cases are the only cases in which</u> *non-existant* <u>an error can be recognized.</u>

III RETURN ACTIONS (which depend on the state of the stack entry.)

A. Normal return (included return with parameters)

    i. If the stack was in state A, execution resumes with the instruction at P.

    ii. If the stack was in state B, P counter offset is computed from the XJ instruction at address P, and execution resumes at the computed address.

    iii. If the stack was in state C, same as ii).

B. Special return ("return and re-execute")

    i. If the stack was in state A, execution resumes with the instruction at P.

    ii. If the stack was in state B, same as i)
    (This is the re-execute case)

    iii. If the stack was in state C, same as A)ii)above.