<table>
<thead>
<tr>
<th>SD.</th>
<th>Fl</th>
<th>Interrupt Data</th>
<th>C-List Length</th>
<th>Next Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>= SD. FINT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD. CC</td>
<td>Subprocess Name (Class Code)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD. ESM</td>
<td>Error Selection Mask</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD. CLST</td>
<td>C-List Unique Name, Not</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD. PTRS</td>
<td>MAX STACK</td>
<td>FATHER_PTR</td>
<td>MAPIN_LIST</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PTR</td>
<td>POINTER</td>
<td>PTR - #0 (addition)</td>
<td></td>
</tr>
<tr>
<td>SD. ORIG</td>
<td>ENTRY</td>
<td>MAP</td>
<td>C-List</td>
<td></td>
</tr>
<tr>
<td></td>
<td>POINT</td>
<td>ORIGIN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD. MAP</td>
<td>COMPILUS MAP # LOGICAL</td>
<td>PTR TO LOGICAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SIZE</td>
<td>MAP ENTRIES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SD. RAFL</td>
<td>RAFL</td>
<td>RA</td>
<td>COMPULS MAP END</td>
<td></td>
</tr>
</tbody>
</table>

compiled map follows immediately

- apply uniformly to all the values with

0 no relocation, points to next SP, rel to beg the SP

1 \( V_1 + V_2 + V_3 + V_4 \); rel B1

2 \( V_2 \) (comment: \( V_1 + V_2 + V_3 + V_4 \) point across a deletion)

3 \( V_1 + V_2 + V_3 \); rel B1

**Flags**

- SD.FINT
- SD.MPE
- SD.UPS
- SD.MAPIN

\( F = 1 \) map in, \( F = 0 \) map out
Map area is a subprogram descriptor.

<table>
<thead>
<tr>
<th>SD.MAP</th>
<th>MP.RAFL +1</th>
<th>MP.CNT</th>
<th>MP.CMAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>compiled map size</td>
<td>logical map entries</td>
<td>logical map references</td>
</tr>
<tr>
<td>+?</td>
<td>RA+FL</td>
<td>RA</td>
<td>space left</td>
</tr>
<tr>
<td>provided</td>
<td>compacted map</td>
<td></td>
<td></td>
</tr>
<tr>
<td>allocated</td>
<td>compacted count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SAD map count</td>
<td>CM_ADDR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECS ADDR</td>
<td>WD COUNT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Compiled map works 2 size of compiled map buffer specified when created.

Logical map area

Same format

Next subprocess follows.

The MP.xxx symbols are mainly for x-referencing; they are relative to the word labeled SD.MAP; i.e., MP.RAFL=1.

F = 1 if map is off, 0 if map is on.

RO = 0 for read/write entry, 1 for read-only entry.

DAE = 1 for DAE entry (only legal on 1st word of compiled map).

LE = 1 for last entry corresponding to a given logical map entry.
accumulated changes to process descriptor

P. SCHED gets bigger

S. INTIM rear clock: set to S. CHARG on swapin

P. TIMER decremented by S. CHARG - S. INTIM at every 3rd if - desched

P. MESSG UNMOT of each message (event)

P. CLOX number of clocks in non-process (≤ 4)

P. MAPESM disappears

P. TRUST disappears

F. ODDP disappears

Flags:
P. P pending for swapin
P. W unchain from each
P. R scheduled (running)
P. I pending interrupt
P. D pending destruction of process

≤ 5 PF. E 0 = real process, 1 = pseudo process
P. C process in core
P. V rescheduled for by arrival of event
P. S descheduled by timer out
P. EE

all these names are only for cross-referencing
P. F. H desched cause being on event

R. MAISIN

PR α-1
**P. PARAM** data word in 1 cell containing the date

X

Block in 2 words, 1st word = 0, 2nd word =

cap in 2 words, 1st

**P. PARAMC** class code of called subp

-2 parameter type bit mask

-1 words used, # cap, area, total data

0 = data word

1 = other, dop or format
<table>
<thead>
<tr>
<th>FLAGS</th>
<th>F-RETURN COUNT</th>
<th>IP LIST ADDR.</th>
<th>P-COUNTER, REL RA OF CURRENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>END OF PATH SUBPROCESS</td>
<td>CURRENT SUBPROCESS</td>
<td>PTR</td>
</tr>
</tbody>
</table>

**FLAGS:** SF, II -

Interrupt inhibit. If an interrupt arrives for the current subprocess (when it is at the top of the stack and interrupts are armed) and the bit is set, the interrupt pending (or is lost if an interrupt is already pending) until the bit is cleared.

The bit is automatically cleared on any subprocess call; it can be explicitly cleared by the current subprocess when it is cleared any pending interrupt will occur immediately.

- SF, PCQ1 = 4
- SF, PCQ2 = 5

The p-counter qualifier. These bits must be adjacent to the return count and are named merely for cross-referencing purposes. Values are:

<table>
<thead>
<tr>
<th>SF, PCQ1</th>
<th>SF, PCQ2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 = about to execute</td>
<td>1 = almost finished</td>
</tr>
<tr>
<td>0 = in middle</td>
<td>= 1 should not occur</td>
</tr>
</tbody>
</table>

These fields are pointers to the first word of the appropriate subprocess descriptors. The pointers are relative to BL.
E40 make PMCSP point to FPSP; FPSP point where PMCSP did MAPIN(FPSP)

Remember error:
1) map off
2) file gone
3) more on pending

E50 PMCSP ← FPSP
make entry in full chain for FPSP, B3 ← B3-2

FPSP = CURRENT?

Yes → E71

No
FPSP ← father of FPSP

E60 Set PMCSP points to FPSP

PMCSP ← FPSP
MAPIN(FPSP)

Remember error:
1 2 3 as above

make entry in full chain
B3 ← B3-2
FPSP = CURRENT?

Yes → E70
FPSP ← father of FPSP

end of chain
E70 set end of my chain flag in $FPSP$

E71 set S:USORA to absolute userRA
set UserRA (old) in $XPACK$
set FL in $XPACK = RA + FL of BOP - RA of counts$
end MA in $XPACK$ being equivalent
set P:CLIST
set up direct ECS access/ony
restore return links
read local list \(\text{if done}\) \(\rightarrow\) E80

map: error? \(\rightarrow\) gee
\[ \text{no} \]
\[ \text{exit to B7 (menu)} \]

E76 exit to B6 with MAPEROURS

E80 maperror? \(\rightarrow\) gee
\[ \text{no} \]
CGONE error \(\rightarrow\) B6
both error \(\rightarrow\) B6
1) Do all subspace fixed?

P. TEMP <- rethink

Why stack him -> cannot talk? -> stack future

Yes
TOSPROC

SYSEXT

Increment function counter: st.
FRET(TOS) ← FRET(TOS)+1

RETP ← 1 (off)

Priority interrupt?

no

.Priority interrupt

TOS almost done?

TOS interrupt

fetch?

no

TOS interrupt

FRET(TOS) > 0?

no

no

TOS interrupt

Fetch X S

got offset

in X S?

no

TOS interrupt

new PC = PC + offset + 1

no

new PC ← 0

TOS interrupt

PC(XPACK) ← new PC

update P, SYSTM, S, SYSTM

display mode

S.RET
USERCAL

display option
update P, USBTIM, s, USRM
fetch PC(XPACK) = CEJ

B4 = Last16 & < 0? yes
no

UCALL1 TOS ← oldflag2, FEAT=0, INJST, PC
INJST < 0 or 3FL - error

Detect (DPO)

UCALL2 IPO = open → error

IPO ∈ IPONo? no
PSCR2 ← off, header, read bit

B1 ← rest of IPO

ipled beyond PL → error

parameter loader?

Initialize XAMBUF, actual parameters, area, etc.

OPINTER
SA2 B1+3 = classcode

SUBCAL?

Y/N action

Y

act v PARAM < 3 classcode

-1
-2
ERRPROC

E: ERROR

ERRPROC

RETLink ← TOSPROC

P:PARAM ← cur#

stick full → KILLPROC

P:PARAM+2 ← current SP points from TOS

B6 ← top of stack

crunch ZCO

B3 ← crunch

goto E:ERROR

search ancestors of current for 31st child

none → error

hit on

found

current TOS puts < limit thru SP

continue

turn off bit

P:PARAM+1 ← on class

entry offset = P:ERROR

goto NEWTOS
<table>
<thead>
<tr>
<th>PROCESS</th>
<th>C</th>
<th>E</th>
<th>X</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Setmess</td>
<td>G</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clmess</td>
<td>G</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timin</td>
<td>G</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Timont</td>
<td>G</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Testib (STIB)</td>
<td>G</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clriib (CLIB)</td>
<td>G</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run</td>
<td>G</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Firearm</td>
<td>G</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Printib (PRINT)</td>
<td>G</td>
<td>✓</td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>Auth</td>
<td></td>
<td></td>
<td>G</td>
<td></td>
</tr>
<tr>
<td>CHIP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Display stack</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;entry&quot;</td>
<td>G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Send interrupt PRINT</td>
<td></td>
<td></td>
<td></td>
<td>G</td>
</tr>
<tr>
<td>Command</td>
<td>C</td>
<td>E</td>
<td>X</td>
<td>T</td>
</tr>
<tr>
<td>---------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Create</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Jumpcal</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Call</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FANCY</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Return</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>SPRET</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Retparam</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Jumptot</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNMONTHS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>MODPC</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSON</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSPSP</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRETURN</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>ERRCALL</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESMSETN (ESMSN)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ESMSET (ESMS)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FINDSUB</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENVIRON</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>STRETCH</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
SYSENT

TOSPROC ✓ ✓ ✓ ✓
PUTRETP ✓ ✓ ✓ ✓
OPINTER ✓ ✓ ✓ ✓
SYSRET ✓ ✓ ✓ ✓
SYSFRET ✓ ✓ ✓ ✓
E.ERROR, E.RAPROC ✓ ✓ ✓ ✓
NEWTOS ✓ ✓ ✓ ✓
<table>
<thead>
<tr>
<th>MAPS</th>
<th>C</th>
<th>E</th>
<th>X</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAPIN</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>MAPOUT</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>MARCHK</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>CALLCMP</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>REFER</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>REFZ</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>DISPLAY (Current)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot; (Sweep)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAPON</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAPOFF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| SWAP       |   |   |   |   |
| SWAPOUT    | ✓ | ✓ |   | ✓ |
| SWAPIN     | ✓ | ✓ |   | ✓ |
| FSWAP      | ✓ | ✓ |   | ✓ |
Routines

✓ TOS Processor (except glue in return parameter code)
✓ SYSRET
✓ New TOS (in, descriptors of callers)
✓ Environ (TOS) (has initialization, some writing)

✓ FINDSUB  
  x3 = class code  
  B7 = netlink
✓ MAPIN  
  BS = I/O processor?
✓ MAPINX
✓ CALLCNP
✓ REFER
✓ REPZ
✓ MAPCHK
✓ Display map entry of S
  "  "  from full path
✓ SWAPIN/OUT (write set up stack) (in TOS)
✓ SWAPIN (call last destroy process, separate out ENV)
Old stuff.*

* (jerk)
run asubprocess(type, pe, pf) local ef1ay, pf1ay, point

startflow = enter point - type
exitflow = flow of

ef1ay? yes -> compute enmosubprocess -> run asubprocess(enter, ef1ay, pf1ay)
no

interrupt pending? yes -> compute interruptsubprocess -> run asubprocess(interrupt, ef1ay, pf1ay)
no

Calluser? yes
point is callafter

operation? yes

 examining order of the operation
Callsubprocess? yes
execute actions? yes

 F-flow? yes

 suspend event channels
new event channels

 event? yes

 compute interrupt subprocess

 interrupt pending? yes

 e-puton? yes

 e-puton? yes

 F-return? yes

 pf = on

 e-puton? yes

 e-putoff? yes

 e-putoff? yes

 e-putoff? yes

 exit
New TOS as class order of called SP, not link

Stack full error? \( \rightarrow \) EP

Determine EP, set EOP, current
Increment stack stuff
Set PCM = about to
f ret = 0
EP counter = EP - n
interrupt inhibit

Error (TOS) \( \rightarrow \) EP

\( n = 0 \)

Parameter from area

\( n = 1 \)

Parameter from area
dep

dep

\( n = 2 \)

Int. data from current SP
go some drop area

return

must

without return if errors are encountered
TOS Processor - Examined (TOS) exists

TOSPROC

- set no parameters
- priority interrupt? - yes \rightarrow \text{NETOS} \rightarrow \text{TOSPROC}
  no

SYSRET

- set no parameters
- \( PCM = \text{about to?} \) - yes \rightarrow \text{Pre計器}(XPACK) \leftarrow \text{Pre計器}(\text{static})
  update clocks
  quantum = \( \frac{x_5}{y_0} \) \rightarrow \text{S.RETU}
  \( x_5 \)

- PCM in middle? - yes \rightarrow \text{return parameter is flagged} \rightarrow \text{fetch EP}
  return count > 0?

- no
  - PCM almost finished?
    - no
      - another order?
        - yes
          - modify = 0 \rightarrow \text{interruptador} \rightarrow \text{EP}
            \text{special UCALL3}
          - no
            - \text{fetch EP}
              - modifier \leftarrow \text{offset from X5}

- no

- \( PCM = \text{about to?} \)
System Call

- `system_call`
- `clock2`
- `set 1st word of top of stack` (VCM = 1N middle)
- `counter = XNCK-1`
- `pointer = 0`
- `set emit from OPINTER = UCALL3`
- `fetch 1st IPO, set up 2nd order --> EP`
- `open file, xmit to UCALL3`
- `OPINTER --> EP`

- `UCALL3` essentially jumps this actual `ACTION` --> EP
- `STU FRBT`
- `STU RET`

- `advance frame select` - area count
- `area count = 0` Then `OUT` - 2xx UCALL3
- another add --> 5xx after rendering...
get event

event yes system return
  no
  deschedule process
  swapout?

set IIB

clear IIB
  test for pending interrupt

swapout

parswap
  fix clocks
  set up TOS
  change X86k around

swapout
  get next process from scheduler
  swapout map? (flag cubre for any map errors)
  write out process description
  timer out? yes deschedule process
  write out clock
  go to cornoc
TOS for process being swapped is always about to execute unless it was hung or missed device, in which case it is in the middle.

**TOS**

Initialize S.INTM, S.QUANT
Read process descriptor

\[
\text{unchain ?} \quad \text{\{destroy ?} \quad \text{event ?}
\]

\[ \text{PCQ} \rightarrow \text{almost finished (should have been "in middle")} \]
\[ \text{clear flags } \leq \text{EOS} \]

\[ \text{そんな (TOS)} \quad \text{special entry} \quad \text{\{} \quad \text{EP} \rightarrow \text{TOS process} \quad \text{\{machine\}} \]
\[ \text{total of accept time} \]

\[ \text{PCM} \rightarrow \text{in middle} \quad \rightarrow \text{TOS process} \]

\[ \text{priority interrupt ?} \quad \text{New TOS} \quad \text{TOS process} \quad \text{maybe set ACM = about 70 ??} \]

Too thin have to rehang a guy because the interrupt which woke him up didn't have priority.

\[ \text{PCM} \rightarrow \text{about to} \]
\[ \text{TOS process (queue flag)} \]
\[ \text{pending ?} \quad \rightarrow \text{S.RECY} \]
\[ \text{PCQ} \rightarrow \text{in middle} ? \quad \rightarrow \text{PCQ} \rightarrow \text{almost ?} \]

\[ \text{TOS proc} \]
EP

return = TOS processor

locate SP to field error

NO SPIERR => EP
do sched

sprom to top

everyone to proper

Codesub(1)

wpm = about to

f_counter = entry - 1

f_return count = 0
codesub(1)

return

SP call action

locate called SP ---＞ EP

New TOS(0) ---＞ TOS processor

SP jump call

locate caller ---＞ EP
do stack

New TOS(1) ---＞ TOS processor