Information

MARK III

ptr types

0 ≡ SD. FIPT	FLAGS	INTERRUPT DATUM	C-LIST LENGTH	NEXT POINTER	0
SD.CC	SUBPROCESS NAME (CLASS CODE)				
SD.ESM	ERROR SELECTION MASK				
SD.CLST	C-LIST UNIQUE NAME, MOT				
SD.PTRS		MAX STACK PTR	FATHER POINTER	MAPIN LIST PTR −70	3 2 X
SD.ORIG		ENTRY POINT	MAP ORIGIN	C-LIST ORIGIN	
SD.MAP	F	COMPILED MAP SIZE	# LOGICAL MAP ENTRIES	PTR TO LOGICAL MAP, REL THIS WORD	
SD.MAP+1 ≡ SD.RAFL	0	RA+FL	RA	SPACE LEFT IN COMPILED MAP BUFF	1 1 0

compiled map follows immediately

Δ - applies uniformly to all SPs unless noted

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

1 $V_1+V_2+V_3+V_4$; rel B1

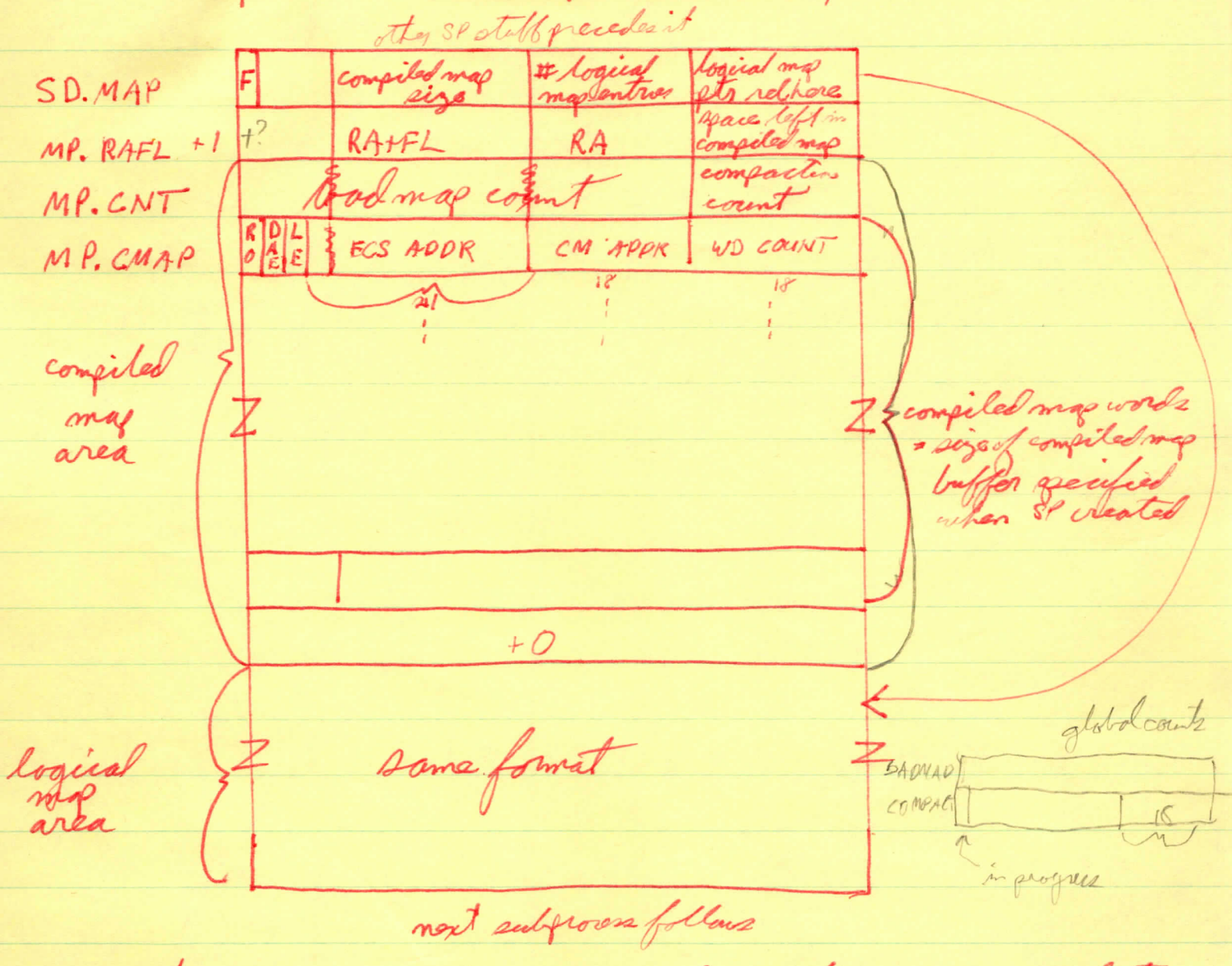
2 ~~V_3 sometimes~~ ; rel B1 to beg the SP; ~~change when they point across a deletion~~
$V_1+V_2+V_3(+V_4$ sometimes)

3 $V_1+V_2+V_3$; rel B1

FLAGS
- SD.FINT — interrupt pending
- SD.FMPE — map error pending
- SD.FDIS — interrupts disarmed
- ~~SD.MPIN~~ — map in

F=1 if map is off

Map area in a subprocess descriptor



the MP.xxx symbols are mainly for x-referencing & are relative to the word labeled SD.MAP; ie 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 log. map entry

accumulated changes to process descriptor

P.SCHED gets bigger

needn't go out to ECS

S. INTIM near clocks: set to S.CHARG on swapin

contiguous with usertime, etc.

P.TIMER - - decremented by S.CHARG-S.INTIM at swapout; if -, desched

P.MESG | UN, MOT of each message (event) |

P.CLOX = number of clocks in user process (=4)

P.MAPESM disappears

P.IRUST disappears

P.OLDP

Flags:

PF.P something pending for swapper

PF.W unchain from each

PF.R scheduled (running)

~~no longer used~~ PF.I pending interrupt

PF.D pending destruction of process

5 ≡ PF.E 0 = real process, 1 = pseudo process

PF.C process in core

PF.V rescheduled by arrival of event?

PF.S descheduled by timer out

↕ PF.EE " for ~~error~~ event

all three names are only for cross-referencing

PF.H ~~Hung~~ descheled 'cause hung on EVCH

~~P.TEMPS~~

P.MAPSIN | 1 | P.IR or -1 |

(set to -1 ~~to be used~~ by swapper at swapin)

P.PARAM data word in 1 cell containing the datum

X block is 2 words, 1st word = 0
2nd word =

cap is 2 words, 1st

0 = data word
1 = other, dap or format

-2 parameter type bit mask

-1

	words used in A area	# cap params	total data params

P.PARAMC class code of called subp

STACK ENTRIES

FLAGS	F-RETURN COUNT	IP LIST ADDR, REL RA OF CURRENT	P-COUNTER, REL RA OF CURRENT	← PTR
+		END OF PATH SUBPROCESS [1]	CURRENT SUBPROCESS [1]	

FLAGS: SF.II - Interrupt inhibit. If an interrupt arrives for the current subprocess (when it is at the top of the stack & has interrupts armed) & this bit is set, the interrupt pends (or is lost if an interrupt is already pending) until the bit is cleared.
The bit is automatically set on any subprocess call; it can be explicitly set & 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 & are named merely for cross-referencing purposes. Values are:

SF.PCQ1 = 1 = about to execute
SF.PCQ2 = 1 = almost finished
both = 0 = in middle
both = 1 should not occur

1 These fields are pointers to the first word of the appropriate subprocess descriptors. The pointers are relative to B1.

flowcharts

ENVIORN

initialize reg, etc (not B6, MCSP!!)

1) end of map chain = -SD.PTRS
   a) in ENVIORN ✓
   b) in SWAP
   c) OMYCOD
2) set MA in 1) OMYCOD
   2) Create process

EOP already in? — yes → (E50) ✓ X (root return to BLDR)

↓ no

E10 → end of mapchain? — yes → (E60) ✓ X (swapin)

↓ no

EOP above MCSP? — yes → (E40) ✓
(EOP RA ≥ MCSP RA+FL?)

↓ no

EOP below MCSP? — no → (E35) ✓
(EOP RA+FL ≤ MCSP RA?)

↓ yes

PMCSP ← MCSP

E20 FPSP already in? — yes → (E50) ✓

↓ no

E30 end of map chain? — yes → (E60) ✓

↓ no

FPSP above MCSP? — yes → (E40) ✓

E35 set MCSP not in core ← 1 ref (EOP collides with MCSP)

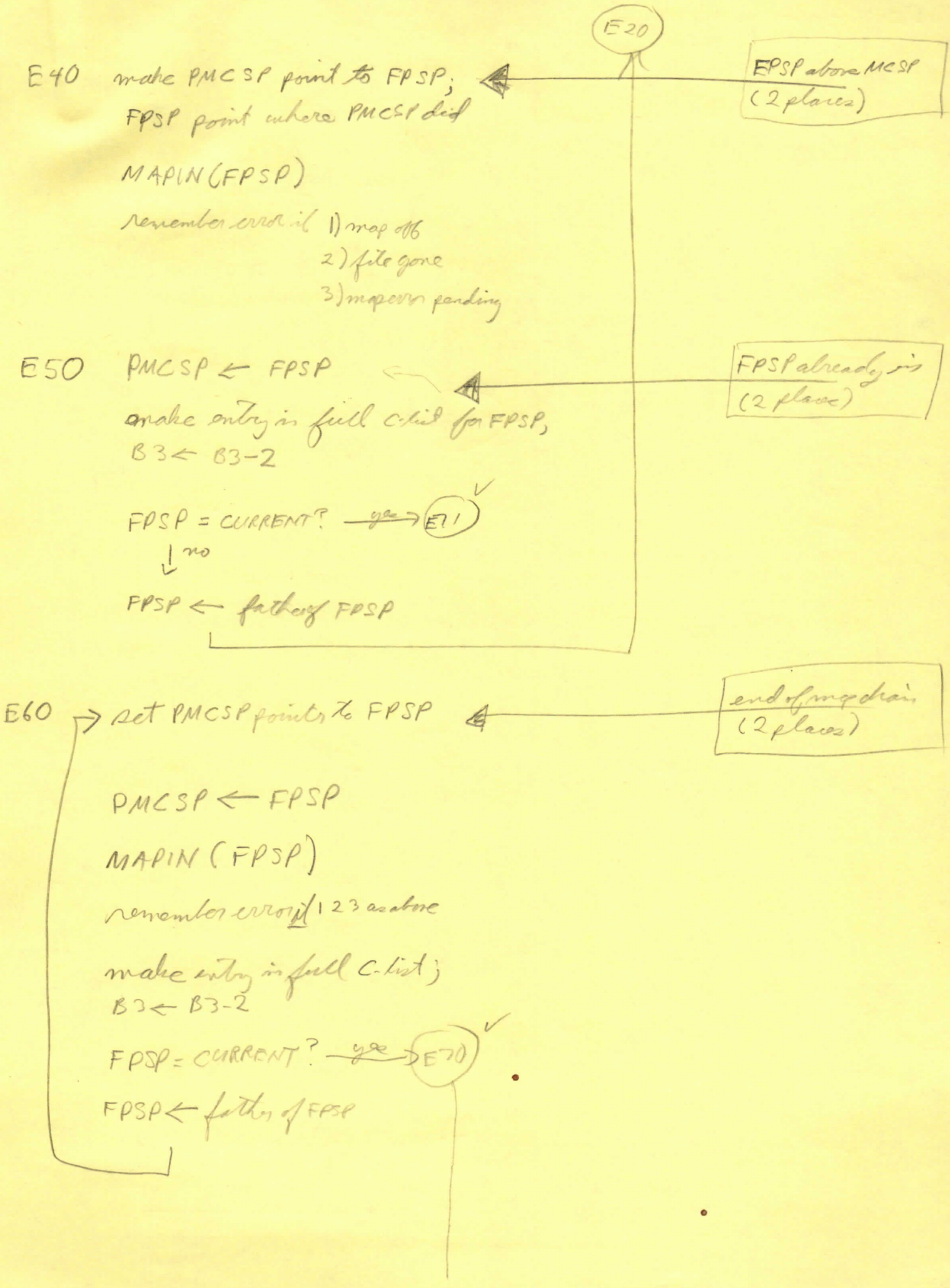
PTR in PMCSP ← PTR from MCSP

MAPOUT (MCSP)

file gone error? — no

↓ yes

(needs help!)
flag old MCSP for pending
map error


E20
E40 make PMCSP point to FPSP;
FPSP point where PMCSP did
MAPIN(FPSP)
remember error if 1) map off
2) file gone
3) maperr pending
FPSP above MCSP
(2 places)
E50 PMCSP ← FPSP
make entry in full C-list for FPSP,
B3 ← B3-2
FPSP = CURRENT? yes E71
no
FPSP ← father of FPSP
FPSP already in
(2 places)
E60 set PMCSP points to FPSP
PMCSP ← FPSP
MAPIN (FPSP)
remember error if 1 2 3 as above
make entry in full C-list;
B3 ← B3-2
FPSP = CURRENT? yes E70
FPSP ← father of FPSP
end of mq drain
(2 places)

E70 set end of map chain flag in FPSP

E71 set S.USERA to absolute user RA

set user RA (abs.) in XPACK

set FL in XPACK = RA+FL of EOP − RA of current

{set MA in XPACK} being rejected out

set P.CLIST

set up direct ECS access if any

restore return links

read local clist (if gone) → E80

map error? — yes

↓ no

exit to B7 (normal)

E7(exit to B6 with MAPERROR

E80 map error? — yes

↓ no

CGONE error exit to B6

BOTH error to B6

NewTOS

1) ~~Is old Subproc fixed?~~

P. TEMP4 ← retlink

subp stack lim > current stack? —no→ stack fullerror
↓ yes

TOSPROC

SYSFRET — increment fretum count in stack
FRET(TOS) ← FRET(TOS)+1

SYSRET
RETP ← 1

TOSPROC — RETP ← 1 (off)

Priority int? — yes → NEWTOS (interrupt subp, TOSPROC)

no ↓

TOSPR2 — TOS about to ex? — yes → TOSPR1 newPC ← PC(TOS)

no ↓

TOS almost bin? — yes →

no ↓

RetPar? — yes → check this: return thru ... error

no ↓

TOSPR4 — FRET(TOS) > 0? — yes → check this: FRET

no ↓

TOSPR7 — Fetch XJ
get offset
is XJ? — no → error

TOSPR8 — newPC = PC(TOS) + offset + 1

newPC negat 0 — yes → error

TOSPR2.5 — PC(XPACK) ← newPC

update P.SYSTEM, S.SYSTEM display word

S_RETN — XJ

USERCAL

display system

update P.USRTIM, S.USRTM

fetch PC(XPACK) = CEJ

B4 = low 8 bits < 0? —yes→ B4 = -low 4 bits

↓ no

UCALL1 TOS ← dbflags, FRET=0, IPLIST, PC

IPLIST < 0 or ≥ FL —error→

Getcap (IPO)

UCALL2 IPO = op? —no→ error

IPO allowed? —no→ error

P.SCR2 ← op's header & mask bits

B1 ← rest of op?

iplist beyond FL —yes→ error

parameterless?

Initializing PARMBUF, actual parameter area, etc.

OPINTER

SA2 B1+3 = class code

Subcall?

yes → ↓ no

action

ect P.PARAMC -1 -2 } class code bitmask

action

ERRPROC

E.ERROR — RETLINK ← TOSPROC

ERRPROC — P.PARAM ← err#

stack full —yes→ KILLPROC

P.PARAM+2 ← current SP ptr from TOS
B6 ← top of path " "
err class ≥ 60 —yes→ error too big goto E.ERROR
B3 ← error class

→ search ancestors of current for SI which bit on —none→ err a error?
no → error error go to E.ERROR
yes → STOPPROC

found ↓

current TOS ptr < limit the SP? —no→ cont
no → continue search

yes ↓

turn off bit

P.PARAM+1 ← err class

entry offset = P.ERROFF

go to NEWTOS

PROCESS	C	E	X	T
Create	✓	✓	✓	
Delete				
Display				
Setmess	G	✓		
Clrmess	G	✓		
Timin	G	✓	✓	
Timout	G	✓		
Setiib (STIIB)	G	✓		
Cliib (CLIIB)	G	✓		
Qrun	G	✓		
Disarm	G	✓		
Priint (PRINT)	G(2any)	G		
~~SCAT~~ Qaths				
CHIP				
Display stack	G			
" " entry	G			
Send interrupt PTINT	G			

SUBPROC	C	E	X	T
Create	✓	✓	✓	
Delete	✓	✓	✓	
Jumpcal	✓	✓		
Call FANCY	✓	✓	✓	
Return	✓	✓	✓	
SPRET	✓	✓	✓	
Retparm	✓	✓	✓	
Jumpret				
FINDNTHS	✓	✓	✓	
MODPC	✓	✓		
FSON	✓	✓		
DSPSP	✓	✓		
FRETURN	✓	✓	?✓	
ERRCALL	✓	✓		
ESMSETH (ESMSN)	✓	✓		
ESMSET (ESMS)	✓	✓		
FINDSUB	✓	✓		
ENVIORN	✓	✓		
Setretp	✓	✓	✓	

SYSENT	C	E	X	T
TOSPROC	✓	✓	✓	
PUTRETP	✓	✓	✓	
OPINTER	✓	✓	✓	
SYSRET	✓	✓	✓	
SYSFRET	✓	✓	✓	
E.ERROR, ERRPROC	✓	✓	✓	
NEWTOS	✓	✓	✓	

MAPS	C	E	X	T
MAPIN	✓	✓	✓	
MAPOUT	✓	✓	✓	
MAPCHK	✓	✓	✓	
CALLCMP	✓	✓	✓	
REFER	✓	✓	✓	
REFZ	✓	✓	✓?	
DISPLAY (Current)	✓	✓		
" (Swap)	✓	✓		
MAPON				
MAPOFF				

SWAP				
SWAPOUT	✓	✓	✓	
SWAPIN	✓	✓	✓	
FSWAP	✓	✓	✓	

Routines

✓ TOS Processor (except glue in return parameter code)
✓ SYSRET
✓ New TOS (n, descriptor of caller)
✓ Environ (TOS) ~~(bit initialization + some write up)~~

~~Locate DD~~

✓ FINDSUB : X3 = classcode
B7 = retlink
✓ MAPIN ✓ ~~B5 = 1 if preserved?~~
✓ MAPOUT
✓ CALLCMP
✓ REFER
✓ REF2
✓ MAPCHK
✓ Display map entry of SP
✓ " " " from full path
✓ SWAPOUT ~~(call but set up stack)(interrupt)~~
✓ SWAPIN (call but destroy process, separate out ENVIORN errors)

Fetch parameters

✓ Return parameters
✓ SYSCALL
✓ SYSFRET
Return Opr
✓ EP
✓ E.ERROR

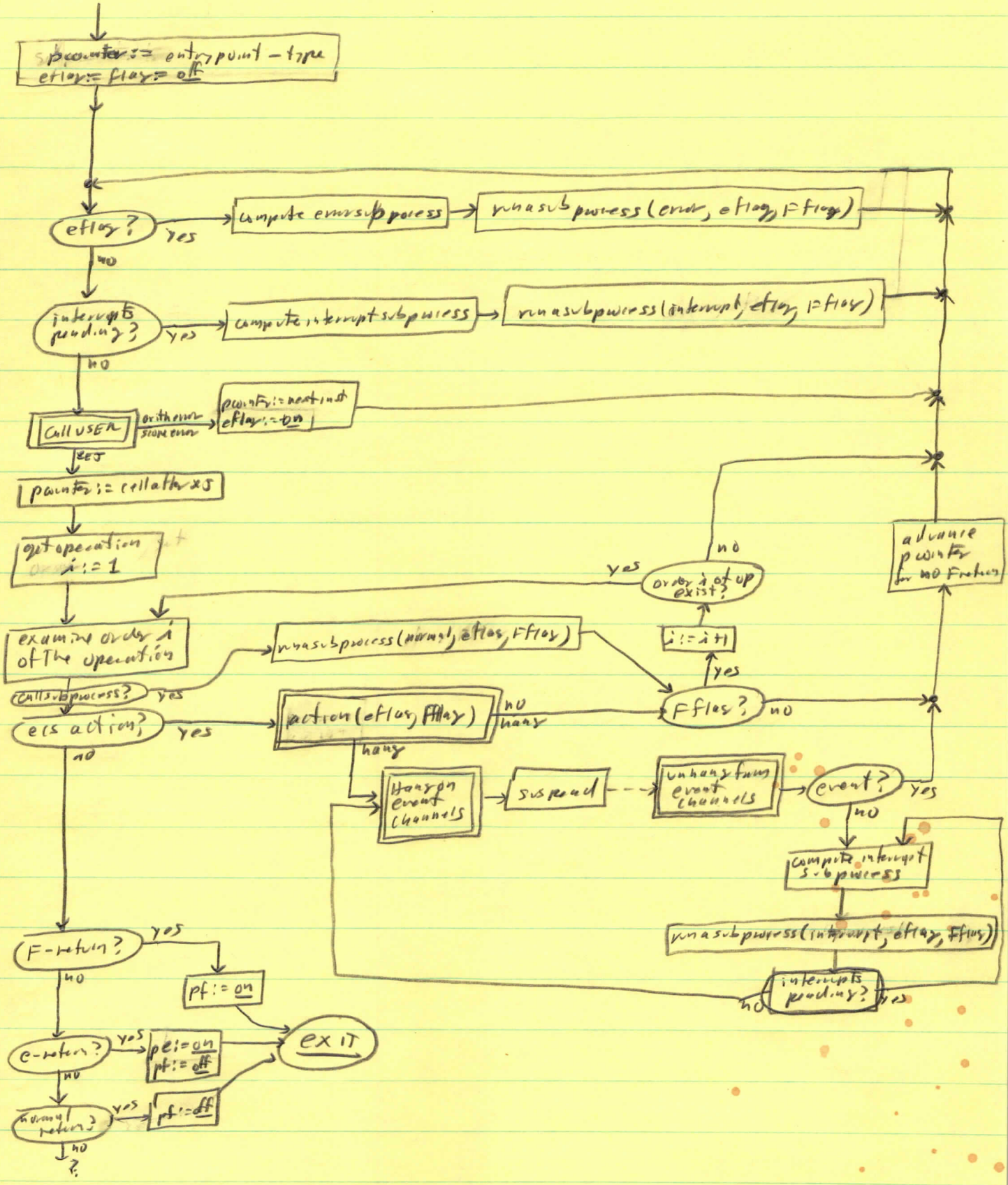
SCOPE error
CHIP error
✓ Subprocess call
✓ Subprocess jump call

Old stuff.*

* (junk)

6/9/70

run a sub process (type, pe, pf) local eflag, Fflag, pointer, i



NEWTOS

New TOS n, ~~descriptor~~ call type (different) B3 of called B2 SP, retlink B7

stack full error? ----> EP

B6 error retlink

{determine EOP, set EOP, current
increment stack stuff
set PCM = about to
f-ret = 0
P-counter = EP-n
interrupt inhibit

Environ (TOS) ----> EP

~~n=0~~

if non-trivial full path, set flow core
~~flow~~ with data relevant to old current SP.

n = 0	n = 1	n = 2
params from ap area ----> EP	params from error dump area	int datum from current SP or some dump area

return

must
~~did not~~ return if errors are encountered

TOS Processor - Execution (TOS) exits

TOSPROC — set no parameters
priority interrupt? - yes → NETOS ---→ TOSPROC
no

SYSRET — set no parameters
→ PCM = about to? - yes → P-counter(XPACK) ← P-counter(stack) (2.5)
update clocks
quantum up - yes → SUSPOUT
no
S.RETU XJ no

PCM = in middle? - yes → return parameters if flagged ---→ EP
return count > 0?
no → PCM = almost finished?
yes → another order?
no → modifier = 0
yes → interpret order 2 ---→ EP
call UCALL3
no

fetch XJ ---→ EP
modifier ← offset from XJ
New P-counter ← P-counter(stack) + modifier + 1 ---→ EP
P-counter(stack) ← New P-counter
PCM = about to
(2.5)

~~F.p:=0~~ ~~#~~ 100

System Call

clocks

set 1st word of top of stack. (OPCM = IH middle)

P-counter = XJACK-1

f-return = 0

~~set ent from OPINTER = UCALL3~~

xx ~~[illegible]~~ fetch IPO, set up 1st order --> EP

opinter xit to UCALL3

OPINTER --> EP

UCALL3 essentially jump thru ACTION <---- entered after processing orders by F-return code

ACTION ----> EP

SYSRET

SYSRET

UCALLX ~~advance to next order~~

~~order exhausted~~

order count = f-return count ? --> yes: UCALL3

another order ? --> yes: xx after reading order

F-return

get event

event —yes→ system return
↓ no

deschedule process

swapout?

set IIB

clear IIB

test for pending interrupt

swapout

FORSWAP – fix clocks
set up TOS
change XFaddrs around

Swapout – get next process from scheduler
swapout map2 (flag subp2 for any map errors)
write out process descriptor
timer out? —yes→ deschedule process
write out clocks
go to swapin

TOS for a process being swapped in is always "about to execute" unless it was hung on an event channel in which case it is "in the middle".

Swapin

Initialize S.INTIM, S.QUANT

Read process descriptor

if P-flag:
- unchain? ---->
- destroy? ---->
- event? ----> PCQ ← almost finished (should have been "in middle")
- clear flags in ECS

environ (TOS) special entry?? --turn off quantum time--> EP ----> TOS processor (no phase change)

total up swap time

~~PCM = in middle? --no--> TOS processor~~

~~priority interrupt? --yes--> New TOS, TOS processor~~

~~maybe set PCM = about to??~~

~~we have to rehang a guy because the interrupt which woke him up didn't have priority~~

~~PCM = about to~~

~~TOS processor (reset flag)~~

pending? --no--> S.RETU

PCQ = in middle? --yes--> PCQ ← about to

↓ no

TOSPROC

EP

E.EKKON return = TOS processor

ERRPROC locate SP to field error --→ NO SUBP ERR --→ EP

turn off bit in ESM

↓ deschedule → awaport

~~reset up bcs, and sp in new top of stack~~

overname to proper ~~EP~~ area

~~callback (1)~~

~~PCM = about to~~

~~f counter = entry - 1~~

~~f return count = 0~~

~~Callback (1)~~

return

New TOS

SP call action

locate called SP --→ EP

New TOS(1) → TOS processor

SP jump call

locate callee --→ EP

dec stack

New TOS(1) → TOS processor