

DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL

(3)	224	SPAWN COMMAND
(4)	411	SPAWN A SUBPROCESS
(5)	928	PROCESS SPAWN INPUT STREAM
(6)	1098	CHECK FOR ASSOCIATED TERMINAL MAILBOX
(7)	1145	PROCESS SPAWN OUTPUT STREAM
(8)	1283	ATTACH COMMAND
(9)	1340	PERFORM ATTACH OPERATION
(10)	1596	CREATE ATTACH REQUEST MAILBOX
(11)	1649	DELETE ATTACH REQUEST MAILBOX
(12)	1670	CREATE TERMINATION MAILBOX
(13)	1740	DELETE TERMINATION MAILBOX
(14)	1765	DEALLOCATE SPWN BLOCKS
(15)	1794	WRITE RETURNED MESSAGE
(16)	1836	WRITE CONTEXT TO SUBPROCESS
(17)	2088	WRITE RECORD TO CONTEXT MAILBOX
(18)	2115	WRITE ALL SYMBOLS IN A SYMBOL TABLE
(19)	2179	CHECK FOR PENDING HANGUP AST
(20)	2199	GET DEVICE NAME
(21)	2237	CREATE OUTPUT MAILBOX
(22)	2345	DELETE WRITE MAILBOX
(23)	2366	WRITE REQUEST AST FROM A SUBPROCESS
(24)	2436	READ AST FROM READING ATTACH REQUEST RESPONSE
(25)	2468	ATTACH REQUEST AST FROM ANOTHER PROCESS
(26)	2531	SUBPROCESS TERMINATION AST ROUTINE
(27)	2720	BROADCAST NOTIFICATION MESSAGE

```

0000 1
0000 2
0000 3
0000 4
0000 5
0000 6
0000 7
0000 8
0000 9
0000 10
0000 11
0000 12
0000 13
0000 14
0000 15
0000 16
0000 17
0000 18
0000 19
0000 20
0000 21
0000 22
0000 23
0000 24
0000 25
0000 26
0000 27
0000 28
0000 29
0000 30
0000 31
0000 32
0000 33
0000 34
0000 35
0000 36
0000 37
0000 38
0000 39
0000 40
0000 41
0000 42
0000 43
0000 44
0000 45
0000 46
0000 47
0000 48
0000 49
0000 50
0000 51
0000 52
0000 53
0000 54
0000 55
0000 56
0000 57

```

```

.TITLE SPAWN - MULTI-PROCESSING COMMANDS
.IDENT 'V04-000'

*****
*
* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
* ALL RIGHTS RESERVED.
*
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
* TRANSFERRED.
*
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
* CORPORATION.
*
* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
*
*****

AUTHOR:
    Tim Halvorsen, May 1981

MODIFIED BY:
    V03-025 SSA0029      Stan Amway      6-Aug-1984
    Propagate WS default correctly. GETJPI item code was
    JPI$_WSSIZE, should be JPI$_DFWSCNT.

    V03-024 HWS0097      Harold Schultz    01-Aug-1984
    Change max. DCL command string length from 132 to
    WRK_C_INPBUFSIZ. Change max. logical name translation
    size from 63 to LNM$C_NAMLENGTH.

    V03-023 HWS0088      Harold Schultz    21-Jul-1984
    Fix output mailbox read operation to correctly handle
    an empty mailbox. Fixes problem where subprocesses
    finish but never wake up parent, while running up huge
    buffered I/O counts.

    V03-022 HWS0064      Harold Schultz    02-May-1984
    When attaching to another process, or spawning a new
    new process, don't arm own attach mailbox attention
    AST until just ready to hibernate. (This is to close
    a window where a newly attached process tries to attach
    back to the old current process before it has a chance
    to marked itself detached.)
    Unconceal output stream logical name.

```

0000	58	:	V03-021	HWS0044	Harold Schultz	30-Mar-1984
0000	59	:		Add SPAWN/TABLE to allow propagation of table name.		
0000	60	:		Use table search in cli name verification.		
0000	61	:		Allow ATTACH to attach to any other process in its		
0000	62	:		job, removing the the restrictions of having the input		
0000	63	:		streams be the same for both processes and that the		
0000	64	:		input device be a terminal.		
0000	65	:		When spawning a process, indicate to \$CREPRC that a		
0000	66	:		CLI is being specified. (if none entered via /CLI, the		
0000	67	:		default cli is specified)		
0000	68	:		Make WRITE AST use DCL\$SPAWNOUT instead of DCL\$MSGOUT.		
0000	69	:		Remove \$CHIPDEF		
0000	70	:				
0000	71	:	V03-020	HWS0017	Harold Schultz	21-Feb-1984
0000	72	:		Always signal subprocess errors - do not set inhibit bit.		
0000	73	:		Use \$BRKTHRUW instead of \$BRDCST.		
0000	74	:		Change /CLI specification to cli name instead of cli file spec.		
0000	75	:				
0000	76	:	V03-019	HWS0002	Harold Schultz	03-Feb-1984
0000	77	:		Do not propagate the protection mask from a private		
0000	78	:		logical name table to a sub-process.		
0000	79	:		Fix logical name hash table processing to process		
0000	80	:		first bucket in table.		
0000	81	:				
0000	82	:	V03-018	PCG0017	Peter Georg	12-Oct-1983
0000	83	:		Do not set final status if image count has changed.		
0000	84	:		Check for associated mbx when doing an attach.		
0000	85	:				
0000	86	:	V03-017	PCG0016	Peter George	22-Sep-1983
0000	87	:		Fix bug in null input file processing.		
0000	88	:				
0000	89	:	V03-016	PCG0015	Peter George	15-Sep-1983
0000	90	:		Slap colon at the end of PPD\$T_INPDVI before using.		
0000	91	:				
0000	92	:	V03-015	PCG0014	Peter George	16-Aug-1983
0000	93	:		Correctly supply address of IOSB to IOSM TT_PROCESS QIO.		
0000	94	:		Be a little more intelligent about when to perform the QIO.		
0000	95	:				
0000	96	:	V03-014	PCG0013	Peter George	27-Jun-1983
0000	97	:		Add /CLI qualifier.		
0000	98	:		Use event flags more consistently.		
0000	99	:		Remove code to propagate old format logical name tables.		
0000	100	:		Specify STSFLG argument to \$CREPRC.		
0000	101	:		Do an IOSM TT_PROCESS set mode QIO whenever an interactive		
0000	102	:		process comes back to life.		
0000	103	:		Remove CTRL/C and out-of-band AST warning message.		
0000	104	:				
0000	105	:	V03-013	PCG0012	Peter George	27-Jun-1983
0000	106	:		Change MOVW in logical name code to MOVL.		
0000	107	:				
0000	108	:	V03-012	PCG0011	Peter George	15-Jun-1983
0000	109	:		Add SPAWN/NOKEYPAD.		
0000	110	:				
0000	111	:	V03-011	PCG0010	Peter George	27-May-1983
0000	112	:		Use DCL\$FORMMSG for logical name message.		
0000	113	:				
0000	114	:	V03-010	RAS0157	Ron Schaefer	27-May-1983

0000	115	:	Add support for new logical names to be passed thru
0000	116	:	to the subprocess.
0000	117	:	
0000	118	:	V03-009 PCG0009 Peter George 27-May-1983
0000	119	:	Fix bug in SPAWN CTRL/Y processing.
0000	120	:	Use DCL\$FORMMSG.
0000	121	:	
0000	122	:	V03-008 PCG0008 Peter George 20-Apr-1983
0000	123	:	Add prompt string and keypad state to context that is
0000	124	:	passed through to the subprocess.
0000	125	:	Make spawn work in command procedures.
0000	126	:	
0000	127	:	V03-007 PCG0005 Peter George 30-Mar-1983
0000	128	:	Have SPAWN/NOTIFY use SPWN_T_PROCESS.
0000	129	:	Fix bug in ATTACH_AST that allows illegal attaches.
0000	130	:	Correctly signal process creation errors.
0000	131	:	Process parsed input/output devices as PPF devices.
0000	132	:	
0000	133	:	V03-006 PCG0005 Peter George 29-Mar-1983
0000	134	:	Sort out CTX_C_KEY* symbols.
0000	135	:	
0000	136	:	V03-005 PCG0004 Peter George 01-Mar-1983
0000	137	:	Call DCL\$GETNVAL. Propagate keypad symbols to subprocess.
0000	138	:	
0000	139	:	V03-004 PCG0003 Peter George 01-Feb-1983
0000	140	:	Clean up code, fix bugs.
0000	141	:	Add /SYMBOLS, /LOGICAL_NAMES, /NOTIFY.
0000	142	:	
0000	143	:	V03-003 PCG0002 Peter George 14-Jan-1983
0000	144	:	Fix SPWN deallocation problem.
0000	145	:	
0000	146	:	V03-002 PCG0001 Peter George 18-Nov-1982
0000	147	:	Store size of allocated TMBX structure in that structure.
0000	148	:	
0000	149	:	V03-001 KDM0002 Kathleen D. Morse 28-Jun-1982
0000	150	:	Added \$IODEF, \$PSLDEF and \$SSDEF.
0000	151	:	
0000	152	:---	

```

0000 154 :
0000 155 : MACRO LIBRARY CALLS
0000 156 :
0000 157 $PRCDEF ;DEFINE CREPRC FLAGS
0000 158 PRCDEF ;DEFINE PROCESS WORK AREA
0000 159 WRKDEF ;DEFINE COMMAND WORK AREA
0000 160 PTRDEF ;DEFINE TOKEN DESCRIPTORS
0000 161 SYMDEF ;DEFINE SYMBOL TABLE ENTRY
0000 162 SPWNDEF ;DEFINE SPAWN LOCAL STORAGE
0000 163 CTXDEF ;DEFINE PROCESS CONTEXT MESSAGES
0000 164 TMBXDEF ;DEFINE TERMINATION MAILBOX STRUCTURE
0000 165 $BRKDEF ;DEFINE BREAKTHRU CLASSES
0000 166 $PPDDEF ;DEFINE PPD FIELDS
0000 167 $DIBDEF ;DEFINE GETDEV INFO BUFFER
0000 168 $JPIDEF ;DEFINE GETJPI ITEM CODES
0000 169 $DVIDEF ;DEFINE GETDVI ITEM CODES
0000 170 $SYIDEF ;DEFINE GETSYI ITEM CODES
0000 171 $PQLDEF ;DEFINE PROCESS QUOTA TYPE CODES
0000 172 $PSLDEF ;DEFINE PROGRAM STATUS LONGWORD FIELDS
0000 173 $IODEF ;DEFINE I/O FUNCTION CODES
0000 174 $SSDEF ;DEFINE SYSTEM STATUS CODES
0000 175 $LOGDEF ;DEFINE LOGICAL NAME ENTRY
0000 176 $LNMDEF ;DEFINE LOGICAL NAME ATTRIBUTES
0000 177 $LNMSTRDEF ;DEFINE LOGICAL NAME STRUCTURES
0000 178 $ACCDDEF ;TERMINATION RECORD FORMAT
0000 179 $CLMSGDEF ;CLI MESSAGE CODES
0000 180 $DEVDEF ;DEFINE DEVICE CHARACTERISTICS
0000 181 $FABDEF ;DEFINE FAB FIELDS
0000 182 $NAMDEF ;DEFINE NAM FIELDS
0000 183 $TT2DEF ;DEFINE DEVDEPEND2 FIELDS
0000 184 $DCDEF ;DEFINE DVI DEVICE CLASSES
0000 185 $$CLITABDEF ;DEFINE MAX PROMPT SIZE
0000 186
00000000 187 .PSECT DCL$ZCODE,BYTE,RD,NOWRT
0000 188
00000010 0000 189 ATTMBX_MAXMSG = 16 ; MAXIMUM SIZE OF ATTACH REQUEST MESSAGE
0000 190
0000 191 LOGINOUT:
0000 192 .ASCIC 'SYS$SYSTEM:LOGINOUT' ; IMAGE TO INITIALIZE DCL
3A 4D 45 54 53 59 53 24 53 59 53 00' 0000
54 55 4F 4E 49 47 4F 4C 000C
13 0000
~ 0014
4C 55 21 5F 53 41 21 00' 0014
07 0014
001C 196 ATTACH_NAME:
5F 48 43 41 54 54 41 24 4C 43 44 00' 001C 197 .ASCIC 'DCL$ATTACH_!XL' ; FAO STRING FOR ATTACH LOGNAME
4C 58 21 0028
0E 001C
002B 198 OUTPUT_NAME:
5F 54 55 50 54 55 4F 24 4C 43 44 00' 002B 199 .ASCIC 'DCL$OUTPUT_!XL' ; FAO STRING FOR OUTPUT LOGNAME
4C 58 21 0037
0E 002B
003A 200
003A 201 SYSS$INPUT:
54 55 50 4E 49 24 53 59 53 00' 003A 202 .ASCIC 'SYS$INPUT' ; DEFAULT INPUT STREAM
09 003A

```

54 55 50 54 55 4F 24 53 59 53 00'	0044	203	SYSS\$OUTPUT:		
0A	0044	204	.ASCIC	'SYSS\$OUTPUT'	; DEFAULT OUTPUT STREAM
3A 4D 45 54 53 59 53 24 53 59 53 00'	004F	205	SYSS\$SYSTEM:		
0B	004F	206	.ASCIC	'SYSS\$SYSTEM:'	; DEVICE SPEC. FOR CLI SPECIFICATION
3A 4C 4E 5F 00'	005B	207	NL:		
04	005B	208	.ASCIC	'_NL:'	; NULL DEVICE
4D 4F 43 2E 00'	0060	209	COM:		
04	0060	210	.ASCIC	'_COM'	; DEFAULT INPUT FILE TYPE
47 4F 4C 2E 00'	0065	211	LOG:		
04	0065	212	.ASCIC	'_LOG'	; DEFAULT OUTPUT FILE TYPE
4C 43 44 00'	006A	213	CLI_NAME:		
03	006A	214	.ASCIC	'DCL'	; KNOWN CLI NAME TABLE.
52 43 4D 00'	006E	215	.ASCIC	'MCR'	; CLI NAME FOR DCL
03	006E	216	.ASCIC	'SHELL'	; CLI NAME FOR MCR
4C 4C 45 48 53 00'	0072	217	.ASCIC	'SHELL'	; CLI NAME FOR SHELL
05	0072	218	.BYTE	0	; END OF TABLE MARKER
00	0078	219			
73 65 63 6F 72 70 62 75 53 07 07 00'	0079	220	NOTIFY_MSG:		
6F 63 20 73 61 68 20 43 41 21 20 73	0085	221	.ASCIC	'Subprocess !AC has completed'	
64 65 74 65 6C 70 6D	0091				
1E	0079				
0000002C	0098	222		NOTIFY_LEN = 28+15+1	

```

0098 224 .SBTTL SPAWN COMMAND
0098 225
0098 226 :+ DCL$SPAWN - SPAWN COMMAND
0098 227
0098 228 : THIS ROUTINE IS CALLED TO EXECUTE THE DCL SPAWN COMMAND. THE SPAWN
0098 229 : COMMAND CREATES A "CLONED" SUBPROCESS WITH THE FOLLOWING CONTEXT COPIED
0098 230 : FROM THE PARENT TO THE SUBPROCESS:
0098 231
0098 232 : 1) ALL CLI SYMBOLS
0098 233 : 2) ALL PROCESS LOGICAL NAMES
0098 234 : 3) DEFAULT DISK AND DIRECTORY
0098 235 : 5) CURRENT PROCESS PRIVILEGES
0098 236 : 6) CURRENT COMMAND VERIFICATION STATE
0098 237 : 7) CURRENT "ON CONTROL" (OUT-OF-BAND) STATE
0098 238 : 8) CURRENT PROMPT STRING
0098 239 : 9) CURRENT KEYPAD STATE
0098 240
0098 241 : NO PROCESS PERMANENT OPEN FILES ARE COPIED, NOR IS ANY IMAGE OR PROCEDURE
0098 242 : CONTEXT. THE SUBPROCESS IS SET TO PROCEDURE LEVEL 0. LOGIN.COM IS NOT
0098 243 : EXECUTED, BOTH BECAUSE THE CONTEXT IS COPIED SEPARATELY AND TO CAUSE THE
0098 244 : SUBPROCESS TO INITIALIZE QUICKLY. THE PARENT IS LEFT IN HIBERNATION STATE
0098 245 : UNTIL THE SUBPROCESS TERMINATES OR TRANSFERS CONTROL BACK TO THE PARENT VIA
0098 246 : THE ATTACH COMMAND.
0098 247
0098 248 : INPUTS:
0098 249
0098 250 : R10 = ADDRESS OF COMMAND WORK AREA
0098 251 : R11 = ADDRESS OF PROCESS WORK AREA
0098 252
0098 253 : OUTPUTS:
0098 254
0098 255 : R0 = STATUS CODE
0098 256
0098 257 : R2-R9 DESTROYED.
0098 258
0098 259 :-
0098 260
0098 261 DCL$SPAWN::
0098 262
0098 263 : ALLOCATE SOME SPACE FOR SPAWN STORAGE
0098 264
0098 265 : MOVZWL #SPWN_C_LENGTH,R1 : LENGTH OF STORAGE TO ALLOCATE
0098 266 : BSBW DCL$[CLDYNMEM : ALLOCATE STORAGE
0098 267 : BLBS R0,10$ : BRANCH IF OK
0098 268 : RSB : IF ERROR DETECTED, REPORT IT
0098 269 10$: MOVL R2,R6 : POINT TO SPWN STORAGE
0098 270 : MOVCS #0,(R1),#0,#SPWN_C_LENGTH,(R6) : ZERO THE BLOCK
0098 271 : : (WITHOUT DESTROYING R1)
0098 272 : MOVW R1,SPWN_W_SIZE(R6) : STORE SIZE OF BLOCK
0098 273 : BISW #SPWN_M_LOG!SPWN_M_WAIT!- : ASSUME /LOG, /WAIT
0098 274 : SPWN_M_CLISYM!SPWN_M_LOGNAM!- : COPY CLI SYMBOLS & LOGNAMES
0098 275 : SPWN_M_KEYPAD,SPWN_W_FLAGS(R6)
0098 276 : BICW #SPWN_M_CLI!SPWN_M_TABLE,- : ASSUME /NOCLI AND /NOTABLE
0098 277 : SPWN_D_FLAGS(R6)
0098 278 : MOVW DCL$[CREF,SPWN_W_PMPTCTRL(R6) : ASSUME /CONTROL
0098 279 : MNEGB #1,SPWN_B_EFNTR6) : DO NOT SET EVENT FLAG ON TERMINATION
0098 280

```

```

00CC 281 :
00CC 282 : PROCESS THE VERB QUALIFIERS ON THE COMMAND LINE
00CC 283 :
FF31' 30 00CC 284 40$: BSBW DCL$GETDVAL : GET NEXT TOKEN
10 50 E9 00CF 285 BLBC R0,43$ : IF EOL, END OF PROCESSING
00 55 D1 00D2 286 CMPL R5,#PTR_K_COMDQUAL : VERB QUALIFIER?
0E 13 00D5 287 BEQL 42$ : BRANCH IF SO
03 55 D1 00D7 288 CMPL R5,#PTR_K_PARAMETR : PARAMETER (COMMAND STRING)?
F0 12 00DA 289 BNEQ 40$ : IF NOT, IGNORE IT
30 A6 51 7D 00DC 290 MOVQ R1,SPWN_Q_CMDSTR(R6) : SAVE DESCRIPTOR OF COMMAND STR'NG
EA 11 00E0 291 BRB 40$
0171' 31 00E2 292 43$: : END OF COMMAND PARSING
FF18' 30 00E5 293 42$: BSBW DCL$GETNVAL : GET QUALIFIER NUMBER
00'8F 51 91 00E8 294 CMPB R1,#CLISK_SPAW_WAIT : /WAIT?
27 13 00EC 295 BEQL 49$
00'8F 51 91 00EE 296 CMPB R1,#CLISK_SPAW_LOG : /LOG?
2F 13 00F2 297 BEQL 47$
00'8F 51 91 00F4 298 CMPB R1,#CLISK_SPAW_SYMB : /SYMBOLS?
37 13 00F8 299 BEQL 50$
00'8F 51 91 00FA 300 CMPB R1,#CLISK_SPAW_LOGI : /LOGICAL_NAMES?
3F 13 00FE 301 BEQL 51$
00'8F 51 91 0100 302 CMPB R1,#CLISK_SPAW_NOTI : /NOTIFY?
49 13 0104 303 BEQL 52$
00'8F 51 91 0106 304 CMPB R1,#CLISK_SPAW_CARR : /CARRIAGE_CONTROL?
53 13 010A 305 BEQL 531$
00'8F 51 91 010C 306 CMPB R1,#CLISK_SPAW_KEYP : /KEYPAD?
60 13 0110 307 BEQL 532$
006D 31 0112 308 BRW 60$ : CHECK OTHER QUALIFIER POSSIBILITIES
0115 309
0115 310 49$: SETBIT SPWN_V_WAIT,SPWN_W_FLAGS(R6) : ASSUME /WAIT
AF 53 00 E1 0119 311 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,40$ : IGNORE IF NOT /NOWAIT
A9 11 011D 312 CLRBIT SPWN_V_WAIT,SPWN_Q_FLAGS(R6) : CLEAR WAIT FLAG
0121 313 BRB 40$
0123 314 47$: SETBIT SPWN_V_LOG,SPWN_W_FLAGS(R6) : ASSUME /LOG
A1 53 00 E1 0127 315 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,40$ : IGNORE IF NOT /NOLOG
012B 316 CLRBIT SPWN_V_LOG,SPWN_W_FLAGS(R6) : CLEAR LOG MESSAGE FLAG
9B 11 012F 317 BRB 40$
0131 318 50$: SETBIT SPWN_V_CLISYM,SPWN_W_FLAGS(R6) : ASSUME /SYMBOLS
93 53 00 E1 0135 319 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,40$ : IGNORE IF NOT /NOSYMBOLS
0139 320 CLRBIT SPWN_V_CLISYM,SPWN_Q_FLAGS(R6) : CLEAR CLISYM FLAG
8D 11 013D 321 BRB 40$
013F 322 51$: SETBIT SPWN_V_LOGNAM,SPWN_W_FLAGS(R6) : ASSUME /LOGICAL_NAMES
5E 53 00 E1 0144 323 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ : IGNORE IF NOT /NOLOGICAL_NAMES
0148 324 CLRBIT SPWN_V_LOGNAM,SPWN_Q_FLAGS(R6) : CLEAR LOGNAM FLAG
57 11 014D 325 BRB 41$
014F 326 52$: SETBIT SPWN_V_NOTIFY,SPWN_W_FLAGS(R6) : ASSUME /NOTIFY
4E 53 00 E1 0154 327 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ : IGNORE IF NOT /NONOTIFY
0158 328 CLRBIT SPWN_V_NOTIFY,SPWN_Q_FLAGS(R6) : CLEAR NOTIFY FLAG
47 11 015D 329 BRB 41$
00A3 C6 00000000'EF B0 015F 330 531$: MOVW DCL$CRLF,SPWN_W_PMPTCTRL(R6) : ASSUME /CONTROL
3A 53 00 E1 0168 331 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ : IGNORE IF NOT /NOCONTROL
00A3 C6 B4 016C 332 CLRW SPWN_W_PMPTCTRL(R6) : SET /NOCONTROL
34 11 0170 333 BRB 41$
0172 334 532$: SETBIT SPWN_V_KEYPAD,SPWN_W_FLAGS(R6) : ASSUME /KEYPAD
2B 53 00 E1 0177 335 BBC #PTR_V_NEGATE-PTR_V_FLAGS,R3,41$ : IGNORE IF NOT /NOKEYPAD
017B 336 CLRBIT SPWN_V_KEYPAD,SPWN_Q_FLAGS(R6) : CLEAR KEYPAD FLAG
24 11 0180 337 BRB 41$

```

00'8F	51	91	0182	338	60\$:	CMPB	R1,#CLISK_SPAW_INPU	:	/INPUT?
	21	13	0182	339		BEQL	45\$		
00'8F	51	91	0186	340		CMPB	R1,#CLISK_SPAW_OUTP	:	/OUTPUT?
	33	13	0188	341		BEQL	46\$		
00'8F	51	91	018C	342		CMPB	R1,#CLISK_SPAW_PROG	:	/PROCESS_NAME?
	45	13	0192	343		BEQL	48\$		
00'8F	51	91	0194	344		CMPB	R1,#CLISK_SPAW_CLI	:	/CLI?
	51	13	0198	345		BEQL	481\$		
00'8F	51	91	019A	346		CMPB	R1,#CLISK_SPAW_TABL	:	/TABLE?
	63	13	019E	347		BEQL	485\$		
00'8F	51	91	01A0	348		CMPB	R1,#CLISK_SPAW_PROM	:	/PROMPT?
	75	13	01A4	349		BEQL	53\$		
	FF23	31	01A6	350	41\$:	BRW	40\$	:	IF NONE OF THE ABOVE QUALS, IGNORE IT
			01A9	351					
			01A9	352	45\$:	CLRBIT	SPWN_V_INPUT,SPWN_W_FLAGS(R6)	:	ASSUME NO VALUE PRESENT
02	54	91	01AE	353		CMPB	R4,#PTR_K_COLON	:	IS A VALUE PRESENT?
	F3	12	01B1	354		BNEQ	41\$	:	IF NOT, USE DEFAULT
			01B3	355		SETBIT	SPWN_V_INPUT,SPWN_W_FLAGS(R6)	:	INDICATE VALUE PRESENT
	FE45'	30	01B8	356		BSBW	DCL\$GETDVAL	:	GET /INPUT VALUE
20	A6	7D	01BB	357		MOVQ	R1,SPWN_Q_INPUT(R6)	:	USE EXPLICIT INPUT STREAM
	E5	11	01BF	358		BRB	41\$		
			01C1	359	46\$:	CLRBIT	SPWN_V_OUTPUT,SPWN_W_FLAGS(R6)	:	ASSUME NO VALUE PRESENT
02	54	91	01C6	360		CMPB	R4,#PTR_K_COLON	:	IS A VALUE PRESENT?
	DB	12	01C9	361		BNEQ	41\$	:	IF NOT, USE DEFAULT
			01CB	362		SETBIT	SPWN_V_OUTPUT,SPWN_W_FLAGS(R6)	:	INDICATE VALUE PRESENT
	FE2D'	30	01D0	363		BSBW	DCL\$GETDVAL	:	GET /OUTPUT VALUE
28	A6	7D	01D3	364		MOVQ	R1,SPWN_Q_OUTPUT(R6)	:	USE EXPLICIT OUTPUT STREAM
	CD	11	01D7	365		BRB	41\$		
	02	54	01D9	366	48\$:	CMPB	R4,#PTR_K_COLON	:	IS A VALUE PRESENT?
	C8	12	01DC	367		BNEQ	41\$	:	IF NOT, USE DEFAULT
	FE1F'	30	01DE	368		BSBW	DCL\$GETDVAL	:	GET VALUE
18	A6	7D	01E1	369		MOVQ	R1,SPWN_Q_PRCNAM(R6)	:	USE EXPLICIT PROCESS NAME
			01E5	370		SETBIT	SPWN_V_PRCNAM,SPWN_W_FLAGS(R6)	:	SET PROCESS NAME SPECIFIED
	BB	12	01E9	371		BNEQ	41\$		
			01EB	372	481\$:	CLRBIT	SPWN_V_CLI,SPWN_W_FLAGS(R6)	:	ASSUME /NOCLI
B2	53	E0	01F0	373		BBS	#PTR_V_NEGATE-PTR_V_FLAGS,R3,41\$	:	FLAG OK AS IS IF /NOCLI
			01F4	374		SETBIT	SPWN_V_CLI,SPWN_W_FLAGS(R6)	:	SET /CLI PRESENT FLAG
	FE04'	30	01F9	375		BSBW	DCL\$GETDVAL	:	GET /CLI VALUE
00C6	C6	7D	01FC	376		MOVQ	R1,SPWN_Q_CLI(R6)	:	SAVE DESCRIPTOR OF CLI NAME
	A3	11	0201	377		BRB	41\$		
			0203	378	485\$:	CLRBIT	SPWN_V_TABLE,SPWN_W_FLAGS(R6)	:	ASSUME /NOTABLE
9A	53	E0	0208	379		BBS	#PTR_V_NEGATE-PTR_V_FLAGS,R3,41\$	:	FLAG OK AS IS IF /NOTABLE
			020C	380		SETBIT	SPWN_V_TABLE,SPWN_W_FLAGS(R6)	:	SET /TABLE PRESENT FLAG
	FDEC'	30	0211	381		BSBW	DCL\$GETDVAL	:	GET /TABLE VALUE
00CE	C6	7D	0214	382		MOVQ	R1,SPWN_Q_TABLE(R6)	:	SAVE DESCRIPTOR OF TABLE NAME
	8B	11	0219	383		BRB	41\$		
			021B	384	53\$:	CLRBIT	SPWN_V_PROMPT,SPWN_W_FLAGS(R6)	:	ASSUME /NOPROMPT
	30	53	0220	385		BLBS	R3,53\$	:	BRANCH IF SO
			0223	386		SETBIT	SPWN_V_PROMPT,SPWN_W_FLAGS(R6)	:	SET /PROMPT
	00'8F	90	0228	387		MOVB	#DCL\$C_PROMPTLEN,-	:	ASSUME NO VALUE SPECIFIED
	00A2	C6	022B	388			SPWN_B_PROMPTLEN(R6)	:	
		DD	022E	389		PUSHL	R4	:	SAVE THE TERMINATOR
00000000'EF	0000'8F	28	0230	390		MOVCS	#DCL\$C_PROMPTLEN,DCL\$CRLF,-	:	GET PROMPT STRING
	00A3	C6	0239	391			SPWN_W_PMPCTRL(R6)	:	
		8ED0	023C	392		POPL	R4	:	RESTORE THE TERMINATOR
	02	54	023F	393		CMPB	R4,#PTR_K_COLON	:	IS A VALUE PRESENT?
		91		394					

```

00A2 C6      0F 12 0242 395      BNEQ 533$      ; USE DEFAULT PROMPT
              FDB9' 30 0244 396      BSBW DCL$GETDVAL ; GET VALUE
              51 03 81 0247 397      ADDB3 #3,R1,SPWN_B PROMPTLEN(R6) ; GET PROMPT LENGTH
              62 20 28 024D 398      MOV C3 #ENT_K_MAX-PROMPT,(R2),- ; GET PROMPT STRING
              00A6 C6 0250 399      SPWN_G-PROMPT(R6) ;
              FE76 31 0253 400 533$: BRW 40$
              0256 401
              0256 402 ;
              0256 403 ; THE PARSING IS COMPLETE, AND ALL OPTIONS ARE SET INTO THE SPWN
              0256 404 ; BLOCK. NOW PERFORM THE SPAWN OPERATION.
              0256 405 ;
              07 10 0256 406 80$: BSBW DCL$SPAWN2 ; PERFORM THE SPAWN OPERATION
              03 50 E9 0258 407      BLBC R0, 90$ ; TEST FOR FAILURE OF SPAWN ITSELF
              50 51 D0 025B 408      MOVL R1, R0 ; GET SUBPROCESS TERMINATION STATUS
              05 025E 409 90$: RSB

```

```

025F 411 .SBTTL SPAWN A SUBPROCESS
025F 412 ---
025F 413 :
025F 414 THIS ROUTINE IS CALLED TO PERFORM THE ACTUAL CREATION OF THE SUBPROCESS.
025F 415 :
025F 416 INPUTS:
025F 417 :
025F 418 R6 = ADDRESS OF SPWN PARAMETER BLOCK
025F 419 R11 = ADDRESS OF PRC AREA
025F 420 :
025F 421 OUTPUTS:
025F 422 :
025F 423 R0 = STATUS
025F 424 R1 = FINAL STATUS FROM COMPLETED SUBPROCESS
025F 425 SPWN_L_PID(R6) = PID OF SUBPROCESS (IF NOWAIT FLAG SPECIFIED)
025F 426 ---
025F 427 :
025F 428 DCL$SPAWN2::
01BC 8F BB 025F 429 PUSHF #^M<R2,R3,R4,R5,R7,R8> ; SAVE REGISTERS
0263 430 :
0263 431 :
0263 432 CONTROL/Y AST'S ARE DISABLED THROUGHOUT THIS COMMAND, TO ENSURE THAT
0263 433 MAILBOXES WHICH ARE CREATED HERE ARE CORRECTLY DELETED, ETC. AS A
0263 434 RESULT, WE MUST PERIODICALLY CHECK THE HANGUP FLAG IN CASE A HANGUP
0263 435 IS DETECTED WHILE WE ARE OPERATING.
0263 436 :
00B4 CB D0 0263 437 MOVL PRC_L_OUTOFBAND(R11),- ; SAVE OUT-OF-BAND ENABLE MASK
58 A6 0267 438 SPWN_L_OUTOFBAND(R6) ;
51 D4 0269 439 CLRL R1 ; DISABLE ALL OUT-OF-BAND AST'S
FD92' 30 026B 440 BSBW DCL$RESET00B ;
026E 441 :
026E 442 :
026E 443 PRE-CLEAR THE TERMINATION EVENT FLAG, IF ONE IS SPECIFIED.
026E 444 :
OF A6 95 026E 445 TSTB SPWN_B_EFN(R6) ; EVENT FLAG SPECIFIED?
OA 19 0271 446 BLSS 2$ ; BRANCH IF NONE
0273 447 $CLREF_S EFN=SPWN_B_EFN(R6) ; PRE-CLEAR IT
027D 448 :
027D 449 :
027D 450 :
027D 451 SAVE THE PREVIOUS ACCESS MODE, FOR PROBING RETURN STATUS AND DELIVERING
027D 452 TERMINATION ASTS.
027D 453 2$:
027F 454 MOVPSL R0 ; GET CURRENT PSL
0282 455 EXTZV #PSL$V_PRVMOD,#PSL$S_PRVMOD,- ; EXTRACT PREVIOUS ACMODE
0284 456 MOVB R0,SPWN_B_ACMODE(R6) ; SAVE THE ACCESS MODE
0288 457 :
0288 458 :
0288 459 ALLOCATE SOME SCRATCH STORAGE ON THE STACK (1 - 1/2 PAGES)
0288 460 :
5E FD00 CE 9E 0288 461 MOVAB -768(SP),SP ; ALLOCATE SCRATCH BUFFER ON STACK
7E 0300 5E DD 028D 462 PUSHL SP ; CONSTRUCT DESCRIPTOR OF BUFFER
58 5E D0 028F 463 MOVZWL #768,-(SP) ;
0294 464 MOVL SP,R8 ; AND POINT TO IT
0297 465 :
0297 466 :
0297 467 FIND A TERMINATION MAILBOX SLOT AND INITIALIZE IT FOR THIS SUBPROCESS.

```

```

0297 468 :
0297 469 :
57 74 AB 9E 0297 470 : ASSUME TMBX_L LINK EQ 0
57 67 D0 0297 471 10$: MOVAB PRC [TMBX(R11),R7 : GET ADDRESS OF TMBX BLOCK LIST HEA
09 12 0297 472 : MOVL TMBX_C LINK(R7),R7 : GET NEXT BLOCK IN LIST
OACE 30 029E 473 : BNEQ 20$ : BRANCH IF NOT END OF LIST
03 50 E8 02A0 474 : BSBW CREATE_TMBX : CREATE NEW MBX IF END OF LIST
00FA 31 02A3 475 : BLBS R0,20$ : CONTINUE IF NO ERROR
04 08 A7 91 02A6 476 20$: BRW 230$ : EXIT IF ERROR
EC 13 02A9 477 : CMPB TMBX_B_REFS(R7),#TMBX_C_MAXREFS : DOES MBX HAVE ANY OPEN SLOTS?
08 A7 96 02AD 478 : BEQL 10$ : NO, LOOK AT NEXT MBX IN LIST
08 A6 04 A7 B0 02AF 479 : INCB TMBX_B_REFS(R7) : YES, GRAB ONE, INCR REFERENCE COUN
02B2 480 : MOVW TMBX_W_UNIT(R7),SPWN_W_UNIT(R6) : COPY THE UNIT NUMBER
02B7 481 :
02B7 482 : CHECK FOR KNOWN CLI'S TO WHICH TO PASS CONTEXT IF /CLI
02B7 483 : WAS SPECIFIED.
02B7 484 :
2F 0C 7E 58 7D 02B7 485 : MOVQ R8,-(SP) : SAVE WORK REGISTERS
52 00C6 C6 7D 02BA 486 : BBC #SPWN_V_CLI,SPWN_W_FLAGS(R6),50$ : BRANCH IF NO CLI WAS SPECIFIED
58 FDA2 CF 9E 02BF 487 : MOVQ SPWN_Q_CLI(R6),R2 : GET CLI DESCRIPTOR
02C9 488 : MOVAB CLI_NAME,R8 : GET TABLE ADDR. OF KNOWN CLI'S
59 68 9A 02C9 489 21$: MOVZBL (R8),R9 : EXTRACT LENGTH
16 13 02CC 490 : BEQL 40$ : IF EQ, SPECIFIED CLI NOT KNOWN
02CE 491 :
00 7E 52 7D 02CE 492 : MOVQ R2,-(SP) : SAVE DESCRIPTOR
01 63 52 2D 02D1 493 : CMPC5 R2,(R3),#0,- : COMPARE TO THIS TABLE ENTRY
A8 59 02D5 494 : R9,1(R8)
11 13 02D8 495 : BEQL 45$ : IF EQ, THIS CLI IS KNOWN
02DA 496 :
58 52 8E 7D 02DA 497 : MOVQ (SP)+,R2 : RESTORE DESCRIPTOR
01 A849 9E 02DD 498 : MOVAB 1(R8)[R9],R8 : SET UP TO CHECK NEXT ENTRY
E5 11 02E2 499 : BRB 21$ : CHECK NEXT TABLE ENTRY
02E4 500 :
02E4 501 40$: SETBIT SPWN_V_NOCTX,SPWN_W_FLAGS(R6) : DON'T SEND CONTEXT TO FOR THIS CLI
52 03 11 02E9 502 : BRB 50$
02EB 503 45$: MOVQ (SP)+,R2 : RESTORE STACK
02EE 504 :
02EE 505 : PROPAGATE EITHER THE SPECIFIED CLI AND TABLE NAMES OR THE DEFAULT NAMES TO
02EE 506 : THE SPAWN CLI AND TABLE SPECIFICATION TABLES.
02EE 507 :
58 00C6 C6 3C 02EE 508 50$: MOVZWL SPWN_Q_CLI(R6),R8 : GET LENGTH OF CLI NAME, IF SPEC.
59 00CE C6 3C 02F3 509 : MOVZWL SPWN_Q_TABLE(R6),R9 : GET LENGTH OF TABLE NAME, IF SPECI
40 0C A6 0F E1 02F8 510 : BBC #SPWN_Q_TABLE,SPWN_W_FLAGS(R6),218$ : SKIP IF NO TABLE SPECIFIED
24 0C A6 0D E1 02FD 511 : BBC #SPWN_V_CLI,SPWN_W_FLAGS(R6),216$ : SKIP IF NO CLI SPECIFIED
0302 512 :
0302 513 : BOTH CLI AND NAME WERE SPECIFIED
0302 514 :
00000000'GF 58 90 0302 515 : MOVB R8,G^CTL$GT SPAWNCLI : SET SIZE FIELD
00CA D6 58 28 0309 516 : MOVG R8,@SPWN_Q_CLI+4(R6),- : MOVE CLI SPECIFICATION
00000001'GF 030E 517 : G^CTL$GT-SPAWNCLI+1
00000000'GF 59 90 0313 518 214$: MOVB R9,G^CTL$GT SPAWNTABLE : SET LENGTH OF TABLE NAME
00D2 D6 59 28 031A 519 : MOVG R9,@SPWN_Q_TABLE+4(R6),- : COPY TABLE NAME
00000001'GF 031F 520 : G^CTL$GT-SPAWNTABLE+1
5F 11 0324 521 : BRB 228$
0326 522 :
0326 523 : JUST TABLE NAME SPECIFIED
0326 524 :

```

```

58 00000000'GF 9A 0326 525 216$: MOVZBL G^CTL$GT_CLINAME,R8 ; GET LENGTH OF CLI NAME
                                58 D6 032D 526 ; INCL R8 ; INCLUDE LENGTH BYTE
00000000'GF 58 28 032F 527 ; MOV R8,G^CTL$GT_CLINAME,- ; GET DEFAULT CLI NAME
00000000'GF 58 0336 528 ; G^CTL$GT_SPAWNCLI
D6 11 033B 529 ; BRB 214$ ; NOW TRANSER TABLE NAME
033D 530 ;
033D 531 ; NO TABLE NAME SPECIFIED. CHECK IF CLI NAME SPECIFIED
033D 532 ;
19 0C A6 OD E1 033D 533 218$: BBC #SPWN_V_CLI,SPWN_W_FLAGS(R6),220$ ; SKIP IF CLI NOT SPECIFIED
00000000'GF 58 94 0342 534 ; CLRB G^CTL$GT_SPAWNTABLE ; ZERO TABLE SPECIFICATION
00000000'GF 58 90 0348 535 ; MOV R8,G^CTL$GT_SPAWNCLI ; SET SIZE FIELD
00CA D6 58 28 034F 536 ; MOV R8,@SPWN_Q_CLI+4(R6),- ; MOVE CLI SPECIFICATION
00000001'GF 58 0354 537 ; G^CTL$GT_SPAWNCLI+1
2A 11 0359 538 ; BRB 228$
035B 539 ;
035B 540 ; NEITHER CLI OR TABLE SPECIFIED. JUST COPY DEFAULTS
035B 541 ;
58 00000000'GF 9A 035B 542 220$: MOVZBL G^CTL$GT_CLINAME,R8 ; GET LENGTH OF CLI NAME
                                58 D6 0362 543 ; INCL R8 ; INCLUDE LENGTH BYTE
00000000'GF 58 28 0364 544 ; MOV R8,G^CTL$GT_CLINAME,- ; COPY CLI NAME
00000000'GF 58 036B 545 ; G^CTL$GT_SPAWNCLI
58 00000000'GF 9A 0370 546 ; MOVZBL G^CTL$GT_TABLENAME,R8 ; GET LENGTH OF TABLE NAME
                                58 D6 0377 547 ; INCL R8 ; INCLUDE LENGTH BYTE
00000000'GF 58 28 0379 548 ; MOV R8,G^CTL$GT_TABLENAME,- ; COPY TABLE NAME
00000000'GF 58 0380 549 ; G^CTL$GT_SPAWNTABLE
58 8E 7D 0385 550 228$: MOVQ (SP)+,R8 ; RESTORE WORK REG.
0388 551 ;
0388 552 ; CREATE A COMMUNICATIONS MAILBOX TO COPY THE CONTEXT TO THE SUBPROCESS
0388 553 ;
SF 0C A6 OE E0 0388 554 ; BBS #SPWN_V_NOCTX,SPWN_W_FLAGS(R6),22$ ; BRANCH IF UNKNOWN CLI WAS SPECIF
038D 555 ; $CREMBX_S_CHAN=SPWN_W_CHAN(R6),- ; CREATE COMMUNICATIONS MAILBOX
038D 556 ; -MAXMSG=#CTX_C_MAXLEN
4C 50 E9 03A3 557 230$: BLBC R0,95$ ; BRANCH ON ERROR
03A6 558 ;
03A6 559 ; GET THE NAME OF THAT MAILBOX USING $GETDVI.
03A6 560 ;
03A6 561 ;
7E D4 03A6 562 ; CLRL -(SP) ; CREATE GETDVI ITEM LIST
10 A6 9F 03A8 563 ; PUSHAB SPWN_Q_MBXNAM(R6) ; ADDRESS OF WORD TO RECEIVE LENGTH
7E 68 7D 03AB 564 ; MOVQ (R8),-(SP) ; SET DESCRIPTOR OF BUFFER
02 AE 20 B0 03AE 565 ; MOVW #DVI$DEVNAM,2(SP) ; REQUEST COMPLETE DEVICE NAME
50 5E D0 03B2 566 ; MOVL SP,R0 ;
03B5 567 ; $GETDVIW S_CHAN=SPWN_W_CHAN(R6),- ; GET MAILBOX DEVICE NAME
03B5 568 ; IOSB=SPWN_Q_IOSB(R6),- ;
03B5 569 ; EFN=#EXE$C_SYSEFN,- ;
03B5 570 ; ITMLST=(R0) ;
5E 10 C0 03D1 571 ; ADDL #4*4,SP ; POP GETDVI ITEM LIST
18 50 E9 03D4 572 ; BLBC R0,95$ ; BRANCH IF ERROR DETECTED
50 38 A6 3C 03D7 573 ; MOVZWL SPWN_Q_IOSB(R6),R0 ; GET IOSB STATUS
14 50 E9 03DB 574 ; BLBC R0,95$ ; BRANCH IF ERROR DETECTED
03DE 575 ;
03DE 576 ;
03DE 577 ; SAVE THE NAME OF THE COMMUNICATIONS MAILBOX IN THE SPWN BLOCK.
03DE 578 ; DEALLOCATE SPACE FOR THE MAILBOX NAME FROM THE SCRATCH STORAGE.
03DE 579 ;
14 A6 04 A8 D0 03DE 580 ; MOVL 4(R8),SPWN_Q_MBXNAM+4(R6) ; SET ADDRESS OF BUFFER
68 10 A6 A2 03E3 581 ; SUBW SPWN_Q_MBXNAM(R6),(R8) ; MARK DEVICE NAME NO LONGER SCRATCH

```

```

04 A8 10 A6 C0 03E7 582 ADDL SPWN_Q_MBXNAM(R6),4(R8) ;
03EC 583
03EC 584 :
03EC 585 : CREATE AN ATTACH MAILBOX TO HANDLE RE-ATTACH REQUESTS TO THIS PROCESS.
03EC 586
08F3 30 03EC 587 22$: BSBW CREATE_ATTMBX ; CREATE OUR ATTACH MAILBOX
03 50 E8 03EF 588 RO,25$- ; BRANCH IF SUCCESS
02CD 31 03F2 589 95$: BRW SPAWN_EXIT ; BRANCH IF ERROR DETECTED
03F5 590
03F5 591 :
03F5 592 : IF /NOTIFY WAS SPECIFIED, CHECK THAT IT IS ALLOWED.
03F5 593
08 E1 03F5 594 25$: BBC #SPWN V NOTIFY,- ; SKIP IF /NONOTIFY
11 0C A6 03F7 595 SPWN_Q_FLAGS(R6),30$ ;
50 00038250 8F D0 03FA 596 MOVL #CLIS_NOTIFY,RO ; ASSUME NOTIFY NOT ALLOWED
EC 68 AB 06 E0 0401 597 BBS #PRC V MODE,PRC_W_FLAGS(R11),95$ ; NOT ALLOWED IN BATCH JOBS
02 E0 0406 598 BBS #SPWN V WAIT,- ; NOT ALLOWED IN /WAIT JOBS
E7 0C A6 0408 599 SPWN_Q_FLAGS(R6),95$ ;
040B 600
040B 601 :
040B 602 : CHECK THAT COMMAND STRING IS NOT TOO LONG.
040B 603
50 00038018 8F D0 040B 604 30$: MOVL #CLIS_BUFOVF,RO ; ASSUME COMMAND BUFFER OVERFLOW
0100 8F 30 A6 B1 0412 605 CMPW SPWN_Q_CMDSTR(R6),#WRK_C_INPBUFSIZ ; COMMAND STRING TOO LONG?
D8 1A 0418 606 BGTRU 95$ ; BRANCH IF SO
041A 607
041A 608 :
041A 609 : CHECK THAT PROMPT STRING IS NOT TOO LONG.
041A 610
50 000388FA 8F D0 041A 611 MOVL #CLIS_STRTOOLNG,RO ; ASSUME PROMPT IS TOO LONG
00A2 C6 91 0421 612 CMPB SPWN_B_PROMPTLEN(R6),- ; PROMPT STRING TOO LONG?
23 0425 613 #ENT_K_MAX_PROMPT+3 ;
CA 1A 0426 614 BGTRU 95$ ; BRANCH IF SO
0428 615
0428 616 :
0428 617 : PROCESS THE INPUT AND OUTPUT STREAMS.
0428 618
02E5 30 0428 619 BSBW VERIFY_INPUT ; VERIFY INPUT STREAM
C4 50 E9 042B 620 BLBC RO,95$- ; BRANCH IF ERROR
04BE 30 042E 621 BSBW VERIFY_OUTPUT ; VERIFY OUTPUT STREAM
BE 50 E9 0431 622 BLBC RO,95$- ; BRANCH IF ERROR
0434 623
0434 624 :
0434 625 : DETERMINE BASE PRIORITY FOR NEW SUBPROCESS FROM CURRENT BASE PRIORITY
0434 626
7E 7C 0434 627 CLRQ -(SP) ; CREATE GETJPI ITEM LIST
48 A6 DF 0436 628 PUSHAL SPWN_L_PRI8(R6) ; SET BUFFER ADDRESS
03090004 8F DD 0439 629 PUSHL #JPI$_PRI8@16+4 ; REQUEST CURRENT BASE PRIORITY
043F 630
043F 631 :
043F 632 : DETERMINE CURRENT IMAGE COUNT (FOR USE BY TERMINATION AST)
043F 633
7E D4 043F 634 CLRL -(SP) ; NO RETLEN ADDRESS
5C A6 DF 0441 635 PUSHAL SPWN_L_IMAGCNT(R6) ; SET BUFFER ADDRESS
041A0004 8F DD 0444 636 PUSHL #JPI$_IMAGECOUNT@16+4 ; REQUEST CURRENT IMAGE COUNT
044A 637
044A 638 :

```

```

044A 639 : DETERMINE CURRENT VALUES OF NON-DEDUCTIBLE QUOTAS, AND CREATE A QUOTA
044A 640 : LIST WHICH WILL BE USED TO SET THE QUOTAS OF THE SUBPROCESS.
044A 641 :
53 60 A6 9E 044A 642 MOVAB SPWN_G_QUOTAS(R6),R3 : GET ADDRESS OF SPACE FOR LIST
7E D4 044E 643 CLRL -(SP) : NO RETLEN ADDRESS
83 01 90 0450 644 MOVB #PQL$_ASTLM,(R3)+ : SET TYPE CODE IN PQL LIST
83 DF 0453 645 PUSHAL (R3)+ : SET BUFFER ADDRESS
04090004 8F DD 0455 646 PUSHHL #JPI$_ASTLM@16+4 : REQUEST CURRENT ASTLM
7E D4 045B 647 CLRL -(SP) : NO RETLEN ADDRESS
83 02 90 045D 648 MOVB #PQL$_BIOLM,(R3)+ : SET TYPE CODE IN PQL LIST
83 DF 0460 649 PUSHAL (R3)+ : SET BUFFER ADDRESS
03100004 8F DD 0462 650 PUSHHL #JPI$_BIOLM@16+4 : REQUEST CURRENT BIOLM
7E D4 0468 651 CLRL -(SP) : NO RETLEN ADDRESS
83 05 90 046A 652 MOVB #PQL$_DIOLM,(R3)+ : SET TYPE CODE IN PQL LIST
83 DF 046D 653 PUSHAL (R3)+ : SET BUFFER ADDRESS
03130004 8F DD 046F 654 PUSHHL #JPI$_DIOLM@16+4 : REQUEST CURRENT DIOLM
7E D 0475 655 CLRL -(SP) : NO RETLEN ADDRESS
83 0B 9 0477 656 MOVB #PQL$_WSDEFAULT,(R3)+ : SET TYPE CODE IN PQL LIST
83 D 047A 657 PUSHAL (R3)+ : SET BUFFER ADDRESS
04030004 8F DD 047C 658 PUSHHL #JPI$_DFWSCNT@16+4 : REQUEST CURRENT WSDEFAULT
7E D4 0482 659 CLRL -(SP) : NO RETLEN ADDRESS
83 0A 90 0484 660 MOVB #PQL$_WSQUOTA,(R3)+ : SET TYPE CODE IN PQL LIST
83 DF 0487 661 PUSHAL (R3)+ : SET BUFFER ADDRESS
04020004 8F DD 0489 662 PUSHHL #JPI$_WSQUOTA@16+4 : REQUEST CURRENT WSQUOTA
7E D4 048F 663 CLRL -(SP) : NO RETLEN ADDRESS
83 0D 90 0491 664 MOVB #PQL$_WSEXTENT,(R3)+ : SET TYPE CODE IN PQL LIST
83 DF 0494 665 PUSHAL (R3)+ : SET BUFFER ADDRESS
04160004 8F DD 0496 666 PUSHHL #JPI$_WSEXTENT@16+4 : REQUEST CURRENT WSEXTENT
83 00 90 049C 667 MOVB #PQL$_LISTEND,(R3)+ : TERMINATE PQL LIST
50 SE D0 049F 668 MOVL SP,R0 : GET ADDRESS OF ITEM LIST
04A2 669 $GETJPIW S ITMLST=(R0),- : GET CURRENT QUOTA VALUES
04A2 670 IOSB=SPWN_Q_IOSB(R6),-
04A2 671 EFN=#EXEST_SYSEFN
SE 00000064 8F C0 04BA 672 ADDL #8*12+4,SP : POP ITEM LIST
04C1 673
04C1 674 :
04C1 675 : INSERT THE CURRENT SPWN BLOCK INTO THE LINKED LIST OF PENDING SUBPROCESSES,
04C1 676 : SO THAT IN CASE THE SUBPROCESS DIES WHILE WE ARE ATTEMPTING TO FEED IT
04C1 677 : CONTEXT, THE SUBPROCESS TERMINATION AST ROUTINE CAN DO SOMETHING ABOUT IT.
04C1 678 :
04C1 679 SETBIT SPWN_V_ACTIVE,SPWN_W_FLAGS(R6) : MARK BLOCK CURRENTLY "ACTIVE"
66 00C0 CB D0 04C6 680 MOVL PRC [ SPWN(R11),SPWN_L_LINK(R6) : INSERT INTO LINKED LIST
00C0 CB 56 D0 04CB 681 MOVL R6,PRC_L_SPWN(R11) : PASS ADDRESS OF SPWN TO TERMINATIO
682 : AST, SO WE CAN FIND OUT IF IT DIES
04D0 683
04D0 684 :
04D0 685 : CONSTRUCT THE PROCESS NAME OF THE FORM <USERNAME>_1. THE TRAILING
04D0 686 : NUMBER WILL BE INCREMENTED UNTIL WE FIND A UNIQUE PROCESS NAME.
04D0 687 :
04D0 688 BBS #SPWN_V_PRCNAM,- : BRANCH IF /PROCESS GIVEN
6F 0C A6 E0 04D2 689 SPWN_Q_FLAGS(R6),SPAWN_PROCESS :
54 01 D0 04D5 690 MOVL #1,R4 : START WITH POSTFIX #1
04D8 691
04D8 692 CONSTRUCT PRCNAM:
53 18 A6 9E 04D8 693 MOVAB SPWN_Q_PRCNAM(R6),R3 : GET ADDRESS OF DESCRIPTOR SPACE
7E D4 04DC 694 CLRL -(SP) : MARK END OF JPI LIST
53 DD 04DE 695 PUSHHL R3 : ADDRESS TO STORE RETURN LENGTH

```

```

02 AE 7E 68 7D 04E0 696      MOVQ      (R8),-(SP)
      0202 8F B0 04E3 697      MOVW      #JPI$_USERNAME,2(SP)
      50 5E D0 04E9 698      MOVL      SP,R0
      04 A3 5E 10 C0 0504 702      $GETJPIW S ITMLST=(R0),-
      04 B3 63 20 3A 0507 703      IOSB=SPWN_Q_IOSB(R6),-
      63 50 C2 0511 704      EFN=#EXE$C_SYSEFN
      7E 04 A8 63 C1 0514 705      ADDL      #4*4,SP
      51 7E 68 63 C3 0519 706      MOVL      4(R8),4(R3)
      51 FAF3 CF 9E 051D 707      LOCC      #^A' (R3),.24(R3)
      51 51 81 9A 0522 708      SUBL      R0,(R3)
      51 7E 50 7D 0525 709      ADDL3     (R3),4(R8),-(SP)
      51 50 5E D0 0528 710      SUBL3     (R3),(R8),-(SP)
      51 50 5E D0 052B 711      MOVAB     PRCNAM,NAME,R1
      51 50 5E D0 052B 712      MOVZBL    (R1)+,R0
      51 50 5E D0 052B 713      MOVQ      R0,-(SP)
      51 50 5E D0 052B 714      MOVL      SP,R0
      51 50 5E D0 052B 715      $FAO_S    CTRSTR=(R0),-
      51 50 5E D0 052B 716      OUTBUF=8(R0),-
      51 50 5E D0 052B 717      OUTLEN=2(R0),-
      51 50 5E D0 052B 718      P1=R3,-
      51 50 5E D0 052B 719      P2=R4
      51 50 5E D0 052B 720      ADDL      #8,SP
      51 50 5E D0 052B 721      MOVQ      (SP)+,(R3)
      51 50 5E D0 052B 722      :
      51 50 5E D0 052B 723      : CREATE THE SUBPROCESS
      51 50 5E D0 052B 724      : SPAWN_PROCESS:
      51 50 5E D0 052B 725      MOVAB     LOGINOUT,R1
      51 50 5E D0 052B 726      MOVZBL    (R1)+,R0
      51 50 5E D0 052B 727      MOVQ      R0,SPWN_Q_CLI(R6)
      51 50 5E D0 052B 728      MOVL      #PRCSM_CLI$SPEC,R1
      51 50 5E D0 052B 729      :
      51 50 5E D0 052B 730      : CONSTRUCT DESCRIPTOR OF IMAGE NAME
      51 50 5E D0 052B 731      :
      51 50 5E D0 052B 732      : SAVE DEFAULT CLI NAME
      51 50 5E D0 052B 733      : ASSUME NON-INTERACTIVE (ALSO
      51 50 5E D0 052B 734      : UNCONDITIONALLY INDICATE THAT
      51 50 5E D0 052B 735      : CLI IS SPECIFIED.
      51 50 5E D0 052B 736      :
      51 50 5E D0 052B 737      : BRANCH IF SO
      51 50 5E D0 052B 738      : SET INTERACTIVE BIT
      51 50 5E D0 052B 739      : CREATE THE PROCESS
      51 50 5E D0 052B 740      : SPECIFY INPUT
      51 50 5E D0 052B 741      : SPECIFY OUTPUT
      51 50 5E D0 052B 742      : ERROR = COMM. MAILBOX NAME
      51 50 5E D0 052B 743      : PROCESS NAME (USERNAME 1)
      51 50 5E D0 052B 744      : PROPAGATE BASE PRIORITY
      51 50 5E D0 052B 745      : PROPAGATE NON-DEDUCTIBLE QUOTAS
      51 50 5E D0 052B 746      : TERMINATION MAILBOX
      51 50 5E D0 052B 747      : RECEIVE PID HERE
      51 50 5E D0 052B 748      : SET/CLEAR INTERACTIVE BIT
      51 50 5E D0 052B 749      : PROCESS NAME ALREADY EXIST?
      51 50 5E D0 052B 750      : IF NOT, PROCESS AS ANY OTHER ERROR
      51 50 5E D0 052B 751      : BRANCH IF /PROCESS GIVEN
      51 50 5E D0 052B 752      :
      51 50 5E D0 052B 753      : INCREMENT NUMBER AFTER PROCESS NAM
      51 50 5E D0 052B 754      : IF ERROR CREATING PROCESS, REPORT
      51 50 5E D0 052B 755      : CLEANUP AND EXIT WITH STATUS
      51 50 5E D0 052B 756      :
      51 50 5E D0 052B 757      :
      51 50 5E D0 052B 758      :
      51 50 5E D0 052B 759      :
      51 50 5E D0 052B 760      :
      51 50 5E D0 052B 761      :
      51 50 5E D0 052B 762      :
      51 50 5E D0 052B 763      :
      51 50 5E D0 052B 764      :
      51 50 5E D0 052B 765      :
      51 50 5E D0 052B 766      :
      51 50 5E D0 052B 767      :
      51 50 5E D0 052B 768      :
      51 50 5E D0 052B 769      :
      51 50 5E D0 052B 770      :
      51 50 5E D0 052B 771      :
      51 50 5E D0 052B 772      :
      51 50 5E D0 052B 773      :
      51 50 5E D0 052B 774      :
      51 50 5E D0 052B 775      :
      51 50 5E D0 052B 776      :
      51 50 5E D0 052B 777      :
      51 50 5E D0 052B 778      :
      51 50 5E D0 052B 779      :
      51 50 5E D0 052B 780      :
      51 50 5E D0 052B 781      :
      51 50 5E D0 052B 782      :
      51 50 5E D0 052B 783      :
      51 50 5E D0 052B 784      :
      51 50 5E D0 052B 785      :
      51 50 5E D0 052B 786      :
      51 50 5E D0 052B 787      :
      51 50 5E D0 052B 788      :
      51 50 5E D0 052B 789      :
      51 50 5E D0 052B 790      :
      51 50 5E D0 052B 791      :
      51 50 5E D0 052B 792      :
      51 50 5E D0 052B 793      :
      51 50 5E D0 052B 794      :
      51 50 5E D0 052B 795      :
      51 50 5E D0 052B 796      :
      51 50 5E D0 052B 797      :
      51 50 5E D0 052B 798      :
      51 50 5E D0 052B 799      :
      51 50 5E D0 052B 800      :
      51 50 5E D0 052B 801      :
      51 50 5E D0 052B 802      :
      51 50 5E D0 052B 803      :
      51 50 5E D0 052B 804      :
      51 50 5E D0 052B 805      :
      51 50 5E D0 052B 806      :
      51 50 5E D0 052B 807      :
      51 50 5E D0 052B 808      :
      51 50 5E D0 052B 809      :
      51 50 5E D0 052B 810      :
      51 50 5E D0 052B 811      :
      51 50 5E D0 052B 812      :
      51 50 5E D0 052B 813      :
      51 50 5E D0 052B 814      :
      51 50 5E D0 052B 815      :
      51 50 5E D0 052B 816      :
      51 50 5E D0 052B 817      :
      51 50 5E D0 052B 818      :
      51 50 5E D0 052B 819      :
      51 50 5E D0 052B 820      :
      51 50 5E D0 052B 821      :
      51 50 5E D0 052B 822      :
      51 50 5E D0 052B 823      :
      51 50 5E D0 052B 824      :
      51 50 5E D0 052B 825      :
      51 50 5E D0 052B 826      :
      51 50 5E D0 052B 827      :
      51 50 5E D0 052B 828      :
      51 50 5E D0 052B 829      :
      51 50 5E D0 052B 830      :
      51 50 5E D0 052B 831      :
      51 50 5E D0 052B 832      :
      51 50 5E D0 052B 833      :
      51 50 5E D0 052B 834      :
      51 50 5E D0 052B 835      :
      51 50 5E D0 052B 836      :
      51 50 5E D0 052B 837      :
      51 50 5E D0 052B 838      :
      51 50 5E D0 052B 839      :
      51 50 5E D0 052B 840      :
      51 50 5E D0 052B 841      :
      51 50 5E D0 052B 842      :
      51 50 5E D0 052B 843      :
      51 50 5E D0 052B 844      :
      51 50 5E D0 052B 845      :
      51 50 5E D0 052B 846      :
      51 50 5E D0 052B 847      :
      51 50 5E D0 052B 848      :
      51 50 5E D0 052B 849      :
      51 50 5E D0 052B 850      :
      51 50 5E D0 052B 851      :
      51 50 5E D0 052B 852      :
      51 50 5E D0 052B 853      :
      51 50 5E D0 052B 854      :
      51 50 5E D0 052B 855      :
      51 50 5E D0 052B 856      :
      51 50 5E D0 052B 857      :
      51 50 5E D0 052B 858      :
      51 50 5E D0 052B 859      :
      51 50 5E D0 052B 860      :
      51 50 5E D0 052B 861      :
      51 50 5E D0 052B 862      :
      51 50 5E D0 052B 863      :
      51 50 5E D0 052B 864      :
      51 50 5E D0 052B 865      :
      51 50 5E D0 052B 866      :
      51 50 5E D0 052B 867      :
      51 50 5E D0 052B 868      :
      51 50 5E D0 052B 869      :
      51 50 5E D0 052B 870      :
      51 50 5E D0 052B 871      :
      51 50 5E D0 052B 872      :
      51 50 5E D0 052B 873      :
      51 50 5E D0 052B 874      :
      51 50 5E D0 052B 875      :
      51 50 5E D0 052B 876      :
      51 50 5E D0 052B 877      :
      51 50 5E D0 052B 878      :
      51 50 5E D0 052B 879      :
      51 50 5E D0 052B 880      :
      51 50 5E D0 052B 881      :
      51 50 5E D0 052B 882      :
      51 50 5E D0 052B 883      :
      51 50 5E D0 052B 884      :
      51 50 5E D0 052B 885      :
      51 50 5E D0 052B 886      :
      51 50 5E D0 052B 887      :
      51 50 5E D0 052B 888      :
      51 50 5E D0 052B 889      :
      51 50 5E D0 052B 890      :
      51 50 5E D0 052B 891      :
      51 50 5E D0 052B 892      :
      51 50 5E D0 052B 893      :
      51 50 5E D0 052B 894      :
      51 50 5E D0 052B 895      :
      51 50 5E D0 052B 896      :
      51 50 5E D0 052B 897      :
      51 50 5E D0 052B 898      :
      51 50 5E D0 052B 899      :
      51 50 5E D0 052B 900      :
      51 50 5E D0 052B 901      :
      51 50 5E D0 052B 902      :
      51 50 5E D0 052B 903      :
      51 50 5E D0 052B 904      :
      51 50 5E D0 052B 905      :
      51 50 5E D0 052B 906      :
      51 50 5E D0 052B 907      :
      51 50 5E D0 052B 908      :
      51 50 5E D0 052B 909      :
      51 50 5E D0 052B 910      :
      51 50 5E D0 052B 911      :
      51 50 5E D0 052B 912      :
      51 50 5E D0 052B 913      :
      51 50 5E D0 052B 914      :
      51 50 5E D0 052B 915      :
      51 50 5E D0 052B 916      :
      51 50 5E D0 052B 917      :
      51 50 5E D0 052B 918      :
      51 50 5E D0 052B 919      :
      51 50 5E D0 052B 920      :
      51 50 5E D0 052B 921      :
      51 50 5E D0 052B 922      :
      51 50 5E D0 052B 923      :
      51 50 5E D0 052B 924      :
      51 50 5E D0 052B 925      :
      51 50 5E D0 052B 926      :
      51 50 5E D0 052B 927      :
      51 50 5E D0 052B 928      :
      51 50 5E D0 052B 929      :
      51 50 5E D0 052B 930      :
      51 50 5E D0 052B 931      :
      51 50 5E D0 052B 932      :
      51 50 5E D0 052B 933      :
      51 50 5E D0 052B 934      :
      51 50 5E D0 052B 935      :
      51 50 5E D0 052B 936      :
      51 50 5E D0 052B 937      :
      51 50 5E D0 052B 938      :
      51 50 5E D0 052B 939      :
      51 50 5E D0 052B 940      :
      51 50 5E D0 052B 941      :
      51 50 5E D0 052B 942      :
      51 50 5E D0 052B 943      :
      51 50 5E D0 052B 944      :
      51 50 5E D0 052B 945      :
      51 50 5E D0 052B 946      :
      51 50 5E D0 052B 947      :
      51 50 5E D0 052B 948      :
      51 50 5E D0 052B 949      :
      51 50 5E D0 052B 950      :
      51 50 5E D0 052B 951      :
      51 50 5E D0 052B 952      :
      51 50 5E D0 052B 953      :
      51 50 5E D0 052B 954      :
      51 50 5E D0 052B 955      :
      51 50 5E D0 052B 956      :
      51 50 5E D0 052B 957      :
      51 50 5E D0 052B 958      :
      51 50 5E D0 052B 959      :
      51 50 5E D0 052B 960      :
      51 50 5E D0 052B 961      :
      51 50 5E D0 052B 962      :
      51 50 5E D0 052B 963      :
      51 50 5E D0 052B 964      :
      51 50 5E D0 052B 965      :
      51 50 5E D0 052B 966      :
      51 50 5E D0 052B 967      :
      51 50 5E D0 052B 968      :
      51 50 5E D0 052B 969      :
      51 50 5E D0 052B 970      :
      51 50 5E D0 052B 971      :
      51 50 5E D0 052B 972      :
      51 50 5E D0 052B 973      :
      51 50 5E D0 052B 974      :
      51 50 5E D0 052B 975      :
      51 50 5E D0 052B 976      :
      51 50 5E D0 052B 977      :
      51 50 5E D0 052B 978      :
      51 50 5E D0 052B 979      :
      51 50 5E D0 052B 980      :
      51 50 5E D0 052B 981      :
      51 50 5E D0 052B 982      :
      51 50 5E D0 052B 983      :
      51 50 5E D0 052B 984      :
      51 50 5E D0 052B 985      :
      51 50 5E D0 052B 986      :
      51 50 5E D0 052B 987      :
      51 50 5E D0 052B 988      :
      51 50 5E D0 052B 989      :
      51 50 5E D0 052B 990      :
      51 50 5E D0 052B 991      :
      51 50 5E D0 052B 992      :
      51 50 5E D0 052B 993      :
      51 50 5E D0 052B 994      :
      51 50 5E D0 052B 995      :
      51 50 5E D0 052B 996      :
      51 50 5E D0 052B 997      :
      51 50 5E D0 052B 998      :
      51 50 5E D0 052B 999      :
      51 50 5E D0 052B 1000     :

```

```

18 A6 90 05AC 753 15$:   MOVB   SPWN_Q_PRCNAM(R6), -      ; GET LENGTH OF PROCESS NAME
0092 C6      05AF 754      SPWN_T_PROCESS(R6)          ;
18 A6 28 05B2 755      MOV C3  SPWN-Q-PRCNAM(R6)        ; COPY THE NAME
1C B6      05B5 756      SPWN-Q-PRCNAM+4(R6), -        ;
0093 C6      05B7 757      SPWN_T_PROCESS+1(R6)        ;
      05BA 758      ;
      05BA 759      ;
      05BA 760      ; WRITE /LOG MESSAGE INDICATING PROCESS WAS CREATED
      05BA 761      ;
      05BA 762      ;
10 0C A6 00 E1 05BA 762      BBC      #SPWN_V_LOG, -      ; BRANCH IF /NOLOG SPECIFIED
18 A6 9F 05BC 763      SPWN-Q_FLAGS(R6), 20$           ;
51 01 9F 05BF 764      PUSHAB  SPWN-Q-PRCNAM(R6)        ; ADDRESS OF PROCESS NAME
50 0003FD01 8F DO 05C2 765      MOVL   #1,RT            ; SET NUMBER OF ARGS
      FA31' 30 DO 05C5 766      MOVL   #CLIS SPAWNED,RO   ; MESSAGE CODE
      05CC 767      BSBW   DCL$FORMMSG                 ; OUTPUT MESSAGE
      05CF 768      ;
      05CF 769      ;
      05CF 770      ; IF /WAIT /LOG THEN WRITE MESSAGE INDICATING SUBPROCESS ATTACHED.
      05CF 771      ; DO THIS BEFORE WE WRITE THE CONTEXT, WHILE WE STILL HAVE CONTROL OVER
      05CF 772      ; THE TERMINAL.
      05CF 773      ;
15 0C A6 02 E1 05CF 774 20$:   BBC      #SPWN_V_WAIT, -      ; SKIP IF /NOWAIT
      05D1 775      SPWN-Q_FLAGS(R6), 30$           ;
10 0C A6 00 E1 05D4 776      BBC      #SPWN_V_LOG, -      ; SKIP IF /NOLOG SPECIFIED
18 A6 9F 05D6 777      SPWN-Q_FLAGS(R6), 30$           ;
51 01 9F 05D9 778      PUSHAB  SPWN-Q-PRCNAM(R6)        ; ADDRESS OF PROCESS NAME
50 0003FD09 8F DO 05DC 779      MOVL   #1,RT            ; SET NUMBER OF ARGS
      FA17' 30 DO 05DF 780      MOVL   #CLIS ATTACHED,RO  ; MESSAGE CODE
      05E6 781      BSBW   DCL$FORMMSG                 ; OUTPUT MESSAGE
      05E9 782      ;
      05E9 783      ;
      05E9 784      ; WRITE CONTEXT TO SUBPROCESS VIA MAILBOX. IF AN ERROR OCCURS AT THIS POINT,
      05E9 785      ; WE MUST DELETE THE PROCESS TO AVOID HAVING IT HANG AROUND FOR NOTHING.
      05E9 786      ;
23 0C A6 0E E0 05E9 787 30$:   BBS      #SPWN_V_NOCTX,SPWN_W_FLAGS(R6),35$ ; BRANCH IF UNKNOWN CLI WAS SPECIF
      088F 30 05EE 788      BSBW   WRITE_CONTEXT        ; WRITE CONTEXT TO SUBPROCESS
1D 50 E8 05F1 789      BLBS   RO,35$                 ; BRANCH IF OK
50 DD 05F4 790      PUSHL   RO                        ; SAVE WRITE STATUS
      05F6 791      $DELPRC_S PIDADR=SPWN_L_SUBPID(R6)  ; DELETE THE SUBPROCESS
01 44 A6 D1 0602 792      CMPL   SPWN_L_STATUS(R6),#1    ; GET TERMINATION STATUS
      04 1B 0606 793      BLEQU  32$                   ; USE IT IF SIGNIFICANT
6E 44 A6 DO 0608 794      MOVL   SPWN_L_STATUS(R6),(SP)  ; (NEQ TO 0 OR 1)
      50 8ED0 060C 795 32$:   POPL   RO                  ; RESTORE STATUS CODE
      6B 11 060F 796      BRB    60$                   ; SIGNAL THE ORIGINAL ERROR, OR THE
      0611 797      ; SUBPROCESS ABNORMAL STATUS (IF ANY)
      0611 798      ;
      0611 799      ;
      0611 800      ; CLEANUP CONTEXT MAILBOX, TO ELIMINATE POOL USAGE AS SOON AS POSSIBLE.
      0611 801      ;
      0611 802      ;
      0611 803 35$:   $DASSGN_S CHAN=SPWN_W_CHAN(R6)    ; CLEANUP MAILBOX (THUS, POOL USAGE)
50 0A A6 B4 061C 803      CLRW   SPWN_W_CHAN(R6)        ; INDICATE CHANNEL WAS "REMOVED"
      01 DO 061F 804      MOVL   #1,RO                  ; IGNORE STATUS FROM DEASSIGN
      0622 805      ; WHILE WAITING FOR RE-ATTACH
      0622 806      ;
      0622 807      ;
      0622 808      ; CREATE /NOWAIT SUBPROCESS.
      0622 809      ;

```

```

51 01 D0 0622 810      MOVL  #1,R1      ; SET NORMAL TERMINATION STATUS
      02 E0 0625 811      BBS   #SPWN_V_WAIT,-      ; BRANCH IF /NOWAIT
03 0C A6 0627 812      BRW   SPWN_Q_FLAGS(R6),9000$ ;
      0095 31 062A 813      SPWN_Q_EXIT
      062D 814 9000$:
      062D 815 :
      062D 816 : CREATE /WAIT SUBPROCESS. MARK THIS (PARENT) PROCESS DETACHED FROM TERMINAL.
      062D 817 :
      07 E1 062D 818      BBC   #SPWN_V_ACTIVE,-      ; BRANCH IF INACTIVE (SUBPROCESS
4A 0C A6 062F 819      SPWN_Q_FLAGS(R6),60$      ; ALREADY TERMINATED)
      0632 820      SETBIT  PRC_V_DETACHED,PRC_W_FLAGS(R11) ; MARK PROCESS DETACHED
      0637 821 :
      0637 822 :
      0637 823 : WAIT FOR SUBPROCESS TO RETURN CONTROL HERE (VIA ATTACH OR TERMINATION)
      0637 824 :
      0883 30 0637 825 50$: BSBW  CHECK_FOR_HANGUP      ; CHECK FOR HANGUP AST
50 7E 7E 063A 826      MOVAQ  -(SP),R0      ; ALLOCATE A TEMP. IOSB
      063D 827      $QIOW_S FUNC=#IOS SETMODE!IOSM_WRTATTN,- ; SET ATTENTION AST ON MAILBOX
      063D 828      CHAN=PRC_Q_ATTMBX(R11),-
      063D 829      EFN=#EXESC_SYSEFN,-
      063D 830      IOSB=(R0),-
      063D 831      P1=ATTACH_AST,-
      063D 832      P2=R11
      08 0C 0663 833      ADDL  #8,SP      ; ADDRESS OF AST ROUTINE
5E 08 0C 0666 834      $HIBER_S      ; PASS ADDRESS OF CLI STORAGE
      07 E1 066D 835      BBC   #SPWN_V_ACTIVE,-      ; CLEAN UP STACK
05 0C A6 066F 836      SPWN_Q_FLAGS(R6),55$      ; HIBERNATE WAITING FOR WAKEUP
      0F E0 0672 837      BBS   #PRC_V_DETACHED,-      ; BRANCH IF SUBPROCESS INACTIVE
      0674 838      PRC_Q_FLAGS(R11),50$      ; (SUBPROCESS HAS BEEN TERMINATED)
      0677 839 55$: CLRBIT  PRC_V_DETACHED,PRC_W_FLAGS(R11) ; BRANCH IF PARENT STILL DETACHED
      067C 840      ; MARK PROCESS ATTACHED TO TERMINAL
      067C 841 :
      067C 842 : WE HAVE RETURNED FROM THE SUBPROCESS EITHER VIA A RE-ATTACH REQUEST TO
      067C 843 : THIS PRGCESS OR VIA SUBPROCESS TERMINATION. IF /LOG WAS SPECIFIED, THEN
      067C 844 : OUTPUT THE RETURNED MESSAGE.
      067C 845 :
      00 E1 067C 846 60$: BBC   #SPWN_V_LOG,-      ; BRANCH IF /NOLOG SPECIFIED
08 0C A6 067E 847      SPWN_Q_FLAGS(R6),70$      ;
      50 DD 0681 848      PUSHL  R0      ; SAVE FINAL STATUS
      0780 30 0683 849      BSBW  RETURNED_MESSAGE      ; WRITE MESSAGE SAYING WE HAVE RETUR
      50 8ED0 0686 850      POPL  R0      ; RESTORE FINAL STATUS
      0689 851 :
      0689 852 :
      0689 853 : IF WE ARE IN AN INTERACTIVE PROCESS, THEN TELL THE TERMINAL DRIVER THAT
      0689 854 : WE NOW OWN THE TERMINAL.
      0689 855 :
      28 68 AB 06 0689 856 70$: BBS   #PRC_V_MODE,PRC_W_FLAGS(R11),80$ ; SKIP IF NOT INTERACTIVE
      50 DD 068E 857      PUSHL  R0      ; SAVE FINAL STATUS
      0690 858      $QIOW_S FUNC=#IOS SETMODE!IOSM TT_PROCESS,- ; ASSUME TERMINAL OWNERSHIP
      0690 859      CHAN=PRC_Q_INPCHAN(R11),-
      0690 860      IOSB=SPWN_Q_IOSB(R6),-
      0690 861      EFN=#EXESC_SYSEFN
      50 8ED0 0683 862      POPL  R0      ; RESTORE FINAL STATUS
      0686 863 :
      0686 864 :
      0686 865 : IF THIS IS A RE-ATTACH THEN SKIP DATA STRUCTURE AND MAILBOX DELETIONS AND
      0686 866 : SET NORMAL TERMINATION STATUS. OTHERWISE, SET THE RETURNED TERMINATION STATUS.

```

```

      51 01 DO 06B6 867 :
      07 EO 06B6 868 80$: MOVL #1,R1 ; SET NORMAL TERMINATION STATUS
      33 0C A6 06B9 869 BBS #SPWN_V_ACTIVE,- ; BRANCH IF CONTROL RETURNED
      51 44 A6 DO 06BB 870 SPWN_Q_FLAGS(R6),RESTORE_CONTEXT; VIA A RE-ATTACH
      06BE 871 MOVL SPWN_L_STATUS(R6),R1 ; GET TERMINATION STATUS
      06C2 872 :
      06C2 873 :
      06C2 874 : THE SPAWN COMMAND IS ESSENTIALLY COMPLETE. WE HAVE RETURNED CONTROL HERE
      06C2 875 : DUE TO SUBPROCESS TERMINATION. THEREFORE, CLEANUP MISC. DATA STRUCTURES AND
      06C2 876 : MAILBOXES AND REPORT ANY ERRORS WHICH WERE DETECTED.
      06C2 877 :
      06C2 878 : IF SUBPROCESS WAS SUCCESSFULLY SPAWNED AND TERMINATED, THEN TMBX WAS
      06C2 879 : DELETED BY THE TERMINATION AST. IF NOT, TRY TO DEASSIGN THE MAILBOX AND
      06C2 880 : DELETE THE DATA STRUCTURE HERE.
      06C2 881 :
      06C2 882 SPAWN_EXIT:
      7E 50 7D 06C2 883 MOVQ R0,-(SP) ; SAVE ERROR AND TERMINATION STATUS
      OD 50 E8 06C5 884 BLBS R0,10$ ; BRANCH IF SUCCESSFUL SPAWN
      07 E1 06C8 885 BBC #SPWN_V_ACTIVE,- ; DID SUBPROCESS TERMINATE?
      08 0C A6 06CA 886 SPWN_Q_FLAGS(R6),10$ ; IF SO, TERMINATION MBX TAKEN CARE
      08 A7 97 06CD 887 DECB TMBX_B_REFS(R7) ; DECR REFERENCE COUNT TO MAILBOX
      03 12 06D0 888 BNEQ 10$ ; BRANCH IF STILL OUTSTANDING USES
      072A 30 06D2 889 BSBW DELETE_TMBX ; DELETE TERMINATION MAILBOX AND TMB
      06D5 890 :
      06D5 891 :
      06D5 892 : IN ALL CASES, DELETE THE ATTACH MAILBOX AND DEASSIGN THE CONTEXT MAILBOX
      06D5 893 : CHANNEL.
      06D5 894 :
      068A 30 06D5 895 10$: BSBW DELETE_ATTMBX ; DELETE THE ATTACH MAILBOX
      OA A6 B4 06D8 896 $DASSGN_S_CHAN=SPWN_W_CHAN(R6) ; DEASSIGN CONTEXT MAILBOX CHANNEL
      06E3 897 CLRW SPWN_W_CHAN(R6) ; INDICATE CHANNEL WAS 'REMOVED'
      06E6 898 :
      06E6 899 :
      06E6 900 : IF /WAIT, THEN DEALLOCATE THE SPWN BLOCK.
      06E6 901 :
      03 02 E1 06E6 902 BBC #SPWN_V_WAIT,- ; BRANCH IF /NOWAIT
      0C 0C A6 06E8 903 SPWN_Q_FLAGS(R6),20$ ;
      072B 30 06EB 904 BSBW DEALLOC_SPWN ; REMOVE INACTIVE SPWN BLOCKS
      50 8E 7D 06EE 905 20$: MOVQ (SP)+,R0 ; RESTORE ERROR AND TERMINATION STAT
      06F1 906 :
      06F1 907 :
      06F1 908 : IN ALL CASES, RESTORE THE PROCESS STATE -
      06F1 909 : STACK, EVENT FLAGS, CONTROL KEY AST'S, SPAWN STATUS, TERMINATION STATUS,
      06F1 910 : AND SAVED REGISTERS.
      06F1 911 :
      06F1 912 RESTORE_CONTEXT:
      5E 0308 CE 9E 06F1 913 MOVAB 768+8(SP),SP ; POP SCRATCH STORAGE OFF STACK
      7E 50 7D 06F6 914 MOVQ R0,-(SP) ; SAVE ERROR AND TERMINATION STATUS
      51 58 A6 DO 06F9 915 MOVL SPWN_L_OUTOFBAND(R6),R1 ; RESTORE OUT-OF-BAND ENABLE MASK
      F900 30 06FD 916 BSBW DCL$RESET00B ; RESTORE AST'S
      01BF 8F BA 0700 917 POPR #^M<R0,R1,R2,R3,R4,R5,R7,R8> ; RESTORE REGISTERS
      0704 918 :
      0704 919 :
      0704 920 : IF A HANGUP IS PENDING, THEN ABORT PROCESS
      0704 921 :
      06 0C E1 0704 922 BBC #PRC_V_HANGUP,- ; IF SET, HANG-UP PENDING
      68 AB 0706 923 PR_Q_FLAGS(R11),20$ ;

```

SPAWN
V04-000

- MULTI-PROCESSING COMMANDS
SPAWN A SUBPROCESS

D 13

16-SEP-1984 00:17:05
4-SEP-1984 23:43:20

VAX/VMS Macro V04-00
[DCL.SRC]SPAWN.MAR;1

Page 19
(4)

SE 08 CO 0709 924 ADDL
F8F1' 31 070C 925 BRW
05 070F 926 20\$: RSB

#8,SP
CCL\$ABORT

; POP TWO PC'S OFF THE STACK
; LOG THE PROCESS OUT
; EXIT

```

0710 928      .SBTTL  PROCESS SPAWN INPUT STREAM
0710 929      :---
0710 930      :
0710 931      : THIS ROUTINE IS CALLED TO PROCESS THE SPAWN INPUT STREAM.
0710 932      :
0710 933      : INPUTS:
0710 934      :
0710 935      :     R6 = ADDRESS OF SPAWN BLOCK
0710 936      :     R8 = DESCRIPTOR OF SCRATCH AREA
0710 937      :     R11 = ADDRESS OF PRC AREA
0710 938      :
0710 939      : OUTPUTS:
0710 940      :
0710 941      :     R0 = STATUS CODE
0710 942      :---
0710 943      :
0710 944      : VERIFY_INPUT:
0710 945      :
0710 946      : DETERMINE IF AN AUTOMATIC LOGOUT SHOULD OCCUR AFTER EXECUTION OF THE
0710 947      : COMMAND STRING.  IF ONLY THE COMMAND STRING WAS SPECIFIED, THEN LOGOUT
0710 948      : AFTER IT IS EXECUTED.  IF BOTH THE COMMAND STRING AND /INPUT WERE SPECIFIED,
0710 949      : THEN CAUSE THE COMMAND STRING TO BE EXECUTED FIRST, FOLLOWED BY THE /INPUT
0710 950      : STREAM.
0710 951      :
0710 952      : BBS      #SPWN_V_INPUT,-      : IF INPUT SPECIFIED,
0710 953      : SPWN_Q_FLAGS(R6),10$      : KEEP AUTOLOGUT FLAG OFF
0710 954      : MOVZBL  SYSS$INPUT,SPWN_Q_INPUT(R6)      : USE 'SYSS$INPUT'
0710 955      : MOVAB   SYSS$INPUT+1,SPWN_Q_INPUT+4(R6)      :
0710 956      : TSTW   SPWN_Q_CMDS$R(R6)      : ANY COMMAND STRING SPECIFIED?
0710 957      : BEQL   10$      : NO, DO NOT SET IMPLIED LOGOUT
0710 958      : SETBIT  SPWN_V_AUTOLOGO,-      : YES, SET IMPLIED LOGOUT FLAG
0710 959      : SPWN_W_FLAGS(R6)      :
0710 960      :
0710 961      :
0710 962      : CHECK THAT INPUT FILE SPECIFICATION IS NOT TOO LONG.
0710 963      :
0710 964      : 10$:  MOVL   #CLIS_INVFILSPE,R0      : ASSUME INVALID FILE SPEC
0710 965      : CMPW   SPWN_Q_INPUT(R6),-      : NAME TOO LONG?
0710 966      :      #NAM$C_MAXRSS      :
0710 967      : BGTRU   25$      : IF SO, EXIT WITH ERROR
0710 968      :
0710 969      :
0710 970      : ALLOCATE FAB AND NAM BLOCKS AND THEN SPARSE THE INPUT FILE SPECIFICATION.
0710 971      :
0710 972      : MOVAB   -FAB$C_BLN-NAM$C_BLN(SP),SP      : ALLOCATE FAB AND NAM BLOCKS
0710 973      : MOVC5   #0,(SP),#0,-      : ZERO FAB AND NAM
0710 974      :      #FAB$C_BLN+NAM$C_BLN,(SP)      :
0710 975      : MOVL   SP,R4      : GET BASE OF FAB
0710 976      : MOVAB   FAB$C_BLN(R4),R5      : GET BASE OF NAM
0710 977      :
0710 978      : ASSUME   FAB$B_BLN EQ FAB$B_BID+1      : INIT FAB
0710 979      : MOVW   #FAB$C_BID+<FAB$C_BLN@8>,-      : SET FAB ID
0710 980      :      FAB$B_BID(R4)      :
0710 981      : SETBIT  FAB$V_PPF,FAB$B_FOP(R4)      : USE PROCESS I/O SEGMENT
0710 982      : MOVB   SPWN_Q_INPUT(R6),FAB$B_FNS(R4)      : SPECIFY FILE NAME
0710 983      : MOVL   SPWN_Q_INPUT+4(R6),FAB$B_FNA(R4)      :
0710 984      : MOVB   COM,FAB$B_DNS(R4)      : SPECIFY DEFAULT FILE TYPE

```

```

30 A4 F8F6 CF 9E 0767 985      MOVAB COM+1,FAB$!_DNA(R4)      :
      28 A4 55 D0 076D 986      MOVL  R5,FAB$!_NAM(R4)      : SPECIFY NAM BLOCK ADDRESS
      6002 8F B0 0771 987      :
      65      0771 988      ASSUME NAM$B_BLN EQ NAM$B_BID+1      : INIT NAM
      08 A5 10 90 0771 989      MOVW  #NAM$C_BID+<NAM$C_BLN@B>,-      : SET NAM ID
      0A A5 FF 8F 90 0775 990      NAM$B_BID(R5)
      0C A5 04 A8 D0 0776 991      MOVW  #NAM$M_NOCONCEAL,NAM$B_NOP(R5)      : GET PAST DOUBLE UNDERSCORES
      90 077A 992      MOVW  #NAM$C_MAXRSS,NAM$B_ESS(R5)      : SPECIFY RESULT BUFFER
      90 077F 993      MOVL  4(R8),NAM$!_ESA(R5)
      0784 994
      0784 995      $PARSE FAB=(R4)      : PARSE THE FILE SPEC
      08 50 E8 078D 996      BLBS  R0,30$      : CONTINUE IF SUCCESSFUL
      90 0790 997
      5E 00B0 CE 9E 0790 998 20$: MOVAB FAB$C_BLN+NAM$C_BLN(SP),SP      : DEALLOCATE FAB AND NAM
      00EB 31 0795 999 25$: BRW  95$      : EXIT WITH ERROR
      0798 1000
      0798 1001 :
      0798 1002 : PROCESS NON-PPF INPUT HERE. IF NOT RECORD-ORIENTED, THEN USE EXPANDED
      0798 1003 : STRING AS INPUT, DO A $SEARCH FOR THE FILE, AND SET SILENT LOGOUT FLAG.
      0798 1004 : IF RECORD-ORIENTED THEN USE DEV FIELD AS INPUT.
      0798 1005
      3F 34 A5 10 E0 0798 1006 30$: BBS  #NAM$V_PPF,NAM$!_FNB(R5),50$      : BRANCH IF PPF FILE
      50 08 A5 9A 079D 1007      MOVZBL NAM$B_ESL(R5),R0      : GET LENGTH OF RESULT STRING
      68 50 C2 07A1 1008      SUBL  R0,(R8)      : PERMANENTLY REMOVE RESULT
      04 A8 50 C0 07A4 1009      ADDL  R0,4(R8)      : FROM SCRATCH AREA
      0D 40 A4 00 E1 07A8 1010      BBC  #DEV$V_REC,FAB$!_DEV(R4),40$      : BRANCH IF NOT REC-ORIENTED
      20 A6 39 A5 9A 07AD 1011      MOVZBL NAM$B_DEV(R5),SPWN_Q_INPUT(R6)      : USE DEVICE AS INPUT
      24 A6 44 A5 D0 07B2 1012      MOVL  NAM$!_DEV(R5),SPWN_Q_INPUT+4(R6)
      0085 31 07B7 1013      BRW  56$      : PROCESS AS PPF DEVICE
      07BA 1014
      07BA 1015 :
      07BA 1016 : INPUT IS A NON-PPF NON-RECORD-ORIENTED DEVICE. $SEARCH FOR THE FILE.
      07BA 1017 : IF FOUND, THEN USE THE ESS AS SYSSINPUT. ALSO, MARK THE PROCESS AS
      07BA 1018 : NON-INTERACTIVE.
      07BA 1019
      20 0A 50 E9 07C3 1020 40$: $SEARCH FAB=(R4)      : SEARCH FOR THE SPECIFIED FILE
      A6 0B A5 9A 07C6 1021      BLBC  R0,20$      : BRANCH IF NOT FOUND
      24 A6 0C A5 D0 07CB 1022      MOVZBL NAM$B_ESL(R5),SPWN_Q_INPUT(R6)      : USE ESS AS INPUT FILE SPEC
      07D0 1023      MOVL  NAM$!_ESA(R5),SPWN_Q_INPUT+4(R6)
      5E 00B0 CE 9E 07D4 1024      SETBIT SPWN_Q_MODE,SPWN_W_FLAGS(R6)      : SET NON-INTERACTIVE SUBPROCESS
      0095 31 07D9 1025 45$: MOVAB FAB$C_BLN+NAM$C_BLN(SP),SP      : DEALLOCATE FAB AND NAM
      07DC 1026      BRW  90$      : DONE WITH PARSING
      07DC 1027
      07DC 1028 :
      07DC 1029 : INPUT IS PPF. IF FROM A RECORD-ORIENTED DEVICE, DO $GETDVI TO GET
      07DC 1030 : THE DEVICE NAME. OTHERWISE, 1) IF EXPLICITLY SPECIFIED THEN RETURN AN
      07DC 1031 : ERROR, 2) IF INTERACTIVE PROCESS THEN USE THE TERMINAL AS INPUT, 3) IF
      07DC 1032 : NON-INTERACTIVE PROCESS THEN USE NL: AS INPUT.
      07DC 1033
      0C 0C A6 0A E1 07DC 1034 50$: BBC  #SPWN_V_INPUT,SPWN_W_FLAGS(R6),52$      : SKIP IF IMPLICIT INPUT
      50 000388EA 8F D0 07E1 1035      MOVL  #CLIS-SPWNIO,R0      : ASSUME ILLEGAL INPUT FILE
      A3 40 A4 00 E1 07E8 1036      BBC  #DEV$V_REC,FAB$!_DEV(R4),20$      : ERROR IF NOT REC-ORIENTED
      07ED 1037
      07ED 1038 :
      07ED 1039 : PROCESS LEGAL NON-RECORD-ORIENTED PPF INPUT.
      07ED 1040
      4D 40 A4 00 E0 07ED 1041 52$: BBS  #DEV$V_REC,FAB$!_DEV(R4),56$      : BRANCH IF REC-ORIENTED

```

```

37 68 AB 06 E0 07F2 1042 BBS #PRC V MODE,PRC W FLAGS(R11),53$ : BRANCH IF NOT INTERACTIVE
51 00000028'8F D0 07F7 1043 MOVL #CTLSAG CLIDATA+PPD$T_INPDV1,R1 : GET ADDRESS OF ASCII DEVICE NAME
55 81 9A 07FE 1044 MOVZBL (R1)+,R5 : GET LENGTH OF DEVICE
7E 54 7D 0801 1045 MOVQ R4,-(SP) : SAVE R4/R5
04 B8 61 55 28 0804 1046 MOVQ R5,(R1),@4(R8) : COPY STRING TO SCRATCH AREA
63 3A 90 0809 1047 MOVQ #^A':'(R3) : ADD COLON TO END OF STRING
54 8E 7D 080C 1048 MOVQ (SP)+,R4 : RESTORE R4/R5
55 D6 080F 1049 INCL R5 : ADD LENGTH OF A COLON
20 A6 55 D0 0811 1050 MOVL R5,SPWN_Q_INPUT(R6) : SAVE DEVICE DESCRIPTOR
24 A6 04 A8 D0 0815 1051 MOVL 4(R8),SPWN_Q_INPUT+4(R6) :
68 55 A2 081A 1052 SUBW R5,(R8) : PERMANENTLY REMOVE DEVICE
04 A8 55 C0 081D 1053 ADDL R5,4(R8) : FROM SCRATCH AREA
00000044'9F D0 0821 1054 MOVL @#CTLSAG CLIDATA+PPD$S_INPDEV,- : GET INPUT DEVICE CHARS
40 A4 : FAB$S_DEV(R4)
26 40 A4 00 E0 0829 1056 BBS #DEV$V_REC,FAB$S_DEV(R4),58$ : BRANCH IF REC-ORIENTED
20 A6 F829 CF 9A 082E 1057 53$: MOVZBL NL,SPWN_Q_INPUT(R6) : SET INPUT TO NL:
24 A6 F824 CF 9E 0834 1058 MOVAB NL+1,SPWN_Q_INPUT+4(R6) :
40 A4 D4 083A 1059 CLRL FAB$S_DEV(R4) : CLEAR DEVICE CHARACTERISTICS
15 11 083D 1060 BRB 58$ :
083F 1061 :
083F 1062 :
083F 1063 : PROCESS REC-ORIENTED PPF INPUT.
083F 1064 :
52 20 A6 7D 083F 1065 56$: MOVQ SPWN_Q_INPUT(R6),R2 : COPY DESCRIPTOR OF DEVICE NAME
54 68 7D 0843 1066 MOVQ (R8),R4 : COPY DESCRIPTOR OF SCRATCH AREA
097D 30 0846 1067 BSBW GET_DEVICE : GET DEVICE NAME
20 A6 54 7D 0849 1068 MOVQ R4,SPWN_Q_INPUT(R6) : SAVE EQUIVALENCE STRING
68 54 A2 084D 1069 SUBW R4,(R8) : PERMANENTLY REMOVE RESULT
04 A8 54 C0 0850 1070 ADDL R4,4(R8) : FROM SCRATCH AREA
0854 1071 :
0854 1072 :
0854 1073 : GET TERMINAL CHARACTERISTICS. DO NOT SPAWN THE SUBPROCESS IF THE TERMINAL
0854 1074 : HAS AN ASSOCIATED MAILBOX.
0854 1075 :
50 40 AE D0 0854 1076 58$: MOVL FAB$S_DEV(SP),R0 : GET DEVICE CHARACTERISTICS
5E 0080 CE 9E 0858 1077 MOVAB FAB$C_BLN+NAME$C_BLN(SP),SP : DEALLOCATE FAB AND NAME
06 50 02 E0 085D 1078 BBS #DEV$V_TRM,R0,59$ : SKIP IF INPUT IS NOT A TERMINAL
0A 11 0861 1079 SETBIT SPWN_V_MODE,SPWN_W_FLAGS(R6) : SET NON-INTERACTIVE SUBPROCESS
51 20 A6 7E 0865 1080 BRB 90$ : ALL DONE
0016 30 0867 1081 59$: MOVQ SPWN_Q_INPUT(R6),R1 : GET ADDRESS OF INPUT STRING
12 50 E9 086B 1082 BSBW CHECK_TRMBX : CHECK FOR MAILBOX
086E 1083 BLBC R0,95$ : BRANCH IF IT EXISTS
0871 1084 :
0871 1085 :
0871 1086 : CHECK THAT INPUT FILE SPEC IS NOT TOO LONG FOR $CREPRC TO HANDLE.
0871 1087 :
50 0038200 8F D0 0871 1088 90$: MOVL #CLIS_INVFILSPE,R0 : ASSUME SPEC IS TOO LONG
20 A6 00FF 8F B1 0878 1089 CMPW #LNMS$_NAMLENGTH,SPWN_Q_INPUT(R6) : IS IT REALLY?
03 1F 087E 1090 BLSSU 95$ : YES, THEN ERROR
0880 1091 :
0880 1092 :
0880 1093 : RETURN WITH STATUS
0880 1094 :
50 01 D0 0880 1095 : SET NORMAL STATUS
05 0883 1096 95$: RSB : EXIT WITH STATUS

```

```

0884 1098 .SBTTL CHECK FOR ASSOCIATED TERMINAL MAILBOX
0884 1099 :---
0884 1100 :
0884 1101 : THIS ROUTINE IS CALLED TO CHECK FOR AN ASSOCIATED TERMINAL MAILBOX.
0884 1102 : SUBPROCESS IS NOT SPAWNED OR ATTACHED IF THE TERMINAL HAS AN ASSOCIATED
0884 1103 : MAILBOX.
0884 1104 :
0884 1105 : INPUTS:
0884 1106 :
0884 1107 : R1 ADDRESS OF DESCRIPTOR OF TERMINAL DEVICE NAME
0884 1108 :
0884 1109 : OUTPUTS:
0884 1110 :
0884 1111 : R0 STATUS CODE
0884 1112 :---
0884 1113 :
0884 1114 CHECK_TRMMBX:
0884 1115 MOVQ R2,-(SP) : SAVE REGISTERS
0884 1116 CLRQ -(SP) : ALLOCATE AN IOSB
0884 1117 MOVAL -(SP),R2 : ALLOCATE LONGWORD FOR CHANNEL NUMB
0884 1118 SUBL #12,SP : ALLOCATE DEVICE CHARACTERISTICS BU
0884 1119 MOVL SP,R3 : SAVE BUFFER ADDRESS
0884 1120 $ASSIGN,S DEVNAM=(R1),- : GET A CHANNEL TO THE TERMINAL
0884 1121 CHAN=(R2)
0884 1122 BLBC R0,90$ : BRANCH IF ERROR
0884 1123 :
0884 1124 $QIOW,S FUNC=#IOS_SENSEMODE,- : GET TERMINAL CHARACTERISTICS
0884 1125 CHAN=(R2),- :
0884 1126 IOSB=4(R2),- :
0884 1127 EFN=#EXESC_SYSEFN,- :
0884 1128 P1=(R3),- :
0884 1129 P2=#12 :
0884 1130 BLBC R0,90$ : BRANCH IF ERROR
0884 1131 MOVZWL 4(R2),R0 : GET IOSB STATUS
0884 1132 BLBC R0,90$ : BRANCH IF ERROR
0884 1133 :
0884 1134 $DASSGN,S CHAN=(R2) : DEASSIGN TERMINAL CHANNEL
0884 1135 BLBC R0,90$ : BRANCH IF ERROR
0884 1136 :
0884 1137 MOVL #CLIS_TRMMBX,R0 : ASSUME ASSOCIATED TERMINAL MBX
0884 1138 BBS #TT2$V_DCL_MAILBX,8(R3),90$ : EXIT WITH ERROR IF TRUE
0884 1139 MOVL #1,R0 : SET SUCCESS
0884 1140 :
0884 1141 90$: ADDL #8+4+12,SP : RESTORE STACK
0884 1142 MOVQ (SP)+,R2 : RESTORE REGISTERS
0884 1143 RSB :

```

```

08EF 1145 .SBTTL PROCESS SPAWN OUTPUT STREAM
08EF 1146 :---
08EF 1147 :
08EF 1148 THIS ROUTINE IS CALLED TO PROCESS THE SPAWN OUTPUT STREAM.
08EF 1149 :
08EF 1150 INPUTS:
08EF 1151 :
08EF 1152 R6 = ADDRESS OF SPAWN BLOCK
08EF 1153 R8 = DESCRIPTOR OF SCRATCH AREA
08EF 1154 R11 = ADDRESS OF PRC AREA
08EF 1155 :
08EF 1156 OUTPUTS:
08EF 1157 :
08EF 1158 R0 = STATUS CODE
08EF 1159 :---
08EF 1160
08EF 1161 VERIFY_OUTPUT:
08EF 1162 :
08EF 1163 IF OUTPUT FILE SPECIFICATION NOT SPECIFIED, THEN USE CURRENT SYSS$OUTPUT
08EF 1164 :
08EF 1165 BBS #SPWN_V_OUTPUT, - ; IF OUTPUT SPECIFIED,
08F1 1166 SPWN_Q_FLAGS(R6), 10$ ; THEN USE IT
28 A6 0C 0C A6 9A 08F4 1167 MOVZBL SYSS$OUTPUT, SPWN_Q_OUTPUT(R6) ; ELSE USE "SYSS$OUTPUT"
2C A6 F74C CF 9E 08FA 1168 MOVAB SYSS$OUTPUT+1, SPWN_Q_OUTPUT+4(R6);
0900 1169
0900 1170 :
0900 1171 CHECK THAT OUTPUT FILE SPECIFICATION IS NOT TOO LONG.
0900 1172 :
50 00038200 8F D0 0900 1173 10$: MOVL #CLIS_INVFILSPE, R0 ; ASSUME INVALID FILE SPEC
00FF 8F 28 A6 B1 0907 1174 CMPW SPWN_Q_OUTPUT(R6), #NAMSC_MAXRSS ; NAME TOO LONG?
5C 1A 090D 1175 BGTRU 25$ ; IF SO, EXIT WITH ERROR
090F 1176
090F 1177 :
090F 1178 ALLOCATE FAB AND NAM BLOCKS AND THEN SPARSE THE OUTPUT FILE SPECIFICATION.
090F 1179 :
5E FF50 CE 9E 090F 1180 MOVAB -FABSC_BLN-NAMSC_BLN(SP), SP ; ALLOCATE FAB AND NAM BLOCKS
00 6E 00 2C 0914 1181 MOVCS #0, (SP), #0, - ; ZERO FAB AND NAM
6E 00B0 8F 0918 1182 #FABSC_BLN+NAMSC_BLN, (SP)
54 5E D0 091C 1183 MOVL SP, R4 ; GET BASE OF FAB
55 50 A4 9E 091F 1184 MOVAB FABSC_BLN(R4), R5 ; GET BASE OF NAM
0923 1185
0923 1186 ASSUME FABSB_BLN EQ FABSB_BID+1 ; INIT FAB
5003 8F B0 0923 1187 MOVW #FABSC_BID+<FABSC_BLN@8>, - ; SET FAB ID
64 0927 1188 FABSB_BID(R4)
0928 1189 SETBIT FABSV_PPF, FABSL_FOP(R4) ; USE PROCESS I/O SEGMENT
34 A4 28 A6 90 092D 1190 MOVB SPWN_Q_OUTPUT(R6), FABSB_FNS(R4) ; SPECIFY FILE NAME
2C A4 2C A6 D0 0932 1191 MOVL SPWN_Q_OUTPUT+4(R6), FABSL_FNA(R4);
35 A4 F72A CF 90 0937 1192 MOVB LOG, FABSB_DNS(R4) ; SPECIFY DEFAULT FILE TYPE
30 A4 F725 CF 9E 093D 1193 MOVAB LOG+1, FABSL_DNA(R4) ;
28 A4 55 D0 0943 1194 MOVL R5, FABSL_NAM(R4) ; SPECIFY NAM BLOCK ADDRESS
0947 1195
0947 1196 ASSUME NAMSB_BLN EQ NAMSB_BID+1 ; INIT NAM
6002 8F B0 0947 1197 MOVW #NAMSC_BID+<NAMSC_BLN@8>, - ; SET NAM ID
65 094B 1198 NAMSB_BID(R5)
08 A5 10 90 094C 1199 MOVB #NAMSM_NOCONCEAL, NAMSB_NOP(R5) ; GET PAST DOUBLE UNDERSCORES
0A A5 FF 8F 90 0950 1200 MOVB #NAMSC_MAXRSS, NAMSB_ESS(R5) ; SPECIFY RESULT BUFFER
0C A5 04 A8 D0 0955 1201 MOVL 4(R8), NAMSL_ESA(R5) ;

```

```

08 50 E8 095A 1202 SPARSE FAB=(R4) ; PARSE THE FILE SPEC
0963 1203 BLBS RO,30$ ; CONTINUE IF SUCCESSFUL
0966 1204
SE 00B0 CE 9E 0966 1205 20$: MOVAB FAB$C_BLN+NAM$C_BLN(SP),SP ; DEALLOCATE FAB AND NAM
0092 31 096B 1206 25$: BRW 95$ ; EXIT WITH ERROR
096E 1207
096E 1208
096E 1209 : PROCESS NON-PPF OUTPUT HERE. IF NOT RECORD-ORIENTED, THEN USE EXPANDED
096E 1210 : STRING AS OUTPUT. IF RECORD-ORIENTED THEN USE DEV FIELD AS OUTPUT.
096E 1211
28 34 A5 10 E0 096E 1212 30$: BBS #NAM$V_PPF,NAM$C_FNB(R5),50$ ; BRANCH IF PPF FILE
50 0B A5 9A 0973 1213 MOVZBL NAM$B_ESL(R5),RO ; GET LENGTH OF RESULT STRING
68 50 C2 0977 1214 SUBL RO,(R8) ; PERMANENTLY REMOVE RESULT
04 A8 50 C0 097A 1215 ADDL RO,4(R8) ; FROM SCRATCH AREA
0C 40 A4 00 E1 097E 1216 #DEV$V_REC,FAB$C_DEV(R4),40$ ; BRANCH IF NOT REC-ORIENTED
28 A6 39 A5 9A 0983 1217 MOVZBL NAM$B_DEV(R5),SPWN_Q_OUTPUT(R6) ; USE DEVICE AS OUTPUT
2C A6 44 A5 D0 0988 1218 MOVL NAM$C_DEV(R5),SPWN_Q_OUTPUT+4(R6);
40 11 098D 1219 BRB 56$ ; PROCESS AS PPF DEVICE
098F 1220
098F 1221
098F 1222 : OUTPUT IS A NON-PPF NON-RECORD-ORIENTED DEVICE. USE THE ESS AS OUTPUT.
098F 1223
28 A6 0B A5 9A 098F 1224 40$: MOVZBL NAM$B_ESL(R5),SPWN_Q_OUTPUT(R6) ; USE ESS AS OUTPUT FILE SPEC
2C A6 0C A5 D0 0994 1225 MOVL NAM$C_ESA(R5),SPWN_Q_OUTPUT+4(R6);
49 11 0999 1226 BRB 90$ ; DONE WITH PARSING
099B 1227
099B 1228
099B 1229 : OUTPUT IS PPF. IF FROM A RECORD-ORIENTED DEVICE, DO $GETDVI TO GET
099B 1230 : THE DEVICE NAME. OTHERWISE, IF EXPLICITLY SPECIFIED THEN RETURN AN
099B 1231 : ERROR ELSE CREATE A MAILBOX THAT WILL CAUSES WRITES TO GO TO THE PARENT'S
099B 1232 : OUTPUT STREAM.
099B 1233
50 0E 0C A6 0B E1 099B 1234 50$: BBC #SPWN_V_OUTPUT,SPWN_W_FLAGS(R6),51$ ; SKIP IF IMPLICIT OUTPUT
000388EA 8F D0 09A0 1235 MOVL #CLIS$-SPWNIO,RO ; ASSUME ILLEGAL OUTPUT FILE
BA 40 A4 00 E1 09A7 1236 BBC #DEV$V_REC,FAB$C_DEV(R4),20$ ; ERROR IF NOT REC-ORIENTED
17 11 09AC 1237 BRB 52$
09AE 1238
09AE 1239
09AE 1240 : IF AT CTRL/Y LEVEL, THEN DON'T USE SYS$OUTPUT. USE THE OUTPUT STREAM
09AE 1241 : SPECIFIED AT PROCESS CREATION TIME.
09AE 1242
12 68 0B E1 09AE 1243 51$: BBC #PRC_V_YLEVEL,- ; IF CTRL/Y LEVEL,
0118 AB 09B0 1244 PRC_Q_FLAGS(R11),52$ ; THEN SPECIAL CASE SYS$OUTPUT
40 CB D0 09B3 1245 MOVL PRC_L_OUTRABCTX(R11),- ; GET DEVICE CHARACTERISTICS
51 011C CB 9E 09B7 1246 FAB$C_DEV(R4)
50 81 9A 09B9 1247 MOVAB PRC_T_OUTDVI(R11),R1 ; GET DEVICE NAME DESCRIPTOR
28 A6 50 7D 09BE 1248 MOVZBL (R1)+,RO ;
09C1 1249 MOVA RO,SPWN_Q_OUTPUT(R6) ; SAVE DEVICE NAME DESCRIPTOR
09C5 1250
09C5 1251
09C5 1252 : PROCESS LEGAL NON-RECORD-ORIENTED OUTPUT.
09C5 1253
05 40 A4 00 E0 09C5 1254 52$: BBS #DEV$V_REC,FAB$C_DEV(R4),56$ ; ERROR IF NOT REC-ORIENTED
083B 30 09CA 1255 BSBW CREATE_OUTMBX ; CREATE OUTPUT MAILBOX
1A 11 09CD 1256 BRB 91$ ; DO NOT SET OUTPUT BIT
09CF 1257
09CF 1258

```

```

09CF 1259 : PROCESS RECORD-ORIENTED OUTPUT.
09CF 1260 :
52 28 A6 7D 09CF 1261 56$: MOVQ SPWN_Q_OUTPUT(R6),R2 : COPY DESCRIPTOR OF DEVICE NAME
54 68 7D 09D3 1262 : MOVQ (R8),R4 : COPY DESCRIPTOR OF SCRATCH AREA
07ED 30 09D6 1263 : BSBW GET_DEVICE : GET DEVICE NAME
28 A6 54 7D 09D9 1264 : MOVQ R4,SPWN_Q_OUTPUT(R6) : SAVE EQUIVALENCE STRING
68 54 A2 09DD 1265 : SUBW R4,(R8) : PERMANENTLY REMOVE RESULT
04 A8 54 C0 09E0 1266 : ADDL R4,4(R8) : FROM SCRATCH AREA
09E4 1267 :
09E4 1268 :
09E4 1269 : CHECK THAT OUTPUT FILE SPEC IS NOT TOO LONG FOR $CREPRC TO HANDLE.
09E4 1270 :
09E4 1271 90$: SETBIT SPWN_V_OUTPUT,SPWN_W_FLAGS(R6) : SET NO MAILBOX IN USE
5E 00B0 CE 9E 09E9 1272 91$: MOVAB FAB$C_BLN+NAM$C_BLN(SP),SP : DEALLOCATE FAB AND NAM
50 00038200 8F D0 09EE 1273 : MOVL #CLIS_INVFILSPEC,R0 : ASSUME SPEC IS TOO LONG
28 A6 00FF 8F B1 09F5 1274 : CMPW #LNMSC_NAMLENGTH,SPWN_Q_OUTPUT(R6) : IS IT REALLY?
03 1F 09FB 1275 : BLSSU 95$ : YES, THEN ERROR
09FD 1276 :
09FD 1277 :
09FD 1278 : EXIT WITH STATUS
09FD 1279 :
50 01 D0 09FD 1280 : MOVL #1,R0 : SET NORMAL STATUS
05 0A00 1281 95$: RSB : EXIT WITH STATUS

```

```

OA01 1283 .SBTTL ATTACH COMMAND
OA01 1284 :+
OA01 1285 : DCL$ATTACH - ATTACH COMMAND
OA01 1286 :
OA01 1287 : THIS ROUTINE IS CALLED TO EXECUTE THE DCL ATTACH COMMAND. THE ATTACH
OA01 1288 : COMMAND TRANSFERS EXCLUSIVE CONTROL OF THE TERMINAL FROM THIS PROCESS
OA01 1289 : TO ANOTHER PROCESS BY MUTUAL COOPERATION. A MESSAGE IS SENT TO THE
OA01 1290 : 'DESTINATION' PROCESS'S ATTACH REQUEST MAILBOX ASKING IF IT WANTS TO
OA01 1291 : HAVE CONTROL OF THE TERMINAL. THE DESTINATION PROCESS RESPONDS WITH
OA01 1292 : A YES/NO REPLY DEPENDING ON ITS STATE. IF IT RESPONDS WITH YES, THEN
OA01 1293 : WE MARK OURSELVES DETACHED, AND WAIT FOR A RE-ATTACH REQUEST.
OA01 1294 :
OA01 1295 : INPUTS:
OA01 1296 :
OA01 1297 : R10 = ADDRESS OF COMMAND WORK AREA
OA01 1298 : R11 = ADDRESS OF PROCESS WORK AREA
OA01 1299 :
OA01 1300 :
OA01 1301 : OUTPUTS:
OA01 1302 :
OA01 1303 : R0 = STATUS CODE
OA01 1304 : -
OA01 1305 :
OA01 1306 DCL$ATTACH::
OA01 1307 :
OA01 1308 : GET THE PROCESS NAME OR PID FROM THE COMMAND LINE
OA01 1309 :
56 D4 OA01 1310 CLRL R6 : MARK NO PROCESS NAME SUPPLIED YET
58 D4 OA03 1311 CLRL R8 : MARK NO PID SUPPLIED YET
F5F8' 30 OA05 1312 40$: BSBW DCL$GETDVAL : GET NEXT TOKEN
38 50 E9 OA08 1313 BLBC R0,50$ : BRANCH IF END OF LINE
00 55 D1 OA0B 1314 CMPL R5,#PTR_K_COMDQUAL : VERB QUALIFIER?
0A 13 OA0E 1315 BEQL 42$ : BRANCH IF SO
03 55 D1 OA10 1316 CMPL R5,#PTR_K_PARAMETR : PARAMETER (PROCESS NAME)?
F0 12 OA13 1317 BNEQ 40$ : IF NOT, IGNORE IT
56 51 7D OA15 1318 MOVQ R1,R6 : SAVE DESCRIPTOR OF PROCESS NAME
EB 11 OA18 1319 BRB 40$
F5E3' 30 OA1A 1320 42$: BSBW DCL$GETNVAL : GET QUALIFIER NUMBER
00'8F 51 91 OA1D 1321 CMPB R1,#CLISK_ATTA_IDEN : /IDENTIFICATION?
E2 12 OA21 1322 BNEQ 40$ : IF NOT, IGNORE IT
02 54 91 OA23 1323 CMPB R4,#PTR_K_COLON : IS A VALUE PRESENT?
DD 12 OA26 1324 BNEQ 40$ : IF NOT, USE DEFAULT
F5D5' 30 OA28 1325 BSBW DCL$GETDVAL : GET /IDENT VALUE
52 51 7D OA2B 1326 MOVQ R1,R2 : PASS DESCRIPTOR OF VALUE STRING
51 00 9A OA2E 1327 MOVZBL #PRC_K_HEX,R1 : DEFAULT RADIX = HEX
F5CC' 30 OA31 1328 BSBW DCL$CNVASCBIN : CONVERT TO BINARY
05 12 OA34 1329 BNEQ 45$ : BRANCH IF CONVERSION ERROR
58 51 D0 OA36 1330 MOVL R1,R8 : SAVE PID AWAY
CA 11 OA39 1331 BRB 40$
05 OA3B 1332 45$: STATUS EXPSYN : EXPRESSION SYNTAX ERROR
0A 10 OA43 1333 RSB
06 50 E9 OA45 1335 50$: BSBW DCL$ATTACH2 : PERFORM ACTUAL ATTACH OPERATION
03EB 30 OA48 1336 BLBC R0,90$ : BRANCH IF ERROR DETECTED
50 01 D0 OA4B 1337 BSBW RETURNED_MESSAGE : WRITE MESSAGE SAYING WE HAVE RETURNED
05 OA4E 1338 90$: MOVL #1,R0 : INDICATE SUCCESS
RSB : RETURN WITH STATUS

```

```

0A4F 1340 .SBTTL PERFORM ATTACH OPERATION
0A4F 1341 :---
0A4F 1342 :
0A4F 1343 : ALTERNATE ENTRY POINT, USED BY ATTACH CLI CALLBACK. NO USE OF THE
0A4F 1344 : WRK AREA MUST BE MADE AFTER THIS POINT, SINCE CLI CALLBACKS OPERATE
0A4F 1345 : WITHOUT ANY SUCH AREA.
0A4F 1346 :
0A4F 1347 : INPUTS:
0A4F 1348 :
0A4F 1349 : R6/R7 = DESCRIPTOR OF PROCESS NAME
0A4F 1350 : R8 = PID OF DESTINATION PROCESS
0A4F 1351 : R11 = ADDRESS OF PRC AREA
0A4F 1352 :
0A4F 1353 : EITHER THE PID OR PROCESS NAME MUST BE SPECIFIED.
0A4F 1354 :
0A4F 1355 : OUTPUTS:
0A4F 1356 :
0A4F 1357 : R0 = FINAL STATUS
0A4F 1358 :
0A4F 1359 :---
0A4F 1360 :
0A4F 1361 DCL$ATTACH2::
0A4F 1362 :
0A4F 1363 : CONTROL/Y AST'S ARE DISABLED THROUGHOUT THIS COMMAND, TO ENSURE THAT
0A4F 1364 : MAILBOXES WHICH ARE CREATED HERE ARE CORRECTLY DELETED, ETC. AS A
0A4F 1365 : RESULT, WE MUST PERIODICALLY CHECK THE HANGUP FLAG IN CASE A HANGUP
0A4F 1366 : IS DETECTED WHILE WE ARE OPERATING.
0A4F 1367 :
00B4 CB DD 0A4F 1368 PUSHL PRC_L_OUTOFBAND(R11) ; SAVE OUT-OF-BAND ENABLE MASK
51 D4 0A53 1369 CLRL R1 ; DISABLE ALL OUT-OF-BAND AST'S
F5A8' 30 0A55 1370 BSBW DCL$RESET00B
0A58 1371 :
0A58 1372 :
0A58 1373 : LOOKUP THE DESTINATION PROCESS, AND IF PRESENT, GET THE PID AND MPID
0A58 1374 :
7E 7C 0A58 1375 CLRQ -(SP) ; ALLOCATE AN IOSB
7E 7C 0A5A 1376 CLRQ -(SP) ; CREATE GETJPI ITEM LIST
F8 AE 9F 0A5C 1377 PUSHAB -2*4(SP) ; SET BUFFER ADDRESS
03190004 8F DD 0A5F 1378 PUSHL #JPI$_PID@16+4 ; REQUEST PID, SET BUFFER LENGTH
7E D4 0A65 1379 CLRL -(SP) ; SETUP MPID REQUEST ITEM
F8 AE 9F 0A67 1380 PUSHAB -2*4(SP) ; SET BUFFER ADDRESS
03250004 8F DD 0A6A 1381 PUSHL #JPI$_MASTER_PID@16+4 ; REQUEST MPID, SET BUFFER LENGTH
01C0 8F BB 0A70 1382 PUSHR #*M<R6,R7,R8> ; PUSH PROCESS NAME DESCRIPTOR, PID
50 5E D0 0A74 1383 MOVL SP,R0 ;
0A77 1384 $GETJPIW S ITMLST=12(R0),- ; GET PID OF THE PROCESS
0A77 1385 IOSB=40(R0),- ;
0A77 1386 EFN=#EXESC $YSEFN,- ;
0A77 1387 PRCNAM=(R0),- ;
0A77 1388 PIDADR=8(R0) ;
5E 0C C0 0A91 1389 ADDL #3*4,SP ; POP PRCNAM DESCRIPTOR, PID
53 8ED0 CA94 1390 POPL R3 ; GET THE DESTINATION MPID
5E 08 C0 0A97 1391 ADDL #2*4,SP ; POINT TO RETURNED DESTINATION PID
52 8ED0 CA9A 1392 POPL R2 ; GET THE DESTINATION PID
5E 0C C0 0A9D 1393 ADDL #3*4,SP ; POP ITEM LIST
06 50 E9 0AA0 1394 BLBC R0,49$ ; BRANCH IF ERROR
50 6E 3C 0AA3 1395 MOVZWL (SP),R0 ; GET IOSB STATUS
06 50 E8 0AA6 1396 BLBS R0,55$ ; BRANCH IF PROCESS FOUND

```

```

5E 08 C0 0AA9 1397 49$: ADDL #8,SP ; POP THE IOSB
0220 31 0AAC 1398 BRW 95$ ; REPORT ERROR
0AAF 1399
0AAF 1400
0AAF 1401 : MAKE SURE WE AREN'T TRYING TO ATTACH TO OURSELVES
0AAF 1402
0AAF 1403 55$: CLRQ -(SP) ; CREATE GETJPI ITEM LIST
F8 AE 9F 0AB1 1404 PUSHAB -2*4(SP) ; SET BUFFER ADDRESS
03190004 8F DD 0AB4 1405 PUSHL #JPI$_PID@16+4 ; REQUEST OUR PID, SET BUFFER LENGTH
7E D4 0ABA 1406 CLRL -(SP) ; SETUP MPID REQUEST ITEM
F8 AE 9F 0ABC 1407 PUSHAB -2*4(SP) ; SET BUFFER ADDRESS
03250004 8F DD 0ABF 1408 PUSHL #JPI$_MASTER_PID@16+4 ; REQUEST OUR MPID, SET BUFFER LENGTH
50 5E D0 0AC5 1409 MOVL SP,R0 ;
0AC8 1410 $GETJPIW S ITMLST=(R0),- ; GET PID OF THIS PROCESS
0AC8 1411 IOSB=28(R0),- ;
0AC8 1412 EFN=#EXESC_SYSEFN ;
5E 54 8ED0 JAE0 1413 POPL R4 ; GET OUR MPID
08 C0 0AE3 1414 ADDL #2*4,SP ; POINT TO OUR RETURNED PID
51 51 8ED0 0AE6 1415 POPL R1 ; GET OUR PID
5E 0C C0 0AE9 1416 ADDL #3*4,SP ; CLEANUP STACK
000388C2 8F D0 0AEC 1417 MOVL #CLIS_REFUSED,R0 ; ASSUME AN ERROR
52 51 D1 0AF3 1418 CMPL R1,R2 ; TRYING TO ATTACH TO OURSELF?
B1 13 0AF6 1419 BEQL 49$ ; IF SO, REJECT THE OPERATION
54 53 D1 0AF8 1420 CMPL R3,R4 ; TRYING TO ATTACH TO A PROCESS OUTSIDE
0AFB 1421 OF OUR JOB?
0AFB 1422 BNEQ 49$ ; IF SO, REJECT THE OPERATION
5E 08 C0 0AFD 1423 ADDL #8,SP ; POP THE IOSB
0B00 1424
0B00 1425 :
0B00 1426 : CREATE ATTACH$PID LOGICAL NAME FOR DESTINATION ATTACH MAILBOX
0B00 1427 :
5E 1E C2 0B00 1428 SUBL #30,SP ; ALLOCATE ROOM FOR NAME
5E DD 0B03 1429 PUSHL SP ; CREATE DESCRIPTOR FOR NAME
1E DD 0B05 1430 PUSHL #30 ;
51 F511 CF 9E 0B07 1431 MOVAB ATTACH_NAME,R1 ; GET ADDRESS OF ASCII FAO STRING
50 81 9A 0B0C 1432 MOVZBL (R1)+,R0 ; CONSTRUCT DESCRIPTOR OF STRING
7E 50 7D 0B0F 1433 MOVQ R0,-(SP) ; PUSH DESCRIPTOR ONTO STACK
50 5E D0 0B12 1434 MOVL SP,R0 ;
0B15 1435 $FAO_S CTRSTR=(R0),- ; CONSTRUCT LOGICAL NAME
0B15 1436 OUTBUF=8(R0),- ;
0B15 1437 OUTLEN=8(R0),- ;
0B15 1438 P1=R2 ; FROM PID FOR THIS PROCESS
5E 08 C0 0B26 1439 ADDL #8,SP ; POP FAO STRING DESCRIPTOR
0B29 1440
0B29 1441 :
0B29 1442 : GET SYSS$INPUT TRANSLATION (TERMINAL DEVICE NAME).
0B29 1443 : IF WE ARE AT CTRL/Y LEVEL, USE PROCESS PERMANENT SYSS$INPUT INSTEAD.
0B29 1444 :
54 10 3C 0B29 1445 60$: MOVZWL #ATTMBX_MAXMSG,R4 ; SET SIZE OF SCRATCH BUFFER
5E 54 C2 0B2C 1446 SUBL R4,SP ; ALLOCATE SCRATCH BUFFER FOR STRING
55 5E D0 0B2F 1447 MOVL SP,R5 ; CREATE DESCRIPTOR OF BUFFER
7E 52 7D 0B32 1448 MOVQ R2,-(SP) ; SAVE R2/R3
0B35 1449 BBC #PRC_V_YLEVEL,- ; IF CTRL/Y LEVEL,
09 68 AB 0B37 1450 PRC_Q_FLAGS(R11),62$ ; THEN SPECIAL CASE SYSS$OUTPUT
53 00000028 8F D0 0B3A 1451 MOVL #CT$AG_CLIDATA+PPD$T_INPDVI,R3 ; GET ADDRESS OF ASCII DEVICE NAME
05 11 0B41 1452 BRB 64$ ; SKIP SYSS$INPUT PROCESSING
53 F4F3 CF 9E 0B43 1453 62$: MOVAB SYSS$INPUT,R3 ; CREATE SYSS$INPUT DESCRIPTOR

```

```

52 83 9A 0B48 1454 64$: MOVZBL (R3)+,R2
0678 30 0B4B 1455 BSBW GET DEVICE
5C 50 E9 0B4E 1456 BLBC R0,651$
0B51 1457 : GET INPUT DEVICE NAME
0B51 1458 : BRANCH IF ERROR
0B51 1459 :
: CHECK IF WE ARE A TERMINAL
7E 7E 7C 0B51 1460 CLRQ -(SP)
51 52 7D 0B53 1461 MOVQ R2,-(SP)
51 5E D0 0B56 1462 MOVL SP,R1
7E 7C 0B59 1463 CLRQ -(SP)
F8 AE 9F 0B5B 1464 PUSHAB -2*4(SP)
00040004 8F D0 0B5E 1465 PUSHL #DVIS_DEVCLASS@16+4
50 5E D0 0B64 1466 MOVL SP,R0
0B67 1467 $GETDVI S,DEVNAM=(R1),-
0B67 1468 IOSB=8(R1),-
0B67 1469 EFN=#EXESC SYSEFN,-
0B67 1470 ITMLST=(R0)
51 6E D0 0B82 1471 MOVL (SP),R1
5E 20 C0 0B85 1472 ADDL #4*8,SP
22 50 E9 0B88 1473 BLBC R0,651$
00000042 8F 51 D1 0B8D 1474 CLRL R0
02 13 0B94 1475 CMPL R1,#DCS_TERM
50 50 D6 0B96 1476 BEQL 641$
52 8E 7D 0B98 1477 INCL R0
0B98 1478 641$: MOVQ (SP)+,R2
0B98 1479
0B98 1480 :
0B98 1481 : CHECK FOR AN ASSOCIATED TERMINAL MAILBOX.
0B98 1482 :
7E 54 7D 0B9B 1483 MOVQ R4,-(SP)
51 5E D0 0B9E 1484 MOVL SP,R1
13 50 E8 0BA1 1485 BLBS R0,66$
0BA4 1486
FCDD 30 0BA4 1487 BSBW CHECK TRMMBX
OD 50 E8 0BA7 1488 BLBS R0,66$
0BAA 1489
011F 31 0BAA 1490 65$: BRW 90$
50 000388C2 8F D0 0BAD 1491 651$: MOVL #CLIS_REFUSED,R0
0115 31 0BB4 1492 BRW 90$
0BB7 1493
0BB7 1494 :
0BB7 1495 : CREATE ATTACH MAILBOX FOR THIS PROCESS TO RECEIVE RE-ATTACH REQUESTS.
0BB7 1496 : IT IS DONE NOW, SO THAT ANY ERRORS DETECTED WILL CAUSE THE ENTIRE
0BB7 1497 : OPERATION TO FAIL, RATHER THAN SENDING THE ATTACH REQUEST TO THE
0BB7 1498 : OTHER PROCESS, AND THEN FINDING OUT WE CAN'T CREATE THE RE-ATTACH
0BB7 1499 : MAILBOX.
0BB7 1500 :
0128 30 0BB7 1501 66$: BSBW CREATE_ATTMBX
ED 50 E9 0BBA 1502 BLBC R0,65$
0BBD 1503 :
0BBD 1504 :
0BBD 1505 : ASSIGN CHANNEL TO ATTACH MAILBOX
0BBD 1506 :
50 7E DE 0BBD 1507 67$: MOVAL -(SP),R0
ORCO 1508 $ASSIGN S,DEVNAM=4*8+ATTMBX_MAXMSG(R0),-
0BC0 1509 CHAN=(R0)
52 8ED0 0BCE 1510 POPL R2
: ADDRESS OF LOGICAL NAME DESCRIPTOR
: ASSIGN CHANNEL TO MAILBOX
: GET CHANNEL OF MAILBOX

```

```

68 50 E9 OBD1 1511 BLBC R0,691$ ; BRANCH IF ERROR
OBD4 1512
OBD4 1513
OBD4 1514 : SEND REQUEST TO DESTINATION ATTACH REQUEST MAILBOX AND GET REPLY
OBD4 1515
54 05E6 30 OBD4 1516 68$: BSBW CHECK_FOR_HANGUP ; CHECK FOR HANGUP AST
7E 7E OBD7 1517 MOVAQ -(SP),R4 ; ALLOCATE IOSB FOR I/O REQUESTS
OBD4 1518 $QIOW_S FUNC=#IOS_WRITEVBLK,- ; WRITE TO MAILBOX
OBD4 1519 CHAN=R2,-
OBD4 1520 EFN=#EXESC_SYSEFN,-
OBD4 1521 IOSB=(R4),-
OBD4 1522 P1=@12(R4),P2=8(R4) ; RECORD = SYSSINPUT TRANSLATION
3B 50 E9 OBF8 1523 BLBC R0,69$ ; BRANCH IF ERROR
50 64 3C OBF8 1524 MOVZWL (R4),R0 ; GET FINAL STATUS FROM IOSB
35 50 E9 OC01 1525 BLBC R0,69$ ; BRANCH IF ERROR
53 7E 7E OC04 1526 MOVAQ -(SP),R3 ; ALLOCATE RECORD BUFFER
04 A3 5B D0 OC07 1527 MOVL R11,4(R3) ; GET PRC AREA ADDRS.
OC08 1528 $QIOW_S FUNC=#IOS_READVBLK,- ; READ RESPONSE FROM MAILBOX
OC08 1529 CHAN=R2,-
OC08 1530 EFN=#EXESC_SYSEFN,-
OC08 1531 IOSB=(R4),-
OC08 1532 ASTADR=READ_AST,-
OC08 1533 ASTPRM=R3,-
OC08 1534 P1=(R3),P2=#4 ; TO LONGWORD BUFFER
53 8ED0 OC30 1535 POPL R3 ; POP RECORD BUFFER
03 50 E9 OC33 1536 BLBC R0,69$ ; BRANCH IF ERROR
50 64 3C OC36 1537 MOVZWL (R4),R0 ; GET FINAL STATUS FROM IOSB
5E 0C C0 OC39 1538 69$: ADDL #12,SP ; POP THE IOSB
7C 50 E9 OC3C 1539 69$: BLBC R0,80$ ; BRANCH IF ERROR
OC3F 1540 $DASSGN_S CHAN=R2 ; DEASSIGN CHANNEL TO ATTACH MAILBOX
OC49 1541
OC49 1542 : IF DESTINATION PROCESS SAYS NO TO ATTACH REQUEST, REPORT IT TO THE USER
OC49 1543
OC49 1544
79 53 E9 OC49 1545 BLBC R3,85$ ; BRANCH IF DESTINATION PROCESS SAYS NO
OC4C 1546
OC4C 1547
OC4C 1548
OC4C 1549 : WAIT FOR PROCESS TO RETURN CONTROL HERE (FROM ATTACH OR TERMINATION).
OC4C 1550 : IF WE ARE IN AN INTERACTIVE PROCESS, THEN TELL THE TERMINAL DRIVER THAT
OC4C 1551 : WE NOW OWN THE TERMINAL.
OC4C 1552
OC4C 1553
50 056E 30 OC4C 1554 70$: BSBW CHECK_FOR_HANGUP ; CHECK FOR HANGUP AST
7E 7E OC4F 1555 MOVAQ -(SP),R0 ; CREATE A TEMP. IOSB
OC52 1556 $QIOW_S FUNC=#IOS_SETMODE!IOSM_WRTATTN,- ; SET ATTN. AST ON MAILBOX
OC52 1557 CHAN=PRC @ ATTMBX(R11),-
OC52 1558 EFN=#EXESC_SYSEFN,-
OC52 1559 IOSB=(R0),-
OC52 1560 P1=ATTACH_AST,-
OC52 1561 P2=R11 ; ADDRESS OF AST ROUTINE
5E 08 C0 OC78 1562 ADDL #8,SP ; PASS ADDRESS OF CLI STORAGE
OC7B 1563 $HIBER_S ; CLEAN UP STACK
OC82 1564 BBS #PRC_V_DETACHED,- ; HIBERNATE WAITING FOR WAKEUP
C5 68 AB E0 OC84 1565 BBS PRC @ FLAGS(R11),70$ ; BRANCH IF STILL DETACHED
06 E0 OC87 1566 BBS #PRC_V_MODE,- ; SKIP IF NOT INTERACTIVE
40 68 AB OC89 1567 PRC @ FLAGS(R11),90$

```

```

50  7E  7C  0C8C  1568      CLRQ  -(SP)                ; ALLOCATE AN IOSB
    SE  DO  0C8E  1569      MOVL  SP,R0                ; GET ADDRESS OF IOSB
    0C91  1570      $QIOW,S  FUNC=#IOS,SETMODE!IOSM,TI,PROCESS,- ; ASSUME TERMINAL OWNERSHIP
    0C91  1571      0C91  1572      CHAN=PRC @ INPCHAN(R11),-
    0C91  1573      0C91  1574      IOSB=(R0),-
    SE  08  C0  0CB3  1574      EFN=#EXESC,_SYSEFN
    50  01  DO  0CB6  1575      ADDL  #8,SP                ; RESTORE THE STACK
    11  11  0CB9  1576      MOVL  #1,R0                ; SET SUCCESSFUL
    0CBB  1577      BRB  90$                ; AND EXIT
    0CBB  1578      ;
    0CBB  1579      ; CLEANUP FROM ERROR DETECTED WHILE HAVING CHANNEL OPEN TO ATTACH MAILBOX
    0CBB  1580      ;
    50  000388C2 8F  DO  0CBB  1581 80$: $DASSGN,S CHAN=R2        ; DEASSIGN CHANNEL TO ATTACH MAILBOX
    0CCC  1582 85$: MOVL  -#CLIS,_REFUSED,R0        ; ATTACH REQUEST REFUSED
    0CCC  1583      ;
    0CCC  1584      ;
    0CCC  1585      ; RESTORE PROCESS CONTEXT
    0CCC  1586      ;
    SE  3E  C0  0CCC  1587 90$: ADDL  #8+ATTMBX_MAXMSG+8+30,SP; CLEAN STACK
    56  50  DO  0CCF  1588 95$: MOVL  R0,R6                ; SAVE FINAL STATUS CODE
    0144 30  0CD2  1589      BSBW  DEALLOC_SPWN        ; REMOVE INACTIVE SPWN BLOCKS
    008A 30  0CD5  1590      BSBW  DELETE_ATTMBX        ; DELETE OUR ATTACH MAILBOX
    51 8ED0  0CD8  1591      POPL  R1                ; RESTORE OUT-OF-BAND ENABLE MASK
    F322' 30  0CDB  1592      BSBW  DCL$RESETOOB        ; RESTORE AST'S
    50  56  DO  0CDE  1593      MOVL  R6,R0                ; RESTORE EXIT STATUS
    05  05  0CE1  1594      RSB                    ; EXIT

```

```

OCE2 1596 .SBTTL CREATE ATTACH REQUEST MAILBOX
OCE2 1597 :---
OCE2 1598 :
OCE2 1599 THIS ROUTINE IS CALLED TO CREATE AN ATTACH REQUEST MAILBOX FOR
OCE2 1600 THIS PROCESS TO RECEIVE ATTACH REQUESTS FROM OTHER PROCESSES.
OCE2 1601 :
OCE2 1602 INPUTS:
OCE2 1603 :
OCE2 1604 R11 = ADDRESS OF PRC AREA
OCE2 1605 :
OCE2 1606 OUTPUTS:
OCE2 1607 :
OCE2 1608 RO = STATUS CODE
OCE2 1609 :---
OCE2 1610
OCE2 1611 CREATE_ATTMBX:
OCE2 1612 PUSH R2 ; SAVE REGISTERS
OCE2 1613 CLRQ -(SP) ; ALLOCATE AN IOSB
OCE2 1614 :
OCE2 1615 CREATE ATTACH MAILBOX AND LOGICAL NAME, SHOULD ANY OTHER PROCESS DESIRE
OCE2 1616 TO RE-ATTACH TO THIS PROCESS LATER ON.
OCE2 1617 :
OCE2 1618 CLRQ -(SP) ; CREATE GETJPI ITEM LIST
OCE2 1619 PUSHAB -2*4(SP) ; SET BUFFER ADDRESS
OCE2 1620 PUSHL #JPI$PID@16+4 ; REQUEST OUR PID, SET BUFFER LENGTH
OCE2 1621 MOVL SP,RO ;
OCE2 1622 $GETJPIW S ITMLST=(RO),- ; GET PID OF THIS PROCESS
OCE2 1623 IOSB=16(RO),- ;
OCE2 1624 EFN=#EXESC_SYSEFN ;
OCE2 1625 POPL R2 ; GET OUR PID
OCE2 1626 ADDL #3*4,SP ; CLEANUP STACK
OCE2 1627 SUBL #30,SP ; ALLOCATE ROOM FOR NAME
OCE2 1628 PUSHL SP ; CREATE DESCRIPTOR FOR NAME
OCE2 1629 PUSHL #30 ;
OCE2 1630 MOVAB ATTACH_NAME,R1 ; GET ADDRESS OF ASCII FAO STRING
OCE2 1631 MOVZBL (R1)+,RO ; CONSTRUCT DESCRIPTOR OF STRING
OCE2 1632 MOVQ RO,-(SP) ; PUSH DESCRIPTOR ONTO STACK
OCE2 1633 MOVL SP,RO ;
OCE2 1634 $FAO_S CTRSTR=(RO),- ; CONSTRUCT LOGICAL NAME
OCE2 1635 OUTBUF=8(RO),- ;
OCE2 1636 OUTLEN=8(RO),- ;
OCE2 1637 P1=R2 ; FROM PID FOR THIS PROCESS
OCE2 1638 ADDL #8,SP ; POP FAO STRING DESCRIPTOR
OCE2 1639 MOVL SP,RO ; ADDRESS OF LOGICAL NAME DESCRIPTOR
OCE2 1640 $CREMBX S (CHAN=PRC W ATTMBX(R11),- ; CREATE ATTACH MAILBOX
OCE2 1641 -BUFQUO=#ATTMBX_MAXMSG,- ; ONLY NEEDS TO HOLD 1 MESSAGE
OCE2 1642 PROMSK=#^B1111T11100001111,- ; ONLY GIVE OWNER R/W ACCESS
OCE2 1643 LOGNAM=(RO) ;
OCE2 1644 ADDL #30+8,SP ; CLEANUP STACK
OCE2 1645 ADDL #8,SP ; POP THE IOSB
OCE2 1646 POPL R2 ; RESTORE REGISTERS
OCE2 1647 RSB

```

52 DD OCE2 1612
7E 7C OCE4 1613
OCE6 1614
OCE6 1615
OCE6 1616
OCE6 1617
OCE6 1618
F8 AE 9F OCE8 1619
03190004 8F DD OCEB 1620
50 5E DO OCF1 1621
OCE4 1622
OCE4 1623
OCE4 1624
52 8ED0 ODOC 1625
5E OC CO ODOF 1626
5E 1E C2 OD12 1627
5E DD OD15 1628
1E DD OD17 1629
51 F2FF CF 9E OD19 1630
50 81 9A OD1E 1631
7E 50 7D OD21 1632
50 5E DO OD24 1633
OD27 1634
OD27 1635
OD27 1636
OD27 1637
5E 08 CO OD38 1638
50 5E DO OD3B 1639
OD3E 1640
OD3E 1641
OD3E 1642
OD3E 1643
5E 26 CO OD58 1644
5E 08 CO OD5B 1645
52 8ED0 OD5E 1646
05 OD61 1647

90\$:

```

OD62 1649      .SBTTL  DELETE ATTACH REQUEST MAILBOX
OD62 1650      :---
OD62 1651      :
OD62 1652      : THIS ROUTINE IS CALLED TO DELETE THE ATTACH REQUEST MAILBOX
OD62 1653      : FOR THIS PROCESS.  THIS IS DONE AFTER WE COME BACK FROM DETACHED
OD62 1654      : STATE, SINCE AFTER THAT POINT, WE CAN NO LONGER ACCEPT ATTACH REQUESTS.
OD62 1655      :
OD62 1656      : INPUTS:
OD62 1657      :
OD62 1658      :      R11 = ADDRESS OF PRC AREA
OD62 1659      :
OD62 1660      : OUTPUTS:
OD62 1661      :
OD62 1662      :      NONE
OD62 1663      :---
OD62 1664      :
OD62 1665      DELETE_ATTMBX:
OD62 1666      $DASSGN_S CHAN=PRC_W_ATTMBX(R11) ;DELETE ATTACH MAILBOX
7A AB  B4 OD6D 1667      CLRW  PRC_W_ATTMBX(R11) ;MARK CHANNEL NO LONGER VALID
05 OD70 1668      RSB

```

```

OD71 1670      .SBTTL CREATE TERMINATION MAILBOX
OD71 1671      :---
OD71 1672      :
OD71 1673      : THIS ROUTINE IS CALLED TO CREATE A TERMINATION MAILBOX FOR
OD71 1674      : THIS PROCESS TO RECEIVE TERMINATION MESSAGE FROM UP TO FOUR OF ITS
OD71 1675      : SPAWNED SUBPROCESSES.
OD71 1676      :
OD71 1677      : INPUTS:
OD71 1678      :
OD71 1679      :     R6 = ADDRESS OF SPWN DATA STRUCTURE
OD71 1680      :     R8 = ADDRESS OF SCRATCH STORAGE
OD71 1681      :     R11 = ADDRESS OF PRC AREA
OD71 1682      :
OD71 1683      : OUTPUTS:
OD71 1684      :
OD71 1685      :     R0 = STATUS CODE
OD71 1686      :     R7 = ADDRESS OF TMBX BLOCK
OD71 1687      : ---
OD71 1688      :
OD71 1689      : CREATE_TMBX:
OD71 1690      :
OD71 1691      :
OD71 1692      : ALLOCATE A NEW TMBX BLOCK AND LINK IT INTO THE LIST.
OD71 1693      :
51 10 3C OD71 1694      : MOVZWL #TMBX_C_LENGTH,R1          ; LENGTH OF STORAGE TO ALLOCATE
      F289' 30 OD74 1695      : BSBW DCL$AELDYNMEM          ; ALLOCATE STORAGE
      7B 50 E9 OD77 1696      : BLBC R0,90$          ; BRANCH ON ERROR
57 52 D0 OD7A 1697      : MOVL R2,R7          ; SET ADDRESS OF TMBX BLOCK
OA A7 51 B0 OD7D 1698      : MOVW R1,TMBX_W_SIZE(R7) ; STORE SIZE OF BLOCK
      08 A7 94 OD81 1699      : CLAB TMBX_B_REI'S(R7)   ; SET REFERENCE COUNT TO ZERO
OC A7 5B D0 OD84 1700      : MOVL R11,TMBX_I_PRC(R7) ; STORE BASE OF PROCESS WORK AREA
      74 AB D0 OD88 1701      : MOVL PRC_L_TMBX(R11),-  ; INSERT BLOCK INTO LINKED LIST
      67
74 AB 57 D0 OD8B 1702      : TMBX_C_LINK(R7)
      OD8C 1703      : MOVL R7,PRC_L_TMBX(R11)
      OD90 1704      :
      OD90 1705      :
      OD90 1706      : CREATE A TRMINATION MAILBOX THAT CAN HOLD AS MANY MESSAGES AS ALLOWED
      OD90 1707      : BROTHERS AND ENABLE THE TERMINATION AST.
      OD90 1708      :
      OD90 1709      : $CREMBX_S CHAN=TMBX_W_CHANNEL(R7),- ; CREATE A TERMINATION MAILBOX
      OD90 1710      : BUFQUO=#TMBX_C_MAXREFS*ACC$C_TERMLEN
      4B 50 E9 ODA8 1711      : BLBC R0,95$          ; BRANCH ON ERROR
      ODA8 1712      : $QIOW_S FUNC=#IOS_SETMODE!IOSM_WRTATTN,- ; SET ATTENTION AST ON MAILBOX
      ODA8 1713      : CHAN=TMBX_W_CHANNEL(R7),-
      ODA8 1714      : IOSB=SPWN_Q_IOSB(R6),-
      ODA8 1715      : EFN=#EXESC_SYSEFN,-
      ODA8 1716      : P1=W^TERMINATION_AST,-
      ODA8 1717      : P2=R7
      23 50 E9 ODD0 1718      : BLBC R0,95$          ; ADDRESS OF AST ROUTINE
50 38 A6 3C ODD3 1719      : MOVZWL SPWN_Q_IOSB(R6),R0 ; PASS TMBX BLOCK ADDRESS
      1C 50 E9 ODD7 1720      : BLBC R0,95$          ; BRANCH IF ERROR
      ODDA 1721      :
      ODDA 1722      :
      ODDA 1723      : GET UNIT NUMBER OF TERMINATION MAILBOX
      ODDA 1724      :
      ODDA 1725      : $GETCHN_S CHAN=TMBX_W_CHANNEL(R7),- ; GET MAILBOX INFORMATION
      ODDA 1726      : PRIBUF=(R8)

```

SPAWN
V04-000

- MULTI-PROCESSING COMMANDS
CREATE TERMINATION MAILBOX

H 14

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 36
(12)

```
04 A7 06 50 E9 ODED 1727 BLBC R0,95$ ; BRANCH ON ERROR
      14 A8 B0 ODF0 1728 MOVW DIB$W_UNIT+8(R8),TMBX_W_UNIT(R7) ; SAVE MAILBOX UNIT NUMBER
      05 ODF5 1729 90$: RSB ; RETURN WITH STATUS
      ODF6 1730
      ODF6 1731
      ODF6 1732 ; DELETE TERMINATION MAILBOX AND TMBX DATA STRUCTURE AND THEN
      ODF6 1733 ; RETURN WITH STATUS
      ODF6 1734
      50 DD ODF6 1735 95$: PUSHL R0 ; SAVE STATUS
0004 30 ODF8 1736 BSBW DELETE_TMBX ; DELETE ALLOCATED TMBX
      50 8ED0 ODFB 1737 POPL R0 ; RESTORE STATUS
      05 ODFF 1738 RSB ; RETURN WITH STATUS
```

```

ODFF 1740      .SBTTL  DELETE TERMINATION MAILBOX
ODFF 1741      :---
ODFF 1742      :
ODFF 1743      : THIS ROUTINE IS CALLED TO DELETE A TERMINATION MAILBOX AND ITS ASSOCIATED
ODFF 1744      : TMBX DATA STRUCTURE.
ODFF 1745      :
ODFF 1746      : INPUTS:
ODFF 1747      :
ODFF 1748      :       R7 = ADDRESS OF TMBX DATA STRUCTURE
ODFF 1749      :       R11 = ADDRESS OF PRC AREA
ODFF 1750      :
ODFF 1751      : OUTPUTS:
ODFF 1752      :
ODFF 1753      :       NONE
ODFF 1754      :---
ODFF 1755      :
ODFF 1756      : DELETE_TMBX:
ODFF 1757      : SDASSGN_S CHAN=TMBX_W_CHANNEL(R7)      ; DELETE TERMINATION MAILBOX
74 AB 67 DO OE0A 1758      : MOVL -TMBX_L_LINK(R7),PRC_L_TMBX(R11) ; REMOVE BLOCK FROM LIST
50 57 DO OE0E 1759      : MOVL R7,R0 ; SET ADDRESS OF BLOCK
51 OA A7 3C OE11 1760      : MOVZWL TMBX_W_SIZE(R7),R1 ; GET SIZE OF BLOCK
    F1E8' 30 OE15 1761      : BSBW DCL$DEADYNMEM ; DEALLOCATE BLOCK
    05 OE18 1762      : RSB ;
    OE19 1763      :

```

```

OE19 1765 .SBTTL DEALLOCATE SPWN BLOCKS
OE19 1766 :---
OE19 1767 :
OE19 1768 : THIS ROUTINE 'S CALLED TO DEALLOCATE THE SPWN BLOCKS OF TERMINATED
OE19 1769 : SUBPROCESSES.
OE19 1770 :
OE19 1771 : INPUTS:
OE19 1772 :
OE19 1773 : R11 = ADDRESS OF PRC AREA
OE19 1774 :
OE19 1775 : OUTPUTS:
OE19 1776 :
OE19 1777 : NONE
OE19 1778 :---
OE19 1779 :
OE19 1780 DEALLOC_SPWN:
50 00C0 CB DE OE19 1781 MOVAL PRC_L_SPWN(R11),R0 : GET ADDR OF FIRST SPWN BLOCK
    51 50 DO OE1E 1782 ASSUME SPWN_L_LINK EQ 0 :
    50 60 DO OE1E 1783 10$: MOVL RO,RT : SAVE ADDR OF PREVIOUS SPWN BLOCK
      OF 13 OE21 1784 MOVL SPWN_L_LINK(R0),R0 : GET NEXT SPWN BLOCK
      07 E0 OE24 1785 BEQL 90$ : BRANCH IF DONE
    F3 0C A0 OE26 1786 BBS #SPWN_V_ACTIVE,- : BRANCH IF STILL IN USE
      60 DO OE28 1787 SPWN_Q_FLAGS(R0),10$ :
      61 OE2B 1788 MOVL SPWN_L_LINK(R0),- : REMOVE BLOCK FROM LIST
    51 04 A0 3C OE2D 1789 SPWN_L_LINK(R1) :
      F1CB' 30 OE2E 1790 MOVZWL SPWN_W_SIZE(R0),R1 : SET LENGTH OF BLOCK
    05 OE32 1791 BSBW DCL$DEADYNMEM : DEALLOCATE BLOCK
    05 OE35 1792 90$: RSB

```

```

OE36 1794 .SBTTL WRITE RETURNED MESSAGE
OE36 1795 :---
OE36 1796 :
OE36 1797 : THIS ROUTINE WRITES A MESSAGE INDICATING THAT TERMINAL CONTROL
OE36 1798 : HAS RETURNED TO THIS PROCESS.
OE36 1799 :
OE36 1800 : INPUTS:
OE36 1801 :
OE36 1802 :     NONE
OE36 1803 :
OE36 1804 : OUTPUTS:
OE36 1805 :
OE36 1806 :     NONE
OE36 1807 : ---
OE36 1808 :
OE36 1809 RETURNED_MESSAGE:
OE36 1810 :
OE36 1811 : OUTPUT MESSAGE SAYING THAT CONTROL OF THE TERMINAL HAS RETURNED
OE36 1812 : TO THIS PROCESS
OE36 1813 :
      5E 10 C2 OE36 1814      SUBL #16,SP      :ALLOCATE SCRATCH BUFFER FOR PRCNAM
      5E 10 DD OE39 1815      PUSHL SP      :CREATE DESCRIPTOR OF BUFFER
      50 5E DD OE3B 1816      PUSHL #16
      7E 50 DO OE3D 1817      MOVL SP,R0
      50 7E D4 OE40 1818      CLRL -(SP)      :MARK END OF JPI LIST
      7E 50 DD OE42 1819      PUSHL R0      :ADDRESS TO STORE RETURN LENGTH
      02 AE 7E 60 7D OE44 1820      MOVQ (R0),-(SP) :DESCRIPTOR OF STRING BUFFER
      031C 8F 80 OE47 1821      MOVW #JPI$_PRCNAM,2(SP) :SET PARAMETER ID TO RETRIEVE
      7E 7C OE4D 1822      CLRG -(SP) :ALLOCATE AN IOSB
      50 5E DO OE4F 1823      MOVL SP,R0
      OE52 1824      $GETJPIW S ITMLST=8(R0),- :GET CURRENT PROCESS NAME
      OE52 1825      IOSB=(R0),-
      OE52 1826      EFN=#EXESC_SYSEFN
      5E 18 C0 OE6A 1827      ADDL #6*4,SP :POP ITEM LIST AND IOSB
      5E DD OE6D 1828      PUSHL SP :ADDRESS OF PROCESS NAME
      51 01 DO OE6F 1829      MOVL #1,R1 :SET NUMBER OF ARGS
      50 0003FD11 8F DO OE72 1830      MOVL #CLIS_RETURNED,R0 :MESSAGE CODE
      F184 30 OE79 1831      BSBW DCL$FORMMSG :OUTPUT MESSAGE
      5E 18 C0 OE7C 1832      ADDL #8*16,SP :CLEAN STACK
      05 OE7F 1833      RSB
      OE80 1834

```

```

OE80 1836 .SBTTL WRITE CONTEXT TO SUBPROCESS
OE80 1837 :---
OE80 1838 :
OE80 1839 : THIS ROUTINE WRITES THE PROCESS CONTEXT RECORDS TO THE SUBPROCESS
OE80 1840 : INITIALIZATION ROUTINE VIA THE CONTEXT MAILBOX. EACH RECORD HAS
OE80 1841 : A TYPE FIELD WHICH DISTINGUISHES THE DIFFERENT TYPES OF INFORMATION
OE80 1842 : WHICH CAN BE PASSED FROM PARENT TO SUBPROCESS.
OE80 1843 :
OE80 1844 : INPUTS:
OE80 1845 :
OE80 1846 : R6 = ADDRESS OF SPWN STORAGE
OE80 1847 :
OE80 1848 : OUTPUTS:
OE80 1849 :
OE80 1850 : R0 = STATUS CODE
OE80 1851 :---
OE80 1852 :
OE80 1853 WRITE_CONTEXT:
OE80 1854 PUSHF #M<R2,R3,R4,R5,R7> ; SAVE REGISTERS
OE84 1855 :
OE84 1856 :
OE84 1857 : CONSTRUCT THE HEADER CONTEXT RECORD, WHICH CONTAINS SUCH THINGS AS PROCESS
OE84 1858 : PRIVILEGES, ETC.
OE84 1859 :
SE FC00 CE 9E OE84 1860 MOVAB -CTX_C_MAXLEN(SP),SP ; ALLOCATE BUFFER FOR CONTEXT RECORD
57 5E D0 OE89 1861 MOVL SP,R7 ; SAVE ADDRESS OF BUFFER
67 00 3C OE8C 1862 MOVZWL #CTX_C_HEADER,CTX_W_TYPE(R7) ; SET TYPE OF RECORD
7E 7C OE8F 1863 CLRQ -(SP) ; SET END OF ITEM LIST, RETLEN ADDRE
02 A7 7F OE91 1864 PUSHAQ CTX_Q_PROCPRIV(R7) ; SET ADDRESS OF BUFFER
02040008 8F DD OE94 1865 PUSHL #JPI$-PROCPRIV@16!8 ; SET GETJPI ITEM CODE AND LENGTH
50 5E D0 OE9A 1866 MOVL SP,R0 ; ADDRESS OF ITEM LIST
OE9D 1867 $GETJPIW S ITMLST=(R0),- ; GET INFORMATION
OE9D 1868 IOSB=SPWN_Q_IOSB(R6),- ;
OE9D 1869 EFN=#EXE$C_SYSEFN ;
SE 10 C0 OE85 1870 ADDL #4*4,SP ; POP GETJPI ITEM LIST
OE A7 94 OE88 1871 CLRB CTX_B_FLAGS(R7) ; CLEAR THE FLAGS BYTE
OE8B 1872 :
04 0C A6 02 E1 OE8B 1873 BBC #SPWN_V_WAIT,SPWN_W_FLAGS(R6),5$ ; BRANCH IF NOT SET
OE8C 1874 SETBIT CTX_V_WAIT,CTX_B_FLAGS(R7) ;
OE8C 1875 :
04 0C A6 03 E1 OE8C 1876 5$: BBC #SPWN_V_AUTOLOGO,SPWN_W_FLAGS(R6),15$ ; BRANCH IF NOT SET
OE8D 1877 SETBIT CTX_V_AUTOLOGO,CTX_B_FLAGS(R7) ;
OE8D 1878 :
04 0C A6 04 E1 OE8D 1879 15$: BBC #SPWN_V_MODE,SPWN_W_FLAGS(R6),16$ ; BRANCH IF NOT SET
OE8E 1880 SETBIT CTX_V_MODE,CTX_B_FLAGS(R7) ;
OE8E 1881 :
04 68 AB 07 E1 OE8E 1882 16$: BBC #PRC_V_VERIFY,PRC_W_FLAGS(R11),161$ ; COPY VERIFICATION FLAG
OE8F 1883 SETBIT CTX_V_VERIFY,CTX_B_FLAGS(R7) ;
OE8F 1884 :
04 00AF CB 07 E1 OE8F 1885 161$: BBC #PRC_V_VERIMAGE,PRC_B_FLAGS2(R11),162$ ; COPY IMAGE VERIFICATION FL
OE8F 1886 SETBIT CTX_V_VERIMAGE,CTX_B_FLAGS(R7) ;
OE8F 1887 :
12 0C A6 09 E0 OE8F 1888 162$: BBC #SPWN_V_PROMPT,SPWN_W_FLAGS(R6),17$ ; BRANCH IF EXPLICIT PROMPT
50 00F0 CB 9A OE8F 1889 MOVZBL PRC_B_PROMPTLEN(R11),R0 ; GET PROMPT LENGTH
OF A7 50 90 OE8F 1890 MOVB R0,CTX_B_PROMPTLEN(R7) ;
00F1 CB 50 28 OE8F 1891 MOVCS R0,PRC_W_PROMPTCTRL(R11),- ; GET PROMPT STRING
10 A7 OE8F 1892 CTX_W_PROMPTCTRL(R7) ;

```

```

50 00A2 C6 11 0EFE 1893      BRB      18$
  OF A7 50 9A 0F00 1894 17$: MOVZBL SPWN_B_PROMPTLEN(R6),R0      ; GET PROMPT LENGTH
00A3 C6 50 90 0F05 1895      MOVB    RO,CTX_B_PROMPTLEN(R7)      ;
  10 A7 28 0F09 1896      MOVCB    RO,SPWN_Q_PROMPTCTRL(R6),-    ; GET PROMPT STRING
      0F0E 1897      CTX_W_PROMPTCTRL(R7)      ;
      0F10 1898      ;
      58 A6 D0 0F10 1899 18$: MOVL    SPWN_L_OUTOFBAND(R6),-    ; GET OUT-OF-BAND AST MASK
      0A A7 0F13 1900      CTX_C_OUTOFBAND(R7)      ;
      0F15 1901      ;
      53 33 D0 0F15 1902      MOVL    #CTX_C_HDRLEN,R3      ; COMPUTE SIZE OF RECORD
      0200 30 0F18 1903      BSBW    WRITE_MAILBOX      ; WRITE THE RECORD TO THE MAILBOX
      23 50 E9 0F1B 1904      BLBC    RO,19$      ; BRANCH IF ERROR DETECTED
      0F1E 1905      ;
      0F1E 1906      ;
      0F1E 1907      ; WRITE THE COMMAND STRING RECORD TO THE MAILBOX
      0F1E 1908      ;
      30 A6 B5 0F1E 1909      TSTW    SPWN_Q_CMDSTR(R6)      ; ANY COMMAND STRING?
      24 13 0F21 1910      BEQL    20$      ; BRANCH IF NOT
      30 A6 B1 0F23 1911      CMPW    SPWN_Q_CMDSTR(R6),-    ; CHECK IF BIGGER THAN OUR BUFFER
03FE 8F 0F26 1912      #CTX_C_MAXLEN-CTX_T_CMDSTR      ;
      06 1B 0F29 1913      BLEQU    10$      ; BRANCH IF OK
03FE 8F 3C 0F2B 1914      MOVZWL    #CTX_C_MAXLEN-CTX_T_CMDSTR,-    ; ELSE, TRUNCATE STRING
      30 A6 0F2F 1915      SPWN_Q_CMDSTR(R6)      ;
      67 01 B0 0F31 1916 10$: MOVW    #CTX_C_CMDSTR,CTX_W_TYPE(R7) ; SET TYPE OF RECORD
      30 A6 28 0F34 1917      MOVCB    SPWN_Q_CMDSTR(R6),-    ;
      34 B6 0F37 1918      @SPWN_Q_CMDSTR+4(R6),-    ;
      02 A7 0F39 1919      CTX_T_CMDSTR(R7)      ;
      53 57 C2 0F3B 1920      SUBL    R7,R3      ; COMPUTE LENGTH OF RECORD
      01DA 30 0F3E 1921      BSBW    WRITE_MAILBOX      ; WRITE THE RECORD TO THE MAILBOX
      03 50 E8 0F41 1922 19$: BLBS    RO,20$      ; BRANCH IF SUCCESS
      01CA 31 0F44 1923      BRW     90$      ;
      0F47 1924      ;
      0F47 1925      ; WRITE THE PROCESS LOGICAL NAMES TO THE MAILBOX
      0F47 1926      ;
      0F47 1927      ;
03 0C A6 06 E0 0F47 1928 20$: BBS     #SPWN_V_LOGNAM,SPWN_W_FLAGS(R6),30$ ; BRANCH IF REQUESTED
      0153 31 0F4C 1929      BRW     40$      ;
      0F4F 1930      ;
      0F4F 1931      ;
      0F4F 1932      ; FIRST WRITE ALL OF THE TABLE NAMES
      0F4F 1933      ;
      67 05 B0 0F4F 1934 30$: MOVW    #CTX_C_LNMTABLE,CTX_W_TYPE(R7) ; SET LOGICAL NAME TABLE
55 00000000 GF D0 0F52 1935      MOVL    G^CTC$GL LNMDIRECT,R5-    ; GET PROCESS DIRECTORY
      55 0C A5 D0 0F59 1936      MOVL    LNMB$TABLE(R5),R5      ; GET IT'S TABLE
      50 11 A5 D0 0F5D 1937      MOVL    LNMT$CHILD(R5),R0      ; GET FIRST TABLE
      55 50 D0 0F61 1938 31$: MOVL    R0,R5      ; GET NEXT TABLE
      54 09 A5 D0 0F64 1939      MOVL    LNMT$NAME(R5),R4      ; GET POINTER TO LNMB
58 10 A4 01 E0 0F68 1940      BBS     #LNMB$V-CONFINE,LNMB$B_FLAGS(R4),33$ ; SKIP SUBTREE IF CONFINED
      01 0B A4 91 0F6D 1941      CMPB    LNMB$B_ACMODE(R4),#PSL$C_EXEC ; PRIV MODE TABLE?
      4C 1B 0F71 1942      BLEQU    32$      ; DON'T COPY IF SO.
04 A7 0B A4 90 0F73 1943      MOVB    LNMB$B_ACMODE(R4),CTX_B_ACMODE(R7) ; COPY ACMODE
      08 88 0F78 1944      BICB3    #LNMB$M_TABLE,-    ;
05 A7 10 A4 D0 0F7A 1945      LNMB$B_FLAGS(R4),CTX_B_TFLAGS(R7) ; COPY NAME FLAGS
08 A7 1D A5 D0 0F7E 1946      MOVL    LNMT$C_BYTESLM(R5),CTX_L_QUOTA(R7) ; COPY QUOTA
      53 0C A7 DE 0F83 1947      MOVAL   CTX_T_LNMTABLE(R7),R3      ; PTR TO BUFFER
      51 11 A4 DE 0F87 1948      MOVAL   LNMB$T_NAME(R4),R1      ; PTR TO NAME
      50 61 9A 0F8B 1949      MOVZBL (R1),R0      ; LENGTH OF NAME

```

```

50 D6 OF8E 1950 INCL R0 ; INCLUDE THE COUNT
30 BB OF90 1951 PUSHR #^M<R4,R5> ; SAVE REGS
63 61 50 28 OF92 1952 MOV C3 R0,(R1),(R3) ; COPY NAME
30 BA OF96 1953 POPR #^M<R4,R5> ; RESTORE REGS
51 0D A5 D0 OF98 1954 MOVL LNMTH$$_PARENT(R5),R1 ; PTR TO PARENT
51 09 A1 D0 OF9C 1955 MOVL LNMTH$$_NAME(R1),R1 ; PTR TO PARENT'S LNMB
51 11 A1 DE OFA0 1956 MOVAL LNMBS$_NAME(R1),R1 ; PTR TO NAME
50 61 9A OFA4 1957 MOVZBL (R1),R0 ; LENGTH OF TABLE NAME
30 D6 OFA7 1958 INCL R0 ; INCLUDE THE COUNT
63 61 50 BB OFA9 1959 PUSHR #^M<R4,R5> ; SAVE REGS
30 28 OFAB 1960 MOV C3 R0,(R1),(R3) ; COPY TABLE NAME
30 BA OFAF 1961 POPR #^M<R4,R5> ; RESTORE REGS
02 A7 53 57 C2 OFB1 1962 SUBL 2 R7,R3 ; COMPUTE SIZE OF CONTEXT RECORD
53 02 A3 OFB4 1963 SUBW 3 #CTX_W_ENTSIZE,R3,CTX_W_ENTSIZE(R7) ; SET SIZE OF ENTRY
01 5F 30 OFB9 1964 BSBW WRITE_MAILBOX ; WRITE THE RECORD TO THE MAILBOX
28 50 E9 OFBC 1965 BLBC R0,92$ ; QUIT ON ERROR
50 11 A5 D0 OFBF 1966 32$: MOVL LNMTH$$_CHILD(R5),R0 ; GET CHILD TABLE PTR
25 12 OFC3 1967 BNEQ 310$ ; AND PASS IT.
50 15 A5 D0 OFC5 1968 33$: MOVL LNMTH$$_SIBLING(R5),R0 ; SIBLING TABLE?
1F 12 OFC9 1969 BNEQ 310$ ; AND PASS IT.
55 0D A5 D0 OFCB 1970 MOVL LNMTH$$_PARENT(R5),R5 ; GET SIBLING OF PARENT TABLE
50 15 A5 D0 OFCF 1971 MOVL LNMTH$$_SIBLING(R5),R0 ; AND PASS IT
15 12 OFD3 1972 BNEQ 310$
OFD5 1973
OFD5 1974
OFD5 1975 ; NOW THAT THE TABLES EXIST, WRITE ALL OF THE LOGICAL NAMES
OFD5 1976
54 67 06 D0 OFD5 1977 MOVL #CTX_C_LNMNAME,CTX_W_TYPE(R7) ; SET LOGICAL NAME TYPE
00000000'GF D0 OFD8 1978 MOVL G^CT$$_LNMHASH,R4 ; GET POINTER TO HASH TABLE
54 64 D2 OFDF 1979 MCOML LNMHSH$_MASK(R4),R4 ; GET SIZE MASK
54 06 D6 OFE2 1980 INCL R4 ; CONVERT TO SIZE
00AC 31 OFE4 1981 34$: BRW 38$ ; LOOP
0127 31 OFE7 1982 92$: BRW 90$ ; HELPER BRANCH
FF74 31 OFEA 1983 310$: BRW 31$ ; HELPER BRANCH
OFED 1985
55 65 D0 OFED 1986 35$: MOVL (R5),R5 ; GET NEXT LINK IN CHAIN
F2 13 OFF0 1987 BEQL 34$ ; GO TO NEXT CHAIN IF NONE
F6 10 A5 01 E0 OFF2 1988 BBS #LNMB$$_CONFINE,LNMB$$_FLAGS(R5),35$ ; SKIP NAME IF CONFINED
F1 10 A5 03 E0 OFF7 1989 BBS #LNMB$$_TABLE,LNMB$$_FLAGS(R5),35$ ; SKIP NAME IF TABLE
01 0B A5 91 OFFC 1990 CMPB LNMB$$_ACMODE(R5),#PSL$$_EXEC ; PRIV MODE TABLE?
04 A7 0B A5 90 1000 1991 BLEQU 35$ ; DON'T COPY IF SO.
05 A7 10 A5 90 1007 1992 MOV B LNMB$$_ACMODE(R5),CTX_B_ACMODE(R7) ; COPY ACMODE
53 07 A7 DE 100C 1994 MCVB LNMB$$_FLAGS(R5),CTX_B_RFLAGS(R7) ; COPY NAME FLAGS
51 0C A5 D0 1010 1995 MOVAL CTX_T_LNMNAME(R7),R3 ; PTR TO BUFFER
51 09 A1 D0 1014 1996 MOVL LNMB$$_TABLE(R5),R1 ; PTR TO TABLE
50 11 A1 9A 1018 1997 MOVL LNMTH$$_NAME(R1),R1 ; PTR TO TABLE'S LNMB
50 D6 101C 1998 MOVZBL LNMBS$_NAME(R1),R0 ; LENGTH OF TABLE NAME
30 BB 101E 1999 INCL R0 ; INCLUDE THE COUNT
63 11 A1 50 28 1020 2000 PUSHR #^M<R4,R5> ; SAVE REGS
30 BA 1025 2001 MOV C3 R0,LNMB$$_NAME(R1),(R3) ; COPY NAME
50 11 A5 9E 1027 2002 POPR #^M<R4,R5> ; RESTORE REGS
52 D4 102B 2003 MOVAB LNMB$$_NAME(R5),R0 ; PTR TO NAME
0E 60 02 E0 102D 2004 CLRL R2 ; COUNT # OF XLATIONS
52 D6 1033 2005 350$: BRB 351$ ; TRY NEXT ENTRY
2006 BBS #LNMX$$_XEND,LNMX$$_FLAGS(R0),352$ ; BR IF END OF LIST
INCL R2 ; COUNT THIS XLATION

```

```

50 04 A0 9E 1035 2007      MOVAB LNMXT XLATION(R0),R0      : POINT TO STRING
    51 80 9A 1039 2008 351$: MOVZBL (R0)+,R1      : GET LENGTH OF THIS STRING
    50 51 C0 103C 2009      ADDL2 R1,R0      : SKIP OVER IT
    11 103F 2010      BRB 350$      : TRY NEXT ENTRY
06 A7 52 90 1041 2011 352$: MOVB R2,CTX_B_TRANCNT(R7)      : SAVE COUNT
51 11 A5 9E 1045 2012      MOVAB LNMBSI_NAME(R5),R1      : ADDR OF NAME AGAIN
    50 D6 1049 2013      INCL R0      : COUNT TERMINATION BYTE
    52 51 C2 104B 2014      SUBL2 R1,R0      : COMPUTE SIZE NEEDED
    0400 C7 9E 104E 2015      MOVAB CTX_C_MAXLEN(R7),R2      : GET END ADDR OF CTX
    52 53 C2 1053 2016      SUBL2 R3,R2      : BUFFER SPACE REMAINING
    52 50 D1 1056 2017      CMPL R0,R2      : NAME ALL FIT?
    18 1A 1059 2018      BGTRU 36$      : NOPE
    30 BB 105B 2019      PUSHR #M<R4,R5>      : SAVE REGS
63 61 50 28 105D 2020      MOVC3 R0,(R1),(R3)      : COPY NAME AND XLATIONS
    30 BA 1061 2021      POPR #M<R4,R5>      : RESTORE REGS
02 A7 53 57 C2 1063 2022      SUBL2 R7,R3      : COMPUTE SIZE OF CONTEXT RECORD
    53 02 A3 1066 2023      SUBW3 #CTX_W_ENTSIZE,R3,CTX_W_ENTSIZE(R7) : SET SIZE OF ENTRY
    00AD 30 1069 2024      BSBW WRITE_MAILBOX      : WRITE THE RECORD TO THE MAILBOX
    75 50 E9 106E 2025      BLBC R0,49$      : QUIT ON ERROR
    1D 11 1071 2026      BRB 37$      : ONTO THE NEXT NAME
    1073 2027      :
    1073 2028      : DEAL WITH RECORDS TOO LONG FOR ONE MAILBOX MESSAGE
    1073 2029      : (FOR NOW, ISSUE THE "SYMBOL TOO LONG" ERROR.
    1073 2030      :
    1073 2031      :
18 0C A6 00 E1 1073 2032 36$: BBC #SPWN_V_LOG,SPWN_W_FLAGS(R6),37$ : BRANCH IF /NOLOG SPECIFIED
    01 A1 9F 1078 2033      PUSHAB 1(R1)      : PUSH DESCRIPTOR OF THE
    7E 61 9A 107B 2034      MOVZBL (R1),-(SP)      : LOGICAL NAME
    5E DD 107E 2035      PUSHL SP      : ADDRESS OF THE DESCRIPTOR
50 00038218 8F DD 1080 2036      MOVL #1,R1      : SET NUMBER OF ARGS
    EF73 30 DD 1083 2037      MOVL #CLIS_SYMTOLNG,R0      : MESSAGE CODE
    5E 08 C0 108A 2038      BSBW DCL$FORMMSG      : OUTPUT MESSAGE
    FF5A 31 C0 108D 2039      ADDL #4*2,SP      : CLEAN STACK
    1093 2040 37$: BRW 35$      : PROCESS NEXT SYMBOL
55 00000000'GF D0 1093 2042 38$: MOVL G^CTL$GL_LNMHASH,R5      : GET ADDR OF HASHTABLE
    55 08 A544 DE 109A 2043      MOVAL LNMHSH$C_BUCKET-4(R5)[R4],R5 : GET ADDR OF BUCKET
    EE 54 F4 109F 2044      SOBGEQ R4,37$      : LOOP OVER ALL CHAINS
    10A2 2045      :
    10A2 2046      :
    10A2 2047      : WRITE THE DCL GLOBAL SYMBOLS TO THE MAILBOX
    10A2 2048      :
16 0C A6 05 E1 10A2 2049 40$: BBC #SPWN_V_CLISYM,SPWN_W_FLAGS(R6),45$ : BRANCH IF NOT REQUESTED
    54 28 AB 9E 10A7 2050      MOVAB PRC_Q_GLOBAL(R11),R4      : GET ADDRESS OF GLOBAL LISTHEAD
    04 A7 00 90 10AB 2051      MOVB #CTX_C_GLOBAL,CTX_B_SYMTAB(R7) : SET WHICH SYMBOL TABLE
    0097 30 10AF 2052      BSBW WRITE_SYMBOLS      : WRITE ALL GLOBAL SYMBOLS
    54 38 AB 9E 10B2 2053      MOVAB PRC_Q_LOCAL(R11),R4      : GET ADDRESS OF LOCAL LISTHEAD
    04 A7 01 90 10B6 2054      MOVB #CTX_C_LOCAL,CTX_B_SYMTAB(R7) : SET WHICH SYMBOL TABLE
    008C 30 10BA 2055      BSBW WRITE_SYMBOLS      : WRITE ALL LOCAL SYMBOLS
    10BD 2056      :
    10BD 2057      :
    10BD 2058      : WRITE THE KEYPAD STATE TO THE MAILBOX
    10BD 2059      :
27 0C A6 0C E1 10BD 2060 45$: BBC #SPWN_V_KEYPAD,SPWN_W_FLAGS(R6),50$ : BRANCH IF NOT REQUESTED
    54 40 AB 9E 10C2 2061      MOVAB PRC_Q_KEYPAD(R11),R4      : GET ADDRESS OF KEYPAD LISTHEAD
    04 A7 02 90 10C6 2062      MOVB #CTX_C_KEYTABL,CTX_B_SYMTAB(R7) : SET WHICH SYMBOL TABLE
    007C 30 10CA 2063      BSBW WRITE_SYMBOLS      : WRITE ALL KEYPAD SYMBOLS

```

```

03 A7 02 67 04 B0 10CD 2064 MOVW #CTX_C_KEYSTATE,CTX_W_TYPE(R7) ; SET TYPE OF RECORD
52 4C AB D0 10D0 2065 MOVL PRC_C_LASTKEY(R1),R2- ; GET ADDR OF ASCII KEY STATE
51 82 9A 10D4 2066 MOVZBL (R2)+,R1 ; GET DESCRIPTOR
A7 51 90 10D7 2067 MOVB R1,CTX_B_KEYLENGTH(R7) ; COPY THE LENGTH
62 51 28 10DB 2068 MOVC R1,(R2),CTX_T_KEYSTATE(R7) ; COPY THE STRING
53 57 C2 10E0 2069 SUBL R7,R3 ; COMPUTE LENGTH OF RECORD
0035 3C 10E3 2070 BSBW WRITE MAILBOX ; WRITE THE RECORD TO THE MAILBOX
28 50 E9 10E6 2071 4Y$: BLBC R0,90$ ; BRANCH IF ERROR DETECTED
10E9 2072
10E9 2073
10E9 2074 : WRITE AN END-OF-FILE TO THE MAILBOX AND WAIT FOR COMPLETION, SO THAT
10E9 2075 : WE DON'T DELETE THE MAILBOX BEFORE THE SUBPROCESS EVEN GETS A CHANCE
10E9 2076 : TO ASSIGN A CHANNEL TO IT.
10E9 2077
10E9 2078 50$: $QIOW_S FUNC=#IOS_WRITEOF, - ; WRITE AN EOF TO MAILBOX
10E9 2079 CHAN=SPWN_W_CHAN(R6),-
10E9 2080 IOSB=SPWN_Q_IOSB(R6),-
10E9 2081 EFN=#EXE$C_SYSEFN
10E9 2082 BLBC R0,90$ ; BRANCH IF ERROR
50 04 50 E9 110A 2083 MOVZWL SPWN_Q_IOSB(R6),R0 ; GET IOSB STATUS
SE 50 38 A6 3C 110D 2084 90$: MOVAB CTX_C_MAXLEN(SP),SP ; DEALLOCATE RECORD BUFFER
0400 CE 9E 1111 2085 POPR #^MZR2,R3,R4,R5,R7> ; RESTORE REGISTERS
00BC 8F BA 1116 2086 RSB
05 111A 2086

```

```

1118 2088 .SBTTL WRITE RECORD TO CONTEXT MAILBOX
1118 2089 :---
1118 2090 :
1118 2091 : WRITE A RECORD TO THE MAILBOX WITHOUT WAITING FOR A READER
1118 2092 :
1118 2093 : INPUTS:
1118 2094 :
1118 2095 : R6 = ADDRESS OF SPWN STORAGE
1118 2096 : R7 = ADDRESS OF RECORD TO BE OUTPUT
1118 2097 : R3 = LENGTH OF RECORD TO BE OUTPUT
1118 2098 :
1118 2099 : OUTPUTS:
1118 2100 :
1118 2101 : R0 = FINAL STATUS
1118 2102 :---
1118 2103 :
009F 30 1118 2104 WRITE_MAILBOX:
1118 2105 BSBW CHECK FOR HANGUP ; CHECK FOR HANGUP AST
1118 2106 $QIOW_S FUNC=#IOS_WRITEVBLK!IOSM_NOW,- ; WRITE TO MAILBOX WITHOUT WAITING
1118 2107 CHAN=SPWN_W CHAN(R6),- ;
1118 2108 EFN=#EXESC SYSEFN,- ;
1118 2109 IOSB=SPWN_Q IOSB(R6),- ;
1118 2110 P1=(R7),P2=R3 ; ADDRESS/LENGTH OF BUFFER
50 04 50 E9 1141 2111 R0,90$ ; BRANCH IF ERROR DETECTED
38 A6 3C 1144 2112 SPWN_Q_IOSB(R6),R0 ; GET FINAL STATUS
05 1148 2113 90$: RSB

```

```

1149 2115 .SBTTL WRITE ALL SYMBOLS IN A SYMBOL TABLE
1149 2116 :---
1149 2117 :
1149 2118 WRITE ALL THE SYMBOL RECORDS TO THE MAILBOX FROM A
1149 2119 SPECIFIED SYMBOL TABLE.
1149 2120 :
1149 2121 INPUTS:
1149 2122 :
1149 2123 CTX_B_SYMTAB(R7) = TYPE OF SYMBOL TABLE
1149 2124 R4 = ADDRESS OF SYMBOL TABLE LISTHEAD
1149 2125 R6 = ADDRESS OF SPWN STORAGE
1149 2126 R7 = ADDRESS OF CONTEXT RECORD BUFFER
1149 2127 :
1149 2128 OUTPUTS:
1149 2129 :
1149 2130 RO-R5 DESTROYED.
1149 2131 :---
1149 2132 :
1149 2133 WRITE_SYMBOLS:
1149 2134 MOVW #CTX_C_CLISYM,CTX_W_TYPE(R7) ; SET TYPE OF RECORD
1149 2135 MOVL R4,R5 ; SAVE FOR END-OF-LIST CHECK
1149 2136 45$: MOVL (R4),R4 ; GET NEXT SYMBOL ENTRY IN LIST
1149 2137 CMPL R4,R5 ; END OF LIST?
1149 2138 BEQL 50$ ; BRANCH IF SO
1149 2139 ASSUME SYM_K_STRING EQ CTX_C_STRING
1149 2140 ASSUME SYM_K_PERM EQ CTX_C_PERM
1149 2141 ASSUME SYM_K_BINARY EQ CTX_C_BINARY
1149 2142 ASSUME SYM_K_KEYPAD EQ CTX_C_KEYPAD
1149 2143 MOVW SYM_B_TYPE(R4),CTX_B_SYMTYPE(R7) ; COPY SYMBOL TYPE CODE
1149 2144 MOVW SYM_B_NONUNIQUE(R4),CTX_B_NONUNIQUE(R7) ; COPY UNIQUENESS POINT
1149 2145 MOVZBL SYM_T_SYMBOL(R4),R0 ; GET LENGTH OF SYMBOL NAME
1149 2146 CMPB SYM_B_TYPE(R4),#SYM_K_BINARY ; BINARY VALUE?
1149 2147 BNEQ 47$ ; BRANCH IF NOT
1149 2148 ADDL #4+1,R0 ; COMPUTE LENGTH OF ASCII NAME + VALUE
1149 2149 BRB 48$
1149 2150 47$: MOVAB SYM_T_SYMBOL+1(R4)[R0],R1 ; ADDRESS OF WORD-COUNTED VALUE STRING
1149 2151 MOVZWL (R1),R1 ; GET LENGTH OF VALUE STRING
1149 2152 MOVAB 3(R1)[R0],R0 ; COMPUTE LENGTH OF ASCII NAME + VALUE
1149 2153 48$: CMPW R0,#<CTX_C_MAXLEN-CTX_T_SYMBOL> ; WILL SYMBOL VALUE FIT?
1149 2154 BRB 60$ ; IF LARGER, THEN NO
1149 2155 PUSHR #*M<R4,R5> ; SAVE REGISTERS
1149 2156 MOVW R0,SYM_T_SYMBOL(R4),CTX_T_SYMBOL(R7) ; MOVE NAME + VALUE
1149 2157 POPR #*M<R4,R5> ; RESTORE REGISTERS
1149 2158 SUBL R7,R3 ; COMPUTE SIZE OF RECORD
1149 2159 SUBW3 #CTX_W_ENTSIZE,R3,CTX_W_ENTSIZE(R7) ; SET SIZE OF ENTRY
1149 2160 BSBB WRITE_MAILBOX ; WRITE RECORD TO MAILBOX
1149 2161 BRB 45$ ; LOOP UNTIL TABLE EXHAUSTED
1149 2162 50$: RSB
1149 2163 :
1149 2164 :
1149 2165 : WE HAVE ENCOUNTERED AN OVERSIZED SYMBOL. ISSUE A WARNING MESSAGE SAYING
1149 2166 : THAT WE ARE IGNORING IT AND THEN PROCEED.
1149 2167 :
1149 2168 60$: BBC #SPWN_V_LOG,SPWN_W_FLAGS(R6),45$ ; BRANCH IF /NOLOG SPECIFIED
1149 2169 MOVAB SYM_T_SYMBOL+1(R4),-(SP) ; PUSH DESCRIPTOR OF THE
1149 2170 MOVZBL SYM_T_SYMBOL(R4),-(SP) ; SYMBOL NAME
1149 2171 PUSHL SP ; ADDRESS OF THE DESCRIPTOR

```

SPAWN
V04-000

- MULTI-PROCESSING COMMANDS
WRITE ALL SYMBOLS IN A SYMBOL TABLE

F 15

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 47
(18)

50	51	01	D0	11AA	2172	MOVL	#1,R1	:	SET NUMBER OF ARGS
	00038218	8F	D0	11AD	2173	MOVL	#CLIS SYMTOOLNG,R0	:	MESSAGE CODE
		EE49	30	11B4	2174	BSBW	DCL\$FORMMSG	:	OUTPUT MESSAGE
	5E	08	C0	11B7	2175	ADDL	#4*2,SP	:	CLEAN STACK
		FF92	31	11BA	2176	BRW	45\$	:	PROCESS NEXT SYMBOL
				11BD	2177				

```

11BD 2179 .SBTTL CHECK FOR PENDING HANGUP AST
11BD 2180 :---
11BD 2181 :
11BD 2182 : THIS ROUTINE IS CALLED TO CHECK IF A HANGUP AST HAS COME IN SINCE
11BD 2183 : WE STARTED THE COMMAND EXECUTION. IF SO, THE PROCESS IS TERMINATED.
11BD 2184 :
11BD 2185 : INPUTS:
11BD 2186 :
11BD 2187 : R11 = ADDRESS OF PRC AREA
11BD 2188 :
11BD 2189 : OUTPUTS:
11BD 2190 :
11BD 2191 : NONE
11BD 2192 :---
11BD 2193 :
11BD 2194 CHECK_FOR_HANGUP:
03 68 AB OC E1 11BD 2195 BBC #PRC_V_HANGUP,PRC_W_FLAGS(R11),90$ : BRANCH IF NO HANGUP PENDING
EE3B' 31 11C2 2196 BRW DCL$RESTART ; ABORT PROCESS
05 11C5 2197 90$: RSB

```

```

11C6 2199 .SBTTL GET DEVICE NAME
11C6 2200 :---
11C6 2201 :
11C6 2202 : THIS ROUTINE CALLS THE $GETDVI SYSTEM SERVICE TO GET THE CANNONICAL NAME OF
11C6 2203 : A SPECIFIED DEVICE. IT PLACES THE DEVICE NAME IN THE SPECIFIED BUFFER AND
11C6 2204 : PROPAGATES THE STATUS RETURNED BY $GETDVI TO THE CALLER.
11C6 2205 :
11C6 2206 : INPUTS:
11C6 2207 :
11C6 2208 : R2/R3 = DESCRIPTOR OF DEVICE NAME
11C6 2209 : R4/R5 = DESCRIPTOR OF RETURN BUFFER
11C6 2210 :
11C6 2211 : OUTPUTS:
11C6 2212 :
11C6 2213 : R0 = STATUS RETURNED BY $GETDVI
11C6 2214 :
11C6 2215 :---
11C6 2216 :
11C6 2217 GET_DEVICE:
11C6 2218 CLRQ -(SP) ; ALLOCATE AN IOSB
11C6 2219 MOVQ R2,-(SP) ; PUSH DEVICE DESCRIPTOR ON STACK
11C6 2220 MOVL SP,R1 ; GET ADDRESS OF DESCRIPTOR
11C6 2221 CLRL -(SP) ; CREATE GETDVI ITEM LIST
11C6 2222 PUSHAB -12(SP) ; RETURN LENGTH ON STACK
11C6 2223 MOVQ R4,-(SP) ; PUSH BUFFER DESCRIPTOR ON STACK
11C6 2224 MOVW #DVI$_DEVNAM,2(SP) ; REQUEST COMPLETE DEVICE NAME
11C6 2225 MOVL SP,R0 ; GET ADDRESS OF ITEM LIST
11C6 2226 $GETDVIW S,DEVNAM=(R1),- ; GET TERMINAL DEVICE NAME
11C6 2227 IOSB=8(R1),- ;
11C6 2228 EFN=#EXESC_SYSEFN,- ;
11C6 2229 ITMLST=(R0) ;
11C6 2230 MOVZWL (SP),R4 ; GET LENGTH OF RESULT
11C6 2231 ADDL #6*4,SP ; RESTORE STACK
11C6 2232 BLBC R0,90$ ; BRANCH IF ERROR
11C6 2233 MOVZWL (SP),R0 ; GET IOSB STATUS
11C6 2234 ADDL #8,SP ; POP THE IOSB
11C6 2235 RSB

```

```

1208 2237 .SBTTL CREATE OUTPUT MAILBOX
1208 2238 ---
1208 2239 :
1208 2240 : THIS ROUTINE IS CALLED TO CREATE AN OUTPUT MAILBOX FOR THIS PROCESS TO
1208 2241 : RECEIVE WRITE REQUESTS FROM ITS SPAWNED SUBPROCESSES.
1208 2242 :
1208 2243 : INPUTS:
1208 2244 :
1208 2245 : R6 = ADDRESS OF SPWN DATA STRUCTURE
1208 2246 : R8 = ADDRESS OF SCRATCH AREA
1208 2247 : R11 = ADDRESS OF PRC DATA STRUCTURE
1208 2248 :
1208 2249 : OUTPUTS:
1208 2250 :
1208 2251 : RO = STATUS CODE
1208 2252 : ---
1208 2253 :
1208 2254 CREATE_OUTMBX:
1208 2255 :
1208 2256 :
1208 2257 : GET PID OF PARENT.
1208 2258 :
1208 2259 CLRQ -(SP)
1208 2260 PUSHAB -8(SP)
1208 2261 PUSHL #JPI$_PID@16+4
1208 2262 MCVL SP,RO
1208 2263 $GETJPIW S ITMLST=(RO),-
1208 2264 EFN=#EXESC SYSEFN,-
1208 2265 IOSB=SPWN_Q_IOSB(R6)
1208 2266 POPL RO
1208 2267 ADDL #3*4,SP
1208 2268 :
1208 2269 :
1208 2270 : CREATE MAILBOX LOGICAL NAME.
1208 2271 :
1208 2272 MOVQ (R8),-(SP)
1208 2273 MOVAB OUTPUT_NAME,R2
1208 2274 MOVZBL (R2)+,R1
1208 2275 MOVQ R1,-(SP)
1208 2276 MOVL SP,R1
1208 2277 $FAO_S CTRSTR=(R1),-
1208 2278 OUTBUF=8(R1),-
1208 2279 OUTLEN=8(R1),-
1208 2280 P1=RO
1208 2281 ADDL #8,SP
1208 2282 MOVQ (SP)+,SPWN_Q_OUTPUT(R6)
1208 2283 SUBL SPWN_Q_OUTPUT(R6),(R8)
1208 2284 ADDL SPWN_Q_OUTPUT(R6),4(R8)
1208 2285 BLBS RO,20$
1208 2286 BRW 91$
1208 2287 10$: BRW 90$
1208 2288 :
1208 2289 :
1208 2290 : CHECK FOR ALREADY EXISTING MAILBOX.
1208 2291 :
1208 2292 20$: INCW PRC_W_OUTMBXREF(R11)
1208 2293 CMPW PRC_W_OUTMBXREF(R11),#1
1208 2294 : INCR THE MAILBOX REF COUNT
1208 2295 : DOES MAILBOX ALREADY EXIST?

03190004 7E 7C 1208 2259 CLRQ -(SP)
F8 AE 9F 120A 2260 PUSHAB -8(SP)
8F DD 120D 2261 PUSHL #JPI$_PID@16+4
50 5E D0 1213 2262 MCVL SP,RO
1216 2263 $GETJPIW S ITMLST=(RO),-
1216 2264 EFN=#EXESC SYSEFN,-
1216 2265 IOSB=SPWN_Q_IOSB(R6)
SE 50 8ED0 122E 2266 POPL RO
CC CO 1231 2267 ADDL #3*4,SP
1234 2268 :
1234 2269 :
1234 2270 : CREATE MAILBOX LOGICAL NAME.
1234 2271 :
1234 2272 MOVQ (R8),-(SP)
52 7E 68 7D 1234 2273 MOVAB OUTPUT_NAME,R2
EDFO CF 9E 1237 2274 MOVZBL (R2)+,R1
51 82 9A 123C 2275 MOVQ R1,-(SP)
7E 51 7D 123F 2276 MOVL SP,R1
51 5E DG 1242 2277 $FAO_S CTRSTR=(R1),-
1245 2278 OUTBUF=8(R1),-
1245 2279 OUTLEN=8(R1),-
1245 2280 P1=RO
SE 08 CO 1256 2281 ADDL #8,SP
28 A6 8E 7D 1259 2282 MOVQ (SP)+,SPWN_Q_OUTPUT(R6)
68 28 A6 C2 125D 2283 SUBL SPWN_Q_OUTPUT(R6),(R8)
04 A8 28 A6 CO 1261 2284 ADDL SPWN_Q_OUTPUT(R6),4(R8)
06 50 E8 1266 2285 BLBS RO,20$
009B 31 1269 2286 BRW 91$
0091 31 126C 2287 10$: BRW 90$
126F 2288 :
126F 2289 :
126F 2290 : CHECK FOR ALREADY EXISTING MAILBOX.
126F 2291 :
126F 2292 20$: INCW PRC_W_OUTMBXREF(R11)
01 00CE CB B6 126F 2293 CMPW PRC_W_OUTMBXREF(R11),#1
00CE CB B1 1273 2294 : INCR THE MAILBOX REF COUNT
1273 2295 : DOES MAILBOX ALREADY EXIST?

```

```

F2 12 1278 2294      BNEQ 10$      ; YES, THEN SKIP MBX CREATION
      127A 2295
      127A 2296
      127A 2297      : GET MAXIMUM MESSAGE SIZE.
      127A 2298
      7E 7C 127A 2299 25$: CLRQ -(SP)      : CREATE THE ITEM LIST
F8 AE 9F 127C 2300      PUSHAB -8(SP)      : SET THE BUFFER ADDRESS
104F0004 8F DD 127F 2301      PUSHL #SYIS_MAXBUF@16+4      : SET THE ITEM CODE AND BUFFER SIZE
50 5E DO 1285 2302      MOVL SP,R0      : GET ADDRESS OF ITEM LIST
      1288 2303      $GETSYIW S ITMLST=(R0),-      : GET MAX BUF SIZE
      1288 2304      EFN=#EXESC_SYSEFN,-
      1288 2305      IOSB=SPWN_Q_IOSB(R6)
      51 8ED0 12A0 2306      POPL R1      : GET THE MAXMSG SIZE
5E 0C CO 12A3 2307      ADDL #3*4,SP      : POP ITEM LIST
50 5F 50 E9 12A6 2308      BLBC R0,95$      : RETURN IF ERROR
50 38 A6 3C 12A9 2309      MOVZWL SPWN_Q_IOSB(R6),R0      : GET IOSB STATUS
00CC CB 51 B0 12AD 2310      BLBC R0,95$      : RETURN IF ERROR
      12B0 2311      MOVW R1,PRC_W_OUTMBXSIZ(R11)      : SAVE THE MAXMSG SIZE
      12B5 2312
      12B5 2313      : CREATE THE OUTPUT MAILBOX.
      12B5 2314
      12B5 2315
      12B5 2316      $CREMBX_S CHAN=PRC_W_OUTMBXCHN(R11),-      : CREATE A WRITE MAILBOX
      12B5 2317      MAXMSG=R1,-
      12B5 2318      BUFQUO=R1,-
      12B5 2319      LOGNAM=SPWN_Q_OUTPUT(R6)
      38 50 E9 12CD 2320      BLBC R0,95$      : BRANCH ON ERROR
      12D0 2321
      12D0 2322      : SET WRITE ATTENTION AST ON THE MAILBOX.
      12D0 2323
      12D0 2324
      12D0 2325      $QIOW_S FUNC=#IOS_SETMODE!IOSM_WRTATTN,-      : SET ATTENTION AST ON MAILBOX
      12D0 2326      CHAN=PRC_W_OUTMBXCHN(R11),-
      12D0 2327      EFN=#EXESC_SYSEFN,-
      12D0 2328      IOSB=SPWN_Q_IOSB(R6),-
      12D0 2329      P1=W^WRITE_AST,-
      12D0 2330      P2=R11
      50 0F 50 E9 12F6 2331      BLBC R0,95$      : ADDRESS OF AST ROUTINE
50 38 A6 3C 12F9 2332      MOVZWL SPWN_Q_IOSB(R6),R0      : PASS PRC BLOCK ADDRESS
50 08 50 E9 12FD 2333      BLBC R0,95$      : BRANCH IF ERROR
      1300 2334 90$: STATUS NORMAL      : GET IOSB STATUS
      1307 2335 91$: RSB      : BRANCH IF ERROR
      1308 2336      : SET NORMAL STATUS
      1308 2337      : RETURN
      1308 2338      : DELETE WRITE MAILBOX AND RETURN WITH STATUS
      1308 2339
      50 DD 1308 2340 95$: PUSHL R0      : SAVE STATUS
0005 30 130A 2341      BSBW DELETE_OUTMBX      : DELETE ALLOCATED MAILBOX
50 8ED0 130D 2342      POPL R0      : RESTORE STATUS
F5 11 1310 2343      BRB 91$      : EXIT

```

```

1312 2345 .SBTTL DELETE WRITE MAILBOX
1313 2346 :---
1314 2347 :
1315 2348 : THIS ROUTINE IS CALLED TO DELETE A WRITE MAILBOX.
1316 2349 :
1317 2350 : INPUTS:
1318 2351 :
1319 2352 : R11 = ADDRESS OF PRC AREA
1320 2353 :
1321 2354 : OUTPUTS:
1322 2355 :
1323 2356 : NONE
1324 2357 :---
1325 2358 :
1326 2359 DELETE_OUTMBX:
1327 2360 DECW PRC_W_OUTMBXREF(R11) ; DECR THE MAILBOX REF COUNT
1328 2361 BNEQ 90$ ; SKIP IF NON-ZERO
1329 2362 $DASSGN_S CHAN=PRC_W_OUTMBXCHN(R11) ; DELETE OUTPUT MAILBOX
1330 2363 CLRW PRC_W_OUTMBXCHN(R11) ; CLEAR MAILBOX CHANNEL NUMBER
1331 2364 90$: RSB ;

```

OOCE CB B7 1312 2360
10 12 1316 2361
OOCA CB B4 1318 2362
05 1324 2363
1328 2364 90\$: RSB

```

1329 2366 .SBTTL WRITE REQUEST AST FROM A SUBPROCESS
1329 2367 :---
1329 2368 :
1329 2369 : THIS AST ROUTINE HANDLES A WRITE REQUEST FROM A SUBPROCESS
1329 2370 : THROUGH THE WRITE MAILBOX ASSOCIATED WITH THIS PROCESS.
1329 2371 :
1329 2372 : INPUTS:
1329 2373 :
1329 2374 : 4 AP) = ADDRESS OF PRC AREA
1329 2375 :
1329 2376 : OUTPUTS:
1329 2377 :
1329 2378 : NON:
1329 2379 :---
1329 2380 :
1329 2381 WRITE_AST:
1329 2382 .WORD ^M<R2,R3,R4,R5,R6,R11>
1329 2383 :
1329 2384 MOVL 4(AP),R11 ; GET ADDRESS OF CLI PROCESS WORK AR
1329 2385 :
1329 2386 :
1329 2387 : IF HANGUP AST IS PENDING, THEN WAKE UP SO THAT WE CAN TERMINATE.
1329 2388 :
1329 2389 BBS #PRC_V_HANGUP,PRC_W_FLAGS(R11),80$ ; WAKE UP IF HANGUP PENDING
1329 2390 :
1329 2391 :
1329 2392 : RE-ENABLE WRITE ATTENTION AST.
1329 2393 :
1329 2394 CLRQ -(SP) ; ALLOCATE AN IOSB
1329 2395 MOVL SP,R4
1329 2396 $QIOW_S FUNC=#IOS SETMODE!IOSM WRTATTN,- ; RE-ENABLE AST
1329 2397 CHAN=PRC_W_OUTMBXCHN(R11),-
1329 2398 EFN=#EXESC_SYSEFN,-
1329 2399 IOSB=(R4),-
1329 2400 P1=WRITE_AST,- ; ADDRESS OF AST ROUTINE
1329 2401 P2=R11 ; ADDRESS OF PRC DATA STRUCTURE
1329 2402 :
1329 2403 :
1329 2404 : ALLOCATE A RECORD BUFFER AND IOSB.
1329 2405 :
1329 2406 MOVZWL PRC_W_OUTMBXSIZ(R11),R5 ; GET SIZE OF RECORD BUFFER
1329 2407 SUBL R5,SP ; ALLOCATE RECORD BUFFER
1329 2408 MOVL SP,R2 ; GET THE BUFFER ADDRESS
1329 2409 :
1329 2410 :
1329 2411 : GET RECORD TO BE WRITTEN FROM THE MAILBOX. IF THERE IS MORE THAN
1329 2412 : ONE MESSAGE IN THE MAILBOX, LOOP UNTIL WE'VE READ AND WRITTEN THEM ALL.
1329 2413 :
1329 2414 10$: $QIOW_S FUNC=#IOS READVBLK!IOSM NOW,- ; READ THE ATTACH MAILBOX
1329 2415 CHAN=PRC_W_OUTMBXCHN(R11),-
1329 2416 EFN=#EXESC_SYSEFN,-
1329 2417 IOSB=(R4),-
1329 2418 P1=(R2),- ; ADDRESS OF BUFFER
1329 2419 P2=R5 ; SIZE OF BUFFER
1329 2420 BLBC R0,30$ ; BRANCH IF DSW ERROR
1329 2421 BLBC (R4),20$ ; BRANCH IF I/O ERROR
1329 2422 CVTWL 2(R4),R1 ; GET LENGTH OF MESSAGE

```

SPAWN
V04-000

- MULTI-PROCESSING COMMANDS
WRITE REQUEST AST FROM A SUBPROCESS

M 15

16-SEP-1984 00:17:05
4-SEP-1984 23:43:20

VAX/VMS Macro V04-00
[DCL.SRC]SPAWN.MAR;1

Page 54
(23)

EC68'	30	1395	2423	BSBW	DCL\$SPAWNOUT	; WRITE THE MESSAGE
CE	11	1398	2424	BRB	10\$	; LOOP UNTIL MAILBOX CLEANED OUT
		139A	2425			
G870 8F	64	B1	139A	2426	20\$: CMPW	(R4),#SS\$_ENDOFFILE
	C7	12	139F	2427	BNEQ	10\$
04	A4	D5	13A1	2428	TSTL	4(R4)
	C2	12	13A4	2429	BNEQ	10\$
			13A6	2430	STATUS	NORMAL
	04		13AD	2431	30\$: RET	
			13AE	2432		
			13AE	2433	80\$: SWAKE_S	
	04		13B9	2434	RET	

; WAS ERROR EOF?
; NO, DISREGARD AND READ AGAIN
; WAS PID SPECIFIED?
; YES, THEN KEEP READING TO EMPTY
; SET NORMAL STATUS
; RETURN
; WAKE UP CURRENT PROCESS

```

13BA 2436 .SBTTL READ AST FROM READING ATTACH REQUEST RESPONSE
13BA 2437 :---
13BA 2438 :
13BA 2439 : THIS AST ROUTINE IS CALLED UPON THE COMPLETION OF READING THE
13BA 2440 : ATTACH REQUEST RESPONSE. IT HANDLES THE PROCESSING OF THE
13BA 2441 : DETACHED FLAG. IT IS DOWN AT AST LEVEL TO INSURE SYNCHRONIZATION
13BA 2442 : BETWEEN THE CURRENT PROCESS AND THE PROCESS TO BE ATTACHED.
13BA 2443 : THE ADDRESS OF A QUAD WORD IS PASSED AS THE AST PARAMETER. THE QUAD WORD
13BA 2444 : CONTAINS THE INPUT BUFFER FROM THE READ COMMAND (1ST LONG WORD) AND
13BA 2445 : THE ADDRESS OF THE PRC TABLE AREA (2ND LONG WORD).
13BA 2446 :
13BA 2447 : INPUTS:
13BA 2448 :
13BA 2449 : 4(SP) = AST PARAMETER
13BA 2450 :
13BA 2451 : OUTPUTS:
13BA 2452 :
13BA 2453 : PRC V DETACHED BIT IN PRC_W_FLAGS SET IF INPUT BUFFER = 1
13BA 2454 : CLEARED OTHERWISE
13BA 2455 :
13BA 2456 :---
13BA 2457 :
0804 13BA 2458 READ_AST:
13BA 2459 .WORD ^M<R2,R11>
13BC 2460 :
52 04 AC D0 13BC 2461 MOVL 4(AP),R2 ; GET ADDR OF QUAD WORD BUFFER
5B 04 A2 D0 13C0 2462 MOVL 4(R2),R11 ; GET ADDR. OF CLI PROCESS WORK AREA
05 62 E9 13C4 2463 BLBC (R2),10$ ; DON'T SET DETACHED BIT IF ATTACH REFUSED
13C7 2464 SETBIT PRC_V_DETACHED,PRC_W_FLAGS(R11) ; MARK CURRENT PROC. DETACHED
13CC 2465 :
04 13CC 2466 10$: RET

```

```

13CD 2468 .SBTTL ATTACH REQUEST AST FROM ANOTHER PROCESS
13CD 2469 :---
13CD 2470 :
13CD 2471 : THIS AST ROUTINE HANDLES AN ATTACH REQUEST FROM ANOTHER PROCESS
13CD 2472 : THROUGH THE ATTACH MAILBOX ASSOCIATED WITH THIS PROCESS.
13CD 2473 :
13CD 2474 : INPUTS:
13CD 2475 :
13CD 2476 : 4(AP) = ADDRESS OF PRC AREA
13CD 2477 :
13CD 2478 : OUTPUTS:
13CD 2479 :
13CD 2480 : NONE
13CD 2481 :---
13CD 2482 :
13CD 2483 ATTACH_AST:
13CD 2484 .WORD ^M<R2,R3,R4,R5,R6,R11>
13CF 2485
5B 04 AC D0 13CF 2486 MOVL 4(AP),R11 ; GET ADDRESS OF CLI PROCESS WORK AREA
56 F0 AE 9E 13D3 2487 MOVAB -ATTMBX_MAXMSG(SP),R6 ; ALLOCATE RECORD BUFFER
5E 56 D0 13D7 2488 MOVL R6,SP ; REMOVE IT FROM THE STACK
13DA 2489
13DA 2490 :
13DA 2491 : IF HANGUP AST IS PENDING, THEN WAKEUP PROCESS SO WE CAN TERMINATE
13DA 2492 :
76 68 AB 0C E0 13DA 2493 BBS #PRC_V_HANGUP,PRC_W_FLAGS(R11),80$ ; WAKE UP IF HANGUP PENDING
13DF 2494
13DF 2495 :
13DF 2496 : GET THE ATTACH REQUEST MESSAGE FROM THE MAILBOX. IF THERE IS MORE THAN
13DF 2497 : ONE MESSAGE IN THE MAILBOX, ONLY HONOR THE LAST ONE.
13DF 2498 :
53 D4 13DF 2499 CLRL R3 ; ASSUME NO MESSAGE READ
7E 7C 13E1 2500 CLRQ -(SP) ; ALLOCATE AN IOSB
52 5E D0 13E3 2501 MOVL SP,R2 ; POINT TO IT
13E6 2502
10$: 13E6 2503 $QIOW_S FUNC=#IOS READVBLK!IOSM_NOW,- ; READ THE ATTACH MAILBOX
13E6 2504 CHAN=PRC @ ATTMBX(R11),-
13E6 2505 EFN=#EXESC_SYSEFN,-
13E6 2506 IOSB=(R2),-
13E6 2507 P1=(R6),P2=#ATTMBX_MAXMSG
09 50 E9 1408 2508 BLBC R0,20$ ; BRANCH IF ERROR
06 62 E9 140B 2509 BLBC (R2),20$ ; BRANCH IF ERROR (EOF)
53 02 A2 32 140E 2510 CVTWL 2(R2),R3 ; SAVE LENGTH OF LAST MESSAGE
D2 11 1412 2511 BRB 10$ ; LOOP UNTIL MAILBOX CLEANED OUT
53 D5 1414 2512 20$: TSTL R3 ; ANY MESSAGE READ?
48 13 1416 2513 BEQL 90$ ; IF NOT, IGNORE ATTENTION AST
02 68 AB 0F E1 141A 2514 CLRL R4 ; ASSUME THE ANSWER IS 'NO'
54 D4 141B 2515 BBC #PRC_V_DETACHED,PRC_W_FLAGS(R11),30$ ; IF NOT DETACHED, SAY NO
52 54 D6 141F 2516 INCL R4 ; ELSE, RESPOND WITH YES
66 54 D0 1421 2517 30$: MOVL SP,R2 ; RESTORE ADDRESS OF IOSB
1424 2518 MOVL R4,(R6) ; MOVE RECORD INTO RECORD BUFFER
1427 2519 $QIOW_S FUNC=#IOS WRITEVBLK,- ; WRITE YES/NO RESPONSE TO MAILBOX
1427 2520 CHAN=PRC @ ATTMBX(R11),-
1427 2521 EFN=#EXESC_SYSEFN,-
1427 2522 IOSB=(R2),-
1427 2523 P1=(R6),P2=#4 ; ONLY A LONGWORD
16 50 E9 1447 2524 BLBC R0,90$ ; BRANCH IF ERROR DETECTED

```

SPAWN
V04-000

- MULTI-PROCESSING COMMANDS
ATTACH REQUEST AST FROM ANOTHER PROCESS

C 16

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 57
(25)

13 62	E9	144A	2525	BLBC	(R2),90\$	: BRANCH IF ERROR DETECTED
10 54	E9	144D	2526	BLBC	R4,90\$	: IF ANSWER WAS NO, THEN EXIT
		1450	2527	CLRBIT	PRC_V_DETACHED,PRC_W_FLAGS(R11)	: MARK NO LONGER DETACHED
		1455	2528	80\$:	SWAKE_S	: WAKE UP CURRENT PROCESS
04		1460	2529	90\$:	RET	

```

1461 2531 .SBTTL SUBPROCESS TERMINATION AST ROUTINE
1461 2532 :---
1461 2533 :
1461 2534 : THIS AST IS CALLED WHEN A MESSAGE IS WRITTEN INTO THE TERMINATION
1461 2535 : MAILBOX, INDICATING THAT A SUBPROCESS HAS GONE AWAY.
1461 2536 :
1461 2537 : INPUTS:
1461 2538 :
1461 2539 : 4(AP) = ADDRESS OF TMBX AREA
1461 2540 :
1461 2541 : OUTPUTS:
1461 2542 :
1461 2543 : NONE
1461 2544 :---
1461 2545 :
000000D3 1461 2546 LGIS_FACILITY = 211 ; LOGINOUT FACILITY CODE
1461 2547 :
09EC 1461 2548 TERMINATION AST:
1461 2549 .WORD ^M<R2,R3,R5,R6,R7,R8,R11>
1463 2550 :
1463 2551 :
1463 2552 : GET THE ADDRESS OF THE TMBX DATA STRUCTURE FROM THE AST ARGUMENT LIST.
1463 2553 : LOOK IN IT FOR THE ADDRESS OF THE PRC DATA STRUCTURE AND THE TERMINATION
1463 2554 : MAILBOX CHANNEL NUMBER.
1463 2555 :
57 04 AC DO 1463 2556 MOVL 4(AP),R7 ; GET ADDRESS OF TMBX BLOCK
5B 0C A7 DO 1467 2557 MOVL TMBX_L_PRC(R7),R11 ; GET ADDRESS OF CLI PROCESS WORK AR
1468 2558 :
1468 2559 :
1468 2560 : READ THE TERMINATION MAILBOX. IF NO MESSAGES ARE LEFT, THEN WE ARE READY TO
1468 2561 : RESET OR DELETE THE MAILBOX AND RETURN FROM THE AST.
1468 2562 :
5E A4 AE 9E 1468 2563 MOVAB -8-ACCSC_TERMLEN(SP),SP ; ALLOCATE BUFFER SPACE FOR
58 5E DO 146F 2564 10$: MOVL SP,R8 ; THE IOSB + RECORD BUFFER
1472 2565 $QIOW_S FUNC=#IOS_READVBLK!IOSM_NOW,- ; READ THE TERMINATION MAILBOX
1472 2566 CHAN=TMBX_W_CHANNEL(R7),-
1472 2567 EFN=#EXESC_SYSEFN,-
1472 2568 IOSB=(R8),-
1472 2569 P1=8(R8),-
1472 2570 P2=#ACCSC_TERMLEN
03 50 E9 1499 2571 BLBC R0,15$ ; BRANCH IF ERROR
03 68 E8 149C 2572 BLBS (R8),20$ ; BRANCH IF MESSAGE READ
00E2 31 149F 2573 15$: BRW 200$ ; MAILBOX CLEANED OUT - DONE
14A2 2574 :
14A2 2575 :
14A2 2576 : SEARCH THE LIST OF OUTSTANDING SUBPROCESS CONTEXT BLOCKS LOOKING FOR
14A2 2577 : A MATCH WITH THE SUBPROCESS WHICH JUST TERMINATED.
14A2 2578 :
14A2 2579 :
55 00C0 CB 9E 14A2 2580 20$: ASSUME SPWN_L LINK EQ 0 ; GET ADDRESS OF SPWN BLOCK LISTHEAD
56 65 DO 14A7 2581 25$: MOVAB PRC[_SPWN(R11),R5 ; GET ADDRESS OF NEXT ENTRY IN LIST
C3 13 14AA 2582 BEQL 10$ ; IF NOT IN LIST, IGNORE MESSAGE
10 A8 D1 14AC 2583 CMPL ACC$L_PID+8(R8),- ; DOES PID MATCH?
40 A6 14AF 2584 SPWN[_SUBPID(R6)
05 13 14B1 2585 BEQL 30$ ; BRANCH IF SO
55 56 DO 14B3 2586 MOVL R6,R5 ; SKIP TO NEXT ENTRY IN LIST
EF 11 14B6 2587 BRB 25$ ; KEEP LOOPING

```

```

14B8 2588
14B8 2589 :
14B8 2590 : THE PROCESS WHICH JUST TERMINATED HAS BEEN FOUND IN THE LIST OF
14B8 2591 : CURRENTLY ACTIVE SPAWNED PROCESSES. TAKE ANY TERMINATION ACTIONS WHICH
14B8 2592 : ARE APPROPRIATE.
14B8 2593 :
14B8 2594 : SET THE TERMINATION STATUS IN THE SPWN BLOCK.
14B8 2595 :
OC A8 DO 14B8 2596 50$: MOVL ACC$L_FINALSTS+8(RB),- : SAVE THE TERMINATION STATUS
44 A6 04 12 14B8 2597 SPWN_L_STATUS(R6) :
44 A6 01 DO 14BD 2598 BNEQ 50$ : BRANCH IF STATUS IS NON-ZERO
14BF 2599 MOVL #1,SPWN_L_STATUS(R6) : IF 'NO STATUS', RETURN SUCCESS
14C3 2600 :
14C3 2601 :
14C3 2602 : GET CURRENT IMAGE COUNT. IF NOT THE SAME AS THE IMAGE COUNT AT THE TIME
14C3 2603 : OF SUBPROCESS CREATION, THEN DO NOT EVEN CONSIDER QUEUEING A TERMINATION,
14C3 2604 : SETTING A TERMINATION EVENT FLAG, OR SETTING TERMINATION STATUS.
14C3 2605 :
F8 AE 7C 14C3 2606 50$: CLRQ -(SP) : CREATE GETJPI ITEM LIST
041A0004 8F DD 14C5 2607 PUSHAB -2*4(SP) : SET BUFFER ADDRESS
50 SE DO 14C8 2608 PUSHL #JPI$_IMAGECOUNT@16+4 : REQUEST IMAGE COUNT, SET BUFFER LE
14CE 2609 MOVL SP,R0 : SET ADDRESS OF ITEM LIST
14D1 2610 $GETJPIW S ITMLST=(R0),- : GET PROCESS IMAGE COUNT
14D1 2611 IOSB=(RB),- :
14D1 2612 EFN=#EXESC_SYSEFN :
SE 51 8ED0 14E8 2613 POPL R1 : GET THE CURRENT IMAGE COUNT
5C A6 51 D1 14EB 2614 ADDL #3*4,SP : CLEANUP STACK
37 12 14EE 2615 CMPL R1,SPWN_L_IMAGCNT(R6) : SAME IMAGE AS ISSUED SPAWN?
14F2 2616 BNEQ 80$ : NO, THEN DO NOT QUEUE AST OR SET E
14F4 2617 :
14F4 2618 :
14F4 2619 : IF USER REQUESTED THAT THE TERMINATION STATUS BE RETURNED, THEN TRY TO RETURN
14F4 2620 : IT NOW.
50 54 A6 DO 14F4 2621 MOVL SPWN_L_STSADR(R6),R0 : SHOULD STATUS BE RETURNED TO CALLE
OD 13 14F8 2622 BEQL 60$ : BRANCH IF NOT
14FA 2623 IFNOWRT #4,@SPWN_L_STSADR(R6),60$,- : SKIP IF NOT WRITABLE
14FA 2624 SPWN_B_ACMODE(R6) :
44 A6 DO 1502 2625 MOVL SPWN_L_STATUS(R6),- : RETURN STATUS TO CALLER
54 B6 1505 2626 @SPWN_L_STSADR(R6) :
1507 2627 :
1507 2628 :
1507 2629 : IF A TERMINATION AST WAS REQUESTED, THEN QUEUE IT UP NOW.
1507 2630 :
4C A6 D5 1507 2631 60$: TSTL SPWN_L_ASTADR(R6) : AST ROUTINE REQUESTED?
10 13 150A 2632 BEQL 70$ : BRANCH IF NOT
150C 2633 $DCLAST_S ASTADR=@SPWN_L_ASTADR(R6),- : QUEUE THE AST
150C 2634 ASTPRM=SPWN_L_ASTPRM(R6),- :
150C 2635 ACMODE=SPWN_B_ACMODE(R6) :
151C 2636 :
151C 2637 :
151C 2638 : IF A TERMINATION EF WAS REQUESTED, THEN SET IT NOW.
151C 2639 :
OF A6 95 151C 2640 70$: TSTB SPWN_B_EFN(R6) : EVENT FLAG REQUESTED?
OA 19 151F 2641 BLSS 80$ : BRANCH IF NOT
1521 2642 $SETEF_S EFN=SPWN_B_EFN(R6) : SET THE EVENT FLAG
152B 2643 :
152B 2644 :

```

```

152B 2645 : PREVENT THE PARENT FROM HANGING TRYING TO SEND ADDITIONAL MESSAGES TO THE
152B 2646 : CONTEXT MAILBOX BY CANCELING AND DEASSIGNING THE CHANNEL TO THAT MAILBOX.
152B 2647 :
152B 2648 80$:  SCANCEL_S CHAN=SPWN_W_CHAN(R6)          : STOP ANY CURRENT WRITES TO MAILBOX
1536 2649      SDASSGN_S CHAN=SPWN_W_CHAN(R6)          : PREVENT FURTHER WRITES TO MAILBOX
0A A6 B4 1541 2650      CLRW   SPWN_W_CHAN(R6)          : INDICATE CHANNEL WAS 'REMOVED'
1544 2651 :
1544 2652 :
1544 2653 : IF /NOTIFY SPECIFIED, THEN BROADCAST NOTIFICATION MESSAGE.
1544 2654 :
1544 2655 90$:  BBC      #SPWN_V_NOTIFY,-              : SKIP IF NOT /NOTIFY
03 OC A6 1546 2656      SPWN_W_FLAGS(R6),95$          :
0067 30 1549 2657      BSBW   NOTIFY_MESSAGE          : BROADCAST NOTIFICATION MESSAGE
154C 2658 :
154C 2659 :
154C 2660 : DECREMENT TERMINATION MAILBOX REFERENCE COUNT.
154C 2661 :
08 A7 97 154C 2662 95$:  DECB      TMBX_B_REFS(R7)          : DECREMENT REFERENCE COUNT TO MAILB
154F 2663 :
154F 2664 :
154F 2665 : DELETE THE WRITE MAILBOX IF ALL THROUGH WITH IT.
154F 2666 :
03 OC A6 154F 2667      BBS      #SPWN_V_OUTPUT,-          : SKIP IF NO MAILBOX
FDBB 30 1551 2668      SPWN_W_FLAGS(R6),97$          :
1554 2669      BSBW   DELETE_OUTMBX          : UPDATE OUTPUT MBX USAGE
1557 2670 :
1557 2671 :
1557 2672 : IF SUBPROCESS WAS CREATED /WAIT, THEN ASSUME THAT THE SUBPROCESS THAT JUST
1557 2673 : DIED WAS ATTACHED TO THE TERMINAL AND IS NOW LOGGING ITSELF OUT. ALSO,
1557 2674 : ASSUME THAT THE PARENT PROCESS WAS DETACHED FROM THE TERMINAL AND SHOULD
1557 2675 : NOW BE AWAKENED TO RECEIVE CONTROL.
1557 2676 :
1557 2677 : NOTE THAT THESE ASSUMPTIONS ARE NOT VALID IN THE CASE OF AN 'ATTACHED'
1557 2678 : COUSIN DELETING ITS 'DETACHED' COUSIN.
1557 2679 :
1557 2680 : ALSO MARK THE SUBPROCESS INACTIVE SO THAT THE PARENT WILL DELETE THIS
1557 2681 : SPWN BLOCK.
1557 2682 :
18 OC A6 1557 2683 97$:  BBC      #SPWN_V_WAIT,-          : SKIP IF /NOWAIT
1559 2684      SPWN_W_FLAGS(R6),100$          :
155C 2685      CLRBIT   PRC_V_DETACHED,PRC_W_FLAGS(R11) : MARK NO LONGER DETACHED
1561 2686      CLRBIT   SPWN_V_ACTIVE,SPWN_W_FLAGS(R6)  : MARK SUBPROCESS INACTIVE
FE'B 31 1566 2687      $WAKE_S          : WAKE UP CURRENT PROCESS
1571 2688      BRW      10$          : GET NEXT MESSAGE FROM MAILBOX
1574 2689 :
1574 2690 :
1574 2691 : IF THE SUBPROCESS WAS CREATED /NOWAIT, THEN REMOVE THE SPWN BLOCK FROM THE
1574 2692 : LIST OF OUTSTANDING PROCESSES, AND DEALLOCATE IT.
1574 2693 :
1574 2694 100$:  ASSUME   SPWN_L_LINK EQ 0          :
65 66 D0 1574 2695      MOVL      (R6),(R5)          : REMOVE FROM LINKED LIST NOW THAT
1577 2696      : THE PROCESS IS NO LONGER ACTIVE
50 56 D0 1577 2697      MOVL      R6,R0          : SET ADDRESS OF BLOCK
51 04 A6 3C 157A 2698      MOVZWL   SPWN_W_SIZE(R6),R1    : SET LENGTH OF BLOCK
EA7F' 30 157E 2699      BSBW      DCL$DEADYNMEM          : DEALLOCATE SPWN BLOCK
FE'EB 31 1581 2700      BRW      10$          : LOOP UNTIL MAILBOX CLEANED OUT
1584 2701 :

```

```

1584 2702 :
1584 2703 : IF REF COUNT FOR THIS MAILBOX IS NOW ZERO, THEN DEASSIGN IT AND REMOVE
1584 2704 : THE TMBX BLOCK FROM THE LINKED LIST. OTHERWISE, RE-ENABLE WRITE ATTENTION
1584 2705 : AST ON THE MAILBOX.
1584 2706 :
08 A7 95 1584 2707 200$: TSTB TMBX_B_REFS(R7) ; TEST REFERENCE COUNT TO MAILBOX
05 12 1587 2708 BNEQ 210$ ; BRANCH IF STILL OUTSTANDING USES
F873 30 1589 2709 BCBW DELETE_TMBX ; DELETE TERMINATION MAILBOX AND TMB
24 11 158C 2710 BRB 220$ ; DO NOT BOTHER RESETTING THE AST
158E 2711 210$: $QIOW,S FUNC=#IOS_SETMODE!IOSM,WRTATTN,- ; RESET ATTENTION AST ON MAILBOX
158E 2712 CHAN=TMBX_W_CHANNEL(R7),-
158E 2713 IOSB=(R8),-
158E 2714 EFN=#EXESC_SYSEFN,-
158E 2715 P1=TERMINATION_AST,-
158E 2716 P2=R7 ; ADDRESS OF AST ROUTINE
04 1582 2717 220$: RET ; PASS ADDRESS OF TMBX BLOCK
1583 2718

```

```

15B3 2720 .SBTTL BROADCAST NOTIFICATION MESSAGE
15B3 2721 :---
15B3 2722 :
15B3 2723 : THIS ROUTINE BROADCASTS A MESSAGE INDICATING THAT A /NOWAIT SUBPROCESS HAS
15B3 2724 : TERMINATED.
15B3 2725 :
15B3 2726 : INPUTS:
15B3 2727 :
15B3 2728 : NONE
15B3 2729 :
15B3 2730 : OUTPUTS:
15B3 2731 :
15B3 2732 : NONE
15B3 2733 :---
15B3 2734 :
15B3 2735 NOTIFY_MESSAGE:
15B3 2736 SUBL #NOTIFY_LEN,SP ; ALLOCATE BUFFER ON THE STACK
15B3 2737 MOVL SP,-(SP) ; BUILD BUFFER DESCRIPTOR
15B3 2738 MOVL #NOTIFY_LEN,-(SP) ;
15B3 2739 MOVL SP,R3 ; SAVE ADDRESS OF DESCRIPTOR
15B3 2740 MOVAB SPWN_T_PROCESS(R6),R2 ; GET PROCESS NAME
15B3 2741 MOVAB NOTIFY_MSG,R1 ; GET ADDRESS OF ASCII MSG
15B3 2742 MOVZBL (R1)+,R0 ; GET LENGTH OF MSG
15B3 2743 MOVQ R0,-(SP) ; PUSH DESCRIPTOR ON STACK
15B3 2744 MOVL SP,R1 ; SAVE ADDRESS OF DESCRIPTOR
15B3 2745 $FAO_S CTRSTR=(R1),- ; NOTIFICATION MESSAGE
15B3 2746 OUTLEN=(R3),- ; ADDR OF RESULT LENGTH
15B3 2747 OUTBUF=(R3),- ; ADDR OF BUFFER DESC
15B3 2748 P1=R2 ; ASCII SUBPROCESS NAME ADDR
15E1 2749 MOVL #CTLSAG CLIDATA+PPDST_INPDVI,R1 ; GET ADDRESS OF ASCII DEVICE NAME
15E8 2750 MOVZBL (R1)+,R0 ; GET LENGTH OF DEVICE
15EB 2751 MOVQ R0,(SP) ; PUT DESCRIPTOR ON STACK
15EE 2752 MOVL SP,R2 ; SAVE ADDRESS OF DESCRIPTOR
15F1 2753 CLRQ -(SP) ;
15F3 2754 MOVL (SP),R0 ; ALLOCATE AN IOSB
15F6 2755
15F6 2756 $BRKTHRUW_S MSGBUF=(R3),- ; BROADCAST THE MESSAGE
15F6 2757 SENDTO=(R2),- ;
15F6 2758 SNDTYP=#BRK$C_DEVICE,- ;
15F6 2759 REQID=#BRK$C_DCL,- ;
15F6 2760 EFN=#31,- ;
15F6 2761 IOSB=(R0) ;
1611 2762
1611 2763 ADDL #8+8+8+NOTIFY_LEN,SP ; RESTORE STACK
1618 2764 RSB
1619 2765
1619 2766 .END

```

SPAWN
Symbol table

- MULTI-PROCESSING COMMANDS

I 16

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 63
(27)

\$\$TMP1	= 00000001			CTX_B_KEYLENGTH	00000002		
\$\$TMP2	= 00000064			CTX_B_NFLAGS	00000005		
\$\$T1	= 00000000			CTX_B_NONUNIQUE	00000006		
\$\$T2	= 00000004			CTX_B_PROMPTLEN	0000000F		
ACCS_TERMINLEN	= 00000054			CTX_B_SYMTAB	00000004		
ACCS_FINALSTS	= 00000004			CTX_B_SYMTYPE	00000005		
ACCS_PID	= 00000008			CTX_B_TFLAGS	00000005		
ATTACH_AST	000013CD	R	02	CTX_B_TRANCNT	00000006		
ATTACH_NAME	0000001C	R	02	CTX_C_BINARY	= 00000002		
ATTMBX_MAXMSG	= 00000010			CTX_C_CLISYM	= 00000003		
BRKSC_DCL	= 00000006			CTX_C_CMDSTR	= 00000001		
BRKSC_DEVICE	= 00000001			CTX_C_GLOBAL	= 00000000		
CHECK_FOR_HANGUP	000011BD	R	02	CTX_C_HDRLEN	00000033		
CHECK_TRMBX	00000884	R	02	CTX_C_HEADER	= 00000000		
CLISK_ATTACH_IDEN	*****	X	02	CTX_C_KEYPAD	= 00000004		
CLISK_SPAW_CARR	*****	X	02	CTX_C_KEYSTATE	= 00000004		
CLISK_SPAW_CLI	*****	X	02	CTX_C_KEYTABL	= 00000002		
CLISK_SPAW_INPU	*****	X	02	CTX_C_LNMNAME	= 00000006		
CLISK_SPAW_KEYP	*****	X	02	CTX_C_LNMTABLE	= 00000005		
CLISK_SPAW_LOG	*****	X	02	CTX_C_LOCAL	= 00000001		
CLISK_SPAW_LOGI	*****	X	02	CTX_C_MAXLEN	= 00000400		
CLISK_SPAW_NOTI	*****	X	02	CTX_C_PERM	= 00000001		
CLISK_SPAW_OUTP	*****	X	02	CTX_C_STRING	= 00000000		
CLISK_SPAW_PROG	*****	X	02	CTX_G_PROMPT	00000013		
CLISK_SPAW_PROM	*****	X	02	CTX_K_HDRLEN	00000033		
CLISK_SPAW_SYMB	*****	X	02	CTX_L_OUTOFBAND	0000000A		
CLISK_SPAW_TABL	*****	X	02	CTX_L_QUOTA	00000008		
CLISK_SPAW_WAIT	*****	X	02	CTX_Q_PROCPRIV	00000002		
CLIS_ATTACHED	= 0003FD09			CTX_T_CMDSTR	00000002		
CLIS_BUFOVF	= 00038018			CTX_T_KEYSTATE	00000003		
CLIS_EXPSYN	= 00038038			CTX_T_LNMNAME	00000007		
CLIS_INVFILSPE	= 00038200			CTX_T_LNMTABLE	0000000C		
CLIS_NORMAL	= 00030001			CTX_T_LOGNAM	00000005		
CLIS_NOTIFY	= 00038250			CTX_T_SYMBOL	00000007		
CLIS_REFUSED	= 000388C2			CTX_V_AUTOLOGO	= 00000000		
CLIS_RETURNED	= 0003FD11			CTX_V_MODE	= 00000001		
CLIS_SPAWNED	= 0003FD01			CTX_V_VERIFY	= 00000002		
CLIS_SPWNIO	= 000388EA			CTX_V_VERIMAGE	= 00000003		
CLIS_STRTOOLNG	= 000388FA			CTX_V_WAIT	= 00000004		
CLIS_SYMTTOOLNG	= 00038218			CTX_W_ENTSIZE	00000002		
CLIS_TRMBX	= 000388F2			CTX_W_PMPCTCTL	00000010		
CLI_NAME	0000006A	R	02	CTX_W_PROT	00000006		
COM	00000060	R	02	CTX_W_TYPE	00000000		
CONSTRUCT_PRCNAM	000004D8	R	02	DCL\$TERM	= 00000042		
CREATE_ATTMBX	00000CE2	R	02	DCL\$ABORT	*****	X	02
CREATE_OUTMBX	00001208	R	02	DCL\$ALLDYNMEM	*****	X	02
CREATE_TMBX	00000D71	R	02	DCL\$ATTACH	00000A01	RG	02
CTLSAG_CLIDATA	*****	X	02	DCL\$ATTACH2	00000A4F	RG	02
CTLSGL_LNMDIRECT	*****	X	02	DCL\$CNVASCIBIN	*****	X	02
CTLSGL_LNMHASH	*****	X	02	DCL\$CRLF	*****	X	02
CTLSGT_CLINAME	*****	X	02	DCL\$C_PROMPTLEN	*****	X	02
CTLSGT_SPAWNCLI	*****	X	02	DCL\$DEADYNMEM	*****	X	02
CTLSGT_SPAWNTABLE	*****	X	02	DCL\$FORMMSG	*****	X	02
CTLSGT_TABLENAME	*****	X	02	DCL\$GETDVAL	*****	X	02
CTX_B_ACMODE	00000004			DCL\$GETNVAL	*****	X	02
CTX_B_CONTINUE	00000012			DCL\$RESETOOB	*****	X	02
CTX_B_FLAGS	0000000E			DCL\$RESTART	*****	X	02

SPAWN
Symbol table

- MULTI-PROCESSING COMMANDS

J 16

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 64
(27)

DCL\$SPAWN	00000098	RG	02	LNMHSHSC_BUCKET	=	000J000C	
DCL\$SPAWN2	0000025F	RG	02	LNMHSHSL_MASK	=	00000000	
DCL\$SPAWNOUT	*****	X	02	LNMTHSL_BYTESLM	=	0000001D	
DEALLOC_SPWN	00000E19	R	02	LNMTHSL_CHILD	=	00000011	
DELETE_ATTMBX	00000D62	R	02	LNMTHSL_NAME	=	00000009	
DELETE_OUTMBX	00001312	R	02	LNMTHSL_PARENT	=	0000000D	
DELETE_TMBX	00000DFF	R	02	LNMTHSL_SIBLING	=	00000015	
DEVSV_REC	= 00000000			LNMXSB_FLAGS	=	00000000	
DEVSV_TRM	= 00000002			LNMXST_XLATION	=	00000004	
DIBSW_UNIT	= 0000000C			LNMXSV_XEND	=	00000002	
DVIS_DEVCLASS	= 00000004			LOG	00000065	R	02
DVIS_DEVNAM	= 00000020			LOGINOUT	00000000	R	02
ENT_R_MAX_PROMPT	= 00000020			NAMSB_BID	=	000J0000	
EXESC_SYSEFN	*****	X	02	NAMSB_BLN	=	00000001	
FABSB_BID	= 00000000			NAMSB_DEV	=	00000039	
FABSB_BLN	= 00000001			NAMSB_ESL	=	0000000B	
FABSB_DNS	= 00000035			NAMSB_ESS	=	0000000A	
FABSB_FNS	= 00000034			NAMSB_NOP	=	00000008	
FABSC_BID	= 00000003			NAMSC_BID	=	00000002	
FABSC_BLN	= 00000050			NAMSC_BLN	=	00000060	
FABSL_DEV	= 00000040			NAMSC_MAXRSS	=	000000FF	
FABSL_DNA	= 00000030			NAMSL_DEV	=	00000044	
FABSL_FNA	= 0000002C			NAMSL_ESA	=	0000000C	
FABSL_FOP	= 00000004			NAMSL_FNB	=	00000034	
FABSL_NAM	= 00000028			NAMSM_NOCONCEAL	=	00000010	
FABSV_PPF	= 00000012			NAMSV_PPF	=	00000010	
GET_DEVICE	000011C6	R	02	NL	0000005B	R	02
IOSM_NOW	= 00000040			NOTIFY_LEN	=	0000002C	
IOSM_TT_PROCESS	= 00002000			NOTIFY_MESSAGE	000015B3	R	02
IOSM_WRTATTN	= 00000100			NOTIFY_MSG	00000079	R	02
IOS_READVBLK	= 00000031			OUTPUT_NAME	0000002B	R	02
IOS_SENSEMODE	= 00000027			PPDSB_KPROCS	0000001C		
IOS_SETMODE	= 00000023			PPDSC_LENTH	00000168		
IOS_WRITEOF	= 00000028			PPDSK_LENTH	00000168		
IOS_WRITEVBLK	= 00000030			PPDSL_INPDEV	00000044		
JPIS_ASTLM	= 00000409			PPDSL_LGI	00000014		
JPIS_BIOLM	= 00000310			PPDSL_LSTSTATUS	00000018		
JPIS_DFWSCNT	= 00000403			PPDSL_OUTDEV	00000064		
JPIS_DIOLM	= 00000313			PPDSL_PRC	00000008		
JPIS_IMAGECOUNT	= 0000041A			PPDSQ_CLIREG	00000004		
JPIS_MASTER_PID	= 00000325			PPDSQ_CLISYMTBL	0000000C		
JPIS_PID	= 00000319			PPDST_FILENAME	00000068		
JPIS_PRCNAM	= 0000031C			PPDST_INPDVI	00000028		
JPIS_PRI8	= 00000309			PPDST_OUTDVI	00000048		
JPIS_PROCPRI8	= 00000204			PPDSW_FLAGS	00000002		
JPIS_USERNAME	= 00000202			PPDSW_INPCHAN	0000001E		
JPIS_WSEXTENT	= 00000416			PPDSW_INPDID	0000003E		
JPIS_WSQUOTA	= 00000402			PPDSW_INPFID	00000038		
LGIS_FACILITY	= 000000D3			PPDSW_INPFI	00000020		
LNMSC_NAMLENGTH	= 000000FF			PPDSW_INPISI	00000022		
LNMB\$B_ACMODE	= 0000000B			PPDSW_OUTDID	0000005E		
LNMB\$B_FLAGS	= 00000010			PPDSW_OUTFID	00000058		
LNMB\$B_TABLE	= 0000000C			PPDSW_OUTIFI	00000024		
LNMB\$M_TABLE	= 00000008			PPDSW_OUTISI	00000026		
LNMB\$T_NAME	= 00000011			PPDSW_SIZE	00000000		
LNMB\$V_CONFIN	= 00000001			PQL\$ASTLM	=	00000001	
LNMB\$V_TABLE	= 00000003			PQL\$BIOLM	=	00000002	

SPAWN
Symbol table

- MULTI-PROCESSING COMMANDS

K 16

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 65
(27)

PQLS_DIOLM = 00000005
PQLS_LISTEND = 00000000
PQLS_WSDEFAULT = 0000000B
PQLS_WSEXTENT = 0000000D
PQLS_WSQUOTA = 0000000A
PRCSM_CLISPEC = 00001000
PRCSM_INTER = 00000400
PRCNAM_NAME = 00000014 R 02
PRC_B_CONTINUE = 000000F3
PRC_B_DEFRADIX = 000000AE
PRC_B_EXMDEPMOD = 000000AD
PRC_B_EXMDEPMID = 000000AC
PRC_B_EXONLYL = 0000012D
PRC_B_FLAGS2 = 000000AF
PRC_B_IMGFLAG = 00000078
PRC_B_OUTFLAGS = 0000012C
PRC_B_PROMPTLEN = 000000F0
PRC_C_LENGTH = 00000534
PRC_G_COMMANDS = 00000133
PRC_G_PROMPT = 000000F4
PRC_K_HEX = 00000000
PRC_K_LENGTH = 00000534
PRC_L_CURRKEY = 00000048
PRC_L_EXMDEPADR = 000000A8
PRC_L_EXTARG = 00000094
PRC_L_EXTBLK = 0000008C
PRC_L_EXTCOD = 0000009C
PRC_L_EXTHND = 00000090
PRC_L_EXTPRM = 00000098
PRC_L_IDFLNK = 000000BC
PRC_L_IMGACTSTS = 00000080
PRC_L_INDCLOCK = 0000007C
PRC_L_INDEPTH = 0000005C
PRC_L_INDFAB = 0000001C
PRC_L_INDINPRAB = 00000014
PRC_L_INDOUTRAB = 00000018
PRC_L_INPRAB = 00000008
PRC_L_LASTKEY = 0000004C
PRC_L_LSTSTATUS = 000000B0
PRC_L_ONCTLY = 000000B8
PRC_L_ONERROR = 0000006C
PRC_L_OUTOFBAND = 000000B4
PRC_L_OUTRAB = 0000000C
PRC_L_OUTRABCTX = 00000118
PRC_L_PPFLIST = 00000070
PRC_L_RECALLPTR = 0000012F
PRC_L_RESTART = 00000058
PRC_L_SAVAP = 00000000
PRC_L_SAVFP = 00000004
PRC_L_SEVERITY = 00000050
PRC_L_SPWN = 000000C0
PRC_L_STACKLM = 000000A4
PRC_L_STACKPT = 000000A0
PRC_L_STATUS = 00000054
PRC_L_STS = 00000084
PRC_L_STV = 00000088
PRC_L_SYMBOL = 00000060

PRC_L_TMBX = 00000074
PRC_L_TRMLIST = 00000010
PRC_Q_ALLOCREG = 00000020
PRC_Q_COMMAND = 000000E0
PRC_Q_FLUSHTIME = 000000D0
PRC_Q_GLOBAL = 00000028
PRC_Q_IMAGENAME = 000000D8
PRC_Q_KEYPAD = 00000040
PRC_Q_LABEL = 00000030
PRC_Q_LOCAL = 00000038
PRC_Q_SAVEPRIV = 000000E8
PRC_T_OUTDVI = 0000011C
PRC_V_DETACHED = 0000000F
PRC_V_HANGUP = 0000000C
PRC_V_MODE = 00000006
PRC_V_VERIFY = 00000007
PRC_V_VERIMAGE = 00000007
PRC_V_YLEVEL = 0000000B
PRC_W_ASTIOSB = 000000C6
PRC_W_ASTRETN = 000000C8
PRC_W_ASTSTATUS = 000000C4
PRC_W_ATTMBX = 0000007A
PRC_W_FLAGS = 00000068
PRC_W_INPCHAN = 00000064
PRC_W_ONLEVEL = 0000006A
PRC_W_OUTIFI = 00000114
PRC_W_OUTISI = 00000116
PRC_W_OUTMBXCHN = 000000CA
PRC_W_OUTMBXREF = 000000CE
PRC_W_OUTMBXSIZ = 000000CC
PRC_W_PMPCTRL = 000000F1
PRC_W_WAITIOSB = 00000066
PSLSC_EXEC = 00000001
PSLSS_PRVMOD = 00000002
PSLSV_PRVMOD = 00000016
PTR_B_LEVEL = 00000004
PTR_B_NUMBER = 00000005
PTR_B_PARMCNT = 00000006
PTR_B_VALUE = 00000000
PTR_C_LENGTH = 0000000C
PTR_K_COLON = 00000002
PTR_K_COMQUAL = 00000000
PTR_K_LENGTH = 0000000C
PTR_K_PARAMETR = 00000003
PTR_L_DESCR = 00000000
PTR_L_ENTITY = 00000008
PTR_V_FLAGS = 00000014
PTR_V_NEGATE = 00000014
READ_AST = 000013BA R 02
RESTORE_CONTEXT = 000006F1 R 02
RETURNED_MESSAGE = 00000E36 R 02
SPAWN_EXIT = 000006C2 R 02
SPAWN_PROCESS = 00000544 R 02
SPWN_B_ACMODE = 0000000E
SPWN_B_CONTINUE = 000000A5
SPWN_B_EFN = 0000000F
SPWN_B_PROMPTLEN = 000000A2

SPAWN
Symbol table

- MULTI-PROCESSING COMMANDS

L 16

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 66
(27)

SPWN_C_LENGTH 000000D6
SPWN_G_PROMPT 000000A6
SPWN_G_QUOTAS 00000060
SPWN_K_LENGTH 000000D6
SPWN_L_ASTADR 0000004C
SPWN_L_ASTPRM 00000050
SPWN_L_IMAGCNT 0000005C
SPWN_L_LINK 00000000
SPWN_L_OUTOFBAND 00000058
SPWN_L_PRI8 00000048
SPWN_L_STATUS 00000044
SPWN_L_STSADR 00000054
SPWN_L_SUBPID 00000040
SPWN_M_CLI = 00002000
SPWN_M_CLISYM = 00000020
SPWN_M_KEYPAD = 00001000
SPWN_M_LOG = 00000001
SPWN_M_LOGNAM = 00000040
SPWN_M_TABLE = 00008000
SPWN_M_WAIT = 00000004
SPWN_Q_CLI 000000C6
SPWN_Q_CMDSTR 00000030
SPWN_Q_INPUT 00000020
SPWN_Q_IOSB 00000038
SPWN_Q_MBXNAM 00000010
SPWN_Q_OUTPUT 00000028
SPWN_Q_PRCNAM 00000018
SPWN_Q_TABLE 000000CE
SPWN_T_PROCESS 00000092
SPWN_V_ACTIVE = 00000007
SPWN_V_AUTOLOGO = 00000003
SPWN_V_CLI = 0000000D
SPWN_V_CLISYM = 00000005
SPWN_V_INPUT = 0000000A
SPWN_V_KEYPAD = 0000000C
SPWN_V_LOG = 00000000
SPWN_V_LOGNAM = 00000006
SPWN_V_MODE = 00000004
SPWN_V_NOCTX = 0000000E
SPWN_V_NOTIFY = 00000008
SPWN_V_OUTPUT = 0000000B
SPWN_V_PRCNAM = 00000001
SPWN_V_PROMPT = 00000009
SPWN_V_TABLE = 0000000F
SPWN_V_WAIT = 00000002
SPWN_W_CHAN 0000000A
SPWN_W_FLAGS 0000000C
SPWN_W_PMPCTRL 000000A3
SPWN_W_SIZE 00000004
SPWN_W_UNIT 00000008
SSS_DUPLNAM = 00000094
SSS_ENDOFFILE = 00000870
SYS_MAXBUF = 0000104F
SYM_B_FLAGS 0000000B
SYM_B_NONUNIQUE 0000000B
SYM_B_TYPE 0000000A
SYM_K_BINARY = 00000002

SYM_K_KEYPAD = 00000004
SYM_K_PERM = 00000001
SYM_K_STPING = 00000000
SYM_L_BL 00000004
SYM_L_FL 00000000
SYM_T_SYMBOL 0000000C
SYM_W_SIZE 00000008
SYSSASSIGN ***** GX 02
SYSSBRKTHRU ***** GX 02
SYSSCANCEL ***** GX 02
SYSSCLREF ***** GX 02
SYSSCREMBX ***** GX 02
SYSSCREPRC ***** GX 02
SYSSDASSGN ***** GX 02
SYSSDCLAST ***** GX 02
SYSSDELPRC ***** GX 02
SYSSFAO ***** X 02
SYSSGETCHN ***** GX 02
SYSSGETDVIW ***** GX 02
SYSSGETJPIW ***** GX 02
SYSSGETSYIW ***** GX 02
SYSSHIBER ***** GX 02
SYSSINPUT 0000003A R 02
SYSSOUTPUT 00000044 R 02
SYSSPARSE ***** GX 02
SYSSQIOW ***** GX 02
SYSSSEARCH ***** GX 02
SYSSSETEF ***** GX 02
SYSSSYSTEM 0000004F R 02
SYSSWAKE ***** GX 02
TERMINATION_AST 00001461 R 02
TMBX_B_REFS 00000008
TMBX_C_LENGTH 00000010
TMBX_C_MAXREFS = 00000004
TMBX_K_LENGTH 00000010
TMBX_L_LINK 00000000
TMBX_L_PRC 0000000C
TMBX_W_CHANNEL 00000006
TMBX_W_SIZE 0000000A
TMBX_W_UNIT 00000004
TT2SV DCL MAILBX = 00000009
VERIFY_INPUT 00000710 R 02
VERIFY_OUTPUT 000008EF R 02
WRITE_AST 00001329 R 02
WRITE_CONTEXT 00000E80 R 02
WRITE_MAILBOX 0000111B R 02
WRITE_SYMBOLS 00001149 R 02
WRK_B_CMDOPT FFFFFFFC3
WRK_B_MAXPARM FFFFFFFD0
WRK_B_MINPARM FFFFFFFD1
WRK_B_PARMCNT FFFFFFFCE
WRK_B_PARMSUM FFFFFFFCF
WRK_B_RECALLCNT FFFFFFFC5
WRK_B_VALLEV FFFFFFFC4
WRK_B_VERBTYP FFFFFFFC2
WRK_C_INBUFSIZ = 00000100
WRK_C_LENGTH FFFFFFF486

SPAWN
Symbol table

- MULTI-PROCESSING COMMANDS

M 16

16-SEP-1984 00:17:05 VAX/VMS Macro V04-00
4-SEP-1984 23:43:20 [DCL.SRC]SPAWN.MAR;1

Page 67
(27)

WRK_G_BUFFER	FFFFFF492
WRK_G_INPBUF	FFFFFF896
WRK_G_RESULT	FFFFFF986
WRK_K_LENGTH	FFFFFF486
WRK_L_CHARPTR	FFFFFF48E
WRK_L_DISALLOW	FFFFFFE6
WRK_L_ERRORRTN	FFFFFF9AE
WRK_L_EXPANDPTR	FFFFFF486
WRK_L_IMAGE	FFFFFFE2
WRK_L_MARKPTR	FFFFFF48A
WRK_L_PAROUT	FFFFFFD2
WRK_L_PMPTADDR	FFFFFF9A2
WRK_L_PROMPTRTN	FFFFFF9A6
WRK_L_PROPTR	FFFFFFC6
WRK_L_QUABLK	FFFFFFCA
WRK_L_READRTN	FFFFFF9AA
WRK_L_RECALLPTR	FFFFFFEA
WRK_L_RSLND	FFFFFFB6
WRK_L_RSLNXT	FFFFFFBA
WRK_L_SAVAP	FFFFFFF8
WRK_L_SAVFP	FFFFFFFC
WRK_L_SAVSP	FFFFFFF4
WRK_L_SIGNALRTN	FFFFFFD6
WRK_L_SPECRTN	FFFFFF9B2
WRK_L_TAB_VEC	FFFFFFDE
WRK_L_VERB	FFFFFFBE
WRK_W_FLAGS	FFFFFFF0
WRK_W_FLAGS2	FFFFFFF2
WRK_W_IMGCHAN	FFFFFFEE
WRK_W_PMPTLEN	FFFFFF9E
SS	= 000000EF

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes												
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE		
\$AB\$\$	FFFFFFFC (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE		
DCL\$ZCODE	00001619 (5657.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	NOWRT	NOVEC	BYTE		

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	16	00:00:00.05	00:00:01.03
Command processing	97	00:00:00.61	00:00:07.31
Pass 1	835	00:00:43.45	00:02:15.40
Symbol table sort	0	00:00:05.26	00:00:14.71
Pass 2	399	00:00:09.91	00:00:32.77
Symbol table output	0	00:00:00.40	00:00:00.95
Psect synopsis output	0	00:00:00.02	00:00:00.38
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1347	00:00:59.71	00:03:12.55

The working set limit was 2700 pages.
240218 bytes (470 pages) of virtual memory were used to buffer the intermediate de.
There were 180 pages of symbol table space allocated to hold 3272 non-local and 181 local symbols.
2766 source lines were read in Pass 1, producing 32 object records in Pass 2.
96 pages of virtual memory were used to define 72 macros.

! Macro library statistics !

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[SYSLIB]SYSBLDMLB.MLB;1	0
\$255\$DUA28:[DCL.OBJ]DCL.MLB;1	14
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	3
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	46
TOTALS (all libraries)	63

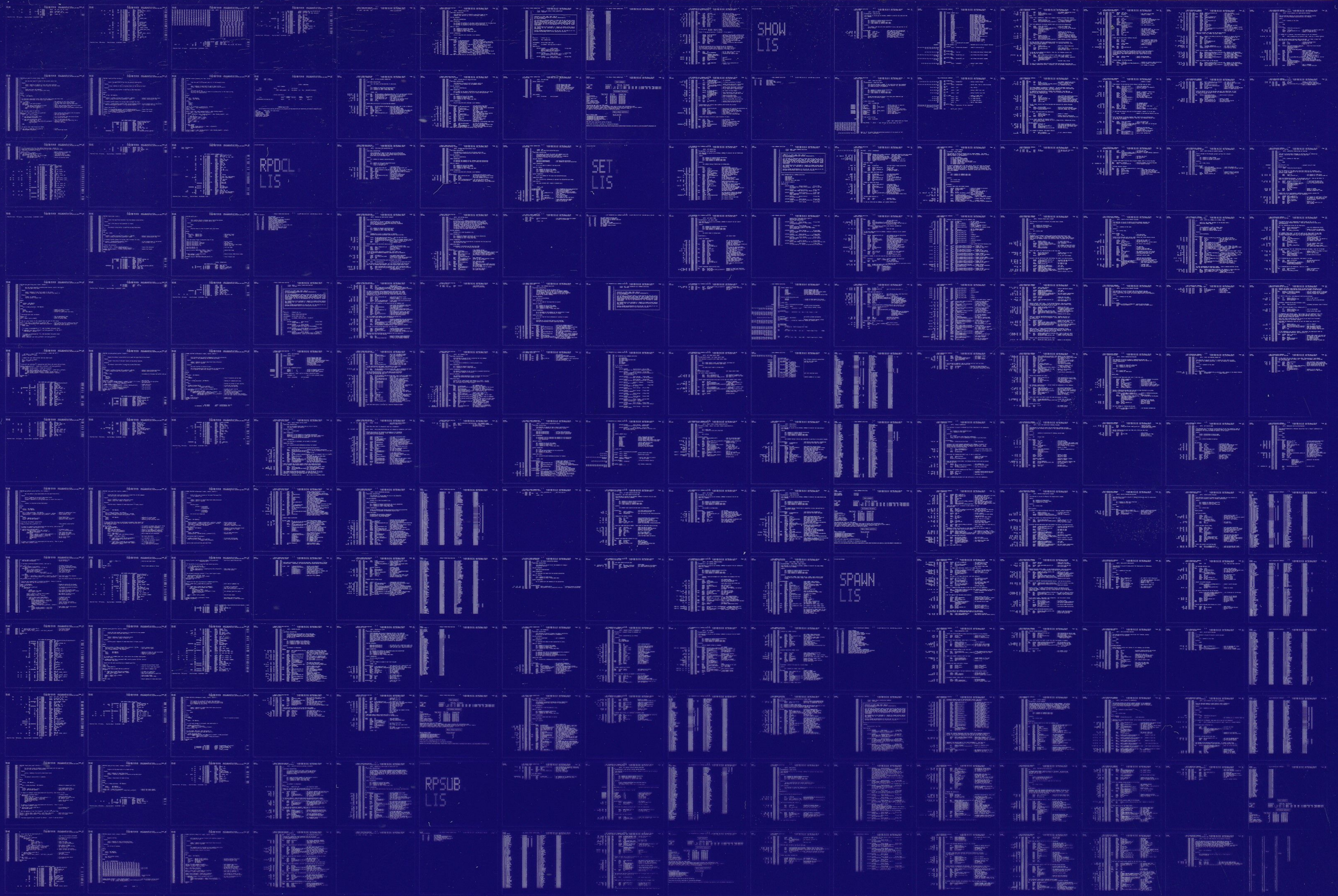
3762 GETS were required to define 63 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS:SPAWN/OBJ=OBJ\$:SPAWN MSRC\$:SPAWN/UPDATE=(ENH\$:SPAWN)+EXECML\$/LIB+LIB\$:DCL/LIB+SYSS\$LIBRARY:SYSBLDMLB/LIB

0073 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0074 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

