

The image is a large, symmetrical, abstract graphic composed of the letters 'S' and 'Y' arranged in a grid-like pattern. The overall shape is a stylized 'W' or a complex letter 'M'. The letters are arranged in a way that creates a sense of depth and perspective, with the 'S' letters forming the outer edges and the 'Y' letters forming the inner structure. The pattern is highly regular and repetitive, with the letters 'S' and 'Y' appearing in a consistent sequence across the entire composition. The background is white, and the letters are black, creating a high-contrast, minimalist aesthetic.

```
PPPPPPPP 000000 SSSSSSSS TTTTTTTTTT EEEEEEEEE EEEEEEEEE FFFFFFFF
PPPPPPPP 000000 SSSSSSSS TTTTTTTTTT EEEEEEEEE FFFFFFFF
PP      PP 00      00 SS      SS      TT      EE      FF
PP      PP 00      00 SS      SS      TT      EE      FF
PP      PP 00      00 SS      SS      TT      EE      FF
PP      PP 00      00 SS      SS      TT      EE      FF
PPPPPPPP 00      00 SSSSSS      TT      EE      FFFFFFFF
PPPPPPPP 00      00 SSSSSS      TT      EE      FFFFFFFF
PP      00      00 SS      SS      TT      EE      FF
PP      00      00 SS      SS      TT      EE      FF
PP      00      00 SS      SS      TT      EE      FF
PP      00      00 SSSSSS      TT      EEEEEEEEE FFFFFFFF
PP      000000 SSSSSSSS      TT      EEEEEEEEE FFFFFFFF
PP      000000 SSSSSSSS      TT      EEEEEEEEE FFFFFFFF
                                     ....
                                     ....
                                     ....
                                     ....
```

```
LL      IIIIII SSSSSSSS
LL      IIIIII SSSSSSSS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SSSSSS
LL      II     SSSSSS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS
```


(1) 56
(1) 90
(1) 224
(1) 267

DECLARATIONS
SCH\$POSTEF - POST EVENT FLAG
CHKWAIT - CHECK FOR WAIT CONDITION SATISFIED
EX\$SHM_CEF_REQ - SEND SHARED MEMORY PROCESSOR REQUEST FOR CEF

```
0000 1 .TITLE POSTEF - POST EVENT FLAGS
0000 2 .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6 *****
0000 7 *
0000 8 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10 * ALL RIGHTS RESERVED.
0000 11 *
0000 12 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17 * TRANSFERRED.
0000 18 *
0000 19 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 * CORPORATION.
0000 22 *
0000 23 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25 *
0000 26 *****
0000 27 *****
0000 28 ++
0000 29 FACILITY: EXECUTIVE, SCHEDULER
0000 30
0000 31 ABSTRACT: POSTS AN EVENT FLAG FOR A PROCESS SPECIFIED BY PID.
0000 32 POSTEF IS AN INTERNAL SERVICE ROUTINE ONLY.
0000 33
0000 34 --
0000 35
0000 36 VERSION:
0000 37
0000 38
0000 39 AUTHOR:
0000 40 R. HUSTVEDT : VERSION
0000 41
0000 42 MODIFIED BY:
0000 43
0000 44 V03-004 KPL0001 Peter Lieberwirth 15-Jan-1984
0000 45 Fix broken word displacement.
0000 46
0000 47 V03-003 KDM0002 Kathleen D. Morse 28-Jun-1982
0000 48 Added $PRDEF.
0000 49
0000 50 V03-002 KDM0001 Kathleen D. Morse 11-May-1982
0000 51 Set the type and size into the pool block allocated
0000 52 via EXESALONONPAGED.
0000 53
0000 54
```



```

0000 56      .SBTTL  DECLARATIONS
0000 57
0000 58 :
0000 59 : INCLUDE FILES:
0000 60 :
0000 61
0000 62      $CEBDEF      : CEB DEFINITIONS
0000 63      $DYNDDEF     : DYNAMIC DATA STRUCTURE TYPE DEFS
0000 64      $FKBDEF      : FORK BLOCK DEFINITIONS
0000 65      $IPLDEF      : IPL DEFINITIONS
0000 66      $IRPDEF      : I/O REQUEST PACKET DEFS
0000 67      $PCBDEF      : PCB DEFINITIONS
0000 68      $PRDEF       : PROCESSOR REGISTERS DEFINITIONS
0000 69      $PRQDEF      : INTER-PROCESSOR REQUEST BLOCK DEFS
0000 70      $SHBDEF      : SHARED MEMORY CONTROL BLOCK DEFS
0000 71      $SSDEF       : STATUS CODE DEFINITIONS
0000 72      $STATEDDEF   : STATE DEFINITIONS
0000 73 :
0000 74 : EXTERNAL SYMBOLS:
0000 75 :
0000 76 :
0000 77 :
0000 78 : MACROS:
0000 79 :
0000 80 :
0000 81 :
0000 82 : EQUATED SYMBOLS:
0000 83 :
0000 84 :
0000 85 :
0000 86 : FIXED DATA:
0000 87      .PSECT  AEXENONPAGED,BYTE      ;NONPAGED EXEC
0000 88

```

```
0000 90 .SBTTL SCH$POSTEF - POST EVENT FLAG
0000 91
0000 92 :++
0000 93 : FUNCTIONAL DESCRIPTION:
0000 94 : POST EVENT FLAG WILL SET AN EVENT FLAG SPECIFIED BY NUMBER
0000 95 : FOR A PROCESS SPECIFIED BY PROCESS ID(PID) AND CAUSE ANY
0000 96 : NECESSARY RESCHEDULING IF THIS EVENT SATISFIES A WAIT
0000 97 : CONDITION FOR THE PROCESS.
0000 98
0000 99 : CALLING SEQUENCE:
0000 100
0000 101 : JSB SCH$POSTEF (ALSO BSB)
0000 102
0000 103 : INPUT PARAMETERS:
0000 104 : R1 - PROCESS IDENTIFICATION (PID)
0000 105 : R2 - PRIORITY INCREMENT CLASS NUMBER
0000 106 : R3 - EVENT FLAG NUMBER
0000 107
0000 108 : IMPLICIT INPUTS:
0000 109 : NONE
0000 110
0000 111 : OUTPUT PARAMETERS:
0000 112 : R0 - COMPLETION STATUS CODE
0000 113 : R4 - PCB ADDRESS OF PROCESS SPECIFIED BY PID
0000 114
0000 115 : IMPLICIT OUTPUTS:
0000 116 : EVENT FLAG BIT SELECTED BY PID AND EVENT FLAG NUMBER
0000 117 : IS SET.
0000 118
0000 119 : COMPLETION CODES:
0000 120 : SSS_NORMAL - NORMAL SUCCESSFUL COMPLETION
0000 121 : SSS_NONEXPR - NON-EXISTENT PROCESS
0000 122 : SSS_WASCLR - SPECIFIED EVENT WAS CLEAR INITIALLY
0000 123 : SSS_WASSET - SPECIFIED EVENT WAS SET INITIALLY
0000 124 : SSS_ILLEFC - ILLEGAL CLUSTER NUMBER
0000 125 : SSS_UNASEFC - UNASSIGNED CLUSTER NUMBER
0000 126
0000 127 : SIDE EFFECTS:
0000 128 : CAN CAUSE RESCHEDULING OF ONE OR MORE PROCESSES.
0000 129 :--
0000 130
0000 131 SCH$POSTEF::
0000 132 : POST EVENT FLAG ENTRY
0003 133 : GET INDEX TO PCB ADDRESS
54 00000000'FF40 D0 0009 134 : BLOCK SYSTEM EVENTS
60 A4 51 D1 0011 135 : AND GET PCB ADDRESS
51 53 03 05 EE 0015 136 : CHECK FOR PID MATCH
2D 19 001C 137 : PROCESS IS GONE
53 FFFFFFFE0 8F CA 001E 138 : GET CLUSTER NUMBER
50 50 A441 DE 0025 139 : BR IF ILLEGAL CLUSTER NUMBER
01 DD 002A 140 : EXTRACT EVENT NUMBER
32 51 F5 002C 141 : GET CLUSTER OR PTR ADDR
002F 142 : ASSUME FLAG WAS CLEAR
002F 143 : CHECK FOR COMMON
13 60 53 E2 002F 144 LOCAL:
51 2E A4 9A 0033 145 : POST LOCAL EF
10$: MOVZBL R3,(R0),15$ : SET FLAG, BR IF ALREADY SET
PCB$B_WFC(R4),R1 : GET NUMBER OF WAIT CLUSTER
```



```
51 50 A441 DE 0037 147 MOVAL PCB$EFC$(R4)[R1],R1 ; AND COMPUTE ITS ADDRESS
      00A5 30 003C 148 BSBW CHKWAIT ; IS WAIT SATISFIED?
      50 8ED0 003F 149 EXITN: POPL R0 ; SET RETURN STATUS CODE
      0042 150 EXITEN: ENBINT ;
      0045 151 EXIT: ;
      05 0045 152 RSB ; RETURN
      0046 153 ;
      6E 09 9A 0046 154 15$: MOVZBL #SS$_WASSET,(SP) ; SET STATUS, FLAG ALREADY SET
      E8 11 0049 155 BRB 10$ ; CONTINUE
      004B 156 .DSABL LSB
      004B 157
      50 00EC 8F 3C 004B 158 ILLEFC: MOVZWL #SS$_ILLEFC,R0 ; SET ERROR STATUS CODE
      FO 11 0050 160 BRB EXITEN ; RETURN ERROR TO CALLER
      0052 161
      50 08E8 8F 3C 0052 162 NONEXPR: MOVZWL #SS$_NONEXPR,R0 ; NONEXISTENT PROCESS
      E9 11 0057 164 BRB EXITEN ; SET NONEXISTENT CODE
      0059 165 ; AND EXIT ENABLING
      0059 166 .ENABL LSB
      0059 167 UNASEFC: MOVZWL #SS$_UNASEFC,12(SP) ; SET ERROR STATUS CODE
      2C 11 005F 169 BRB 30$ ; RESET REGISTERS
      0061 170
      0061 171 COMMON: ; COMMON EVENT POST
      7E 54 DD 0061 172 PUSHL R4 ; SAVE R4
      55 7E 55 7D 0063 173 MOVQ R5,-(SP) ; SAVE R5,R6
      60 14 C1 C066 174 ADDL3 #CEB$_WQFL,(R0),R5 ; POINT TO HEAD OF WAIT QUEUE
      ED 18 006A 175 BGEQ UNASEFC ; BR IF CLUSTER IS NOT ASSIGNED
      54 65 D0 006C 176 MOVL (R5),R4 ; GET HEAD OF WAIT QUEUE
      F6 A5 2D 91 006F 177 CMPB #DYN$_SLAVCEB,<CEB$_TYPE-CEB$_WQFL>(R5) ; IS IT A SLAVE CEB?
      26 13 0073 178 BEQL 50$ ; BR IF IT IS A SLAVE
      1B FC A5 53 E6 0075 179 BBSSI R3,<CEB$_EFC-CEB$_WQFL>(R5),40$ ; SET EVENT FLAG; CHECK STATUS
      55 54 D1 007A 180 20$: CMPL R4,R5 ; CHECK FOR END OF WAIT QUEUE
      OE 13 007D 181 BEQL 30$ ; DONE
      56 64 D0 007F 182 MOVL (R4),R6 ; SAVE REAL FLINK
      50 FC A5 DE 0082 183 MOVAL <CEB$_EFC-CEB$_WQFL>(R5),R0 ; POINT TO EVENT FLAG MASK
      61 10 0086 184 BSBB EXE$CHRWAIT2 ; CHECK FOR WAIT SATISFIED
      54 56 D0 0088 185 MOVL R6,R4 ; NEXT IN WAIT QUEUE
      ED 11 008B 186 BRB 20$ ; CONTINUE
      55 8E 7D 008D 187 30$: MOVQ (SP)+,R5 ; RESTORE R5,R6
      54 8E D0 0090 188 MOVL (SP)+,R4 ; AND R4
      AA 11 0093 189 BRB EXITN ; AND EXIT
      0095 190
      OC AE 09 9A 0095 191 40$: MOVZBL #SS$_WASSET,12(SP) ; SET STATUS, FLAG ALREADY SET
      DF 11 0099 192 BRB 20$ ; CONTINUE
      009B 193
      009B 194 ; SET SHARED MEMORY COMMON EVENT FLAG.
      009B 195
      009B 196
      50 DD 009B 197 50$: PUSHL R0 ; SAVE R0
      7E 51 7D 009D 198 MOVQ R1,-(SP) ; SAVE REGISTERS
      50 60 D0 00A0 199 MOVL (R0),R0 ; GET ADR OF SLAVE CEB
      52 40 A0 D0 00A3 200 MOVL CEB$_MASTER(R0),R2 ; GET ADR OF MASTER CEB
      32 10 A2 53 E6 00A7 201 BBSSI R3,CEB$_EFC(R2),110$ ; SET THE MASTER COPY FIRST/QUIT IF SET
      51 1D A2 9A 00AC 202 MOVZBL CEB$_PROCCNT(R2),R1 ; GET MAX # OF SLAVE CEB'S
      51 D7 00B0 203 DECL R1 ; CONVERT COUNT OF PORTS INTO INDEX
```


38	A241	D5	00B2	204	60\$:	TSTL	CEB\$L_VASLAVE1(R2)[R1]	:	IS THERE A SLAVE CEB?
	1B	13	00B6	205		BEQL	90\$	:	BR IF NO SLAVE, DON'T SEND PROC MSG
38	A241	50	D1	00B8	206	CMPL	R0,CEB\$L_VASLAVE1(R2)[R1]	:	IS THIS THE CURRENT PROCESSOR?
	0F	13	00BD	207		BEQL	80\$	:	BR TO SET FLAG/DON'T SEND MSG TO SELF
	3F	BB	00BF	208		PUSHR	#^M<R0,R1,R2,R3,R4,R5>	:	SAVE REGISTERS
	41	10	00C1	209		BSBB	EX\$SHM_CEF_REQ	:	SEND PROCESSOR REQUEST FOR CEF COPY
	04	50	E8	00C3	210	BLBS	R0,70\$	:	BR IF SUCCESSFUL
30	AE	50	D0	00C6	211	MOVL	R0,48(SP)	:	REMEMBER ERROR CODE IN R0
	3F	BA	00CA	212	70\$:	POPR	#^M<R0,R1,R2,R3,R4,R5>	:	RESTORE REGISTERS
	05	11	00CC	213		BRB	90\$	:	CONTINUE CHECKING SLAVE CEB'S
10	A0	10	A2	D0	00CE	214	80\$:	MOVL	CEB\$L_EFC(R2),CEB\$L_EFC(R0)
	DC	51	F4	00D3	215	90\$:	SOBGEQ	R1,60\$	:
	51	8E	7D	00D6	216	100\$:	MOVQ	(SP)+,R1	:
	50	8ED0	00D9	217		POPL	R0	:	RESTORE EVENT FLAG NUMBER
	9C	11	00DC	218		BRB	20\$	:	RESTORE R0
			00DE	219				:	JOIN COMMON CODE
18	AE	09	9A	00DE	220	110\$:	MOVZBL	#SS\$_WASSET,24(SP)	:
	F2	11	00E2	221		BRB	100\$	:	SET STATUS CODE, FLAG ALREADY SET
			00E4	222		.DSABL	LSB	:	CONTINUE


```
00E4 224 .SBTTL CHKWAIT - CHECK FOR WAIT CONDITION SATISFIED
00E4 225
00E4 226 :++
00E4 227 : FUNCTIONAL DESCRIPTION:
00E4 228 :   CHKWAIT CHECKS THE WAIT CONDITION SPECIFIED IN THE PCB
00E4 229 :   AND RESTARTS THE PREOCESS IF IT IS SATISFIED.
00E4 230 :
00E4 231 : CALLING SEQUENCE:
00E4 232 :   JSB/BSB CHKWAIT
00E4 233 :
00E4 234 : INPUT PARAMETERS:
00E4 235 :   R0 - EVENT FLAG CLUSTER ADDRESS
00E4 236 :   R1 - ADDRESS OF EVENT FLAG CLUSTER THAT PROCESS IS WAITING ON
00E4 237 :   R2 - PRIORITY INCREMENT CLASS
00E4 238 :   R4 - PCB ADDRESS
00E4 239 :
00E4 240 : SIDE EFFECTS:
00E4 241 :   IF THE WAIT CONDITION IS SATISFIED, THE PROCESS WILL RESUME
00E4 242 :   EXECUTION. R3 IS DESTROYED.
00E4 243 :
00E4 244 :--
00E4 245
00E4 246 CHKWAIT: ; CHECK FOR WAIT SATISFIED
00E4 247 .ENABL LSB ;
00E4 248 CMPL R0,R1 ; IS PROCESS WAITING FOR THIS
00E7 249 BNEQ 20$ ; CLUSTER? BR IF NOT
00E9 250 EXE$CHKWAIT2: ;
00E9 251 BICL3 PCB$L_EFWM(R4),(R0),R3 ; TEST WITH MASK
00EE 252 BEQL 20$ ; WAIT NOT SATISFIED
00F0 253 BBS #PCB$V_WALL,PCB$L_STS(R4),30$ ; CHK FOR WAIT ANY
00F5 254 :
00F5 255 : AT THIS POINT:
00F5 256 :   R2 - PRIORITY INCREMENT CLASS NUMBER
00F5 257 :   R4 - PCB ADDRESS OF PROCESS TO ACTIVATE
00F5 258 :
00F5 259 10$: RPTEVT EVENT ; REPORT EVENT SET
00F9 260 20$: RSB ; RETURN
00FA 261 30$: MCOML R3,R3 ; INVERT MASKED EVENTS
00FD 262 CMPL PCB$L_EFWM(R4),R3 ; CHECK FOR ALL
0101 263 BEQL 10$ ; WAIT NOW SATISFIED
0103 264 RSB ; RETURN
0104 265 .DSABL LSB
```

51 50 D1 00E4 248
10 12 00E7 249
53 60 4C A4 CB 00E9 251
09 13 00EE 252
05 24 A4 OD E0 00F0 253
00F5 254
00F5 255
00F5 256
00F5 257
00F5 258
00F5 259
05 00F9 260
53 53 D2 00FA 261
4C A4 D1 00FD 262
F2 13 0101 263
05 0103 264
0104 265

0104 267 .SBTTL EXESSHM_CEF_REQ - SEND SHARED MEMORY PROCESSOR REQUEST FOR CEF

0104 268 :++
0104 269 : FUNCTIONAL DESCRIPTION:
0104 270 :

0104 271 : THIS ROUTINE SENDS A MESSAGE TO ANOTHER PROCESSOR VIA A SHARED MEMORY
0104 272 : REQUESTING IT TO UPDATE ITS SLAVE CEB. THIS IS DONE WHENEVER A COMMON
0104 273 : EVENT FLAG IN SHARED MEMORY GOES FROM CLEAR TO SET. THIS ROUTINE MUST
0104 274 : BE ENTERED VIA A JSB-TYPE INSTRUCTION SO THAT THE CO-ROUTINE CALLS
0104 275 : BETWEEN ITSELF AND MASREQUEST WORK CORRECTLY.
0104 276 :

0104 277 : FIRST, A FORK BLOCK IS ALLOCATED. ALL INFORMATION TO BE CONTAINED IN
0104 278 : THE INTER-PROCESSOR REQUEST BLOCK IS STORED IN THIS FORK BLOCK.
0104 279 : MASREQUEST IS THEN CALLED TO ALLOCATE A REQUEST BLOCK. IF THERE IS NO
0104 280 : FREE BLOCK, MASREQUEST RETURNS TO THE CALLER OF SHM_CEF_REQ, I.E.,
0104 281 : SCH\$POSTEF. LATER ON, WHEN A REQUEST BLOCK IS FREED UP, A CO-ROUTINE
0104 282 : CALL RETURNS TO SHM_CEF_REQ WITH THE ADDRESS OF THE REQUEST BLOCK AND
0104 283 : THE ADDRESS OF THE FORK BLOCK. SHM_CEF_REQ THEN INITIALIZES THE
0104 284 : REQUEST BLOCK, DEALLOCATES THE FORK BLOCK, AND RETURNS TO MASREQUEST.
0104 285 : THE OTHER PROCESSOR WILL HANDLE DEALLOCATION OF THE REQUEST BLOCK.
0104 286 :

0104 287 : THE SHARED MEMORY MASTER COMMON EVENT BLOCK MAY BE RELEASED AND
0104 288 : RE-USED DURING THIS ASYNCHRONOUS PROCESS. THIS DOES NOT MATTER
0104 289 : SINCE AN EXTRANEIOUS COPY OF THE FLAGS FROM THE MASTER CEB TO THE
0104 290 : SLAVE CEB WILL HURT NOTHING. THE RECEIVING LOGIC MERELY EXITS
0104 291 : IF NO SLAVE CEB EXISTS WHEN THE PRQ IS RECEIVED.
0104 292 :

0104 293 : THIS ROUTINE MAY OR MAY NOT RETURN EXECUTION TO SCH\$POSTEF. IF THERE IS
0104 294 : AN ERROR ALLOCATING A FORK BLOCK, THIS ERROR IS RETURNED TO SCH\$POSTEF.
0104 295 : ONCE MASREQUEST IS CALLED, THE FLOW OF EXECUTION MAY PROCEED DIRECTLY
0104 296 : FROM MASREQUEST BACK TO SCH\$POSTEF. THIS OCCURS IF THERE IS NO REQUEST
0104 297 : PACKET IMMEDIATELY AVAILABLE. IT IS ASSUMED THAT A PACKET WILL
0104 298 : EVENTUALLY BECOME AVAILABLE AND THEREFORE A SUCCESSFUL STATUS IS
0104 299 : RETURNED TO SCH\$POSTEF. WHEN THE PACKET BECOMES AVAILABLE, THE REST OF
0104 300 : SHM_CEF_REQ IS EXECUTED AS A FORK PROCESS. ANY ERRORS ENCOUNTERED AT
0104 301 : THIS TIME ARE NOT REPORTED TO THE USER AS THEY OCCUR ASYNCHRONOUSLY
0104 302 : TO HIS REQUEST.
0104 303 :

0104 304 : IF THERE IS A REQUEST PACKET AVAILABLE, THEN THE FLOW OF EXECUTION
0104 305 : RETURNS FROM MASREQUEST TO SHM_CEF_REQ VIA A CO-ROUTINE CALL.
0104 306 : SHM_CEF_REQ INITIALIZES THE REQUEST BLOCK AND RELEASES THE FORK
0104 307 : BLOCK. THEN IT RETURNS TO MASREQUEST. MASREQUEST PLACES THE REQUEST
0104 308 : ON THE APPROPRIATE QUEUE, INTERRUPTS THE APPROPRIATE PORT, AND RETURNS
0104 309 : TO SCH\$POSTEF.
0104 310 :

0104 311 : IF MASREQUEST ENCOUNTERS AN ERROR (SUCH AS A BAD QUEUE HEADER), IT
0104 312 : RETURNS THIS ERROR CODE TO SHM_CEF_REQ VIA AN RSB INSTEAD OF A
0104 313 : CO-ROUTINE CALL. SHM_CEF_REQ RELEASES THE FORK BLOCK AND RETURNS,
0104 314 : ONLY NOT TO MASREQUEST, IN THIS CASE, BUT TO SCH\$POSTEF. SCH\$POSTEF
0104 315 : WILL RETURN THIS ERROR CODE TO THE USER. NOTE THAT THE SYSTEM
0104 316 : SERVICE WILL HAVE PARTIAL SUCCESS AS THE MASTER CEB FLAG WILL HAVE
0104 317 : BEEN UPDATED.
0104 318 :

0104 319 : CALLING SEQUENCE:

0104 320 :
0104 321 : JSB EXESSHM_CEF_REQ
0104 322 :

0104 323 : INPUT PARAMETERS:


```
0104 324 :
0104 325 : RO - VA OF SLAVE CEB FOR PORT THAT THE REQUEST IS FROM
0104 326 : R1 - PORT # TO SEND THE REQUEST TO
0104 327 : R2 - ADDRESS OF THE MASTER CEB IN SHARED MEMORY
0104 328 :
0104 329 : IMPLICIT PARAMETERS:
0104 330 :
0104 331 : THE SHARED MEMORY DATA STRUCTURES.
0104 332 :
0104 333 : OUTPUT PARAMETERS:
0104 334 :
0104 335 : RO - COMPLETION STATUS CODE
0104 336 : R1 - R5 DESTROYED
0104 337 :
0104 338 : COMPLETION CODES:
0104 339 :
0104 340 : $$$ NORMAL - SUCCESSFUL COMPLETION
0104 341 : $$$ BADQUEUEHDR - UNABLE TO ALLOCATE INTER-PROCESSOR REQUEST BLOCK
0104 342 : OTHERS FROM ALLOCATION/DEALLOCATION ROUTINES.
0104 343 : --
0104 344 :
0104 345 : EXE$SHM_CEF_REQ: ; RTN TO SEND INTER-PROCESSOR REQUEST
0104 346 :
0104 347 : .ENABL LSB
0104 348 : ASSUME IRP$C LENGTH GT <FKB$C_LENGTH + 12>
0104 349 : PUSHR #^M<R0,R1,R2,R3> ; SAVE REGISTERS
51 00C4 8F BB 0106 350 : MOVZWL #IRP$C_LENGTH,R1 ; ALLOCATE IRP INSTEAD OF FKB FOR SPEED
FEF2' 30 010B 351 : BSBW EXE$ALONONPAGED ; GET A FORK BLOCK BUT DO NOT WAIT
010E 352 : ; FOR THE RESOURCE IF NONE IS AVAILABLE
05 50 E8 010E 353 : BLBS R0,$$ ; BR IF SUCCESSFUL, BLOCK ALLOCATED
6E 50 D0 0111 354 : MOVL R0,(SP) ; REMEMBER ERROR CODE
54 11 0114 355 : BRB RETURN ; RETURN ERROR CODE TO CALLER
08 A2 51 B0 0116 356 5$: MOVW R1,FKB$W_SIZE(R2) ; SET SIZE OF BLOCK ALLOCATED
0A A2 08 90 011A 357 : MOVB #DYN$C_FRK,FKB$B_TYPE(R2) ; SET TYPE TO FORK BLOCK
55 52 D0 011E 358 : MOVL R2,R5 ; SAVE ADDRESS OF FKB
OF BA 0121 359 : POPR #^M<R0,R1,R2,R3> ; RESTORE REGISTERS
0123 360 :
0123 361 : INITIALIZE THE FORK BLOCK WITH EVERYTHING NEEDED TO INITIALIZE THE
0123 362 : INTER-PROCESSOR REQUEST PACKET AS EXECUTION MAY CHANGE TO A FORK PROCESS.
0123 363 :
08 A5 06 90 0123 364 : MOVB #IPL$_QUEUEAST,FKB$B_FIPL(R5) ; SET FORK IPL
18 A5 51 B0 0127 365 : MOVW R1,<FKB$C_LENGTH>(R5) ; SET PORT # REQUEST IS SENT TO
54 38 A0 D0 012B 366 : MOVL CEB$S_SHBTR0,R4 ; GET ADR OF SH MEM CONTROL BLOCK
1A A5 15 A4 9B 012F 367 : MOVZBW SHB$B_PORT(R4),<FKB$C_LENGTH+2>(R5) ; SET PORT # SENDING REQUEST
20 A5 00 9A 0134 368 : MOVZBL #PRQ$C_SETEF,<FKB$C_LENGTH+8>(R5) ; REMEMBER SUB-TYPE OF PRQ BLK
1C A5 3C A0 3C 0138 369 : MOVZWL CEB$W_INDXX(R0),<FKB$C_LENGTH+4>(R5) ; SET INDEX TO MASTER CEB
54 1C A4 D0 013D 370 : MOVL SHB$S_ADP(R4),R4 ; ADR OF ADAPTOR CONTROL BLOCK
0141 371 :
0141 372 : NOW ACQUIRE AN INTER-PROCESSOR REQUEST BLOCK. EXECUTION THROUGH THE
0141 373 : FOLLOWING LINES OF CODE MAY BE ASYNCHRONOUS IF NO BLOCK IS CURRENTLY
0141 374 : AVAILABLE. ASYNCHRONOUS EXECUTION WILL BE DONE AS A FORK PROCESS.
0141 375 :
00000000'EF 16 0141 376 20$: JSB MAS$REQUEST ; ALLOCATE A REQUEST BLOCK
0147 377 : ; R4 AND R5 MUST BE PRESERVED THRU RSB
OF BB 0147 378 : PUSHR #^M<R0,R1,R2,R3> ; SAVE REGISTERS AS SET BY MAS$REQUEST
18 50 E9 0149 379 : BLBC R0,ERROR_PRQ ; BR IF ERROR ALLOCATING REQUEST BLOCK
014C 380 :
```

```
014C 381 ; R2 - INTER-PROCESSOR REQUEST BLOCK ADDRESS
014C 382 ; R5 - FORK BLOCK ADDRESS
014C 383 ;
014C 384
18 A2 18 A5 D0 014C 385 ASSUME PRQ$W FR PORT EQ <PRQ$W TO PORT + 2>
0B A2 0B A5 90 0151 386 MOVL <FKB$C_LENGTH>(R5),PRQ$W TO PORT(R2) ; SET PORT #'S
1C A2 00 B0 0156 387 MOVW FKB$B_FIPL(R5),FKB$B_FIPL(R2) ; SET REQUEST HANDLER IPL
20 A2 20 A5 B0 015A 388 MOVW #PRQ$C_EXEC,PRQ$W_DISPATCH(R2) ; SET EXECUTIVE REQUEST TYPE
24 A2 1C A5 D0 015F 389 MOVW <FKB$C_LENGTH+8>(R5),PRQ$W_REQTYPE(R2) ; SET REQUEST SUB-TYPE
50 55 D0 0164 390 MOVW <FKB$C_LENGTH+4>(R5),PRQ$W_PARAM(R2) ; SET INDEX TO MASTER CEB
FE96' 30 0167 391 ERROR_PRQ: MOVL R5,R0 ; SET ADR OF FORK BLOCK TO RELEASE
OF BA 016A 392 RETURN: BSBW EXE$DEANONPAGED ; RELEASE FORK BLOCK
05 016C 393 POPR #^M<R0,R1,R2,R3> ; RESTORE REGISTERS AND ERROR CODE
016D 394 RSB ; RETURN TO MAS$REQUEST OR SCH$POSTEF
016D 396
016D 397 .DSABL LSB
016D 398
016D 399 .END
```


POSTEF
Symbol table

- POST EVENT FLAGS

G 7

16-SEP-1984 00:57:27 VAX/VMS Macro V04-00
5-SEP-1984 03:46:19 [SYS.SRC]POSTEF.MAR;1

Page 10
(1)

CEBSB_PROCCNT	= 0000001D		
CEBSB_TYPE	= 0000000A		
CEBSL_EFC	= 00000010		
CEBSL_MASTER	= 00000040		
CEBSL_SHB	= 00000038		
CEBSL_VASLAVE1	= 00000038		
CEBSL_WQFL	= 00000014		
CEBSW_INDXX	= 0000003C		
CHKWAIT	000000E4	R	02
COMMON	00000061	R	02
DYN\$C_FRK	= 00000008		
DYN\$C_SLAVCEB	= 0000002D		
ERROR_PRQ	00000164	R	02
EVT\$ EVENT	*****	X	02
EXE\$ALONONPAGED	*****	X	02
EXE\$CHKWAIT2	000000E9	RG	02
EXE\$DEANONPAGED	*****	X	02
EXE\$SHM_CEF_REQ	00000104	R	02
EXIT	00000045	R	02
EXITEN	00000042	R	02
EXITN	0000003F	R	02
FKBSB_FIPL	= 0000000B		
FKBSB_TYPE	= 0000000A		
FKB\$C_LENGTH	= 00000018		
FKBSW_SIZE	= 00000008		
ILLEFT	0000004B	R	02
IPL\$_QUEUEAST	= 00000006		
IPL\$_SYNCH	= 00000008		
IRP\$C_LENGTH	= 000000C4		
LOCAL	0000002F	R	02
MASREQUEST	*****	X	02
NONEXPR	00000052	R	02
PCBSB_WEFC	= 0000002E		
PCBSL_EFCS	= 00000050		
PCBSL_EFWM	= 0000004C		
PCBSL_PID	= 00000060		
PCBSL_STS	= 00000024		
PCBSV_WALL	= 0000000D		
PR\$ IPL	= 00000012		
PRQ\$C_EXEC	= 00000000		
PRQ\$C_SETEF	= 00000000		
PRQ\$C_PARAM	= 00000024		
PRQ\$W_DISPATCH	= 0000001C		
PRQ\$W_FR_PORT	= 0000001A		
PRQ\$W_REQTYPE	= 00000020		
PRQ\$W_TO_PORT	= 00000018		
RETURN	0000016A	R	02
SCH\$GL_PCBVEC	*****	X	02
SCH\$POSTEF	00000000	RG	02
SCH\$RSE	*****	X	02
SHBSB_PORT	= 00000015		
SHBSL_ADP	= 0000001C		
SS\$_ILLEFT	= 000000EC		
SS\$_NONEXPR	= 000008E8		
SS\$_UNASEFC	= 00000234		
SS\$_WASCLR	= 00000001		
SS\$_WASSET	= 00000009		

UNASEFC

00000059 R 02

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
AEXENONPAGED	0000016D (365.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:01.22
Command processing	105	00:00:00.55	00:00:04.96
Pass 1	308	00:00:09.76	00:00:33.82
Symbol table sort	0	00:00:01.53	00:00:04.07
Pass 2	85	00:00:01.95	00:00:05.69
Symbol table output	8	00:00:00.08	00:00:00.28
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	539	00:00:13.97	00:00:50.06

The working set limit was 1200 pages.
55437 bytes (109 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1035 non-local and 17 local symbols.
399 source lines were read in Pass 1, producing 13 object records in Pass 2.
22 pages of virtual memory were used to define 21 macros.

! Macro library statistics !

Macro library name	Macros defined
\$_255\$DUA28:[SYS.OBJ]LIB.MLB;1	12
\$_255\$DUA28:[SYSLIB]STARLET.MLB;2	6
TOTALS (all libraries)	18

1150 GETS were required to define 18 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:POSTEF/OBJ=OBJ\$:POSTEF MSRC\$:POSTEF/UPDATE=(ENH\$:POSTEF)+EXECML\$/LIB

0379 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY