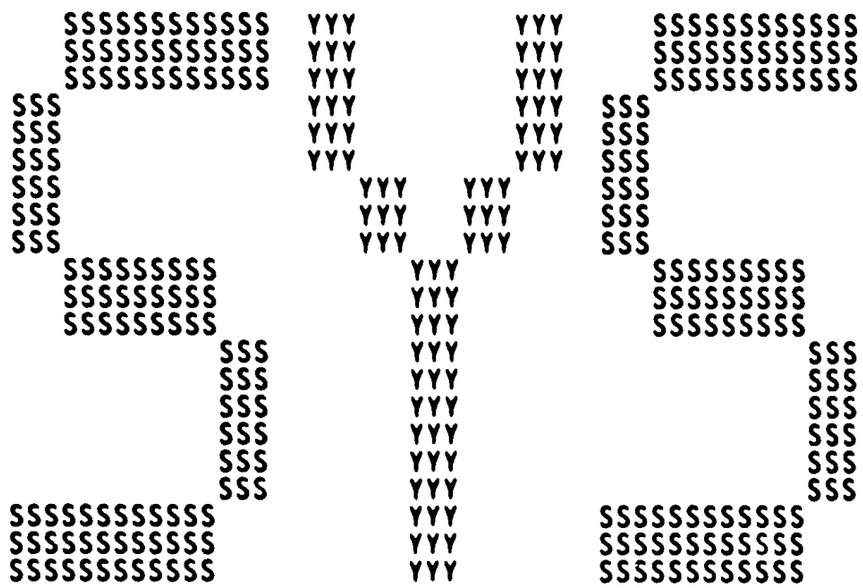

FILEID**FORKCNTRL


```
FFFFFFFFFF    000000    RRRRRRRR    KK      KK    CCCCCCCC  NN      NN  TTTTTTTTTT  RRRRRRRR    LL
FFFFFFFFFF    000000    RRRRRRRR    KK      KK    CCCCCCCC  NN      NN  TTTTTTTTTT  RRRRRRRR    LL
FF          00      00  RR      RR  KK      KK  CC          NN      NN      TT      RR      RR  LL
FF          00      00  RR      RR  KK      KK  CC          NN      NN      TT      RR      RR  LL
FF          00      00  RR      RR  KK    KK    CC          NNNN    NN      TT      RR      RR  LL
FF          00      00  RR      RR  KK    KK    CC          NNNN    NN      TT      RR      RR  LL
FFFFFFFF    00      00  RRRRRRRR    KKKKKK      CC          NN  NN  NN      TT      RRRRRRRR    LL
FFFFFFFF    00      00  RRRRRRRR    KKKKKK      CC          NN  NN  NN      TT      RRRRRRRR    LL
FF          00      00  RR  RR      KK    KK    CC          NN    NNNN      TT      RR  RR      LL
FF          00      00  RR  RR      KK    KK    CC          NN    NNNN      TT      RR  RR      LL
FF          00      00  RR    RR    KK      KK  CC          NN      NN      TT      RR    RR    LL
FF          00      00  RR    RR    KK      KK  CC          NN      NN      TT      RR    RR    LL          ....
FF            000000    RR      RR  KK      KK    CCCCCCCC  NN      NN      TT      RR      RR  LLLLLLLLLL  ....
FF            000000    RR      RR  KK      KK    CCCCCCCC  NN      NN      TT      RR      RR  LLLLLLLLLL  ....
                                                                                                           ....

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
```

Table of contents

(1) 48 CREATE FORK-AND-WAIT PROCESS
(1) 91 CREATE I/O DRIVER FORK PROCESS
(2) 114 CREATE FORK PROCESS
(3) 142 SOFTWARE INTERRUPT ENTRY POINTS
(4) 190 SOFTWARE INTERRUPT FORK DISPATCHER

GLOB
Symb

PCB$

```
0000        .TITLE  FORKCNTRL - FORK PROCESS CONTROL
0000        .IDENT  'V04-000'
0000
0000  ;
0000  ;****************************************************************************
0000  ;*                                                                          *
0000  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000  ;*  ALL RIGHTS RESERVED.                                                    *
0000  ;*                                                                          *
0000  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000  ;*  TRANSFERRED.                                                            *
0000  ;*                                                                          *
0000  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000  ;*  CORPORATION.                                                            *
0000  ;*                                                                          *
0000  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000  ;*                                                                          *
0000  ;*                                                                          *
0000  ;****************************************************************************
0000  ;
0000  ; D. N. CUTLER 9-AUG-76
0000  ;
0000  ; MODIFIED BY:
0000  ;
0000  ;       V03-002 DWT0158         David W. Thiel          27-DEC-1983
0000  ;               Speed up fork processing a bit by doing fewer interrupts
0000  ;               and REMQUE's.
0000  ;
0000  ;       V03-001 ROW0176         Ralph O. Weber          4-APR-1983
0000  ;               Add EXE$FORK_WAIT, the entry fork-and-wait routine which is
0000  ;               the object of the FORK_WAIT macro.
0000  ;
0000  ;
0000  ; MACRO LIBRARY CALLS
0000  ;
0000
0000        $FKBDEF                         ;DEFINE FKB OFFSETS
0000        $PRDEF                          ;DEFINE PROCESSOR REGISTERS
0000        $UCBDEF                         ;DEFINE UCB OFFSETS
```

```
                     0000     48          .SBTTL  CREATE FORK-AND-WAIT PROCESS
                     0000     49 ;+
                     0000     50 ; EXE$FORK_WAIT - CREATE FORK-AND-WAIT PROCESS
                     0000     51 ;
                     0000     52 ; FUNCTIONAL DESCRIPTION:
                     0000     53 ;
                     0000     54 ;       This routine is called to fold a fork context into a fork block and
                     0000     55 ;       place that block on the fork-and-wait work queue, EXE$GL_FKWAITFL.
                     0000     56 ;       From zero to one seconds later, EXE$TIMEOUT will remove all entries in
                     0000     57 ;       the fork-and-wait work queue and restart the fork thread saved here.
                     0000     58 ;
                     0000     59 ; INPUTS:
                     0000     60 ;
                     0000     61 ;       00(SP)  = Return address of the caller
                     0000     62 ;       04(SP)  = Return address of the caller's caller
                     0000     63 ;
                     0000     64 ;       R5      = Address of a fork block
                     0000     65 ;       R3 & R4 = Saved fork context registers
                     0000     66 ;
                     0000     67 ;       IPL >= IPL$_SYNCH
                     0000     68 ;
                     0000     69 ;
                     0000     70 ; OUTPUTS (to caller's caller):
                     0000     71 ;
                     0000     72 ;       All registers and IPL are returned as they were when this routine was
                     0000     73 ;       called.  Depending upon actions taken by the caller this may be
                     0000     74 ;       equivalent to destroying R0 through R5 and altering IPL.
                     0000     75 ;
                     0000     76 ; OUTPUTS (to caller after fork-and-wait completed):
                     0000     77 ;
                     0000     78 ;       R3 through R5 preserved.
                     0000     79 ;       All other registers destroyed.
                     0000     80 ;       IPL = FKB$B_FIPL(R5)
                     0000     81 ;-
                     0000     82
                     0000     83 EXE$FORK_WAIT::
                     0000     84
                     0000     85          ASSUME  FKB$L_FR4 EQ <FKB$L_FR3 + 4>
   10 A5    53    7D  0000     86          MOVQ    R3, FKB$L_FR3(R5)               ; Save fork context registers.
      0C A5 8ED0      0004     87          POPL    FKB$L_FPC(R5)                   ; Save fork context PC.
0000'DF    65    0E  0008     88          INSQUE  (R5), @W^EXE$GL_FKWAITBL        ; Put fork block on work queue.
              05    000D     89          RSB                                     ; Return to caller's caller.
```

```
                     000E     91          .SBTTL  CREATE I/O DRIVER FORK PROCESS
                     000E     92 ;+
                     000E     93 ; EXE$IOFORK - CREATE I/O DRIVER FORK PROCESS
                     000E     94 ;
                     000E     95 ; THIS ROUTINE IS CALLED BY AN I/O DRIVER TO CREATE A FORK PROCESS.
                     000E     96 ;
                     000E     97 ; INPUTS:
                     000E     98 ;
                     000E     99 ;       00(SP) = RETURN ADDRESS OF CALLER.
                     000E    100 ;       04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                     000E    101 ;
                     000E    102 ;       R5 = UCB ADDRESS OF DEVICE UNIT.
                     000E    103 ;
                     000E    104 ; OUTPUTS:
                     000E    105 ;
                     000E    106 ;       ***TBS***
                     000E    107 ;-
                     000E    108
                00000000    109          .PSECT  A$EXENONPAGED,LONG
                    0000    110 EXE$IOFORK::                                ;CREATE I/O DRIVER FORK PROCESS
          64 A5   01   AA   0000    111          BICW    #UCB$M_TIM,UCB$W_STS(R5);DISABLE TIMEOUT AND FALL THROUGH
                    0004    112                                              ;TO EXE$FORK
```

```
                                0004   114          .SBTTL  CREATE FORK PROCESS
                                0004   115 ;+
                                0004   116 ; EXE$FORK - CREATE FORK PROCESS
                                0004   117 ;
                                0004   118 ; THIS ROUTINE IS CALLED TO CREATE A FORK PROCESS.
                                0004   119 ;
                                0004   120 ; INPUTS:
                                0004   121 ;
                                0004   122 ;       00(SP) = RETURN ADDRESS OF CALLER.
                                0004   123 ;       04(SP) = RETURN ADDRESS OF CALLER'S CALLER.
                                0004   124 ;
                                0004   125 ;       R5 = ADDRESS OF FORK BLOCK.
                                0004   126 ;
                                0004   127 ; OUTPUTS:
                                0004   128 ;
                                0004   129 ;       ***TBS***
                                0004   130 ;-
                                0004   131
                                0004   132 EXE$FORK::                                ;CREATE FORK PROCESS
          10 A5    53    7D     0004   133          MOVQ    R3,FKB$L_FR3(R5)        ;SAVE REGISTERS R3 AND R4
               0C A5  8ED0     0008   134          POPL    FKB$L_FPC(R5)           ;SET FORK PROCESS PC
          54    0B A5    9A     000C   135          MOVZBL  FKB$B_FIPL(R5),R4       ;GET FORK IPL
     53    FFD0'CF44    7E     0010   136          MOVAQ   W^SWI$GL_FQFL-<6*8>[R4],R3 ;GET ADDRESS OF FORK QUEUE LISTHEAD
          04 B3    65    0E     0016   137          INSQUE  (R5),@4(R3)              ;INSERT FORK BLOCK IN FORK QUEUE
                   03    12     001A   138          BNEQ    10$                     ;IF QUEUE ALREADY POPULATED AVOID EXTRA INTE
                                001C   139          SOFTINT R4                      ;INITIATE SOFTWARE INTERRUPT
                         05     001F   140 10$:     RSB                             ; AND RETURN
```

```
                                                H  2

                     0020  142          .SBTTL  SOFTWARE INTERRUPT ENTRY POINTS
                     0020  143 ;+
                     0020  144 ; THE APPROPRIATE ENTRY POINT IS AUTOMATICALLY VECTORED TO WHEN THE SOFTWARE
                     0020  145 ; INTERRUPT PRIORITY ARBITRATION LOGIC IN THE CENTRAL PROCESSOR DETECTS A
                     0020  146 ; PENDING INTERRUPT AT LEVEL 6, 8, 9, 10, OR 11 AND THE CURRENT PRIORITY LEVEL
                     0020  147 ; IS LOWER THAN THE PENDING LEVEL. THE STATE OF THE STACK ON ENTRY IS:
                     0020  148 ;
                     0020  149 ;          00(SP) = INTERRUPT PC.
                     0020  150 ;          04(SP) = INTERRUPT PSL.
                     0020  151 ;
                     0020  152 ; THERE IS AN ENTRY POINT FOR EACH FORK LEVEL SO THAT THE HARDWARE IPL REGISTER
                     0020  153 ; WILL NOT HAVE TO BE READ TO DETERMINE THE CURRENT PROCESSOR IPL.  READING THE
                     0020  154 ; IPL REGISTER IS SLOW ENOUGH TO WARRANT THE ADDITIONAL CODE.
                     0020  155 ;-
                     0020  156
                     0020  157          .ALIGN  LONG                        ;ENTRY POINT MUST BE LONGWORD ALIGNED
                     0020  158 EXE$FRKIPL6DSP::                            ;FORK IPL 6 ENTRY POINT
           56    DD  0020  159          PUSHL   R6                          ;SAVE R6.
 56  0000'CF    7E  0022  160          MOVAQ   W^SWI$GL_FQFL,R6            ;GET ADDRESS OF FORK QUEUE LISTHEAD
           2F    11  0027  161          BRB     EXE$FORKDSPTH               ;BRANCH TO COMMON CODE
                     0029  162
                     0029  163          .ALIGN  LONG                        ;ENTRY POINT MUST BE LONGWORD ALIGNED
                     002C  164 EXE$FRKIPL9DSP::                            ;FORK IPL 9 ENTRY POINT
           56    DD  002C  165          PUSHL   R6                          ;SAVE R6.
 56  0018'CF    7E  002E  166          MOVAQ   W^SWI$GL_FQFL+24,R6         ;GET ADDRESS OF FORK QUEUE LISTHEAD
           23    11  0033  167          BRB     EXE$FORKDSPTH               ;BRANCH TO COMMON CODE
                     0035  168
                     0035  169          .ALIGN  LONG                        ;ENTRY POINT MUST BE LONGWORD ALIGNED
                     0038  170 EXE$FRKIPL10DSP::                           ;FORK IPL 10 ENTRY POINT
           56    DD  0038  171          PUSHL   R6                          ;SAVE R6.
 56  0020'CF    7E  003A  172          MOVAQ   W^SWI$GL_FQFL+32,R6         ;GET ADDRESS OF FORK QUEUE LISTHEAD
           17    11  003F  173          BRB     EXE$FORKDSPTH               ;BRANCH TO COMMON CODE
                     0041  174
                     0041  175          .ALIGN  LONG                        ;ENTRY POINT MUST BE LONGWORD ALIGNED
                     0044  176 EXE$FRKIPL11DSP::                           ;FORK IPL 11 ENTRY POINT
           56    DD  0044  177          PUSHL   R6                          ;SAVE R6.
 56  0028'CF    7E  0046  178          MOVAQ   W^SWI$GL_FQFL+40,R6         ;GET ADDRESS OF FORK QUEUE LISTHEAD
           0B    11  004B  179          BRB     EXE$FORKDSPTH               ;BRANCH TO COMMON CODE
                     004D  180
                     004D  181          .ALIGN  LONG                        ;ENTRY POINT MUST BE LONGWORD ALIGNED
                     0050  182 EXE$FRKIPL8DSP::                            ;FORK IPL 8 ENTRY POINT
           56    DD  0050  183          PUSHL   R6                          ;SAVE R6.
 56  0010'CF    7E  0052  184          MOVAQ   W^SWI$GL_FQFL+16,R6         ;GET ADDRESS OF FORK QUEUE LISTHEAD
           01    0057  185          NOP                                 ;PAD OUT TO LONGWORD BOUNDARY
                     0058  186          ;
                     0058  187          ; DROP THROUGH TO COMMON CODE
                     0058  188          ;
```

```
                     0058   190          .SBTTL  SOFTWARE INTERRUPT FORK DISPATCHER
                     0058   191 ;+
                     0058   192 ; EXE$FORKDSPTH - SOFTWARE INTERRUPT FORK DISPATCHER
                     0058   193 ;
                     0058   194 ; FOR EACH OF THE LEVELS 6, 8, 9, 10, AND 11, THERE EXISTS A QUEUE OF FORK
                     0058   195 ; BLOCKS WAITING TO BE PROCESSED. WHEN A FORK BLOCK IS ENTERED IN ITS
                     0058   196 ; CORRESPONDING QUEUE AND IT IS THE FIRST TO BE ENTERED (I. E. THE QUEUE
                     0058   197 ; WAS PREVIOUSLY EMPTY), A SOFTWARE INTERRUPT IS REQUESTED FOR THAT LEVEL.
                     0058   198 ; THE FORK DISPATCHER GAINS CONTROL AND EMPTIES THE QUEUE ONE ENTRY AT A
                     0058   199 ; TIME. AS EACH FORK IS DISPATCHED, REGISTERS R3 AND R4 ARE RESTORED FROM
                     0058   200 ; THE FORK BLOCK AND THE FORK PROCESS IS CALLED VIA A JSB INSTRUCTION.
                     0058   201 ; ON RETURN, THE FORK DISPATCHER RETRIEVES THE NEXT ENTRY FROM THE QUEUE
                     0058   202 ; AND REPEATS THE DISPATCHING OPERATION. THIS PROCESS CONTINUES UNTIL
                     0058   203 ; THERE ARE NO MORE FORKS TO DISPATCH AT WHICH TIME THE INTERRUPT IS
                     0058   204 ; DISMISSED.
                     0058   205 ;
                     0058   206 ;
                     0058   207 ;-
                     0058   208          .ALIGN  LONG
                     0058   209 EXE$FORKDSPTH::                          ;SOFTWARE INTERRUPT FORK DISPATCHER
55   DD              0058   210          PUSHL   R5                      ;SAVE R5 .
54   DD              005A   211          PUSHL   R4                      ;SAVE R4  .
53   DD              005C   212          PUSHL   R3                      ;SAVE R3   . PUSHLS ARE FASTEST!
52   DD              005E   213          PUSHL   R2                      ;SAVE R2  .
51   DD              0060   214          PUSHL   R1                      ;SAVE R1 .
50   DD              0062   215          PUSHL   R0                      ;SAVE R0.
07   11              0064   216          BRB     20$                     ;BRANCH TO BODY OF DISPATCHER
                     0066   217 ;
                     0066   218 ; DISPATCH FORK PROCESS WHEN QUEUE IS NOT YET EMPTY
                     0066   219 ; DISPATCH FORK PROCESS WITH:
                     0066   220 ;
                     0066   221 ;       R0 THRU R2 = SCRATCH REGISTERS.
                     0066   222 ;       R3 AND R4 = RESTORED FROM FORK BLOCK.
                     0066   223 ;       R5 = ADDRESS OF FORK BLOCK.
                     0066   224 ;
53   10 A5   7D      0066   225 10$:     MOVQ    FKB$L_FR3(R5),R3        ;RESTORE REGISTERS R3 AND R4
     0C B5   16      006A   226          JSB     @FKB$L_FPC(R5)          ;DISPATCH FORK
55   00 B6   0F      006D   227 20$:     REMQUE  @(R6),R5                ;REMOVE NEXT ENTRY FROM FORK QUEUE
        F3   12      0071   228          BNEQ    10$                     ;BRANCH IF QUEUE NOT YET EMPTY
        07   1D      0073   229          BVS     30$                     ;IF VS NO ENTRY REMOVED
                     0075   230                                          ;HERE WHEN LAST ENTRY DEQUEUED
                     0075   231 ;
                     0075   232 ; DISPATCH LAST ENTRY IN THE QUEUE
                     0075   233 ; DISPATCH FORK PROCESS WITH:
                     0075   234 ;
                     0075   235 ;       R0 THRU R2 = SCRATCH REGISTERS.
                     0075   236 ;       R3 AND R4 = RESTORED FROM FORK BLOCK.
                     0075   237 ;       R5 = ADDRESS OF FORK BLOCK.
                     0075   238 ;
53   10 A5   7D      0075   239          MOVQ    FKB$L_FR3(R5),R3        ;RESTORE REGISTERS R3 AND R4
     0C B5   16      0079   240          JSB     @FKB$L_FPC(R5)          ;DISPATCH FORK
   007F 8F   BA      007C   241 30$:     POPR    #^M<R0,R1,R2,R3,R4,R5,R6> ;RESTORE FORK PROCESS REGISTER SET
                02   0080   242          REI                             ;DISMISS INTERRUPT
                     0081   243
                     0081   244          .END
```

```
EXE$FORK                        00000004 RG    03
EXE$FORKDSPTH                   00000058 RG    03
EXE$FORK_WAIT                   00000000 RG    01
EXE$FRKIPL10DSP                 00000038 RG    03
EXE$FRKIPL11DSP                 00000044 RG    03
EXE$FRKIPL6DSP                  00000020 RG    03
EXE$FRKIPL8DSP                  00000050 RG    03
EXE$FRKIPL9DSP                  0000002C RG    03
EXE$GL_FKWAITBL                 ********   X   01
EXE$IOFORK                      00000000 RG    03
FKB$B_FIPL                =     0000000B
FKB$L_FPC                 =     0000000C
FKB$L_FR3                 =     00000010
FKB$L_FR4                 =     00000014
PR$_SIRR                  =     00000014
SWI$GL_FQFL                     ********   X   03
UCB$M_TIM                 =     00000001
UCB$W_STS                 =     00000064
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	0000000E (14.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
A$EXENONPAGED	00000081 (129.)	03 (3.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.03	00:00:02.11
Command processing	105	00:00:00.50	00:00:04.83
Pass 1	204	00:00:04.28	00:00:13.35
Symbol table sort	0	00:00:00.69	00:00:01.50
Pass 2	60	00:00:00.91	00:00:03.11
Symbol table output	4	00:00:00.04	00:00:00.04
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	405	00:00:06.48	00:00:24.97

The working set limit was 1200 pages.
23495 bytes (46 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 445 non-local and 4 local symbols.
244 source lines were read in Pass 1, producing 14 object records in Pass 2.
12 pages of virtual memory were used to define 11 macros.

```
                                  +--------------------------+
                                  ! Macro library statistics !
                                  +--------------------------+

Macro library name                 Macros defined
------------------                 --------------
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                3
_$255$DUA28:[SYSLIB]STARLET.MLB;2             5
TOTALS (all libraries)                        8

510 GETS were required to define 8 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:FORKCNTRL/OBJ=OBJ$:FORKCNTRL MSRC$:FORKCNTRL/UPDATE=(ENH$:FORKCNTRL)+EXECML$/LIB
```

FORKCNTRL
LIS

FILINIWCB
LIS

IMGDECODE
LIS

INIT
LIS

IOLOCK
LIS

IODEF
LIS

IOCIOPOST
LIS

GLOBALS
LIS

IMGMAPISD
LIS