DECUS
PROGRAM LIBRARY

DECUS NO.	8-921
TITLE	OS/8 VAX Handler
AUTHOR	Jim van Zee
COMPANY	Laboratory Data Systems, Seattle, Washington
DATE	B, August 1980
SOURCE LANGUAGE	PAL8

new
8-921

OS/8 VAX Handler

Version: B, August 1980

Author: Jim van Zee, Laboratory Data Systems, Seattle, Washington

Operating System: OS/8, OS/78, DS/8, PS/8, OS/12

Source Language: PAL8

Special Hardware Required: Serial line communications interface.

The VAX handler allows any PDP-8 or PDP-12 to send files to (or receive files from) a VAX computer. No special programs are required to do this: the handler may be used with any OS/8 program and presumes only the existence of the $COPY command on the VAX. Only ASCII data may be transferred, but commands as well as data may be sent to the VAX, hence an OS/8 program can use this handler to do 'parallel processing' on data collected by the '8 or '12.

Restrictions: OS/8 FORTRAN-IV programs using this handler for input can only put output files on the system device.

SLV 42,43

The VAX Handler

The VAX handler allows OS/8 programs to transmit data back and forth to the VAX computer using only the $COPY command and the VAX terminal (TT) handler. Linefeeds are suppressed during output to avoid double-spacing (the VAX doesn't ignore linefeed on input!), and the CTRL/Z at the end of the OS/8 file is sent as the last character, thereby terminating the $COPY command automatically. (This limits file transfers to one-at-a-time (you can use PIP to merge several files) but it is a 'cleaner' mode of operation than requiring the user to manually enter the 'EOF' terminator after the copy operation.) To copy files from OS/8 to the VAX, do the following:

1) Login on the VAX
2) Issue a '$ COPY TT: filespec' command
3) Return to OS/8
4) Issue a '.COPY VAX:<filename' command

There are other possibilities, of course: any OS/8 program can be used to send data to the VAX: handler after steps 1 and 2 have been completed, the CUSP used by the COPY command (FOTP) being but one example. Other examples are: PIP (the PUNCH command invokes PIP), DIRECT, CREF, SRCCOM or any language (FOCAL, BASIC, FORTRAN, etc.). Just remember that the first CTRL/Z encountered will terminate the VAX $COPY command, effectively 'disabling' the handler. Subsequent data is then interpreted as VAX DCL commands with unpredictable results!

Sending data in the other direction (from the VAX to OS/8) is just a little bit trickier since OS/8 does not do 'overlapped I/O' and so must stop the VAX whenever the input buffer fills up. It does this by sending a 'XOFF' (CTRL/S), but since the VAX does not respond immediately, it saves up to 8 'extra' characters in a buffer within the handler. The need to keep an impure copy of the handler in memory for the duration of the input transfer means that FORTRAN IV programs can only copy files from the VAX to the system device, since otherwise the VAX: handler will be swapped out occasionally, causing the loss of the characters in the buffer. There are no restrictions on the use of this handler for output.

The VAX command sequence for sending data to the '8 is essentially just the opposite of that used above: '$COPY filespec TT:'. However it is quite obviously necessary to delay execution of this command until the OS/8 handler is ready to receive the data. This is done by simply not terminating the command (i.e. you don't type 'RETURN'!) - allowing the handler itself to send the terminator when it is finally ready for the data. This sequence causes the first characters to be the 'CR/LF' which is echoed in response to the terminator; hence the handler has been constructed to ignore the very first line of input.

The second problem encountered when sending data from the VAX to OS/8 is how to determine when the transmission is over. The VAX does not send any sort of EOF character (although it expects one itself!), so the only general way is to have a timeout loop: if the VAX: handler has to wait for more than approximately 0.3 seconds for a character it assumes the transmission is over and takes the 'EOF' return, which typically instructs the calling program to close the output file. (This delay is adjustable via a 'SET VAX LOC 300' command: change '-1' to '-N') A result of this method of EOF detection is that the file almost always ends up with the DCL prompt which is sent just as soon as the COPY command is finished. This, however, can be easily avoided by creating a small DCL command file which follows the $COPY command with a 2 second $WAIT. This ensures that the handler will time-out after the last data character and before the prompt. The following command file (called 'SENDS.COM') is suggested for this purpose:

```
$ COPY 'P1 TT:
$ WAIT 0::2
```

This set of commands can then be invcked typing '@SEND8 filespec', or even more conveniently, by a 'SEND8 filespec' command if you define the symbol 'SEND8' to be equal to '@SEND8' (just say 'SEND8:=@SEND8'). Just remember not to type the 'RETURN' key at the end of the command - otherwise the data will be sent before the handler is ready! To transmit files from the VAX to any OS/8 program, just do the following:

1) Login on the VAX
2) Type 'SEND8 filespec' (Note: don't type 'RETURN'!)
3) Return to OS/8
4) Type 'PUNCH filename<VAX:' (Note: you can't use 'COPY'!)

Again, the last step can be replaced by any sort of call to the VAX handler for input, for example, the SRCCOM program can be used to compare an OS/8 file with one stored on the VAX system. For the most part the handler overcomes the differences in the way files are structured on the two systems, but there is -one- anomoly in the VAX (V1.6) terminal handler which cannot be eliminated: it will not send back a 'formfeed', but instead simulates it with an appropriate number of linefeeds, based upon the setting of the 'PAGE' parameter for your terminal. Setting 'PAGE=0' eliminates the spurious LF's, but does not eliminate the need to edit the OS/8 file later on to put in the missing page markers. Fortunately V2.0 has a new 'SET TERM/MECHFORM' option which should eliminate this problem.

Other parameters which should be set are: TAB, HOSTSYNC, TTSYNC and WIDTH. The latter is especially important when sending files TO the VAX, since (quite surprisingly!) the terminal handler forces all -input- lines to break at the WIDTH specification. Thus if you are using the VAX: handler to create a listing file for the printer you will want to be sure to set WIDTH=140 or so - otherwise the file will be generated with lines no longer than 80 characters!

It is, of course, possible to send commands as well as data to the VAX, so with the proper sort of programming one can implement a 'dual processing' system. A feature of the handler which might be useful in this regard is that it does not transmit nulls (nulls are interpreted by the VAX as command terminators!). Thus interactive FOCAL programs, for example, can use the 'Output Buffer' command to send short commands to the VAX without having to constantly reload the handler.

To assemble the handler for use on your system, you must first know what device codes are used by the interface: the default values are '40' for 'receive', and '41' for 'transmit', which are the values used by the PDP-12 'dataphone' interface. These values should be edited into the source file, or put in a separate source file ahead of the file VAX.PA. Only 'INDVC' needs to be defined if the values are consecutive, otherwise define 'OUTDVC' as well. Once the handler is installed in the system there is no (easy) way to change the device codes since the 'SET DVC' command only works for handlers with a single DVC, while the 'SET CODE' command (which changes pairs of DVCs) only works for DEC's SLU handlers.

Having created the appropriate binary file, you can either use BUILD to install it, or if you have Version 3 of the SET program, you can .LOAD the binary, save it as a .HN ('handler') file, and then use the 'SET HANDLER' command to swap it with another handler which is already installed in your system. Do not expect the '/L' option in SET to list the VAX handler, however, since it has only limited intelligence. Similarly, RESORC thinks the name is 'NA' instead of 'VAX', 'device type' being '41'. The easiest way to fix this is to use FUTIL to find the sixbit string 'CDP' (card punch) in RESORC.SV, and replace it with 'VAX'.

```
                        /25 COMMUNICATIONS HANDLER FOR THE VAX          -JVZ-

                        / THIS HANDLER ALLOWS A PDP8/12 TO COMMUNICATE WITH THE
                        /VAX TERMINAL HANDLER, USING THE XON/XOFF PROTOCOL (TT-
                        /SYNC AND HOSTSYNC) TO CONTROL TRANSMISSION.  LINEFEEDS
                        /ARE SUPPRESSED DURING OUTPUT TO AVOID 'DOUBLE-SPACING'
                        /SINCE THE VAX DOES NOT HANDLE CR/LF PROPERLY!  UP TO 8
                        /CHARACTERS ARE SAVED BETWEEN CALLS,  SO PROGRAMS WHICH
                        /SWAP HANDLERS (I.E. FORTRAN IV), COULD LOSE CHARACTERS
                        /ONCE IN A WHILE DURING INPUT.  CTRL/Z -IS SENT- DURING
                        /OUTPUT (WHICH CLOSES THE VAX FILE), & INPUT IS STARTED
                        /BY SENDING A 'NULL' TO TERMINATE THE DCL COMMAND LINE.

                        /THE 'END-OF-FILE' IS DETERMINED BY A TIME-OUT LOOP, SO
                        /TO AVOID GETTING THE COMMAND PROMPT IN THE FILE, IT IS
                        /CONVENIENT TO HAVE A 'WAIT' COMMAND FOLLOWING THE COPY
                        /WHICH WILL CAUSE THE HANDLER TO TIME-OUT AFTER RECEIPT
                        /OF JUST THE DATA FILE.    THIS SET OF COMMANDS CAN THEN
                        /BE EQUATED TO THE SYMBOL 'SEND8' FOR CONVENIENT USAGE:

                        /                $ COPY 'P1 TT:
                        /                $ WAIT 0::2

                        /REMEMBER NOT TO TERMINATE THE 'SEND8' COMMAND (I.E. DO
                        /NOT TYPE 'RETURN'!).  THE HANDLER WILL SEND THE TERMI-
                        /NATOR WHEN IT IS READY FOR THE DATA.

                0002    VERSION="B&77

                        IFNDEF  INDVC <INDVC=40>/STANDARD SLU DEVICE CODE
                        IFNDEF  OUTDVC <OUTDVC=INDVC+1>

                6401    KKSF=   INDVC^10+IOT 1  /SKIP ON INPUT FLAG
                6402    KKCC=   INDVC^10+IOT 2  /CLEAR I.F.  AND A.C.
                6404    KKRS=   INDVC^10+IOT 4  /'OR' INPUT DATA WITH A.C.
                6406    KKRB=   KKCC KKRS       /COMBINATION OF '2' AND '4'

                6411    TTSF=   OUTDVC^10+IOT 1 /SKIP ON DONE FLAG
                6412    TTCF=   OUTDVC^10+IOT 2 /CLEAR D.F.
                6416    TTLS=   OUTDVC^10+IOT 6 /XMIT CHAR, LEAVE IN THE A.C.

                0000            PAGE 0
00000   7777            -1              /ONE ENTRY POINT
00001   2601    DEVICE  VAX
00002   3000
00003   2601    DEVICE  VAX
00004   3000
00005   0410            410             /DCB WORD (R/W/NONFILE, DEV 41)
00006   4000            VAX&177+4000    /ENTRY POINT (PLUS 2-PAGE FLAG)
00007   0000    ZBLOCK  2

                0200            PAGE 1
```

```
 53  00200  0002  VAX,    VERSION          /START AT THE VERY TOP
 54  00201  7326          CLA STL RTL      /2=CDI-CDF
 55  00202  6214          RDF
 56  00203  1342          TAD CDF0
 57  00204  3276          DCA CDIX         /SAVE RETURN 'CDI'
 58  00205  7130          STL RAR          /4000
 59  00206  1600          TAD I VAX        /PUT R/W BIT IN THE LINK
 60  00207  0221  XON,    AND V70          /GET BUFFER FIELD
 61  00210  1342          TAD CDF0
 62  00211  3226          DCA TEMP         /PUT INLINE FOR ONCE-ONLY USE
 63  00212  1311          TAD V3700
 64  00213  0600          AND I VAX        /STRIP PAGE COUNT
 65  00214  2200          ISZ VAX          /BUMP TO BUFFER ADDRESS
 66  00215  7450          SNA
 67  00216  5273          JMP VXIT         /IGNORE INITIALIZATION CALLS
 68  00217  7041          CIA
 69  00220  3301          DCA DWC          /SAVE DOUBLE-WORD COUNT
 70
 71                 / WE FALL THROUGH THE CONSTANTS BELOW WITH AC=0
 72
 73                 IFNZRO  .&177-21       <'XON' & 'XOFF' WILL BE WRONG>
 74  00221  0070  V70,    70
 75  00222  0155  XSW,    -7623            /OR -7621
 76  00223  0177  V177,   177
 77
 78  00224  1600          TAD I VAX        /GET BUFFER ADDRESS
 79  00225  3300          DCA BAD
 80  00226  7402  TEMP,   HLT              /CDF TO B.F.
 81  00227  7420          SNL              /READ OR WRITE?
 82  00230  5374          JMP READ
 83
 84  00231  7006  WRIT,   RTL              /SHIFT GUARD BIT
 85  00232  7006          RTL
 86  00233  7510          SPA              /TEST IT
 87  00234  5247          JMP WN03         /TIME FOR THE THIRD BYTE
 88  00235  3226          DCA TEMP
 89  00236  1700          TAD I BAD        /GET FIRST OR SECOND BYTE
 90  00237  4316          JMS PUTC
 91  00240  1700          TAD I BAD        /EXTRACT PART OF THIRD
 92  00241  2300          ISZ BAD
 93  00242  7400  V7400,  7400             /NOP
 94  00243  0242          AND .-1
 95  00244  7104          CLL RAL          /PACK THE BITS TOGETHER
 96  00245  1226          TAD TEMP
 97  00246  5231          JMP WRIT         /SHIFT INTO PROPER POSITION
 98
 99  00247  4316  WN03,   JMS PUTC         /AC HAS SHIFT REGISTER
100  00250  2301          ISZ DWC          /DONE?
101  00251  5231          JMP WRIT         /NO
102  00252  5273          JMP VXIT         /YES
```

```
00253  0000  INP,    0                 /HOLDS ADDRESS OF INPUT ROUTINE
00254  4653          JMS I INP         /GET BYTE NO. 1
00255  3700          DCA I BAD
00256  4653          JMS I INP         /GET BYTE NO. 2
00257  3316          DCA PUTC
00260  4653          JMS I INP         /GET BYTE NO. 3
00261  4302          JMS SAVE          /STORE FIRST WORD
00262  1316          TAD PUTC
00263  3700          DCA I BAD
00264  1226          TAD TEMP
00265  4302          JMS SAVE          /STORE SECOND WORD
00266  2301          ISZ DWC
00267  5254          JMP INP+1         /DO THREE MORE
00270  1321          TAD XOFF
00271  4316          JMS PUTC
00272  4653          JMS I INP         /TEST IF WE SHOULD SEND XOFF

00273  2200  VXIT,   ISZ VAX           /MOVE PAST BUFFER ADDRESS
00274  2200          ISZ VAX           / "     "   BLOCK NUMBER
00275  2200          ISZ VAX           / "     "   ERROR RETURN
               /      TTCF              /BE NICE TO 'ION' PROGRAMS
00276  7402  CDIX,   HLT               /CDF CIF C.F.
00277  5600          JMP I VAX

               IFNZRO .&177-100        <'V3700' WON'T BE CORRECT>
00300  0000  BAD,    0                 /BUFFER ADDRESS
00301  0000  DWC,    0                 /BUFFER SIZE (%2)

00302  0000  SAVE,   0                 /STORE 1-1/2 BYTES IN 1 WORD
00303  7006          RTL
00304  7006          RTL               /SHIFT HALF-BYTE TO BITS 0-3
00305  3226          DCA TEMP
00306  1226          TAD TEMP
00307  0242          AND V7400
00310  1700          TAD I BAD         /MERGE WITH BITS 4-11
00311  3700  V3700,  DCA I BAD
00312  2300          ISZ BAD
00313  0175  M7603,  -7603             /CTRL/C TEST PROTECTS 'ISZ'
00314  7100          CLL
00315  5702          JMP I SAVE

00316  0000  PUTC,   0                 /OUTPUT ROUTINE
00317  3302          DCA SAVE          /SAVE CHARACTER
00320  1302          TAD SAVE
00321  0223  XOFF,   AND V177
00322  7440          SZA
00323  1366          TAD MLF           /CHECK FOR LF, NULLS
00324  7650          SNA CLA
00325  5364          JMP PXIT          /DON'T SEND EITHER ONE
```

```
152  00326  1302          TAD SAVE        /SEND IT AWAY
153  00327  6416          TTLS
154  00330  0223          AND V177
155  00331  1367          TAD M32         /CTRL/Z?
156  00332  7650          SNA CLA
157  00333  5273          JMP VXIT        /LOGICAL END OF BUFFER
158
159  00334  1344  EXON,   TAD V7600       /WATCH FOR ^C, ^X, ^Q
160  00335  6034          KRS             /'OR'
161  00336  1313          TAD M7603
162  00337  7650          SNA CLA         /CTRL/C?
163  00340  6031          KSF             /YES: FLAG UP?
164  00341  5344          JMP V7600       /NO
165  00342  6201  CDF0,   CDF             /BE SURE TO DO THIS!
166  00343  5744          JMP I V7600     /RECALL OS/8 K.B.M.
167
168  00344  7600  V7600,  7600
169  00345  1344          TAD V7600       /SAME IDEA
170  00346  6404          KKRS
171  00347  1222          TAD XSW         /EITHER '-XOFF' OR '-XON'
172  00350  7650          SNA CLA
173  00351  6401          KKSF
174  00352  5357          JMP PCHK
175
176  00353  1222          TAD XSW         /TOGGLE BETWEEN THE TWO
177  00354  7112          CLL RTR
178  00355  7026          CML RTL         /SORT OF CLEVER WAY TO DO IT
179  00356  3222          DCA XSW
180
181  00357  7126  PCHK,   STL RTL         /2
182  00360  0222          AND XSW         /CHECK FOR 'XOFF' CONDITION
183  00361  7650          SNA CLA
184  00362  6411          TTSF
185  00363  5334          JMP EXON        /VAX IS BUSY AT THE MOMENT
186  00364  7120  PXIT,   STL             /SET GUARD BIT
187  00365  5716          JMP I PUTC
188
189  00366  7766  MLF,    -12
190  00367  7746  M32,    -32
191
192         0374          *VAX!174        /PUT THIS AT THE VERY END
193  00374  1207  READ,   TAD XON
194  00375  4316          JMS PUTC        /START THE VAX TALKING TO US
195  00376  7130          STL RAR         /4000
196  00377  4253          JMS INP         /PUTS ADDRESS OF PAGE 2 IN INP
197
198         0400          PAGE
```

```
                        /THIS PAGE HAS THE INPUT ROUTINE WHICH HAS TO RECOGNIZE
                        /WHEN THE VAX HAS FINISHED SENDING DATA SO IT CAN PUT A
                        /CTRL/Z IN THE BUFFER, FILL IT OUT WITH NULLS, AND TAKE
                        /THE ERROR RETURN WITHOUT SENDING 'XOFF'.  THIS ROUTINE
                        /ALSO TURNS MULTIPLE LF'S INTO MULTIPLE CRLF'S AND BUF-
                        /FERS ANY EXTRA CHARACTERS WHICH ARE RECEIVED AFTER THE
                        /VAX HAS BEEN SENT AN 'XOFF'.

                        /THERE IS ONE PROBLEM: THE VAX TURNS FORMFEEDS INTO THE
                        /PROPER NUMBER OF LINEFEEDS (BASED ON THE PAGE SIZE) SO
                        /WE DO NOT GET BACK EXACTLY WHAT WE SENT.  (BEST TO USE
                        /A PAGE SIZE OF 0.)  NOTE: THIS HAS BEEN FIXED IN V2.0!
                        /JUST SET THE 'MECHFORM' OPTION FOR YOUR TERMINAL.

00400  0577  GETC,    (177            /GETS US V377
00401  7440           SZA
00402  5216           JMP NO1         /FIRST CALL
00403  7430           SZL
00404  5240           JMP EOB         /END OF BUFFER
00405  1347           TAD CHAR
00406  7710           SPA CLA         /READ FROM SILO?
00407  2210           ISZ .+1
00410  7402  HOLD,    HLT             /'TAD SILO'
00411  7450           SNA
00412  4301  BOB,     JMS FETCH
00413  0231           AND V377        /ONLY KEEP 8 BITS
00414  7100           CLL
00415  5600           JMP I GETC      /RETURN CHARACTER

00416  3347  NO1,     DCA CHAR        /4000
00417  1240           TAD EOB
00420  3210           DCA HOLD        /INITIALIZE SILO
00421  1301           TAD FETCH       /FIRST CALL AFTER EOF?
00422  7640           SZA CLA
00423  5207           JMP HOLD-1      /RETURN SAVED CHARACTERS
00424  3210           DCA HOLD
00425  6416           TTLS            /TERMINATE VAX COMMAND

00426  4264  GET2,    JMS CTRLC       /SKIP THE COMMAND LINE
00427  5226           JMP .-1
00430  6406           KKRB
00431  0377  V377,    AND (177        /FOUND THE LF YET?
00432  1376           TAD (-12
00433  7640           SZA CLA
00434  5226           JMP GET2
00435  4264           JMS CTRLC       /WAIT FOR FIRST CHARACTER
00436  5235           JMP .-1
00437  5212           JMP BOB         /IT TAKES A WHILE SOMETIMES...
```

```
247   00440  1347  EOB,    TAD CHAR        /REACHED EOF YET?
248   00441  7650          SNA CLA
249   00442  5257          JMP DONE        /YES
250   00443  1216          TAD NO1
251   00444  3252          DCA INST        /GET SILO POINTER
252
253   00445  4301  BLOOP,  JMS FETCH       /GET THE NEXT CHARACTER
254   00446  3210          DCA HOLD
255   00447  1210          TAD HOLD        /REACHED THE END YET?
256   00450  7440          SZA
257   00451  2252          ISZ .+1
258   00452  7402  INST,   HLT             /'DCA SILO'
259   00453  1210          TAD HOLD
260   00454  7640          SZA CLA         /DONE?
261   00455  5245          JMP BLOOP
262   00456  5600          JMP I GETC      /NO
263
264   00457  3301  DONE,   DCA FETCH       /SET EOF FLAG
265   00460  1375          TAD (21         /YES
266   00461  6416          TTLS            /SEND 'XON'
267   00462  2200          ISZ GETC        /TAKE THE ERROR EXIT
268   00463  5600          JMP I GETC
269
270   00464  0000  CTRLC,  0
271   00465  6401          KKSF
272   00466  7610          SKP CLA
273   00467  2264          ISZ CTRLC
274   00470  1374          TAD (7600
275   00471  6034          KRS             /CHECK KEYBOARD
276   00472  1373          TAD (-7603
277   00473  7650          SNA CLA
278   00474  6031          KSF
279   00475  5664          JMP I CTRLC
280   00476  6201          CDF
281   00477  5774          JMP I (7600
282
283                IFNZRO  .&177-100       <DIDN'T GET THIS AT '300'>
284   00500  7777  DELAY,  -1              /ADJUST THIS IF NECESSARY
```

```
00501  0000  FETCH,  0
00502  1347          TAD CHAR          /CHECK FOR EOF
00503  2346          ISZ FLAG          /TURN CRLF,LF INTO CRLF,CRLF
00504  7650          SNA CLA
00505  5701          JMP I FETCH
00506  3345          DCA TIMOUT        /INITIALIZE TIMEOUT LOOP
00507  1300          TAD DELAY
00510  3344          DCA TIMER

00511  4264  WAIT,   JMS CTRLC
00512  5335          JMP LOOP
00513  6406          KKRB              /GET CHARACTER
00514  3347          DCA CHAR
00515  1347          TAD CHAR
00516  0377          AND (177          /KEEP NULLS OUT OF BUFR
00517  7450          SNA
00520  5306          JMP WAIT-3
00521  1376          TAD (-12          /WATCH THOSE LINEFEEDS!
00522  7640          SZA CLA
00523  5333          JMP LOOP-2
00524  7160          STL CMA
00525  1346          TAD FLAG          /WAS LAST CHAR A LF TOO?
00526  7650          SNA CLA
00527  7160          STL CMA
00530  3346          DCA FLAG          /YES, REPLACE WITH CR/LF
00531  7430          SZL
00532  1372          TAD (3            /CR-LF
00533  1347          TAD CHAR
00534  5701          JMP I FETCH

00535  2344  LOOP,   ISZ TIMER         /ROUGHLY 100 MSEC
00536  5311          JMP WAIT
00537  2345          ISZ TIMOUT
00540  5307          JMP WAIT-2
00541  3347          DCA CHAR          /SET EOF FLAG
00542  1371          TAD (32
00543  5701          JMP I FETCH       /RETURN CTRL/Z

00544  0000  TIMER,  0
00545  0000  TIMOUT, 0

00546  0000  FLAG,   0                 /MULTIPLE-LF FLAG
00547  0000  CHAR,   0                 /JUST BEFORE SILO
00550  0000  SILO,   ZBLOCK 20         /SHOULD BE ENOUGH

00571  0032
00572  0003
00573  0175
00574  7600
00575  0021
00576  7766
00577  0177

                    $END$
```

```
BAD     0300        XON     0207
BLOOP   0445        XSW     0222
BOB     0412
CDFO    0342
CDIX    0276
CHAR    0547
CTRLC   0464
DELAY   0500
DONE    0457
DWC     0301
EOB     0440
EXON    0334
FETCH   0501
FLAG    0546
GETC    0400
GET2    0426
HOLD    0410
INDVC   0040
INP     0253
INST    0452
KKCC    6402
KKRB    6406
KKRS    6404
KKSF    6401
LOOP    0535
MLF     0366
M32     0367
M7603   0313
NO1     0416
OUTDVC  0041
PCHK    0357
PUTC    0316
PXIT    0364
READ    0374
SAVE    0302
SILO    0550
TEMP    0226
TIMER   0544
TIMOUT  0545
TTCF    6412
TTLS    6416
TTSF    6411
VAX     0200
VERSIO  0002
VXIT    0273
V177    0223
V3700   0311
V377    0431
V70     0221
V7400   0242
V7600   0344
WAIT    0511
WNO3    0247
WRIT    0231
XOFF    0321
```

V6B