Kermit in FORTH#

original C=64 Sourcecode

ftp://kermit.columbia.edu/kermit/c/

About the Kermit Protcol, see Kermit (protocol) in Wikipedia

KERMIT C-64, v1.5#

Robert W. Detenbeck, University of Vermont.

KERMIT is a protocol for transferring files between different computers. This KERMIT program was written in the FORTH language, for a Commodore C64 computer with a Commodore 1541 disk drive (device 8) and a Commodore 1600 modem on the RS-232 port (device 2). Other drives and modems may work if they use the same device numbers. The program itself uses about 30K; disk buffers for "save" and "help" commands extend downward from $9FFF; and there are scratch string areas in $CB00-$CFFF. This program has been tested only under limited conditions. It has been run at 300 baud, but the nature of the interrupt-driven RS-232 interface suggests that 1200 or even 2400 baud should work without loss of data, though perhaps not at the efficiency of a machine-language program. Multiple-file transfers with wild-card construction cause some problems in the transfer of filenames from Kermit-20.

The "help" files, named scr96, scr97 and scr98, give details of this program's special characteristics. Several special features were required by the C64's use of Commodore's own form of ASCII code, and its lack of certain ASCII capabilities. ASCII backslash prints as a vertical bar, chr$(221), but is transmitted by the "pound" key; the tab and formfeed characters are stored in C64 files as chr$(220) and chr$(219).

The KERMIT "connect" command converts the C64 into a simple terminal, in which the C64 "delete" button transmits the ASCII code 127, "DEL". If the host mainframe requires a backspace, ASCII code 8, the F5 key should be used. The terminal program contained herein responds correctly to a backspace character received from the modem.

You have received a "turnkey" copy of the FORTH program, in which the source screens and programming system are inaccessible. This copy may be recopied and distributed without restriction. In fact, the "new" command of this KERMIT facilitates such copying. If you want a copy of the original FORTH screens, with a printing program to device 4, you may obtain one by sending a blank disk in a stamped, self-addressed mailer to the author, Robert W. Detenbeck, Department of Physics, University of Vermont, Burlington, VT 05405. They were written to be used with version A of C64-FORTH, sold by Performance Micro Products, 770 Dedham Street-S2, Canton, MA 02021. Slight modifications would be required to use the screens with the newer version B, a pure FORTH-79 standard. Because the screens of C64-FORTH are 25 x 40, extensive retyping would be necessary to format them for readability on a standard 16 x 64 FORTH screen.

Information about KERMIT can be obtained from the following sources: Frank da Cruz and Bill Catchings, "Kermit: A File-Transfer Protocol for Universities" BYTE, vol. 9, no. 6, June 1984, p. 255 (Part 1) BYTE, vol. 9, no. 7, July 1984, p. 143 (Part 2) KERMIT USERS GUIDE, 4th ed., available for $5.00 from KERMIT Distribution, Columbia University Center for Computing Activities, 7th Floor, Watson Laboratory, 612 West 115th Street, New York, N.Y. 10025. For those near the University of Vermont, the USERS GUIDE can be purchased from the Academic Computing Center in the Cook Building.

The file C644TH.SCR is an ASCII, printable file made from the FORTH screens of C64-KERMIT, v1.5 , with carriage returns added to its 40-character lines for printing, and with a few inter-screen heading lines added for identification. These screens were written for the original (version A) Performance Micro Products C64-FORTH. They will need modification for use with other versions of FORTH, and some screens will be irrelevant to other systems, especially those dealing with I/O. This file is meant to be used by knowledgeable FORTH programmers who wish to adapt and improve this KERMIT program for their own private systems.

R. Detenbeck, Physics Dept., University of VT, Burlington, VT 05405

```
( ---
( --- SCREEN # 5 ---
( ---
( TYPE DEFINITIONS 070984)

: VAR VARIABLE ;    ( SHORTHAND)
: 0VAR 0 VARIABLE ; ( INITS VAR TO 0)
: CONST CONSTANT ;  ( SHORTHAND)

: ASC 32 WORD HERE 1+ C@ ;
  ( LEAVES ASCII EQUIVALENT OF FOLLOWING
    SINGLE CHARACTER ON STACK. USEAGE:
    ASC A CONST 'A' , STORES 65 IN 'A')
-->
```

```
( ---
( --- SCREEN # 6 ---
( ---
( STRING FUNCTIONS 071784)

: ADD$ ( A$ C$ --- ) ( C$=C$+A$ )
  SWAP COUNT 3 PICK COUNT DUP ROT + SWAP
  3 PICK + 5 PICK C! SWAP CMOVE DROP ;

: CONCAT$ ( A$ B$ C$ ---) ( C$=A$+B$ )
  0 OVER ! ( ZEROES C$ )
  SWAP ROT 3 PICK ADD$ SWAP ADD$ ;

: TYPE$ COUNT TYPE ; ( NAME$ ---)

: TO$ ( A$ B$ ---) ( COPIES A$ OVER B$)
  0 OVER ! ADD$ ; -->
```

```
( ---
( --- SCREEN # 7 ---
( ---
( MORE STRING FUNCTIONS 060584)

: WORD$   ( MAKES STRING CALLED 'ABCD')
  <BUILDS              ( WORD$ 'ABCD' )
  2 0 DO         ( BACKUP PTR TO 2D ' )
    BEGIN
      IN @ 1- DUP
      0< IF ." ERROR -- NO '" ENDIF
      DUP IN !
      BLK @ IF
        BLK @ BLOCK
      ELSE
        TIB @
      THEN
      + C@ 39 =
    UNTIL
  LOOP
  1 ALLOT 39 WORD HERE C@ HERE 1- C!
  HERE C@ 1+ ALLOT
  DOES> 1+ ;

: STR$ ( N --- ADDR)  ( ADDR OF STRING)
  0 <# #S 32 HOLD #>  ( FROM N IN PAD.)
  OVER 1- C! 1- ;     ( ADD COUNT BYTE)
-->


( ---
( --- SCREEN # 8 ---
( ---
( MORE STRING FUNCTIONS 070484)

3 CONSTANT SWD

: =$ ( ADDR1 ADDR --- FLAG )
   COUNT ROT COUNT ROT     ( A A1 N1 N)
   SWD MIN SWAP SWD MIN  ( A A1 N1' N')
   OVER = NOT IF
    2DROP DROP 0 EXIT   ( N'<>N1', F=0)
   THEN 1 SWAP                ( A A1 N)
   0 DO                       ( A A1 1)
    OVER I + C@            ( A A1 1 C1)
    4 PICK I + C@        ( A A1 1 C1 C)
    = NOT IF                ( A1 A2 1)
```

```
      DROP 0 LEAVE           ( A1 A2 0)
    THEN
   LOOP                    ( A1 A2 1/0)
   ROT ROT 2DROP ;

-->




( ---
( --- SCREEN # 9 ---
( ---
( CASE: CHARLES EAKER-> LH-> RD 012284)
: CASE ?COMP CSP @ SP@ CSP ! 4 ;
  IMMEDIATE

: OF 4 ?PAIRS COMPILE OVER COMPILE =
  COMPILE 0BRANCH HERE 0 , COMPILE DROP
  5 ; IMMEDIATE

: ENDOF 5 ?PAIRS COMPILE BRANCH HERE
  0 , SWAP 2 ~[COMPILE] THEN 4 ;
  IMMEDIATE

: ENDCASE 4 ?PAIRS COMPILE DROP
  BEGIN
   SP@ CSP @ = NOT WHILE
    2 ~[COMPILE] THEN
  REPEAT CSP ! ; IMMEDIATE

: ANY DUP ; ( USE IN 'ANY OF ... ENDOF')

( THIS ONE FOR STRINGS)
: OF$ 4 ?PAIRS COMPILE OVER COMPILE =$
  COMPILE 0BRANCH HERE 0 , COMPILE DROP
  5 ; IMMEDIATE
-->



( ---
( --- SCREEN # 10 ---
( ---
( TABLE-BUILDING WORDS 070984 )

( THESE BUILD TABLES OF N+1 ENTRIES:
  0,1,...,N FROM LISTS IN THE INPUT
  STREAM )

: BCONVERT ( N --- ) ( BYTES )
    1+ 0 DO
      BL WORD HERE NUMBER DROP C,
    LOOP ;

( USAGE:
  N BTABLE NAME B0 B1 B2 ... BN )
```

```
: BTABLE <BUILDS BCONVERT
    DOES> + C@ ;

 -->




( ---
( --- SCREEN # 11 ---
( ---
( COMMAND AND VALUE TABLES 052684 )
15 VARIABLE $LEN ( LENGTH OF NAMES IN
  TABLES) : $LEN@+ $LEN @ 1+ ;

: CMDBLD ( N ---) ( BUILD COMMAND TABLE)
  1+ 0 DO ( N+1 EA, WORD$ + CFA )
   BL WORD $LEN@+ ALLOT ( WORD$ )
   FIND DUP 0= 0 ?ERROR , ( CFA )
  LOOP ;

: CMDTBL ( N CMDTBL NAME WORD$ WORD ...)
  <BUILDS DUP C, ( N-> 0TH LOC)
    CMDBLD ( N+1 ENTRIES: WORD$ + CFA )
  DOES> ( --- ADDR OF N ) ;

: () ( ADDR I --- ADDR ENTRY )
    OVER C@ OVER <     ( OVERRANGE? )
    OVER 0< OR IF      ( NEGATIVE?  )
     CR ." ? COMMAND RANGE ERROR"
     ABORT
    THEN $LEN@+ 2+ * SWAP DUP ROT + 1+ ;

( USAGE: NAME I <> CMDEXE )
: CMDEXE $LEN@+ + @ EXECUTE DROP ;
-->



( ---
( --- SCREEN # 12 ---
( ---
( COMMAND AND VALUE TABLES 020184 )
WORD$ '?'

: LIST? ( CMDTBL --- ) ( PRINT NAMES )
  HERE '?' =$ HERE C@ 0 = OR IF ( ?/NUL)
   ." SELECT FROM:" CR
   DUP C@ 1+ 0 DO
     I () SPACE TYPE$ CR
   LOOP
   ." SELECTION: " QUERY CR BL WORD
  THEN ;
: VALBLD ( N --- ) ( BUILD VALUE TABLE)
```

```
  1+ 0 DO ( N+1 EA, WORD$ + VALUE)
   BL WORD $LEN@+ ALLOT ( WORD$ )
   BL WORD HERE NUMBER DROP , ( VALUE )
  LOOP ;

: VALTBL ( N VALTBL NAME WORD$ VALUE...)
  <BUILDS DUP C, ( N-> 0TH LOC)
    VALBLD ( N+1 ENTRIES: WORD$ + VALUE)
  DOES> ( --- ADDR OF N ) ;

( USAGE: NAME I <> VALGET )
: VALGET $LEN@+ + @ ;
 -->


( ---
( --- SCREEN # 13 ---
( ---
( ASCII/BOOLEAN CONSTANTS KERMIT 061884)

ASC A CONST <A>     ASC B CONST <B>
ASC C CONST <C>     ASC D CONST <D>
ASC E CONST <E>     ASC F CONST <F>
ASC G CONST <G>     ASC H CONST <H>
ASC I CONST <I>     ASC J CONST <J>
ASC K CONST <K>     ASC L CONST <L>
ASC M CONST <M>     ASC N CONST <N>
ASC O CONST <O>     ASC P CONST <P>
ASC Q CONST <Q>     ASC R CONST <R>
ASC S CONST <S>     ASC T CONST <T>
ASC U CONST <U>     ASC V CONST <V>
ASC W CONST <W>     ASC X CONST <X>
ASC Y CONST <Y>     ASC Z CONST <Z>
ASC ? CONST <?>     ASC # CONST <#>
ASC " CONST <">

1 CONST TRUE
0 CONST FALSE

-->




( ---
( --- SCREEN # 14 ---
( ---
( LCIN ASCII-> CBMASCII 071884)

( TO INPUT FROM MODEM TO C64-ASCII FILES
  AND TO TERMINAL SCREEN WHEN BOTH UPPER
  AND LOWER CASE CHARACTERS ARE PRESENT)

( INPUT BYTE = 0-127)
( 8=BACKSPACE->157=CRSR LEFT)
( 92=BACKSLASH->221=VERT BAR, NOT \)
( 9=TAB->220)          ( 12=FF->219)
```

```
127 BTABLE LCIN
00 00 00 00 00 00 00 00 157 220
00 00 219 13 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00
00 00 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
060 061 062 063 064 193 194 195 196 197
198 199 200 201 202 203 204 205 206 207
208 209 210 211 212 213 214 215 216 217
218 091 221 093 094 095 096 065 066 067
68 69 70 71 72 73 74 75 76 77
78 79 80 81 82 83 84 85 86 87
88 89 90 00 00 00 00 00        -->


( ---
( --- SCREEN # 15 ---
( ---
( UCIN ASCII-> CBMASCII 071884)

( TO INPUT FROM MODEM TO C64-ASCII FILES
  AND TO TERMINAL SCREEN WHEN ONLY UPPER
  CASE CHARACTERS ARE DESIRED IN C64)

( INPUT BYTE = 0-127)
( 8=BACKSPACE->157=CRSR LEFT)
( 92=BACKSLASH->221=VERT BAR, NOT \)
( 9=TAB->220)         ( 12=FF->219)

127 BTABLE UCIN
00 00 00 00 00 00 0 0 157 220
00 00 219 13 00 00 00 00 00 0
00 00 00 00 00 00 00 00 00 00
00 00 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69
70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89
90 91 221 93 94 95 96 65 66 67
68 69 70 71 72 73 74 75 76 77
78 79 80 81 82 83 84 85 86 87
88 89 90 00 00 00 00 00  -->


( ---
( --- SCREEN # 16 ---
( ---
( LCOUT C64ASCII-> ASCII 111384)
255 BTABLE LCOUT
00 00 02 03 04 05 06 07 08 09
00 11 12 13 14 15 16 17 18 19
127 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
060 061 062 063 064 097 098 099 100 101
102 103 104 105 106 107 108 109 110 111
```

```
112 113 114 115 116 117 118 119 120 121
122 091 092 093 094 095 096 065 066 067
68 69 70 71 72 73 74 75 76 77
78 79 80 81 82 83 84 85 86 87
88 89 90 00 00 00 00 00 00 00
00 00 00 17 19 08 27 18 20 10
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
00 00 00 00 00 00 00 00 00 00
00 00 00 65 66 67 68 69 70 71
72 73 74 75 76 77 78 79 80 81
82 83 84 85 86 87 88 89 90 12
09 92 00 00 00 00 00 00 00 00
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0                    -->


( ---
( --- SCREEN # 17 ---
( ---
( UCOUT C64ASCII-> ASCII 111384)
255 BTABLE UCOUT
00 00 02 03 04 05 06 07 08 09
00 11 12 13 14 15 16 17 18 19
127 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69
70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 65 66 67
68 69 70 71 72 73 74 75 76 77
78 79 80 81 82 83 84 88 86 87
88 89 90 00 00 00 00 00 00 00
000 000 000 017 019 008 027 018 020 010
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
00 00 00 00 00 00 00 00 00 00
00 00 00 65 66 67 68 69 70 71
72 73 74 75 76 77 78 79 80 81
82 83 84 85 86 87 88 89 90 12
09 92 00 00 00 00 00 00 00 00
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0                    -->


( ---
( --- SCREEN # 18 ---
( ---
( KERMIT CONSTANTS AND VARIABLES 071684)
( NOTE: CONSTANTS ARE ->ASCII<- VALUES)

 94 CONST MAXPACK  ( MAX PACKET LENGTH)
  1 CONST <SOH>    ( START OF HEADER)
 32 CONST <SP>     ( ASCII SPACE)
 13 CONST <CR>     ( CARRIAGE RETURN)
 10 CONST <LF>     ( LINE FEED)
127 CONST <DEL>    ( DELETE/RUBOUT)
```

```
  5 CONST MAXTRY   ( TIMES TO RETRY)
140 CONST MYESC ( CONNECT-ESCAPE CHAR)

<#> VAR MYQUOTE ( QUOTE CHAR I'LL USE)
  0 VAR MYPAD   ( NO. PAD CHARS I NEED)
  0 VAR MYPCHAR ( PAD CHAR I NEED)
<CR> VAR MYEOL  ( ENDOLINE CHAR I NEED)
 10 VAR MYTIME  ( SEC AFTER WHICH I)
                ( SHOULD BE TIMED OUT)
  5 VAR DELAY   ( SEC DELAY BEFORE I
                ( SEND INIT PACKET)
127 VAR BITS    ( MASK FOR 7-BIT/8-BIT)

-->




( ---
( --- SCREEN # 19 ---
( ---
( KERMIT CONSTANTS AND VARIABLES 070884)
( VARIABLES FROM 'C' 8080 VERSION)

0VAR SIZE ( SIZE OF PRESENT DATA)
0VAR N ( MESSAGE NUMBER)

MAXPACK VAR RPSIZ ( MAX RECEIVE PACKET)
MAXPACK VAR SPSIZ    ( MAX SEND PACKET)
0  VAR SPAD ( HOW MUCH PADDING TO SEND)
10 VAR TIMINT ( SEC TIME AFTER WHICH I)
               ( NUDGE OTHER KERMIT)
0VAR NUMTRY ( TIMES THIS PACKET RETRIED)
0VAR OLDTRY ( TIMES PREVIOUS PACKET
               RETRIED)
0VAR KSTATE ( PRESENT STATE OF THE
               KERMIT AUTOMATON)
0VAR PADCHAR ( PAD CHARACTER TO SEND)
<CR> VAR EOL ( END OF LINE TO SEND)
<#>  VAR QUOTE ( QUOTE CHAR IN INCOMING
             DATA)
0VAR FLWRN ( FLAG, 1=> FILE-WARNING ON)
-->




( ---
( --- SCREEN # 20 ---
( ---
( KERMIT TABLES, SETUP/ERROR 071584)
0VAR KERR     0VAR KTEMP

: PAUSE ."  -ANY KEY CONTINUES-"
  KEYIN DROP ;

: UNKERR 0 KERR !  ( RESET ERROR FLAG )
  LEAVE ;   ( MIGHT AS WELL OMIT LOOPS)
```

```
: DOKERR 1 KERR ! ; ( SET ERROR FLAG )

: TSTKERR ( TEST ERROR FLAG, WARN )
  KERR @ IF CR IN @ KTEMP !
  0 IN !
  BLK @ IF BLK @ BLOCK ELSE TIB @ THEN
  KTEMP @ 0 DO DUP I + C@ EMIT LOOP DROP
  ." <--ERROR?" CR ." NO ACTION" CR
  SP! PAUSE ABORT THEN ;

: XSETUP ( ADDR --- ADDR N+1 0 )
  ( SETUP FOR SEARCH OF VAL/CMDTBL )
  DOKERR          ( PRESET ERROR FLAG )
  BL WORD              ( INPUT STRING )
  LIST?               ( IS IT A '?' ? )
  DUP C@ 1+ 0 ; -->    ( SETUP DO-LOOP)


( ---
( --- SCREEN # 21 ---
( ---
( TERMINAL CONSTS/VARS/ROUTINES 120284)

0VAR ITSPEED 0VAR TSPEED ( BAUD RATE)
0VAR ITWORD 0VAR TWORD ( WORD LENGTH)
0VAR ITSTOP 0VAR TSTOP ( STOP BITS)
0VAR ITSHK  0VAR TSHK  ( HANDSHAKE MODE)
0VAR ITDUP  0VAR TDUP  ( ECHO/DUPLEX)
0VAR ITPAR  0VAR TPAR  ( PARITY)
0VAR TCTL   0VAR TCOM  ( CTL, CMD WORDS)
0VAR ICHRSET ( GRAPHICS/TEXT CHAR SET)
0VAR IBORDER ( BORDER COLOR, 0-15)
0VAR IBKGND ( BACKGROUND COLOR, 0-15)
0VAR ICHRCLR ( CHARACTER COLOR, 0-15)

: CLR ." " ;          ( CLEARS SCREEN)

: DWN ( N ---)  ( STARTS A LINE N DOWN)
  ." " -DUP IF 0 DO ." " LOOP THEN
  39 0 DO BL EMIT LOOP
  39 0 DO ." " LOOP ; -->




( ---
( --- SCREEN # 22 ---
( ---
( TERMINAL SCREEN PRINT 120284 )

( SPECIAL ROUTINE TO AVOID QUOTE MODE
  AND TO SHOW CURSOR WHERE PRINT WILL
  NEXT OCCUR )

: BYTEPRINT ( B ---)  ( SCREEN PRINT)
```

```
  DUP IF             ( ONLY IF NONZERO)
  0 212 C!       ( TURN OFF QUOTE MODE)
             ( FIRST REMOVE OLD CURSOR)
  0 646 C@ 32 59923 CALLR 2DROP DROP
  EMIT      ( THEN PRINT ONE CHARACTER)
                  ( AND ADD NEW CURSOR)

  0 646 C@ 121 59923 CALLR 2DROP DROP
  ELSE DROP THEN ;

 -->
```

```
( ---
( --- SCREEN # 23 ---
( ---
( PACKET VARIABLES 063084)
( MEMTOP MUST BE < $CB00 )
HEX
CB80 CONST PACKET ( KERMIT SEND BUFFER)
 0VAR NN       ( COUNTER FOR BYTES-OUT)
 0VAR II     ( COUNTER FOR SEND BUFFER)
 0VAR BUFFADDR   ( SEND BUFFER ADDRESS)
CB00 CONST RECPKT ( KERMIT RECEIVE BUF)
DECIMAL

0VAR <KEY> ( KEYBOARD CHARACTER)

0VAR MOD$ ( MODEM INPUT STRING, 1 CHAR)
 MOD$ 1+ CONST MOD$1  ( ACCESS TO BYTE)

0VAR MOU$   ( MODEM OUT STRING. 1 CHAR)

0VAR RTYPE      ( PACKET TYPE, RECEIVE)
0VAR RCK           ( CHECKSUM, RECEIVE)
0VAR RNUM     ( NUMBER OF PACKET, RECV)
0VAR RLEN     ( LENGTH OF PACKET, RECV)

0VAR TNUM     ( NUMBER OF PACKET, SEND)
0VAR TLEN     ( LENGTH OF PACKET, SEND)
-->
```

```
( ---
( --- SCREEN # 24 ---
( ---
( FILE CONSTANTS AND VARIABLES 071784)
( EXPECTS MEMTOP < $CB00 )

HEX
CD00 CONST FBUFF    ( C64-FILES BUFFER)
```

```
  FF CONST FBUFFLEN   ( LENGTH OF IT)
CC00 CONST RNAME$  ( NAME OF READ-FILE)
CC40 CONST WNAME$ ( NAME OF WRITE-FILE)
CC80 CONST FNAME$   ( NAME OF EXT FILE)
CCC0 CONST BUF1 ( BUFFER FOR FILENAMES)
DECIMAL

0VAR FEOF ( EOF FOUND, END THIS BUFFER)
0VAR FEOLB   ( END OF LAST FILE BUFFER)
0VAR FBFPTR ( PTR INTO FILE I/O BUFFER)
0VAR FILTYP ( TYPE OF FILE, TRANSLATE?)

WORD$ 'S0:'
WORD$ '0:'
WORD$ ',R'
WORD$ ',W'
WORD$ ',P'
WORD$ ',S'
-->



( ---
( --- SCREEN # 25 ---
( ---
( DISK BYTE FETCH DGET# 060884)
( XXXX* = C64-BASIC KERNAL ROUTINES )
( REPLACE GET# FOR DISK READ, FIX BUGS)

: READST ( --- B ) ( RETURN STATUS BYTE)
  0 0 0 65463 CALLR          ( READST*)
  ROT ROT 2DROP ;

: UNTALK 65451 CALL ;         ( UNTLK*)

: DGET# ( FILE# ADDR COUNT --- )
  3 PICK 0 SWAP 0 65478 CALLR ( CHKIN*)
  DROP 2DROP 1+ 1 DO
   0 0 0 65445 CALLR          ( ACPTR*)
   4 PICK I + C! 2DROP    ( STORE BYTE)
   I OVER C!    ( STORE CURRENT LENGTH)
   READST IF       ( CHECK STATUS BYTE)
    LEAVE
   THEN
  LOOP 2DROP UNTALK
  READST ST ! CKST DROP ;   ( RD,CK ST)

-->



( ---
( --- SCREEN # 26 ---
( ---
( FILE I/O 062984)   ( INPUT UTILITIES)

: FILLFBUFF ( ---) ( FILL EMPTY BUFFER)
  18 FBUFF 1- COUNT DGET#
```

```
  CKRDSTAT ( EOF?  WORSE?)
  FEOF ! ( SAVE IN EOF-FLAG)
  1 FBFPTR ! ; ( RESET BUFFER POINTER)

: FGET ( --- B )
  ( GET ONE BYTE FROM FILE BUFFER AND )
  ( SET EOLB FLAG IF LAST BYTE IN FILE)
  FBFPTR @ FBUFF + C@       ( GET BYTE)
  FBFPTR @ 1+ DUP FBFPTR !   ( INC PTR)
  FBUFF C@ > IF         ( NEED REFILL?)
   FEOF @ IF  ( -YES, END LAST BUFFER?)
    1 FEOLB !          ( SET EOLB FLAG)
   ELSE
    FILLFBUFF    ( -NO, REFILL BUFFER)
   THEN
  THEN ;
-->




( ---
( --- SCREEN # 27 ---
( ---
( FILE I/O 070384)     ( OPEN FOR READ)

: FRCLOSE 18 CLOSE      ( CLOSE FILE)
  CLOSCHN ;

: FROPEN ( --- F)
  '0:' RNAME$ BUF1 CONCAT$
  ',R' BUF1 ADD$         ( "0:NAME,S,R)
  OPENCHN
  18 8 18 BUF1 COUNT OPEN
  ?FILE DUP    ( F=0=>OK,1=> NOT FOUND)
  IF
   3 DWN RNAME$ TYPE$ ."  NOT FOUND"
   FRCLOSE ABORTIO         ( SHUT DOWN)
   1 FEOLB !        ( AND SET EOF FLAG)
  ELSE
   FILLFBUFF     ( FOUND: FILL BUFFER,)
   0 FEOLB !     ( AND RESET EOF FLAG )
  THEN ;
 -->




( ---
( --- SCREEN # 28 ---
( ---
( FILE I/O 072484)  ( OUTPUT UTILITIES)
: EMPTYFBUFF     ( EMPTIES FILE BUFFER)
  19 FBUFF COUNT PRINT#
```

```
  CKWRSTAT 0 FBUFF C! ; ( CHECK, RESET)

: FPUT ( B ---)   ( ONE BYTE TO BUFFER)
  FBUFF COUNT + C!    ( STORE THE BYTE)
  FBUFF C@ 1+ DUP FBUFF C!     ( COUNT)
  FBUFFLEN < NOT IF EMPTYFBUFF THEN ;

: NAMETEST ( NAME IN WNAME$) ( --- F)
  WNAME$ C@ IF 0           ( F=0 => OK)
  WNAME$ C@ 0 DO    ( F>0 => BAD CHARS)
    WNAME$ 1+ I + C@ 127 AND UCIN
    DUP 46 < IF SWAP 1+ SWAP THEN
    DUP 90 > IF SWAP 1+ SWAP THEN
    DUP 57 > OVER 65 < AND IF
      SWAP 1+ SWAP THEN
    OVER IF DROP BL THEN
    WNAME$ 1+ I + C!
  LOOP FILTYP @ CASE ( APPEND ,P OR ,S)
   <H> OF ',P' ENDOF
   <B> OF ',P' ENDOF
   ANY OF ',S' ENDOF
  ENDCASE WNAME$ ADD$ ELSE 1 THEN ; -->


( ---
( --- SCREEN # 29 ---
( ---
( FILE I/O 071384)    ( OPEN FOR WRITE)
: FWOPEN ( NAME IN NAME$ ) ( --- )
  '0:' WNAME$ BUF1 CONCAT$
  ',W' BUF1 ADD$      ( "0:WNAME,X,W" )
  OPENCHN
  19 8 19 BUF1 COUNT OPEN
  CKWRSTAT   ( FILE ALREADY EXISTS?...)
  FBUFF FBUFFLEN 1+ 0 FILL ;  ( 0 BUFF)

: WCLOSE 19 CLOSE CLOSCHN ;

: FWCLOSE   ( EMPTY BUFFER, CLOSE FILE)
  FBUFF C@ 0= NOT IF  ( STUFF IN BUFF?)
   EMPTYFBUFF         ( -YES, EMPTY IT)
  THEN WCLOSE ;

: FWTEST ( --- F) ( 1=> NO FILE, SAFE)
  '0:' WNAME$ BUF1 CONCAT$
  OPENCHN
  19 8 19 BUF1 COUNT OPEN
  ?FILE
  WCLOSE ;
 -->




( ---
( --- SCREEN # 30 ---
( ---
( SET/SHOW FILE-TRANSLATION 071184)
```

```
( FILE TYPES:
SEQUENTIAL:
  T = TEXT = LC/UC, TRANSLATE DATA
  UC-ONLY = UC/GRAPHICS, TRANSLATE DATA
  A = ANSI/ASCII, 7-BIT, NO TRANSLATION
PROGRAM:
  B = BINARY, 8-BIT, NO TRANSLATION
  H = HEX NIBBLE CODING, 8-BIT BINARY )

 0VAR ITYPE

 4 VALTBL VTYPE TEXT 84     UC-ONLY 85
             ASCII-7 65    BINARY-8 66
                 HEX 72

: STYPE ( N ---)
  ITYPE ! VTYPE ITYPE @ ()
  VALGET FILTYP ! DROP
  FILTYP @ <B> = IF 255 ELSE 127 THEN
  BITS ! ;

 0 STYPE              ( DEFAULT = TEXT)
 -->


( ---
( --- SCREEN # 31 ---
( ---
( SET/SHOW FILE-TRANSLATION 071184)

: XTYPE ( SELECT FILE TYPE FROM KBD)
  VTYPE XSETUP
  DO I () HERE =$ IF
     I STYPE UNKERR
  THEN LOOP DROP TSTKERR ;

: SHOWTYPE
  VTYPE ITYPE @ ()
  ." FILE TYPE = " TYPE$ CR DROP
  FILTYP @ CASE
   <H> OF ."   <PROGRAM>" ENDOF
   <B> OF ."   <PROGRAM>" ENDOF
   ANY OF ."   <SEQUENTIAL>" ENDOF
  ENDCASE CR ;

 -->









( ---
( --- SCREEN # 32 ---
( ---
( TERMINAL PARAMETER SETUP 020484)
```

```
( SET BAUD RATE )

3 VALTBL VTSPEED
  110 3 300 6 1200 8 2400 10

: STSPEED ( N --- )
    ITSPEED ! VTSPEED ITSPEED @ ()
    VALGET TSPEED ! DROP ;
1 STSPEED

( SET WORD LENGTH )

1 VALTBL VTWORD 8-BITS 0 7-BITS 32

: STWORD ( N --- )
    ITWORD ! VTWORD ITWORD @ ()
    VALGET TWORD ! DROP ;
0 STWORD
-->




( ---
( --- SCREEN # 33 ---
( ---
( TERMINAL PARAMETER SETUP 020484)

( SET STOP BITS, NORMALLY 1 )

1 VALTBL VTSTOP 1 0 2 128

: STSTOP ( N --- )
    ITSTOP ! VTSTOP ITSTOP @ () VALGET
    TSTOP ! DROP ;
0 STSTOP

( SET HANDSHAKE MODE )

1 VALTBL VTSHK 3-LINE 0 X-LINE 1

: STSHK ( N --- )
    ITSHK ! VTSHK ITSHK @ () VALGET
    TSHK ! DROP ;
0 STSHK

-->




( ---
( --- SCREEN # 34 ---
( ---
```

```
( TERMINAL PARAMETER SETUP 020484 )

( SET ECHO MODE, NORMALLY FULL DUPLEX )

1 VALTBL VTDUP FULL 0 HALF 16

: STDUP ( N --- )
    ITDUP ! VTDUP ITDUP @ () VALGET
    TDUP ! DROP ;
0 STDUP

( SET PARITY, NORMALLY NO )

4 VALTBL VTPAR NO 0 ODD 32 EVEN 96
               MARK 160 SPACE 224

: STPAR ( N --- )

    ITPAR ! VTPAR ITPAR @ () VALGET
    TPAR ! DROP ;
0 STPAR
-->




( ---
( --- SCREEN # 35 ---
( ---
( TERMINAL PARAMETER SETUP 071784 )

   ( CONSTRUCT CONTROL, COMMAND WORDS )

: STCTL
   TSPEED @ TWORD @ + TSTOP @ + TCTL ! ;

: STCOM
   TSHK @ TDUP @ + TPAR @ + TCOM ! ;

        ( BUILD RS-232 OPENING STRING )

CREATE TOPN$ 0 , 0 C,        ( 3 BYTES)

: BLDTOPN$
    STCTL STCOM
    TOPN$            ( OPENING STRING )
    2 OVER C!          ( NO. OF BYTES )
    1+ TCTL C@ OVER C!    ( CTRL BYTE )
    1+ TCOM C@ SWAP C! ;   ( CMD BYTE )
-->




( ---
```

```
( --- SCREEN # 36 ---
( ---
( TERMINAL PARAMETER SETUP 020484 )

: XTSPEED ( SELECT BAUD RATE )
    VTSPEED XSETUP
    DO I () HERE =$ IF
       I STSPEED UNKERR
    THEN LOOP DROP TSTKERR ;

: XTWORD ( SELECT WORD LENGTH )
    VTWORD XSETUP
    DO I () HERE =$ IF
       I STWORD UNKERR
    THEN LOOP DROP TSTKERR ;

: XTSTOP ( SELECT NO. OF STOP BITS )
    VTSTOP XSETUP
    DO I () HERE =$ IF
       I STSTOP UNKERR
    THEN LOOP DROP TSTKERR ;

: XTSHK ( SELECT HANDSHAKE MODE )
    VTSHK XSETUP
    DO I () HERE =$ IF
       I STSHK UNKERR
    THEN LOOP DROP TSTKERR ;
-->

( ---
( --- SCREEN # 37 ---
( ---
( TERMINAL PARAMETER SETUP 020484)

: XTDUP ( SELECT ECHO/DUPLEX MODE )
  VTDUP XSETUP
  DO I () HERE =$ IF
     I STDUP UNKERR
  THEN LOOP DROP TSTKERR ;

: XTPAR ( SELECT PARITY )
  VTPAR XSETUP
  DO I () HERE =$ IF
     I STPAR UNKERR
  THEN LOOP DROP TSTKERR ;

-->
```

```
( ---
( --- SCREEN # 38 ---
( ---
( TERMINAL PARAMETER SETUP 063084)

1 VALTBL VCHRSET UC/GRAPHIC 21 LC/UC 23

: SCHRSET ( N --- )
  ICHRSET ! VCHRSET ICHRSET @ () VALGET
  53272 ! DROP ;

0 SCHRSET

: XCHRSET ( SELECT CHARACTER SET )
    VCHRSET XSETUP
    DO I () HERE =$ IF
       I SCHRSET UNKERR
    THEN LOOP DROP TSTKERR ;

-->
```

```
( ---
( --- SCREEN # 39 ---
( ---
( TERMINAL PARAMETER SETUP 020484)

: NUMIN BL WORD HERE NUMBER DROP ;

: SBORDER ( N ---)
   DUP 53280 C! IBORDER ! ;
14 SBORDER

: XBORDER NUMIN 15 AND SBORDER ;

: SBKGND ( N ---)
   DUP 53281 C! IBKGND ! ;
1 SBKGND

: XBKGND NUMIN 15 AND SBKGND ;

: SCHRCLR ( N ---)
   DUP 646 C! ICHRCLR ! ;
6 SCHRCLR

: XCHRCLR NUMIN 15 AND SCHRCLR ;

-->
```

```
( ---
( --- SCREEN # 40 ---
( ---
( TERMINAL PARAMETER STORAGE 070284)
: STOR ( BLKADDR INDEX VALADDR ---)
  C@ 0 <# BL HOLD # # #> DROP
  SWAP 3 * 3 PICK + 3 CMOVE ;

: RECL ( --- VALUE) NUMIN ;

: XTST 95 BLOCK
  0 ITSPEED STOR 1 ITWORD STOR
  2 ITSTOP STOR  3 ITSHK STOR
  4 ITDUP STOR   5 ITPAR STOR
  6 ICHRSET STOR 7 ICHRCLR STOR
  8 IBKGND STOR  9 IBORDER STOR
  DROP UPDATE ;
: XTSTORE XTST FLUSH ;

: XTRECALL BLK @ IN @ 95 BLK ! 0 IN !
  RECL  STSPEED  RECL  STWORD
  RECL  STSTOP   RECL  STSHK
  RECL  STDUP    RECL  STPAR

  RECL  SCHRSET  RECL  SCHRCLR
  RECL  SBKGND   RECL  SBORDER
  IN ! BLK ! ;
 -->



( ---
( --- SCREEN # 41 ---
( ---
( TERMINAL-PARAMETER SET 071484)

( 'SET TERMINAL' OPTIONS)

11 CMDTBL CTERMINAL
  SPEED XTSPEED
  WORD XTWORD
  STOPBITS XTSTOP
  HANDSHAKE XTSHK
  ECHO XTDUP
  PARITY XTPAR
  SAVE XTSTORE
  RESTORE XTRECALL
  CASE XCHRSET
  CHARCOLOR XCHRCLR
  BORDER XBORDER
  BACKGROUND XBKGND

: XTERMINAL
  CTERMINAL XSETUP
  DO I () HERE =$ IF
    CTERMINAL I () CMDEXE UNKERR
  THEN LOOP DROP TSTKERR ;
```

```
-->


( ---
( --- SCREEN # 42 ---
( ---
( SHOW TERMINAL-PARAMETERS 020484 )

: SHOWTERM ( --- )
  CR ." TERMINAL PARAMETERS:" CR
  VTSPEED ITSPEED @ ()
    ."  SPEED = " TYPE$ CR DROP
  VTWORD ITWORD @ ()
    ."  WORD LENGTH = " TYPE$ CR DROP
  VTSTOP ITSTOP @ ()
    ."  STOPBITS = " TYPE$ CR DROP
  VTSHK ITSHK @ ()
    ."  HANDSHAKE MODE = " TYPE$ CR DROP
  VTDUP ITDUP @ ()
    ."  ECHO = " TYPE$ ."  DUPLEX" CR
       DROP
  VTPAR ITPAR @ ()
    ."  PARITY = " TYPE$ CR DROP
  VCHRSET ICHRSET @ ()
    ."  CHARACTER SET = " TYPE$ CR
       DROP
  ." CHARACTER COLOR = " ICHRCLR ? CR
  ." BORDER COLOR = " IBORDER ? CR
  ." BACKGROUND COLOR = " IBKGND ? CR ;
-->



( ---
( --- SCREEN # 43 ---
( ---
( SET KERMIT OPTIONS 070884)

: XSPKTLEN NUMIN
  MAXPACK MIN SPSIZ ! ;

: XSPADDIN NUMIN SPAD ! ;

: XSPADCHR NUMIN PADCHAR ! ;

: XSTIMOUT NUMIN MYTIME ! ;

: XSENDLIN NUMIN EOL ! ;

: XSQUOTE BL WORD HERE 1+ C@
  LCOUT MYQUOTE ! ;

0VAR IFLWRN
: XFLWRNOFF 0 IFLWRN ! ;
: XFLWRNON 1 IFLWRN ! ;
-->
```

```
( ---
( --- SCREEN # 44 ---
( ---
( SET KERMIT OPTIONS 070884)

: XRPKTLEN NUMIN
  MAXPACK MIN RPSIZ ! ;


: XRPADDIN NUMIN MYPAD ! ;

: XRPADCHR NUMIN MYPCHAR ! ;

: XRTIMOUT NUMIN TIMINT ! ;

: XRENDLIN NUMIN MYEOL ! ;

: XRQUOTE BL WORD HERE 1+ C@
  LCOUT QUOTE ! ;

: XDELAY  NUMIN DELAY ! ;

-->
```

```
( ---
( --- SCREEN # 45 ---
( ---
( SET KERMIT OPTIONS 070284)

: XSDST 95 BLOCK 40 +
  0 SPSIZ STOR      1 SPAD STOR
  2 PADCHAR STOR    3 MYTIME STOR
  4 EOL STOR        5 MYQUOTE STOR
  DROP UPDATE ;

: XSDSTORE XSDST FLUSH ;

: XSDRECALL BLK @ IN @ 95 BLK ! 40 IN !
  RECL SPSIZ   !
  RECL SPAD    !
  RECL PADCHAR !
  RECL MYTIME  !
  RECL EOL     !
  RECL MYQUOTE !
  IN ! BLK ! ;

 -->
```

```
( ---
( --- SCREEN # 46 ---
( ---
( SET KERMIT OPTIONS, STO/RECALL 070284)

: XRCST 95 BLOCK 80 +
  0 RPSIZ STOR     1 MYPAD STOR
  2 MYPCHAR STOR   3 TIMINT STOR
  4 MYEOL STOR     5 QUOTE STOR
  DROP UPDATE FLUSH ;

: XRCSTORE XRCST FLUSH ;

: XRCRECALL BLK @ IN @ 95 BLK ! 80 IN !
  RECL RPSIZ   !

  RECL MYPAD   !
  RECL MYPCHAR !
  RECL TIMINT  !
  RECL MYEOL   !
  RECL QUOTE   !
  IN ! BLK ! ;

: RECALL XRCRECALL XSDRECALL XTRECALL ;

: STORE XRCST XSDST XTST
  FLUSH ;
 -->



( ---
( --- SCREEN # 47 ---
( ---
( SET KERMIT OPTIONS 071484)
7 CMDTBL CSEND
    LENGTH-PACKET XSPKTLEN
    PADDING       XSPADDIN
    CHAR-PAD      XSPADCHR
    TIMEOUT       XSTIMOUT
    END-OF-LINE   XSENDLIN
    QUOTE         XSQUOTE
    SAVE          XSDSTORE
    RESTORE       XSDRECALL
7 CMDTBL CRECEIVE
    LENGTH-PACKET XRPKTLEN
    PADDING       XRPADDIN
    CHAR-PAD      XRPADCHR
    TIMEOUT       XRTIMOUT
    END-OF-LINE   XRENDLIN
    QUOTE         XRQUOTE
    SAVE          XRCSTORE
    RESTORE       XRCRECALL
```

```
1 CMDTBL CFLWRN
    OFF XFLWRNOFF    ON XFLWRNON

 -->



( ---
( --- SCREEN # 48 ---
( ---
( SET KERMIT OPTIONS 062984)

: XSEND ( SET SEND PARAMETERS)
  CSEND XSETUP
  DO I () HERE =$ IF
    CSEND I () CMDEXE UNKERR
  THEN LOOP DROP TSTKERR ;

: XRECEIVE ( SET RECEIVE PARAMETERS)
  CRECEIVE XSETUP
  DO I () HERE =$ IF
    CRECEIVE I () CMDEXE UNKERR
  THEN LOOP DROP TSTKERR ;

: XFLWRN ( SET FILE-WARNING ON/OFF)
  CFLWRN XSETUP DO I () HERE =$ IF
    CFLWRN I () CMDEXE UNKERR
  THEN LOOP DROP TSTKERR ;

-->







( ---
( --- SCREEN # 49 ---
( ---
( SHOW KERMIT OPTIONS 062984)

: SHOWSEND
   CR ." SEND PARAMETERS:"
   CR ."  PACKET-LENGTH =    " SPSIZ ?
   CR ."  NO. PAD CHARS =    " SPAD ?
   CR ."  PAD CHARACTER =    " PADCHAR ?
   CR ."  TIMEOUT INTERVAL = " MYTIME ?
   CR ."  END-OF-LINE CHAR = " EOL ?
   CR ."  QUOTE CHARACTER =  " MYQUOTE
      @ UCIN EMIT CR ;

: SHOWRECEIVE
   CR ." RECEIVE PARAMETERS:"
   CR ."  PACKET-LENGTH =    " RPSIZ ?
   CR ."  NO. PAD CHARS =    " MYPAD ?
   CR ."  PAD CHARACTER =    " MYPCHAR ?
   CR ."  TIMEOUT INTERVAL = " TIMINT ?
   CR ."  END-OF-LINE CHAR = " MYEOL ?
```

```
   CR ."  QUOTE CHARACTER =  " QUOTE
      @ UCIN EMIT CR ;

-->



( ---
( --- SCREEN # 50 ---
( ---
( SHOW KERMIT OPTIONS 062984)

: SHOWFLWRN CR ." FILE WARNING IS "
    CFLWRN IFLWRN @ () TYPE$ CR DROP ;

: SHOWDELAY CR
  ." INITIAL DELAY INTERVAL = "
  DELAY ? ." SEC" CR ;
-->









( ---
( --- SCREEN # 51 ---
( ---
( SET/SHOW COMMAND TABLES 071284)

5 CMDTBL CSET
  TERMINAL       XTERMINAL
  SEND           XSEND
  RECEIVE        XRECEIVE
  FILE-TYPE      XTYPE
  WARNING-FILE   XFLWRN
  DELAY          XDELAY

5 CMDTBL CSHOW
  TERMINAL       SHOWTERM
  SEND           SHOWSEND
  RECEIVE        SHOWRECEIVE
  FILE-TYPE      SHOWTYPE
  WARNING-FILE   SHOWFLWRN
  DELAY          SHOWDELAY
```

```
-->

( ---
( --- SCREEN # 52 ---
( ---
( KERMIT SET/SHOW ROUTINES 063084)

: SET
  CSET XSETUP

  DO I () HERE =$ IF
    CSET I () CMDEXE UNKERR
  THEN LOOP DROP TSTKERR ;

: SHOW
  CSHOW XSETUP
  DO I () HERE =$ IF
    CSHOW I () CMDEXE UNKERR
  THEN LOOP DROP TSTKERR ;

-->

( ---
( --- SCREEN # 53 ---
( ---
( 1/60-SEC TIMER WORDS 110384)

0 VARIABLE TIVAR

: TICSET                  ( TICKS --- )
  ABS MINUS
  DUP )HI TIVAR C! )LO TIVAR 1+ C! ;

: INTSET                ( SECONDS --- )
  60 * TICSET ;                 ( 60T )

: TINTSET TIMINT @ INTSET ;

: TI0 TIVAR @ 161 ! ;   ( PRESET TIMER)

: TIGET ( --- F)   ( CHECK FOR TIMEOUT)
```

```
  161 C@ 128 < ;

: WAIT DELAY @ INTSET TI0
  BEGIN TIGET UNTIL ;

 -->




( ---
( --- SCREEN # 54 ---
( ---
( CONNECT-DEFINITIONS 110384)

( MEMTOP SHOULD BE BELOW $CB00 )

HEX CF00 CONSTANT BF1
    CE00 CONSTANT BF2 DECIMAL

: CLRIN           ( CLEAR INPUT BUFFER)
  BF1 256 0 FILL 0 667 ! ;

: CLROUT         ( CLEAR OUTPUT BUFFER)
  BF2 256 0 FILL 0 669 ! ;

: MODOPEN         ( OPEN MODEM FOR I/O)
   BLDTOPN$   ( SET CURRENT PARAMETERS)
   2 2 0 TOPN$ COUNT OPEN
   BF1 247 ! BF2 249 ! ( SET BUFF PTRS)
   CLRIN CLROUT ;   ( RESET BUFF INDEX)

: MODCLOSE BEGIN 669 C@ 670 C@ = UNTIL
   2 CLOSE ; ( WAIT TILL BUFFER EMPTY!)

 -->




( ---
( --- SCREEN # 55 ---
( ---
( TERMINAL BREAK-KEY SIMULATION 111084)

: TOGGEL  ( REVERSE RS-232 OUTPUT LINE)
  56576 DUP C@ 4 XOR SWAP C! ;

: BREAK  ( 1/4-SEC BREAK ON RS-232 OUT)
  ."  ~[BREAK] "
  30 TICSET
  MODCLOSE
  TOGGEL
  TI0 BEGIN TIGET UNTIL
  TOGGEL
  MODOPEN ;

 -->
```

```
( ---
( --- SCREEN # 56 ---
( ---
( CONNECT-ESCAPE SEQUENCES 110384)

( B=KEYBOARD BYTE, B1=OUTPUT,140=>NULL)

( FLAG=1 AND B1>0 SENDS B1 TO MODEM)
: FL0 0 0 ;   ( EXIT TO RESET CHAR SET)
: FL1 -1 0 ;        ( DISCONNECT MODEM)
: FL2 140 1 ; ( NULL OUTPUT, NO EFFECT)

: ?ESC  ( B --- B1 F,  F=1 => TRANSMIT)
  1 OVER MYESC = IF 2DROP ." ~[F8]"
  KEYIN DUP EMIT CASE
  <U> OF 0 SCHRSET FL0 ENDOF ( UC-ONLY)
  <L> OF 1 SCHRSET FL0 ENDOF    ( TEXT)
  <C> OF FL1 ENDOF        ( DISCONNECT)
  <B> OF BREAK FL2 ENDOF     ( 'BREAK')
  <S> OF SHOWSEND SHOWRECEIVE
         SHOWFLWRN SHOWDELAY
         FL2 ENDOF   ( SHOW PARAMETERS)
  <T> OF SHOWTERM FL2 ENDOF
  <X> OF FL1 ENDOF         ( SAME AS C)
  <?> OF CR ." OPTIONS: "  ( SHOW THEM)
      ." U L B C S T X ?" CR FL2 ENDOF
  ENDCASE THEN ; -->



( ---
( --- SCREEN # 57 ---
( ---
( TERMINAL I/O LOOPS 112584)

: GETKB                        ( --- B)
  GET TDUP @ IF
    DUP BYTEPRINT

  THEN ;

: GETMOD                       ( --- B)
  2 MOD$ 1 GET#
  MOD$ 1+ C@ 127 AND ;

: PUTMOD                       ( B ---)
  DUP IF
```

```
    <KEY> C!
    2 <KEY> 1 PRINT#
  ELSE
    DROP
  THEN ;

 -->




( ---
( --- SCREEN # 58 ---
( ---
( TERMINAL I/O LOOPS 111084)

: UCIO            ( UPPERCASE TERMINAL)
  BEGIN
    GETKB ?ESC
  WHILE
    UCOUT PUTMOD
    GETMOD
    UCIN BYTEPRINT
  REPEAT ;

: LCIO            ( LOWERCASE TERMINAL)
  BEGIN
    GETKB ?ESC
  WHILE
    LCOUT PUTMOD
    GETMOD
    LCIN BYTEPRINT
  REPEAT ;

 -->




( ---
( --- SCREEN # 59 ---
( ---
( TERMINAL I/O LOOPS 111084)

: TERMLOOP
  BEGIN
  53272 C@ 2 AND IF ( CHAR SET?)
    LCIO
  ELSE
    UCIO
  THEN     ( HERE IF ?ESC GIVES 0 FLAG)
  0< UNTIL ;    ( LOOP UNLESS NEGATIVE)

: CONNECT ( ACT AS TERMINAL, W ESC KEY)
```

```
  ."  *CONNECT TO REMOTE: F8-C RETURNS"
  CR MODOPEN   ( OPEN MODEM CONNECTION)
  TERMLOOP
  MODCLOSE CR ( CLOSE MODEM CONNECTION)
  ."  *RETURN TO LOCAL KERMIT" CR ;

-->




( ---
( --- SCREEN # 60 ---
( ---
( PACKET UTILITY ROUTINES 071584)

( CONVERTS CTRL TO PRINT CHAR BY +32)
: TOCHAR ( N1 --- N2) <SP> + ;

( UNDOES TOCHAR)
: UNCHAR ( N2 --- N1) <SP> - ;

( CONVERTS CTRL <-> PRINT CHAR BY
  TOGGLING CTRL BIT)
: CTL ( N1 --- N2) 64 XOR ;

: PRNTE         ( PRINTS ERROR PACKETS)
  9 DWN ." MSG FROM HOST:" CR
  PACKET RLEN @ 0 DO DUP I + C@
  UCIN EMIT LOOP DROP ;

: DISPL ." PKT # = " N ?
        ."  TRIES = " NUMTRY ? ;

 -->




( ---
( --- SCREEN # 61 ---
( ---
( PACKET UTILITY ROUTINES 071384)

( FILL EXT DATAARRAY W SENDINIT PARAMS)
: SPARA ( ADDATA ---)
   RPSIZ   @ TOCHAR OVER     C!
   MYTIME  @ TOCHAR OVER 1 + C!
   MYPAD   @ TOCHAR OVER 2 + C!
   MYPCHAR @ CTL    OVER 3 + C!
   MYEOL   @ TOCHAR OVER 4 + C!
   MYQUOTE @        SWAP 5 + C! ;
```

```
( GET SENDINIT STUFF FROM EXT DATAARRAY)
: RPARA ( ADDATA ---)
   DUP     C@ UNCHAR SPSIZ !
   DUP 1 + C@ UNCHAR TIMINT !
   DUP 2 + C@ UNCHAR SPAD !
   DUP 3 + C@ CTL    PADCHAR !
   DUP 4 + C@ UNCHAR EOL !
       5 + C@        QUOTE ! ;

: MESSG1 5 DWN ." TRANSFER COMPLETE" ;
: MESSG2 5 DWN ." ??TRANSFER FAILED" ;
 -->



( ---
( --- SCREEN # 62 ---
( ---
( SEND/RECEIVE PACKET WORDS 071084)

: NUMTRY+>?
   NUMTRY @ 1 NUMTRY +! MAXTRY > ;

: OLDTRY+>?
   OLDTRY @ 1 OLDTRY +! MAXTRY > ;

: KSTATE@ KSTATE @ ;
: KSTATE! KSTATE ! ;

: EOLSET
   EOL @ 0= IF <CR> EOL ! THEN ;

: QUOTSET
   QUOTE @ 0= IF <#> QUOTE ! THEN ;

: 0NUMTRY
   0 NUMTRY ! ;

: N+ N @ 1+ 64 MOD N ! ;
: N@ N @ ;

: INITIALIZE 0NUMTRY 0 OLDTRY ! 0 N ! ;
 -->



( ---
( --- SCREEN # 63 ---
( ---
( RECEIVE-PACKET ~[PART A] 071184)

: T7 MOD$1 C@ 127 AND ; ( 7BIT, NO SUM)
: T7+ T7 SWAP OVER + SWAP ;  ( W CKSUM)
: T8 MOD$1 C@ ;              ( 8BIT TM)
: T8+ T8 SWAP OVER + SWAP ; ( LIKE TM+)

( MGET WAITS FOR NON-ZERO INPUT BYTE. )
( MGET NORMALLY RETURNS   F=0, B>32,  )
( BUT A NEW PACKET CAUSES F=0, B=<SOH>)
```

```
( AND A TIMEOUT CAUSES    F=1, B=<SOH>)

: MGET ( --- F B )
  0               ( RESET TIMEOUT FLAG)
  BEGIN
    2 MOD$ 1 GET# MOD$1 C@  ( F B --- )
    TIGET IF       ( TIMEOUT ON ALARM )
      2DROP 1 <SOH> DUP MOD$1 C!
    THEN
  UNTIL              ( LOOP UNTIL B>0 )
  T7 ;              ( RETURN 7BIT BYTE)
 -->




( ---
( --- SCREEN # 64 ---
( ---
( RECEIVE-PACKET ~[PART B] 071184)
: WAITHEADER ( --- FLD=1/CKSUM=0 )
  TI0            ( RESET TIMEOUT CLOCK)
  BEGIN MGET ( F B ---) ( AWAIT PACKET)
  <SOH> = NOT WHILE DROP REPEAT
  IF 7 255          ( IS IT A TIMEOUT?)
  ELSE 1 0 THEN ;   ( OR A NEW PACKET?)

: GETHEADER  ( FLD/CKSUM --- FLD/CKSUM)
  MGET <SOH> = IF ( NORMALLY NOT <SOH>)
   IF 2DROP 7 255   ( IS IT A TIMEOUT?)
   ELSE 2DROP 0 0 THEN ( A NEW PACKET?)
  ELSE DROP THEN ;      ( DISCARD FLAG)

: FILLPACKET ( FLD/CKSUM --- FLD/CKSUM)
  3 PICK @  ( GET # BYTES = PKT LENGTH)
  -DUP IF
   6 PICK DUP ROT + SWAP DO
    MGET <SOH> = IF  ( TIMEOUT OR NEW?)
     IF 2DROP 7 255 ELSE 2DROP 0 0 THEN
    ELSE
     DROP T8+ I C! ( STORE 7/8BIT DATA)
    THEN
   LOOP
  THEN ; -->


( ---
( --- SCREEN # 65 ---
( ---
( RECEIVE-PACKET ~[PART C] 071184)

: RPACK ( ADDATA ADNUM ADLEN --- TYPE)
  WAITHEADER ( --- /FLD=1/CHKSUM=0/ )
  BEGIN ( LOOP ON FIELD NUMBER, FLD)
   OVER CASE
    1 OF GETHEADER        ( GET LENGTH)
     T7+ UNCHAR 3 -
     4 PICK ! ENDOF
```

```
    2 OF GETHEADER        ( GET NUMBER)
     T7+ UNCHAR
     5 PICK ! ENDOF
    3 OF GETHEADER          ( GET TYPE)
     T7+ RTYPE ! ENDOF
    4 OF FILLPACKET ENDOF   ( GET DATA)
    5 OF GETHEADER      ( GET CHECKSUM)
     T7 UNCHAR RCK !
     DUP 192 AND 64 / + 63 AND ENDOF
   ENDCASE SWAP 1+ SWAP OVER 5 >
  UNTIL DUP RCK @ = IF DROP ELSE
    4 DWN 255 =
    IF   ." ? TIMEOUT  "
    ELSE ." ? CKSUM ERR"
    THEN 0 RTYPE !
  THEN 2DROP 2DROP CLRIN RTYPE @ ; -->


( ---
( --- SCREEN # 66 ---
( ---
( SEND-PACKET ~[PART A] 071584)

: MPUT ( B ---)( OUTPUT 1 CHAR TO MODEM)
  MOU$ C! 2 MOU$ 1 PRINT# ;
 -->
```



```
( ---
( --- SCREEN # 67 ---
( ---
( SEND-PACKET ~[PART B] 071584)

: SPACK ( ADDDATA TYPE NUM LEN --- )
   SPAD @ -DUP IF
     0 DO PADCHAR @ MPUT LOOP
   THEN <SOH>                MPUT
( LENGTH OUTPUT; START CHECKSUM       )
   DUP 3 + TOCHAR DUP        MPUT
```

```
( NUM OUTPUT; STACK HAS ON TOP:       )
( .../TYPE/NUM/LEN/CHECKSUM/ ---      )
   ROT TOCHAR   DUP ROT + SWAP MPUT
( TYPE OUTPUT                         )
   ROT          DUP ROT + SWAP MPUT
( AT DATA OUTPUT STACK HAS ON TOP:    )
( /ADDDATA/LEN/CHECKSUM/ ---          )
   SWAP DUP IF
     3 PICK + ROT DO
       I C@     DUP ROT + SWAP MPUT
     LOOP
   ELSE DROP SWAP DROP
   THEN DUP 192 AND 64 / + 63 AND
     TOCHAR ( CHECKSUM OUT ) MPUT
   EOL @ MPUT ; -->




( ---
( --- SCREEN # 68 ---
( ---
( BUFFER-FILL ~[PART A] 072984 )
0VAR BTEMP ( 7/8BIT BYTE STORAGE)

: BOUT ( B --- BN...B1 N )
  DUP  <SP>  < OVER <DEL> = OR
  OVER MYQUOTE @ = OR IF ( SPEC'L CHR?)
   CASE
    MYQUOTE @ OF BTEMP @ BITS @ AND
      MYQUOTE @ 2 ENDOF ( QUOTE ITSELF)
    <CR> OF FILTYP @
      <B> = IF      ( NO CRLF IN 8-BIT)
        BTEMP @ CTL MYQUOTE @ 2
      ELSE      ( CR/LF IN 7-BIT MODES)
        74 MYQUOTE @ 77 MYQUOTE @ 4
      THEN ENDOF
    <LF> OF FILTYP @    ( REDUNDANT LF)
      <B> = IF BTEMP @ CTL MYQUOTE @ 2
      ELSE 0 THEN ENDOF
    ANY OF BTEMP @ BITS @ AND
      CTL MYQUOTE @ 2 ENDOF
   ENDCASE
  ELSE              ( NORMAL CHARACTER)
   DROP
   BTEMP @ BITS @ AND 1
  THEN ; -->


( ---
( --- SCREEN # 69 ---
( ---
( BUFFER-FILL ~[PART B] 071884)

: HEXOUT ( B --- B2 B1 2 )
  HEX 0 <# # # #> DROP
  DUP 1+ C@ SWAP C@ DECIMAL 2 ;

: UCOOUT ( B --- BN...B1 N)
```

```
  UCOUT DUP BTEMP ! BOUT ;

: TEXTOUT ( B --- BN...B1 N)
  LCOUT DUP BTEMP ! BOUT ;

: ASCOUT ( B --- BN...B1 N)
  127 AND DUP BTEMP ! BOUT ;

: BINOUT DUP BTEMP ! 127 AND BOUT ;

-->
```

```
( ---
( --- SCREEN # 70 ---
( ---
( BUFFER-FILL ~[PART C] 071684)

: BUFILL ( BUFFADDR --- SIZE )
  BUFFADDR ! 0 II !
  BEGIN ( IF NOT EOF AND II < MAXLEN-4 )
   FEOLB @ 0= II @ SPSIZ @ 9 - < AND
  WHILE
   FGET
   FILTYP @ CASE   ( 1 OR 2 BYTES OUT)
     <T> OF TEXTOUT ENDOF      ( LC/UC)
     <U> OF UCOOUT ENDOF     ( UC-ONLY)
     <H> OF HEXOUT ENDOF   ( HEX CODED)
     <B> OF BINOUT ENDOF ( 8BIT BINARY)
     ANY OF ASCOUT ENDOF ( ASCII, ETC.)
   ENDCASE ( --- BN...B1 N ---)
   -DUP IF 0 DO
     BUFFADDR @ II @ + C! 1 II +!
   LOOP THEN
  REPEAT
  II @ ; ( RETURNS SIZE=0 FOR EOF )
 -->
```

```
( ---
( --- SCREEN # 71 ---
( ---
( BUFFER-EMPTY ~[PART A] 111384)

 CREATE HBUF 1 , 0 ,  ( CONVERSION BUF)

: HEXIN ( LEN ADDR PTR CHR1 --- BYTE)
  UCIN HBUF DUP C@ + C!   ( STORE BYTE)
```

```
  HBUF DUP C@ 1 > IF    ( 2 BYTES YET?)
    HEX NUMBER DROP DECIMAL FPUT
    1 HBUF C!
  ELSE
    DUP C@ 1+ SWAP C!
  THEN ;

: TRPUT ( B ---) ( TRANSLATE, FILE CHR)
  BITS @ AND
  FILTYP @ CASE
    <T> OF LCIN ?DUP IF FPUT THEN ENDOF
    <U> OF UCIN ?DUP IF FPUT THEN ENDOF
    <H> OF HEXIN ENDOF   ( IN 2, OUT 1)
    ANY OF FPUT ENDOF   ( ASCII/BINARY)
  ENDCASE ;

: BUMP ( ADDR/PTR --- ADDR/PTR/B7)
  OVER OVER + C@
  DUP BTEMP ! 127 AND ; -->


( ---
( --- SCREEN # 72 ---
( ---
( BUFFER-EMPTY ~[PART B] 111384)

: BUFEMP ( LEN/BUFFADDR ---)
  0 ( LEN/BUFFADDR/PTR=0 ---)
  BEGIN
  DUP 4 PICK <         ( PTR < LENGTH?)
  WHILE
   BUMP              ( GET A CHARACTER)
   QUOTE @ = IF               ( QUOTE?)
     1+ BUMP      ( GET NEXT CHARACTER)
     QUOTE @ = IF             ( QUOTE?)
       BTEMP @ TRPUT  ( JUST PUT QUOTE)
     ELSE
       BTEMP @ CTL TRPUT      ( FIX IT)
     THEN
   ELSE
     BTEMP @ TRPUT
   THEN
   1+                  ( INCREMENT PTR)
  REPEAT DROP 2DROP ; -->




( ---
( --- SCREEN # 73 ---
( ---
( GET NAME OF FILE TO SEND 071784)

: GNAME ( STR$ ---) ( GET STR FROM KBD)
  <"> WORD HERE SWAP TO$ ;
```

```
: SNAME ( --- ) ( GET NAME$ FROM KBD)
   RNAME$ GNAME
   BEGIN ( DON'T QUIT WITHOUT NAME$)
    RNAME$ C@ 0= WHILE
     CR ." FILENAME: "
     QUERY RNAME$ GNAME
   REPEAT ;

: OUTNAME  ( TRANSLATE NAME$ TO ASCII )
  RNAME$ C@ FNAME$ C!     ( BYTE COUNT)
  RNAME$ C@ 1+ 1 DO   ( TRANSFER BYTES)
    RNAME$ I + C@
    UCOUT       ( ALWAYS IN UPPER CASE)
    FNAME$ I + C!
  LOOP ;
 -->




( ---
( --- SCREEN # 74 ---
( ---
( GET FILE-NAME TO RECEIVE 110384)

: GETFIL                ( NAMESIZE ---)
  WNAME$ C@ 0= IF         ( IF NO NAME)
   DUP WNAME$ C! 0 DO ( GET FROM F-PKT)
   I PACKET + C@ UCIN I 1+ WNAME$ + C!
   LOOP
  ELSE DROP THEN
  BEGIN BEGIN NAMETEST WHILE
      3 DWN ." ILLEGAL FILENAME" CR
      ." NEW NAME: " QUERY
      WNAME$ GNAME
    REPEAT 1               ( OK SO FAR)
    IFLWRN @ IF     ( FILE-WARNING ON?)
     BEGIN FWTEST 0= WHILE
       3 DWN WNAME$ TYPE$ ."  EXISTS" CR
       ." NEW NAME: " QUERY WNAME$ GNAME
       DROP 0            ( MUST RETEST)
     REPEAT
    THEN
  UNTIL
  IFLWRN @ 0= IF
    'S0:' WNAME$ BUF1 CONCAT$ OPENCHN
    15 BUF1 COUNT PRINT# CLOSCHN
  THEN FWOPEN ;  -->



( ---
( --- SCREEN # 75 ---
( ---
( SEND INIT-PACKET 070384)

: SINIT-ACK RECPKT RPARA EOLSET QUOTSET
  0NUMTRY N+ FROPEN 0= IF <F>
```

```
  ELSE <A> THEN ;

: SINIT
  CLRIN     ( CLEAR OUT OLD NAKS, JUNK)
  2 DWN RNAME$ TYPE$
  NUMTRY+>? IF <A> EXIT THEN   ( ABORT)
  PACKET SPARA       ( SEND 'S' PACKET)
  PACKET <S> N@ 6 SPACK
  RECPKT TNUM TLEN RPACK
  CASE
   <N> OF   ( NAK FOR N+1 => ACK FOR N)
     TNUM @ N@ 1+ - IF KSTATE@ ELSE
       SINIT-ACK THEN ENDOF
   <Y> OF ( ACKNOWLEDGES GOOD PACKET N)
     TNUM @ N@ - IF KSTATE@ ELSE
       SINIT-ACK THEN ENDOF
   <E> OF PRNTE <A> ENDOF ( ERR-PACKET)
   FALSE OF KSTATE@ ENDOF ( CKSM ERROR)
   ANY OF <A> ENDOF     ( UNRECOGNIZED)
  ENDCASE ;
 -->


( ---
( --- SCREEN # 76 ---
( ---
( SEND FILE-PACKET 070384)

: SFILE-ACK 0NUMTRY N+ PACKET BUFILL
  SIZE ! <D> ;    ( READY TO SEND DATA)

: SFILE
  NUMTRY+>? IF <A> EXIT THEN   ( ABORT)
  FNAME$ COUNT <F> N@ ROT SPACK
  RECPKT TNUM TLEN RPACK
  CASE          ( BASED ON PACKET TYPE)
   <N> OF   ( NAK FOR N+1 => ACK FOR N)
    TNUM @ N@ 1+ - IF KSTATE@ ELSE
       SFILE-ACK THEN ENDOF
   <Y> OF  ( ACKNOWLEDGE GOOD PACKET N)
    TNUM @ N@ - IF KSTATE@ ELSE
       SFILE-ACK THEN ENDOF
   <E> OF PRNTE <A> ENDOF ( ERR-PACKET)
   FALSE OF KSTATE@ ENDOF ( CKSM ERROR)
   ANY OF <A> ENDOF ( UNRECOGNIZED)
   ENDCASE ;

 -->




( ---
( --- SCREEN # 77 ---
( ---
( SEND DATA-PACKET 070384)

: SDATA-ACK 0NUMTRY N+    ( RESET ETC.)
```

```
  PACKET BUFILL DUP SIZE !   ( REFILL )
  IF <D> ELSE <Z> THEN ;       ( EOF? )

: SDATA
  NUMTRY+>? IF <A> EXIT THEN   ( ABORT)
  PACKET <D> N@ SIZE @ SPACK
  RECPKT TNUM TLEN RPACK
  CASE          ( BASED ON PACKET TYPE)
   <N> OF   ( NAK FOR N+1 => ACK FOR N)
    TNUM @ N@ 1+ - IF KSTATE@ ELSE
       SDATA-ACK THEN ENDOF
   <Y> OF  ( ACKNOWLEDGE GOOD PACKET N)
    TNUM @ N@ - IF KSTATE@ ELSE
       SDATA-ACK THEN ENDOF
   <E> OF PRNTE <A> ENDOF ( ERR-PACKET)
   FALSE OF KSTATE@ ENDOF ( CKSM ERROR)
   ANY OF <A> ENDOF     ( UNRECOGNIZED)
   ENDCASE ;
 -->


( ---
( --- SCREEN # 78 ---
( ---
( SEND EOF-PACKET 070384)

: SEOF-ACK 0NUMTRY N+     ( RESET ETC.)
  ( GNXTFL) 0   ( NO GET-NEXT-FILE YET)
  IF <F> ELSE <B> THEN ;  ( EOT?      )

: SEOF
  NUMTRY+>? IF <A> EXIT THEN   ( ABORT)
  PACKET <Z> N@ 0 SPACK
  RECPKT TNUM TLEN RPACK
  CASE          ( BASED ON PACKET TYPE)
   <N> OF   ( NAK FOR N+1 => ACK FOR N)
    TNUM @ N@ 1+ - IF KSTATE@ ELSE
       SEOF-ACK THEN ENDOF
   <Y> OF  ( ACKNOWLEDGE GOOD PACKET N)
    TNUM @ N@ - IF KSTATE@ ELSE
       SEOF-ACK THEN ENDOF
   <E> OF PRNTE <A> ENDOF ( ERR-PACKET)
   FALSE OF KSTATE@ ENDOF ( CKSM ERROR)
   ANY OF <A> ENDOF     ( UNRECOGNIZED)
   ENDCASE ;
 -->

( ---
( --- SCREEN # 79 ---
( ---
( SEND BREAK-PACKET <EOT> 070384)

: SBREAK-ACK 0NUMTRY N+   ( RESET ETC.)
  <C> ;         ( SWITCH STATE TO 'C' )

: SBREAK
  NUMTRY+>? IF <A> EXIT THEN   ( ABORT)
  PACKET <B> N@ 0 SPACK
  RECPKT TNUM TLEN RPACK
```

```
  CASE          ( BASED ON PACKET TYPE)
   <N> OF   ( NAK FOR N+1 => ACK FOR N)
    TNUM @ N@ 1+ - IF KSTATE@ ELSE
       SBREAK-ACK THEN ENDOF
   <Y> OF  ( ACKNOWLEDGE GOOD PACKET N)
    TNUM @ N@ - IF KSTATE@ ELSE
       SBREAK-ACK THEN ENDOF
   <E> OF PRNTE <A> ENDOF ( ERR-PACKET)
   FALSE OF KSTATE@ ENDOF ( CKSM ERROR)
   ANY OF <A> ENDOF ( UNRECOGNIZED)
   ENDCASE ;
 -->

( ---
( --- SCREEN # 80 ---
( ---
( SEND: MAJOR ROUTINES 071584)
: SENDSW     ( STATE SWITCHER FOR SEND)
  CLR 1 DWN ." SENDING " <S> KSTATE !
  WAIT                          ( DELAY)
  INITIALIZE TINTSET
  BEGIN DISPL            ( SHOW STATUS)
   KSTATE@ CASE
     <D> OF SDATA  KSTATE! ENDOF
     <F> OF SFILE  KSTATE! ENDOF
     <Z> OF SEOF   KSTATE! ENDOF
     <S> OF SINIT  KSTATE! ENDOF
     <B> OF SBREAK KSTATE! ENDOF
     <C> OF TRUE  EXIT     ENDOF
     <A> OF FALSE EXIT     ENDOF
     ANY OF FALSE EXIT     ENDOF
   ENDCASE
  AGAIN ;

: SEND                   ( SEND A FILE )
  SNAME OUTNAME         ( GET FILE NAME)
  MODOPEN SENDSW            ( SEND FILE)
  IF   MESSG1                 ( SUCCESS)
  ELSE MESSG2                 ( FAILURE)
  THEN FRCLOSE MODCLOSE ABORTIO
  CR PAUSE CLR ; -->


( ---
( --- SCREEN # 81 ---
( ---
( RECEIVE INIT-PACKET 070784)

: RINIT-ACK
 RNUM @ N !              ( SYNC PACKETS)
 PACKET RPARA           ( GET INIT DATA)
 PACKET SPARA          ( SEND INIT DATA)
 PACKET <Y> N@ 6 SPACK
 NUMTRY @ OLDTRY !    ( SAVE TRY COUNT)
 0NUMTRY                ( START NEW ONE)
 N+              ( INCREMENT PACKET NO.)
 <F> ;                     ( RETURN 'F')

: RINIT
```

```
 NUMTRY+>? IF <A> EXIT THEN    ( ABORT)
 PACKET RNUM RLEN RPACK   ( GET PACKET)
 CASE                          ( TYPE?)
  <S> OF RINIT-ACK ENDOF ( INIT-PACKET)
  <E> OF PRNTE <A> ENDOF  ( ERR-PACKET)
  FALSE OF KSTATE@ ENDOF  ( CHKSUM ERR)
  ANY OF <A> ENDOF      ( UNRECOGNIZED)
 ENDCASE ;

 -->


( ---
( --- SCREEN # 82 ---
( ---
( RECEIVE FILE-PACKET ~[PART A] 063084)

: RFILE-S ( IT WAS ANOTHER INIT-PACKET)
  OLDTRY+>? IF <A> EXIT THEN
  RNUM @ N@ 1- = IF     ( PREV PKT # ?)
   PACKET SPARA  ( GET INIT PARAMETERS)
   PACKET <Y> RNUM @ 6 SPACK     ( ACK)
   0NUMTRY           ( RESTART COUNTER)
   KSTATE@               ( KEEP STATE)
  ELSE <A> THEN ;              ( ABORT)

: RFILE-Z  ( IT WAS AN END-OF-FILE PKT)
  OLDTRY+>? IF <A> EXIT THEN
  RNUM @ N@ 1- = IF        ( PREV PKT?)
   0 <Y> RNUM @ 0 SPACK          ( ACK)
   0NUMTRY
   KSTATE@
  ELSE <A> THEN ;              ( ABORT)

 -->

( ---
( --- SCREEN # 83 ---
( ---
( RECEIVE FILE-PACKET ~[PART B] 071584)

: RFILE-F  ( IT WAS A FILE-NAME PACKET)
  RNUM @ N@ = IF  ( CORRECT PACKET # ?)
   RLEN @ GETFIL     ( -YES: OPEN FILE)
   2 DWN WNAME$ TYPE$
   0 <Y> N@ 0 SPACK              ( ACK)
   NUMTRY @ OLDTRY !  ( SAVE TRY COUNT)
   0NUMTRY             ( RESTART COUNT)
   N+                ( INCREMENT PKT #)
   <D>                ( GET-DATA STATE)
  ELSE <A> THEN ;              ( ABORT)

: RFILE-B      ( IT WAS A BREAK-PACKET)
  RNUM @ N@ = IF   ( CORRECT PACKET #?)
   0 <Y> N@ 0 SPACK        ( -YES: ACK)
   <C>         ( RETURN 'C' = COMPLETE)
  ELSE
   <A>              ( OTHERWISE, ABORT)
  THEN ;
```

```
 -->


( ---
( --- SCREEN # 84 ---
( ---
( RECEIVE FILE-PACKET ~[PART C] 070384)

: RFILE
  NUMTRY+>? IF
   <A> EXIT                ( ABORT IF)
  THEN
  PACKET RNUM RLEN RPACK  ( GET PACKET)
  CASE                  ( PACKET TYPE?)
   <S> OF RFILE-S ENDOF  ( INIT-PACKET)
   <Z> OF RFILE-Z ENDOF   ( EOF-PACKET)
   <F> OF RFILE-F ENDOF  ( FILE-PACKET)
   <B> OF RFILE-B ENDOF ( BREAK-PACKET)
   <E> OF PRNTE <A> ENDOF ( ERR-PACKET)
   FALSE OF KSTATE@ ENDOF ( CHKSUM ERR)
   ANY OF <A> ENDOF      ( UNRECOGNIZED)
  ENDCASE ;

 -->

( ---
( --- SCREEN # 85 ---
( ---
( RECEIVE DATA-PACKET ~[PART A] 063084)

: RDATA-D       ( IT WAS A DATA PACKET)
  RNUM @ N@ = IF       ( RIGHT PKT # ?)
   RLEN @ PACKET BUFEMP ( BUFFER->FILE)
   0 <Y> N@ 0 SPACK             ( ACK)
   NUMTRY @ OLDTRY !  ( SAVE TRY COUNT)
   0NUMTRY                    ( RESET)
   N+                     ( INCR PKT #)
   <D>                    ( MORE DATA?)
  ELSE
   OLDTRY+>? IF <A> EXIT THEN
   RNUM @ N@ 1- = IF    ( PREV PKT # ?)
    PACKET SPARA
    PACKET <Y> RNUM @ 6 SPACK    ( ACK)
    0NUMTRY
    KSTATE@
   ELSE <A> THEN ( SORRY, WRONG NUMBER)
  THEN ;

 -->


( ---
( --- SCREEN # 86 ---
( ---
( RECEIVE DATA-PACKET ~[PART B] 072484)

: RDATA-F       ( IT WAS A FILE PACKET)
  OLDTRY+>? IF <A> EXIT THEN
```

```
  RNUM @ N@ 1- = IF     ( PREV PKT # ?)
   0 <Y> N@ 0 SPACK              ( ACK)
   0NUMTRY                    ( RESET)
   KSTATE@                ( KEEP STATE)
  ELSE <A> THEN ;

: RDATA-Z     ( IT WAS AN <EOF> PACKET)
  RNUM @ N@ = IF       ( RIGHT PKT # ?)
   0 <Y> N@ 0 SPACK              ( ACK)
   FWCLOSE           ( CLOSE DISK FILE)
   NUMTRY @ OLDTRY !  ( SAVE TRY COUNT)
   0NUMTRY                    ( RESET)
   N+                      ( INCR PKT #)
   0 WNAME$ !   ( FORGET OLD FILE NAME)
   <F>                 ( ANOTHER FILE?)
  ELSE <A> THEN ;

 -->

( ---
( --- SCREEN # 87 ---
( ---
( RECEIVE DATA-PACKET ~[PART C] 070384)

: RDATA
  NUMTRY+>? IF <A> EXIT THEN
  PACKET RNUM RLEN RPACK  ( GET PACKET)
  CASE
   <D> OF RDATA-D ENDOF  ( DATA-PACKET)
   <F> OF RDATA-F ENDOF  ( FILE-PACKET)
   <Z> OF RDATA-Z ENDOF   ( EOF-PACKET)
   <E> OF PRNTE <A> ENDOF ( ERR-PACKET)
   FALSE OF KSTATE@ ENDOF ( CHKSUM ERR)
   ANY OF <A> ENDOF     ( UNRECOGNIZED)
  ENDCASE ;

 -->


( ---
( --- SCREEN # 88 ---
( ---
( RECEIVE: MAJOR ROUTINES 071584)

: RECSW   ( STATE SWITCHER FOR RECEIVE)
  CLR 1 DWN ." RECEIVING " <R> KSTATE!
  INITIALIZE TINTSET
  BEGIN DISPL            ( SHOW STATUS)
   KSTATE@ CASE
    <D> OF RDATA KSTATE! ENDOF  ( DATA)
    <F> OF RFILE KSTATE! ENDOF ( FNAME)
    <R> OF RINIT KSTATE! ENDOF  ( INIT)
    <C> OF TRUE  EXIT    ENDOF  ( DONE)
    <A> OF FALSE EXIT    ENDOF ( ABORT)
    ANY OF FALSE EXIT    ENDOF ( ?????)
   ENDCASE
  AGAIN ;

: RECV               ( RECEIVE A FILE)
```

```
  WNAME$ GNAME         ( GET FILE NAME)
  MODOPEN RECSW          ( ACCEPT FILE)
  IF   MESSG1                ( SUCCESS)
  ELSE MESSG2                ( FAILURE)
  THEN WCLOSE MODCLOSE ABORTIO
  CR PAUSE CLR ; -->



( ---
( --- SCREEN # 89 ---
( ---
( SERVER COMMAND: GET 071584)
: RSERV
  FNAME$ COUNT <R> 0 ROT SPACK RINIT ;

: GETSW
  CLR 1 DWN ." RECEIVING" <R> KSTATE!
  INITIALIZE TINTSET
  BEGIN DISPL            ( SHOW STATUS)
   KSTATE@ CASE
    <R> OF RSERV KSTATE! ENDOF
    <F> OF RFILE KSTATE! ENDOF
    <D> OF RDATA KSTATE! ENDOF
    <C> OF TRUE EXIT     ENDOF
    <A> OF FALSE EXIT    ENDOF
    ANY OF FALSE EXIT    ENDOF
   ENDCASE
  AGAIN ;

: GETF    ( GET FILENAME FROM KEYBOARD)
  SNAME OUTNAME RNAME$ WNAME$ TO$
  MODOPEN GETSW ( GET FILE FROM SERVER)
  IF   MESSG1                ( SUCCESS)
  ELSE MESSG2                ( FAILURE)
  THEN WCLOSE MODCLOSE ABORTIO
  CR PAUSE CLR ; -->


( ---
( --- SCREEN # 90 ---
( ---
( SERVER COMMANDS 071584)
<L> VAR L
: LOGOUT
  MODOPEN CLR TINTSET
  L <G> 0 1 SPACK  ( SEND A 'GL' PKT)
  RECPKT TNUM TLEN RPACK
  CASE
   <E> OF PRNTE FALSE ENDOF
   <Y> OF TRUE ENDOF
   ANY OF FALSE ENDOF
  ENDCASE
  CASE
   TRUE OF 5 DWN ." LOGGED OUT" ENDOF
   FALSE OF MESSG2 ENDOF
  ENDCASE
  MODCLOSE CLOSCHN ABORTIO
```

```
  CR PAUSE CLR ;
 -->



( ---
( --- SCREEN # 91 ---
( ---
( KERMIT LOCAL COMMANDS 071584)

: KQUIT CLOSCHN ABORTIO ABORT ;

: XDC ~[COMPILE] DC ;

: HELP CLR
  99 96 DO CLR
     24 0 DO I J .LINE CR LOOP
     PAUSE CLR
  LOOP ;

: NEW ( MOVES KERMIT FILES TO NEW DISK)
  ." INSERT, THEN PUSH ANY KEY:"
  CR ."  SOURCE DISK"
  KEYIN DROP
  99 95 DO I BLOCK UPDATE LOOP
  CR ."  DESTINATION DISK"
  KEYIN DROP FLUSH
  ~[COMPILE] SAVESYSTEM ;

 -->



( ---
( --- SCREEN # 92 ---
( ---
( KERMIT COMMAND TABLE 071784)

14 CMDTBL CKERMIT
   CONNECT   CONNECT    SEND    SEND
   RECEIVE   RECV       GET     GETF
   BYE       LOGOUT     HELP    HELP
   EXIT      BASIC      QUIT    KQUIT
   SET       SET        SHOW    SHOW
   SAVE      STORE      RESTORE RECALL
   DIRECTORY DIR        DISK    XDC
   NEW       NEW
 -->



( ---
( --- SCREEN # 93 ---
( ---
( KERMIT INTERACTS WITH USER 111384)

: KERMIT                ( ENDLESS LOOP)
  0 BLK ! -1 WARNING !   ( FOR TURNKEY)
  CLR 1 DWN
  ." KERMIT-C64/V1.5"
```

```
  CR ." R. DETENBECK"
  CR ." DEPT. OF PHYSICS"
  CR ." UNIVERSITY OF VERMONT"
  10 DWN
  BEGIN
    CR ." KERMIT-C64> " QUERY CR
    CKERMIT XSETUP
    DO
      I () HERE =$ IF
        CKERMIT I () CMDEXE UNKERR
      THEN
    LOOP DROP TSTKERR
  AGAIN ;



( ---
( --- SCREEN # 94 ---
( ---
( MOVE SCREENS 21-92 120284)

EMPTY-BUFFERS
30 BUFFERS
: MOVEIT ( N1 N2 --- )
  SWAP 1- SWAP
  BEGIN
  CR ." INSERT SOURCE DISK, TYPE 'A' "
  KEYIN DUP EMIT 65 = UNTIL
  DO
    CR ." COPY " I . ."  TO " I 1+ .
    I DUP 1+ COPY
  -1 +LOOP
  BEGIN
  CR ." INSERT DEST DISK, TYPE 'B' "
  KEYIN DUP EMIT 66 = UNTIL
  FLUSH ;

80 92 MOVEIT
67 79 MOVEIT
54 66 MOVEIT
41 53 MOVEIT
28 40 MOVEIT
21 27 MOVEIT



( ---
( --- SCREEN # 95 ---
( ---
01 00 00 00 00 00 00 06 01 14
96 00 00 10 13 35
96 00 00 10 13 35



( ---
( --- SCREEN # 96 ---
( ---
           KERMIT-C64, V1.5
```

```
EQUIPMENT: COMMODORE C64, 1541, 1600
SOURCE LANGUAGE: FORTH
FILE TYPES HANDLED:

 TYPE...INTERNAL FORM..TRANSMITTED FORM

 TEXT   7-BIT CBMASCII    7-BIT ASCII
        (SEQUENTIAL)      TRANSLATION*
 UC     ~[ SAME, BUT UPPERCASE ONLY ]*
 ASCII  7-BIT ASCII       7-BIT ASCII
        (SEQUENTIAL)     LITERAL COPY**
 HEX    8-BIT BINARY   ENCODED AS ASCII
        (PROGRAM)         HEX NIBBLES
 BINARY 8-BIT BINARY      8-BIT BINARY
        (PROGRAM)        LITERAL COPY**

* CBM <-> ASCII CONVERSIONS:
  BKSLSH/TAB/FF -> CHR$(221/220/219)
  ON INPUT; REVERSED ON OUTPUT.
**KERMIT PROTOCOL: QUOTE CTRL CHAR, BUT
  NO 8TH-BIT QUOTE OR REPEAT QUOTE. SEQ
  FILES: CR INTERNAL, CRLF TRANSMITTED.



( ---
( --- SCREEN # 97 ---
( ---
      PRIMARY KERMIT COMMANDS (V1.5)

BYE         = LOGOUT FROM A SERVER.
CONNECT     = BECOME A DUMB TERMINAL.
DIRECTORY   = SHOW 1541 DISK DIRECTORY.
DISK "XX"   = SEND COMMAND XX TO 1541
              DISK ON CHANNEL 15.
EXIT        = RETURN TO BASIC.
GET "XX"    = RECEIVE THE FILE XX
              FROM A SERVER.
HELP        = DISPLAY THESE SCREENS.
NEW "XX"    = PREPARE NEW KERMIT DISK.
              SAVE KERMIT AS XX.
SEND "XX"   = SEND THE FILE NAMED XX.
RECEIVE     = RECEIVE THE FILE WITH THE
              NAME SPECIFIED BY SENDER.
RECEIVE "XX"= RECEIVE FILE AND STORE
              WITH NAME XX.
RESTORE     = SET KERMIT PARAMETERS
              FROM FILE 'SCR95'.
SAVE        = SAVE KERMIT PARAMETERS
              IN FILE 'SCR95'.
SET         = ALTER KERMIT PARAMETERS.
SHOW        = SHOW KERMIT PARAMETERS.

( ---
( --- SCREEN # 98 ---
( ---
TERMINAL INFORMATION (V1.5):
 SETUP HOST COMPUTER FOR DUMB TERMINAL
```

WHICH ACCEPTS BACKSPACE (E.G., ADM3A).

SPECIAL FUNCTION KEYS IN TERMINAL MODE:
DEL => 127 = RUBOUT/DELETE
F1 => 17 = CTRL-Q = DC1 = XON
F3 => 19 = CTRL-S = DC3 = XOFF
F5 =>   8 = CTRL-H = BACKSPACE
F7 => 27 = CTRL-] = ESC
F2 => 18 = CTRL-R = DC2
F4 => 20 = CTRL-T = DC4
F6 => 10 = CTRL-J = LF
F8 => KERMIT TERMINAL-ESCAPE CHARACTER

OPTIONS WITH F8: (STRIKE KEY AFTER F8)
F8-U => SET SCREEN TO UPPERCASE MODE
F8-L => SET SCREEN TO LC/UC MODE
F8-B => SEND 'BREAK' TO HOST
F8-C => DISCONNECT, RETURN TO KERMIT
F8-X => SAME AS F8-C
F8-T => SHOW TERMINAL PARAMETERS
F8-S => SHOW KERMIT PARAMETERS
F8-? => SHOW THESE OPTIONS