0000000  LIST  X
0000000  NAME  VTEXT  MACRO
0000006  X.J  MACRO  LOC
0000018  LOAD  12/13/82,13/LOC,30/1
0000024  EN7  &1
0000026  JD  F80
0000029  EN31
0000034  X.J  MACRO  LOC,PTh anxious
0000046  LOAD  12/13/82,13/LOC,13/1,13/2
0000052  EN7  &1
0000054  JD  F80
0000057  EN31
0000064  CALL  MACRO  LOC
0000070  EN7  &1
0000072  JD  F80
0000075  EN31
0000084  MACRO  MACRO  NAME
0000088  BUS  &
0000090  NAME  EN9
0000092  LOAD  1/1,20/C,NAME,30/CX,MAST
0000098  EN31
0000106  MACRO  MACRO  NAME
0000110  BUS  &
0000112  NAME  EN9
0000114  LOAD  1/1,20/C,NAME,30/CX,MAST
0000118  EN31
0000128  MACRO  MACRO  NAME
0000132  TEC  &
0000134  LOAD  65/6
0000136  TEC  &
0000138  LOAD  55/5
0000142  TEC  &
0000144  LOAD  55/5
0000148  TEC  &
MACRO NAME+X*FLAG*CMAD*LASTP1*RO

MACRO FOR CLIST INDEX OF OBJECT TO RETURN TO CALLER

MACRO FOR CLIST INDEX OF OBJECT TO BE OBTAINED AT TIME OF THIS SUBPROCESS CONSTRUCTION

SUB PROCESS DESCRIPTOR

DATA 1
DATA 3
DATA HLOC-LINE
DATA HLOC-READ
DATA HLOC-LINEI
DATA HLOC-LINE
DATA HLEF-LINESF
DATA 2
DATA 16
ITEMS FL
ITEMS MAIN
ITEMS CLST07
ITEMS SORKZ
TRYBUF 8587 180 (USED BY GRAYCOE)

REGAREA 8587 94a (HOLDS USERS REGISTERS DURING CALL)

ITEMS PARAM*1

FAKENES 8587 1

DATA ERROR DURING ERROR CALL HANDLING

ERRORS 8587 94a REGISTER SAVE AREA DURING ERROR CALL

TOPNFW 8587 3 TOP OF STACK DURING ERROR CALL

END OF SCRATCH AREA

SCRSTY 8587 9

FIXED ID LISTS

READ

ITEMS CX,CXREF
ITEMS REGAREA

READ

ITEMS CX,CXREF
ITEMS REGAREA

RETURN

RETURN

RETURN

TITLE MAIN-CODE

THIS IS MY INTERFACE TO JIM GRAY'S GRAYCOE

THE CALLS ON GRAYCOE WERE CONSTRUCTED
TO IMIT THE OLD ASCII XTTEXT FILE PROVIDED
BY JIM GRAY

THE EDIT CALL HAS MODELED ON CODE IN THE OLD READ
IF TYPEOUT 3
THEN TTYOUT 4
END IF

SUPPORT E_Support

E.P. 0 X 1 E.P. 1
E.P. 1 X 1 E.P. 1,1
E.P. 1,1 MYCAP UPPER
TTY0 A/1
TTY1 TTY1

E.P. 1,2 MYCAP UPPER
TTY0 A/2
TTY1 TTY1

E.P. 2,1

THEN CUBE HANDLES GETTING A LINE FROM TTY

IF TYPEIN 1
THEN TTYOUT 2
IF TYPEIN 2
THEN TTYOUT 2
END IF
END IF

THEN CUBE HANDLES SENDING A LINE TO TTY

OUTPUT A CHARACTER
EDIT A LINE

TTYOUT 3
TTYOUT 4

END INPUT
THIS CODE SENDS A SINGLE CHAR TO TTY

TYPEOUT CALL PARANO
BY-CHAR X1
GET-TTY TTYBUFF
SET-POSITION E111
END PUTTTY
EJECT

THIS CODE ADAPTED FROM THE OLD EDIT CODE
( PROBABLY WRITTEN BY JIM GRAY )

THE FOLLOWING COMMENTS ARE QUOTED FROM THE
HEADER OF THE EDIT CODE IN THE OLD READ:

THE LINE IS 1ST CHECKED FOR VALIDITY
IT IS TRUNCATED AT 1ST NON GRAPHIC
CHARACTER AP AT 80 CHARACTERS, WHICH EVER
COMES FIRST.

UNHAPPILY IT DOESNT DO BLANK COMPRESSION

THE FOLLOWING COMMENT WAS ALSO THERE, BUT I DONT
KNOW ITS SIGNIFICANCE

UNLIKE GETLINE, IT UPDATES THE STRING
DESCRIPTOR IN THE CALLERS SPACE SO THAT
IT IS CORRECT.

TTYENT 811
PARA=201
9X5 X1
9X4 TTYBUFF+OLD=3+3
9X3 1: **SET 4,STEP
8X3 

+ 9X1 A1+1
9X6 X1
9X6 R1=M
9X7 R2=M
9T R3=M-1

SET READY TO CALL GETC7
TTYBUFF+OLD+1 SET A,LENGTH
9X4 82: NOTE THE FINAL PUT OF A CR WILL SET LENGTH
9X3 TOOLS R,ERROR SET R,RETURN
+ 9X7 8:1 SET R,RETURN
j0 GETC7 GET A CHARACTER
+ 9X1 X1=05 IS IT GRAPHIC
9G X1=GETC7 IF SO, GET ANOTHER
9LINE TOO LONG OR NON GRAPHIC ENCOUNTERED

TOOLS 8X4 X4=1 STEP BACK ONE
ERROR CALL AREA

PANIC EQU 1
ERRORCLASS EQU 4
ERRORCLASS+1 EQU ERRORCLASS+1
ERROR X: ESAVE  SAVE REGISTERS
    SETBCK  RE SET ERROR SELECTION MASK
    ERRORCLASS CHECK ERROR CLASS FOR PANIC
    X1 = ERRORCLASS
    70  X1 + INTERRUPT
    PNSTK  READ TOP OF STACK (IN CASE UNWANTED ERROR)
    X1  ERESTOR  RESTORE REGISTERS
    X4  EFA TH  NOW CONVERT ERROR TO F.RETURN

ESAVE MYCAP SAV E REGS

ERESTOR MYCAP RESTORE REGS

ESAVE MYCAP  SAVE REGS

NOSTK MYCAP DISCARD DATA

FARTH MYCAP  F.RETURN

IF HAVE BEEN INTERRUPTED

INTERRUPT X1  FIXDC  SETS P.COUNTER TO INTERRUPT CODE
    X1  ERESTOR  RETURN TO INTERRUPTED VERSION
<table>
<thead>
<tr>
<th>INTCODE</th>
<th>SAV</th>
<th>ERRNUM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X1, INTCODE</td>
<td>&quot;MILD PANIC&quot;</td>
</tr>
<tr>
<td></td>
<td>X1</td>
<td>&quot;MAJOR PANIC&quot;</td>
</tr>
</tbody>
</table>

| INTCODE | Y1       | INTERR | "MILD PANIC" |

<table>
<thead>
<tr>
<th>FIXINC</th>
<th>HVACP</th>
<th>MOORE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ITENS</td>
<td>CX, INCL</td>
</tr>
<tr>
<td></td>
<td>ITENS</td>
<td>INTCODE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INTERP</th>
<th>HVACP</th>
<th>USERED</th>
<th>&quot;MILD PANIC&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ITENS</td>
<td>E, PANIC</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HEADS</th>
<th>ITENS</th>
<th>CX, DCS</th>
<th>&quot;MAJOR PANIC&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TITLE</td>
<td>V1SC SUBROUTINES</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CRASH</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ERR</th>
<th>X1, SAVE</th>
<th>JD, DIE</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DEBUG</th>
<th>X1, SAVE</th>
<th>JD, BY</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>RESTORE</th>
<th>HVACP</th>
<th>RESTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ITENS</td>
<td>REGAREA</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HEAD</th>
<th>ITENS</th>
<th>CX, HEAD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SAVE</td>
<td>HVACP, SAVE</td>
</tr>
</tbody>
</table>

| DIE     | X1, RETAIN, UP |

<table>
<thead>
<tr>
<th>STOP</th>
<th>PS, EJECT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TITLE</td>
</tr>
</tbody>
</table>


**WHAT FOLLOWS IS COPIED VIA THE EDITOR FROM**
GRAYCO, XTEXT ON OCT 22, 1976

**FIRST WE FIX UP XJ FOR GRAYCO**

<table>
<thead>
<tr>
<th>XJ</th>
<th>MACRO</th>
<th>LOC</th>
<th>12/12/81</th>
<th>10/LOC</th>
</tr>
</thead>
</table>

**NOW WE FIX UP CALLUMS FOR NOMPASS**

<table>
<thead>
<tr>
<th>COL</th>
<th>FUNCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRINT</td>
<td>COUNT</td>
</tr>
<tr>
<td>FINY</td>
<td>DIF</td>
</tr>
<tr>
<td>LNMK</td>
<td>DIFF</td>
</tr>
<tr>
<td>ITY</td>
<td>HASECT</td>
</tr>
<tr>
<td>CHAP</td>
<td>DIFF</td>
</tr>
<tr>
<td>EP360</td>
<td>HASECT</td>
</tr>
<tr>
<td>COUNT</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>LENGTH</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>SCRATCH</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>STEP</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>OUT</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>RETURN</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>DIFF</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>OUTLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>HASECT</td>
<td>INLEN</td>
</tr>
<tr>
<td>OUTLEN</td>
<td>INLEN</td>
</tr>
<tr>
<td>INLEN</td>
<td>INLEN</td>
</tr>
</tbody>
</table>
```
WRITEOUT 1
OUTBASE 1
OUTCHKS 1
OUTTEXT 1
BREAKOUT 1
ICFLAG 1
TABS 1
OLD 2
NEW 3
IF -ABS.RASEPT,1
ELSE 0
IF -ABS.RASEPT,1
END RASEPT
LOC *0.DIFF
MAX EQN 47
ALNC EQN 48
SKPANCH OUT 49
ESCHAM EQN 41
LEFDTR EQN 41
RTXTRD EQN 41
HACKHID EQN 49
HELCHID EQN 4178
LJ EQN 1439
CD EQN 1439
ESCC EQN 1739
ESC EQN 1739
TITLE PP AND CP SIGNALS
ONE EQN 2
HEAD EQN CX.DEAU MODIFIED
WRITE EQN CX.WRITE PRIM
SEND EQN CX.SEND GRAYCODE
HANG EQN CX.HANG ABSOLUTE

*****FILE CAPABILITY INDEX COMES TO IN 112
*****THE INDEX OF THE CP TO PP EVENT CHANNEL IS 112
*****THE INDEX OF THE PP TO CP CHANNEL IS 112

MODEM1 EQN 151,000 MORE OUTPUT
MODEM2 EQN 351,000 MORE INPUT
PULSE EQN 251,000
RESUME EQN 361,000 RESUME HANG ECHO
TALEF EQN 461,000 NO BREAK
TALESP EQN 461,01P ALL BREAK
TALEPS OUT 461,02R BREAK ON NON GRAPHIC
TALEPS EQN 461,03R BREAK ON NON ALPHA MERIC
TALEPS EQN 461,03R BREAK ON NON NUMERIC
MOREJUMP EQN 461,00R MORE DATA IN INPUT BUFFER
SYSCALL *1020 ADD
+ 30_SCRATCH RITY+AD
+ 0 SCRATCH
- 0 PEN 311,40/8
END SYSCALL
SENDỰN 208B PAMAM
SY,OUT PAMAM
```
SA,OUT  R,ITY*ACTUAL
SYSCALL  SENDVENT
END  SENDVENT

END  SET  4001A  END OF WORD AND OF MESSAGE
END  SET  4002A  SET OUTPUT ACTIVE OFF IF PO
END  SET  4003A  END OF WORD NEXT CHARACTER
IS IN NEXT WORD

DEFERED  SET  4007A  ECHO'S DEFERRED BEYOND THIS
NULL  SET  4000A  NULL CHARACTER
LOST  SET  4009A  LOST SOME CHARACTERS HERE
VENTOR  SET  4005A  REENTERED HARD ECHO ON REQU
VENTOUT  SET  4006A  WENT OUT OF ECHO MODE ON RE
PURGED  SET  6210A  PURGED CM INPUT BUFFER HERE

TITLE  INITIALIZE
EN  6/4/700311000402801490B

TTY,0  SA,OUT  1
SV,OUT  9,ONE
SA,OUT  R1,SOFT.ECHO
SV,OUT  96
SA,OUT  X1+NOUTPUNT  INITIALIZE OUT COUNT
SA,OUT  RTNEEDEM
SA,OUT  61,TAFTAKAR
SA,OUT  61,ICF46
SV,OUT  A,ONE
SA,OUT  R1,FORCE
SA,OUT  X1,NOUTUT4
SA,OUT  X1,NEUTETN4
SA,OUT  R1,NOUTGETE2
SA,OUT  R1,NOUTGET4
SA,OUT  X1,NOUTPUT4
SA,OUT  R1,NOLD
SA,OUT  R1,NOLD
SA,OUT  SCRATCH
SA,OUT  SCRATCH
SA,OUT  X1,TARAS
SA,OUT  X1,OUT4,ONE
SA,OUT  R1,NOLD
SA,OUT  R1,NOLD
SA,OUT  R1,NOLD
SA,OUT  X1,NEW3,11
SA,OUT  R1,NEW
SA,OUT  X1,HE,ONE
SA,OUT  X2,OUT
SV,OUT  X1,OUT
GETCH2: \( \text{COUNT} \) \( \text{LENGTH} \) \( \text{CHAR} \) \( \text{SCRATCH} \) \( \text{OUT} \)

GETCH7: \( \text{COUNT} \) \( \text{LENGTH} \) \( \text{CHAR} \) \( \text{SCRATCH} \) \( \text{FAULT} \)

GETCH1: \( \text{SCRATCH} \) \( \text{OUT} \) \( \text{CHAR} \) \( \text{SCRATCH} \) \( \text{OUT} \) \( \text{CHAR} \) \( \text{SCRATCH} \) \( \text{OUT} \)

TITLE \( \text{GET N 7 BIT CHARACTER FROM A STRING} \)

EFFECT

GETS THE \( x \)_LENGTH+1 CHARACTER FROM THE STRING
STARTING AT \( c(x) \)_LENGTH+1 AND RETURNS IT IF
ADJUSTED WITH LEADING ZEROS IN \( x \)_CHAR
INCREMENTS \( x \)_COUNT AND \( x \)_LENGTH BY \( r \)_STEP
RETURNS \( t \)_FAULTR\_RETURN IF \( x \)_LENGTH IS GREATER THAN MAX OR LESS THAN
THEN \( t \)_RETURN IS THRU \( r \)_ERROR AND NO CHARACTER
IS RETURNED
UPDATES \( x \)_CHAR \( x \)_LENGTH \( x \)_COUNT
DESTROYS \( x \)_SCRATCH \( x \)_SCRATCH \( x \)_SCRATCH \( x \)_OUT

TITLE \( \text{PUT A CHARACTER INTO THE TTY BUFFER} \)

EFFECT

PUT CHAR THE 8 BIT RIGHT ADJUSTED CHARACTER TO BE
PUT TTY A POINTER TO THE BASE OF THE TTY BUFFER

PUT HANDS WHEN THE BUFFER IS FULL
PUT WAITS UNTIL THE BUFFER WAS FORMERLY EMPTY
ALL THIS IS ACTUALLY DONE BY FLUSH WHICH IS AN INTERRUP
OF OUT
FOUR OUT OF FIVE TIMES PUT TAKES TENTH MICROSECONDS
UNLESS FLUSHING IS REQUIRED PUT TAKES THIRTY MICROSECONDS
PUT THE LAST CHARACTER OF EACH BUFFER WORD
IF FLOWING IS REQUIRED IT MAY TAKE FOR EVER

OUT DESTROY X, SCRATCH X, OUT
A, SCRATCH A, OUT
THE ONLY BIT PATTERN THAT CANNOT BE SET IS 77778
INITIALIZATION AND FLUSHING MUST FORCE THE IMAGE OF T
PUT DESTRO Y TO 777777777777777777778

PUTTTYT Y, OUT
PUTTTY SC RATCH X, CHAP & A
PUTTTY OUT X, OUT + SCRATCH
PUTTTY SC RATCH EQ
PUTTTY OUT X, OUT + SC RATCH, #1
PUTTTY SC RATCH SP
PUTTTY OUT X, OUT + X, SCRATCH
PUTTTY OUT

PUTTTY X, OUT
PUTTTY SC RATCH X, TTY + NEED EOM
PUTTTY OUT X, CHAP & A, OUT
PUTTTY SC RATCH + SCRATCH + TTY
PUTTTY OUT X, SCRATCH, E7
PUTTTY OUT X, SCRATCH + TTY + OUTPUT
PUTTTY SC RATCH + SCRATCH + TTY
PUTTTY OUT X, SCRATCH + TTY + OUTPUT
PUTTTY SC RATCH + SCRATCH + TTY
PUTTTY OUT X, SCRATCH, E7
PUTTTY SC RATCH + SCRATCH + TTY
PUTTTY OUT X, SCRATCH, E7
PUTTTY SC RATCH + SCRATCH
SKIPPY
PUTTTY OUT X, SCRATCH + SKIPPY
PUTTTY OUT
SKIPPY
PUTTTY OUT X, SCRATCH + SKIPPY
PUTTTY OUT X, SCRATCH + SKIPPY
PUTTTY OUT X, SCRATCH
PUTTTY OUT X, SCRATCH + SKIPPY
PUTTTY OUT X, SCRATCH + TTY + OUTLIM
PUTTTY TTY + OUTLIM
PUTTTY OUT X, SCRATCH + TTY + OUTLIM
PUTTTY OUT X, SCRATCH + TTY + OUTLIM

PUTTTY Y, OUT
PUTTTY SC RATCH X, CHAP & A
PUTTTY OUT X, OUT + SCRATCH
PUTTTY SC RATCH EQ
PUTTTY OUT X, OUT + SC RATCH, #1
PUTTTY SC RATCH SP
PUTTTY OUT X, OUT + X, SCRATCH
PUTTTY OUT

PUTTTY X, OUT
PUTTTY SC RATCH X, TTY + NEED EOM
PUTTTY OUT X, CHAP & A, OUT
PUTTTY SC RATCH + SCRATCH + TTY
PUTTTY OUT X, SCRATCH, E7
PUTTTY OUT X, SCRATCH + TTY + OUTPUT
PUTTTY SC RATCH + SCRATCH + TTY
PUTTTY OUT X, SCRATCH + TTY + OUTPUT
PUTTTY SC RATCH + SCRATCH + TTY
PUTTTY OUT X, SCRATCH, E7
PUTTTY SC RATCH + SCRATCH + TTY
PUTTTY OUT X, SCRATCH, E7
PUTTTY SC RATCH + SCRATCH
SKIPPY
PUTTTY OUT X, SCRATCH + SKIPPY
PUTTTY OUT
SKIPPY
PUTTTY OUT X, SCRATCH + SKIPPY
PUTTTY OUT X, SCRATCH + SKIPPY
PUTTTY OUT X, SCRATCH
PUTTTY OUT X, SCRATCH + SKIPPY
PUTTTY OUT X, SCRATCH + TTY + OUTLIM
PUTTTY TTY + OUTLIM
PUTTTY OUT X, SCRATCH + TTY + OUTLIM
PUTTTY OUT X, SCRATCH + TTY + OUTLIM

TRANSLATE FROM INTERNAL TO

PICK UP THE BUFFER WORD
MAKE ROOM FOR NEXT CHARACTER
RESTORE THE BUFFER WORD
IF THERE ARE MORE SLOTS RET
STEP OUT COUNT FOR SWAPPING

REPLACE

RETURN

SKIPPY

PLACE

REPLACE

PLACE

REPLACE

PLACE

REPLACE

PLACE
The 12 HT character might be adjusted with leading zeros to be gotten.

TTY: a pointer to the base of the virtual TTY buffer.

X*SCRATCH working registers.

X*OUT

If the virtual buffer is empty, the TTY is called when the
data is forced into the hard echo. Then the virtual
TTY buffer is refreshed from the actual TTY buffer by
forcing the buffer until the actual buffer is non-empty. An
flushes the actual buffer into the virtual buffer.

X*SCRATCH does not interpret any of the characters.

X*CHAP, A*SCRATCH, A*OUT

GETTTY

CALCHAP 0.TTY=INGET

A*SCRATCH 0.TTY=INPUT.

X*SCRATCH Y*SCRATCH=X*CHAR

X*SCRATCH=GETTTY

Force the PP back into hard echo.

Flush output.

Read input.

And return thru R*RETURN.

PuSH

INFORcF.

G*SCRATCH 0.TTY=NEEDEOF.

Y*RETURN

Y*RETURN 13.

READIN

SYSCALL 0.INPUT.

GETTTY

X*SCRATCH 0.TTY=INGET

Y*OUT

X*SCRATCH 0.TTY=INPUT

X*SCRATCH Y*SCRATCH=X*OUT

X*SCRATCH=READIN.

X*SCRATCH 0.TTY=SOFTECHO

79.

X*SCRATCH=READIN.

SENDVT BEGIND.

X*RETURN

SYSCALL HANGIN.

X*RETURN

X*RETURN

X*OUT

Y*RETURN

Y*RETURN

SYSCALL A*READIN.

SENDVT Y*RETURN

GETTTY: CALCHAP 0.TTY=INGET.

X*SCRATCH X*CHAR=TTY.

PICK UP THE ACTIVE WORD IF

X*CHAR=TTY.

SHIFT CHARACTER INTO LOWER.

X*OUT 77778.

12 BITS AND SET UP MASK.
ECHO STRATEGY...

ECHO ANY NONBREAK CHARACTER WHEN SOFT ECHO IS ON
NEVER ECHO A BREAK CHARACTER
THE CALLING PROGRAM MAY ECHO A BREAK CHARACTER OR ECH
A TRANSLATION OF IT.
IN ANY CASE BEFORE GETTING A CHARACTER SEE IF THE TTY
MUFFED IS EMPTY, IF SO MIX IF THE SOFTECHO FLAG IS UP
CALL THE TRAPFORCER.
THE TRAPFORCER DOES THE FOLLOWING THINGS
(a) IF NEEDTO THEN PUTCOPY (EOM)
(b) IF TTYOUT NONEMPTY THEN FLUSH
(c) SEND A RESUME ECHO EVENT AT THE DP
(d) RETURN

ENTFDN THRU RETURN WITH ALENGTH SET
SETS ITER RETURN BREAK
88:COUNT 0
LOOP
PY:SCRATCH X:LENGTH-MAX
P1:ST Y:SCRATCH+SETS1
SR:RETURN BREAK
JP:PUTCOPY
BREAK1
P1:OUT X:CHAR+140B
Y:CHAR = Y:OUT X:CHAR
SR:RETURN BREAK
JP:PUTCOPY
BREAK
P1:SCRATCH X:CHAR
SR:SCRATCH A:TTY BREAK+1
TEST FOR BREAK CHARACTER
BREAK1
SR:SCRATCH A:TTY+1
X:SCRATCH = X:SCRATCH+1
SR:SCRATCH A:TTY+1
X:SCRATCH = X:SCRATCH+1
GE
Y:SCRATCH+POP
SR:COUNT B:COUNT+1
SR:SCRATCH A:TTY+1
88:SOFTECHO IS ON ECHO IT
TP Y:SCRATCH LOOP
SR:RETURN LOOP
SP:PUTCOPY
GETST
P1:CHAR = X:CHAR
PAP
TITLE PUT A STRING TO THE TELETEYPE
LEFT
P1: THE STRING WITH DESCRIPTOR C(A:LENGTH+1) (A+LENGTH) TO THE
POINTER TO R:TTY IF THE STRING ENDS IN A OR THE AD
LE, IF THE LENGTH OF THE STRING EXCEEDS MAX TRUNCATE
AT THAT POINT, PUTFL Flushes THE OUTPUT BUFFER
RETURN THRU X:RETURN
CLEARADS A:CHAR X:CHAR+X:LENGTH+1 X:SCRATCH+1 X:SCRATCH X
S:ERROR 88:COUNT A:OUT A:OUT
PUT HANDLES BOTH SQUEEZE AND UNSQUEEZE TEXT
IF ERROR NON-ZERO THEN LINE IS FLUSHED
PUTF
P1:FIN
88:COUNT X:LENGTH
88:COUNT = B:COUNT
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOVE</td>
<td>Move data from one location to another</td>
</tr>
<tr>
<td>SCRATCH</td>
<td>Scratch register for temporary storage</td>
</tr>
<tr>
<td>COUNT</td>
<td>Count the number of characters processed</td>
</tr>
<tr>
<td>LENLENGTH</td>
<td>Length of the string being processed</td>
</tr>
<tr>
<td>RETURN</td>
<td>Return from subroutine or function</td>
</tr>
<tr>
<td>SKIP</td>
<td>Skip to the next instruction if condition is not met</td>
</tr>
<tr>
<td>LEFTHTI</td>
<td>Left justify text (make it fit)</td>
</tr>
<tr>
<td>COPY</td>
<td>Copy data from one location to another</td>
</tr>
<tr>
<td>SCAN</td>
<td>Scan characters from a string</td>
</tr>
<tr>
<td>TAB</td>
<td>Tab character placement</td>
</tr>
<tr>
<td>RECURSE</td>
<td>Recurse to the same subroutine or function</td>
</tr>
<tr>
<td>CONCAT</td>
<td>Concatenate two strings</td>
</tr>
<tr>
<td>CONCAT</td>
<td>Concatenate strings with spaces</td>
</tr>
<tr>
<td>PACK</td>
<td>Pack data into a specified location</td>
</tr>
<tr>
<td>TABL</td>
<td>Table lookup</td>
</tr>
<tr>
<td>RELI</td>
<td>Reli instruction</td>
</tr>
<tr>
<td>RETURN</td>
<td>Return from subroutine or function</td>
</tr>
<tr>
<td>SKIP</td>
<td>Skip to the next instruction if condition is not met</td>
</tr>
<tr>
<td>PRINT</td>
<td>Print string</td>
</tr>
<tr>
<td>COUNT</td>
<td>Count the number of characters processed</td>
</tr>
<tr>
<td>BUMP</td>
<td>Bump pointer to the next character</td>
</tr>
<tr>
<td>SCRATCH flags</td>
<td>Flags for scratch register</td>
</tr>
<tr>
<td>SWITCH</td>
<td>Switch between different modes</td>
</tr>
<tr>
<td>INSTRUCTION</td>
<td>Instruction used in the program</td>
</tr>
<tr>
<td>DATA</td>
<td>Data used in the program</td>
</tr>
<tr>
<td>STRING</td>
<td>String of data used in the program</td>
</tr>
<tr>
<td>VARIABLE</td>
<td>Variable used in the program</td>
</tr>
<tr>
<td>REGISTER</td>
<td>Register used in the program</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>Address of data used in the program</td>
</tr>
<tr>
<td>FUNCTION</td>
<td>Function called from the program</td>
</tr>
<tr>
<td>SUBROUTINE</td>
<td>Subroutine called from the program</td>
</tr>
<tr>
<td>MACRO</td>
<td>Macro defined in the program</td>
</tr>
<tr>
<td>CALL</td>
<td>Call to subroutine or function</td>
</tr>
<tr>
<td>RET</td>
<td>Return from subroutine or function</td>
</tr>
<tr>
<td>Continued</td>
<td>Continued from previous instruction</td>
</tr>
<tr>
<td>END</td>
<td>End of program</td>
</tr>
<tr>
<td>PROGRAM</td>
<td>Program being executed</td>
</tr>
<tr>
<td>PARAMETER</td>
<td>Parameter passed to the program</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>Default value used in the program</td>
</tr>
<tr>
<td>DEFAULT VALUE</td>
<td>Default value for a parameter</td>
</tr>
<tr>
<td>DEFAULT STRING</td>
<td>Default string for a parameter</td>
</tr>
<tr>
<td>DEFAULT REGISTER</td>
<td>Default register for a parameter</td>
</tr>
<tr>
<td>DEFAULT ADDRESS</td>
<td>Default address for a parameter</td>
</tr>
<tr>
<td>DEFAULT FUNCTION</td>
<td>Default function used in the program</td>
</tr>
<tr>
<td>DEFAULT SUBROUTINE</td>
<td>Default subroutine called from the program</td>
</tr>
<tr>
<td>DEFAULT MACRO</td>
<td>Default macro defined in the program</td>
</tr>
<tr>
<td>DEFAULT CALL</td>
<td>Default call to a subroutine or function</td>
</tr>
<tr>
<td>DEFAULT RET</td>
<td>Default return from subroutine or function</td>
</tr>
<tr>
<td>CONTINUE</td>
<td>Continue from previous instruction</td>
</tr>
<tr>
<td>EXIT</td>
<td>Exit from the program</td>
</tr>
<tr>
<td>SUBPROGRAM</td>
<td>Subprogram called from the program</td>
</tr>
<tr>
<td>SUBROUTINE CALL</td>
<td>Call to a subroutine from the program</td>
</tr>
<tr>
<td>MACRO DEFINITION</td>
<td>Macro definition defined in the program</td>
</tr>
<tr>
<td>CALL OPTIONS</td>
<td>Options for calls to subroutines or functions</td>
</tr>
<tr>
<td>RETURNCODE</td>
<td>Return code from a subroutine or function</td>
</tr>
<tr>
<td>CONTINUE CODE</td>
<td>Code to continue from previous instruction</td>
</tr>
<tr>
<td>PROGRAM END</td>
<td>End of the program</td>
</tr>
<tr>
<td>PARAMETER OPTIONS</td>
<td>Options for parameters passed to the program</td>
</tr>
<tr>
<td>DEFAULT OPTIONS</td>
<td>Options for default values used in the program</td>
</tr>
<tr>
<td>DEFAULT MACRO CODE</td>
<td>Macro code defined in the program</td>
</tr>
<tr>
<td>DEFAULT CALL CODE</td>
<td>Code for default calls to subroutines or functions</td>
</tr>
<tr>
<td>DEFAULT RETURNCODE</td>
<td>Return code for default returns from subroutine or function</td>
</tr>
<tr>
<td>CONTINUE CODE</td>
<td>Code to continue from previous instruction</td>
</tr>
<tr>
<td>EXIT CODE</td>
<td>Code for exiting the program</td>
</tr>
<tr>
<td>SUBPROGRAM OPTIONS</td>
<td>Options for subprograms called from the program</td>
</tr>
<tr>
<td>SUBROUTINE CALL OPTIONS</td>
<td>Options for calls to subroutines from the program</td>
</tr>
<tr>
<td>MACRO DEFINITION CODE</td>
<td>Macro definition code defined in the program</td>
</tr>
<tr>
<td>CALL OPTIONS CODE</td>
<td>Code for options for calls to subroutines or functions</td>
</tr>
<tr>
<td>RETURNCODE CODE</td>
<td>Code for return codes from subroutines or functions</td>
</tr>
<tr>
<td>CONTINUE CODE CODE</td>
<td>Code for options to continue from previous instructions</td>
</tr>
<tr>
<td>EXIT CODE CODE</td>
<td>Code for exiting the program from previous instructions</td>
</tr>
<tr>
<td>SUBPROGRAM OPTIONS CODE</td>
<td>Options for subprograms called from the program</td>
</tr>
<tr>
<td>SUBROUTINE CALL OPTIONS CODE</td>
<td>Options for calls to subroutines from the program</td>
</tr>
<tr>
<td>MACRO DEFINITION CODE CODE</td>
<td>Macro definition code defined in the program</td>
</tr>
<tr>
<td>CALL OPTIONS CODE CODE</td>
<td>Code for options for calls to subroutines or functions</td>
</tr>
<tr>
<td>RETURNCODE CODE CODE</td>
<td>Code for return codes from subroutines or functions</td>
</tr>
<tr>
<td>CONTINUE CODE CODE CODE</td>
<td>Code for options to continue from previous instructions</td>
</tr>
<tr>
<td>EXIT CODE CODE CODE</td>
<td>Code for exiting the program from previous instructions</td>
</tr>
</tbody>
</table>
VEL
S0, RETURN

ECHO

FMT

RMT

NEW

LEN

REP

MD

DATA 2

DATA 1

END

REPL

REWIN""