CONSOLE COMPUTER DOCUMENTATION

CDDT COMMANDS

<ADRS>::= 1 OR MORE HEX DIGITS. ONLY THE LAST FOUR TYPED ARE USED.

<DATA>::= 1 OR MORE HEX DIGITS. ONLY THE LAST FOUR TYPED ARE USED.

$ ::= (ESC)

<ADRS>/ PRINTS OUT DATA AT LOCATION <ADRS>. IF <ADRS> IS LEFT OUT, PRINTS OUT LAST LOCATION EXAMINED. THE LOCATION IS NOW OPEN, AND MAY BE MODIFIED BY TYPING <DATA> (CR). (LF) INSTEAD OF (CR) MODIFIES THE LOCATION, THEN OPENS THE NEXT. (^) WORKS LIKE (LF) BUT MOVES BACKWARD.

<ADRS>$G SETS THE STARTING ADDRESS TO <ADRS>, THEN STARTS THERE. $G WITHOUT <ADRS> STARTS AT THE ADDRESS SET BY <ADRS>$G.

$P PROCEEDS FROM THE LOCATION PRINTED OUT WHEN THE BRK HAPPENED.

A BRK INSTRUCTION (OPCODE= 00) GETS BACK TO CDDT, WHICH PRINTS OUT THE PC (2 LOCATIONS PAST THE BRK NOPCODE).

<ADRS>$P SETS THE PC TO <ADRS>, THEN PROCEEDS.

n$L → CONNECTS THE TERMINAL TO THE 10 via cc line n, n=2 is default CDDT IS EAVESDROPPING ON WHAT COMES BACK, AND TAKES ANTHING BETWEEN PAIRS OF { } AS HEX LOAD DATA.

PF1 ~~BREAK~~ GETS OUT OF $L MODE.

CONSOLE COMPUTER LANGUAGE SYNTAX

DEFINE	< NAME > < DEFINITION >	SETS THE DEFINITION OF < NAME > TO BE THE REST OF WHAT YOU TYPED. DOES A LITTLE COMPILING OF BUILT-IN FUNCTION REFERENCES.
MACRO	(< ARGLIST >)	GENERALLY USED AS THE FIRST THING AFTER < NAME > IN DEFINE. ROUGHLY THE SAME AS LAMDA IN LISP. SAVES CURRENT DEFINITIONS OF ALL THE SYMBOLS IN < ARGLIST > THEN BINDS THESE SYMBOLS TO THE VALUE OF THE CORRESPONDING NEXT ITEMS IN THE FUNCTION CALL. WHEN THE FUNCTION EXIST, THE OLD DEFINITIONS ARE RESTORED.

^L

```
→ DEFUN < NAME > (< ARGLIST >) < DEFINTION >
                                                COMBINES FUNCTIONS OF "DEFINE"
                                                AND "MACRO" -- SAVES TYPING.

SYN         < NAME 1 > < NAME 2 >               SETS THE DEFINITION OF < NAME 1 > THE
                                                SAME AS THAT OF < NAME 2 >. EQUIVALENT
                                                TO REMANING THE DEF. OF < NAME 2 >
                                                EXCEPT THAT THE OLD NAME STILL EXIST.

LOCAL        (< ARGLST >)                       SIMILAR TO MACRO IN SAVING THE CURRENT
                                                DEFINITIONS OF < ARGLST >. MAKES TEMP-
                                                ORY VARIABLES WITH INITALLY NULL DEFI-
                                                NITIONS. A RETURN OF THE FUNCTIONS RE-
                                                STORS THE PREVIOUS DEFINITIONS. DOES
                                                SOME OF WHAT A LISP PROG DOES.


            ( NOTE IN EFFECT, ALL EXPRESSIONS ARE PROG'S. EACH FUNCTION GOBBLES
              THE ARGUMENTS IT NEEDS FROM THE INPUT LIST. WHEN IT RETURNS, EVAL
              CONTINUES DOWN WHATEVER IS LEFT OF THE LIST . )


SETQ         < ARGL > < REST OF LIST >          CALLS EVAL TO EVALUATE < REST OF LIST >
                                                THEN SETS THE VALUE OF < ARGL > TO THE
                                                RESULT.

COND         (( < PRED 1 > < DO 1 >)
                          .
                          .
                          .
              ( < PRED N > < DO N >))           Very much like a LISP COND. Returns NIL
                                                IF ALL THE PREDICATES WERE NIL, OTHER-
                                                WISE THE VALUE OF THE FIRST < DO X >
                                                CORRESPONDING TO A NON-NIL < PRED X >.
                                                Example:

                                                [define FOO macro(X)
                                                crlf printv X printv " is "
                                                printv (cond    ((lt X) "negative")
                                                                ((eq X) "zero")
                                                                ((gt X) "positive")
                                                        )
                                                printv "." ()
                                                ];FOO


TALK                                            CONNECTS THE CC'S TERMINAL TO THE
                                                SECOND SERIAL LINE. THE INTERPERTER
                                                IS DISCONNECTED. THE ~~BREAK~~ KEY GETS YOU
                                                OUT OF THE MODE.     PF1
^L
```

LOAD		CONNECTS THE KEY BOARD TO THE OUT BOUND SERIAL LINE WITH THE INTERPETER LISTEN-ING TO WHAT COMMS BACK AND ECHOING TO SCREEN. BREAK GETS YOU OUT OF THIS MODE.
CRLF		PRINTS A CR AND LF , FOR FORMATTING THE OUTPUT OF A MACRO.
HALT		GOES TO THE DEBUGGER (EITHER THE IN - CIRCUIT EMULATOR OR THE FUTURE CONSULE DDT). IF YOU LET THE DEBUGGER CONTINUE, THIS IS A FUNCTION OF NO ARGUMENTS THAT RETURNS NIL.
EXIT		FORSES THE INTERPRETER TO POP OUT OF WHATEVER FUNCTION IT IS IN. SAME EFFECT AS THE BREAK KEY.
SUBF	< VARIABLE > < MASK >	SPECIAL DATA TYPE. THE < MASK > DEFINES A FIELD OF CONTIGUOUS BITS INSIDE THE < VARIABLE >. IF THE SUBF IS BEING EVALUATED, THOSE BITS ARE EXTRACTED AND RIGHT-JUSTIFIED. IF THE SUBF IS THE FIRST ARGUMENT OF SETQ, THE VALUE BEING SET IS SHIFTED LEFT AND STUFFED INTO THE SPECIFIED BIT POSITIONS OF < VARIABLE >. IN THE LATTER CASE, THE VARIABLE MUST BE A SIMPLE VARIABLE (NOT AN I/O INTERFACE OR SHIFT REGIS-TERS OR ANOTHER SUBQ). SETQ OF A SUBF MODIFIES THE CURRENT VALUE OF THE < VARIABLE >, SO YOU MUST BE CAREFUL ABOUT SETTING OTHER THINGS EQUAL TO IT SINCE SETQ ONLY COPIES POINTERS TO VALUES. THE COPY FUNCTION IS THERE TO HELP.
WHILE	< PRED > < REST OF LIST >	IF < PRED > IS NIL, DOES NOTHING AND RETURNS NIL. OTHERWISE CALLS EVAL TO EVALUATE THE REST, THEN REPEATS.
COPY	< EXPR >	MAKES A COPY OF THE VALUE OF < EXPR >, RETURNS THE COPY AS A RESULT. USEFUL FOR SAVING VALUES OF VARIABLES THAT ARE SUBJECT TO MODIFICATION BY SUBF OPERAT-IONS.

^L

EQ, NE, GT, GE, LT, LE		PREDICATES
	(< PRED > < EXPR>)	COMPARES VALUE OF < EXPR WITH ZERO >, RETURNS VALUE OF < EXPR > (NON-NIL) IF TRUE, NIL IF FALSE.
	(< PRED > <EXPR 1> <EXPR 2>)	SUBTRACTS VALUE OF < EXPR 2 > FROM VALUE OF < EXPR 1 > THEN COMPARES RESULT WITH ZERO.
	(ADD < EXPR 1 > < EXPR 2 >) (SUB < EXPR 1 > < EXPR 2 >)	ADD OR SUBTRACT. IF < EXPR 2 > IS OMITTED, THE VALUE 1 (ONE) IS USED IN ITS PLACE.
PRINTV	< EXPR >	PRINTS THE VALUE OF < EXPR >. (THIS MAY BE A LITERIAL STRING). SUPPRESSES LEADING ZEROS.
LPRINT	< EXPR >	SAME AS PRINTV BUT DOES NOT SUPPRESS LEADING ZEROS.
WPRINT	< EXPR 1 > < EXPR 2 >	PRINTS THE VALUE OF < EXPR >, PADDING TO THE LEFT WITH ZEROS TO MAKE AT LEAST < EXPR > DIGITS.
MASK	< EXPR 1 > < EXPR 2 >	[NOTE - < EXPR 1 > HAD BETTER BE LESS THAN < EXPR 2 >]. MAKES A BIT MASK WITH < EXPR 1 > - < EXPR 2 > + 1 ONES AND < EXPR 2 > TRAILING ZEROS. IF < EXPR 2 > TRAILING IS OMITTED IT IS TAKEN TO BE THE SAME AS < EXPR 1 >.

^L

MISCELLANEOUS ADVANCED CC OPERATIONS

LOADING MICROCODE VIA THE DATA LINK

1. type BLAST
2. Type LOADMM n (cr)
   ... n is # of data link line.
3. "TYPE TENEX COMMAND TO PRINT FILE" will appear on terminal
4. hit PF1
5. type TALK (cr)
6. type TTYPE {name of .MLD file containing ucode}
7. hit PF1
8. type LOAD (cr) (cr)
9. when done PF1

LINE SPEED SETTINGS (IN CDDT)

```
        TYPE      ( line # )(speed code)(esc)R
                  EXAMPLE: 1D(esc)R ,  sets line one to 9600

        19200            E
         9600            D
         4800            C
         2400            B
         1200            8        MODEM
          300            6

VT100   SETUP            0000 0011 XXXX XXXX
```

^L

LOADING F4 MACROCODE VIA DATA LINK

1. Type

   ```
         TALK n
   ...where n is # data link line.
   ```

2. Type

   ```
         CSVLOD{CR}
   ```

   This loads the program which sends macrocode files to the F4. It asks whether you want to load the entire file or do an "incremental" load starting at a specified address. Then you tell it the name of the .SAV file to be loaded, and it asks for the name of an output file. At this point you type

   ```
         {PF1}
         LOAD{CR}{CR}
   ```

   and the loading commences. You may stop the load before reaching the end of the file by typing {PF1} when you have had enough; otherwise type {PF1} to return to CCL at the end.

   NOTA BENE:
   This procedure does not load regions of the program consisting of 2 or more consecutive words of 0's. If the program being loaded depends on arrays of 0's, memory should be cleared before loading.

CHANGING SPEED OF COMPUTER LINK

TTYSPD <line> <code>

1. Type HALT to get CDDT
2. To CDDT, type E012/
3. This is the address of a PAIR of registers in the serial interface chip. The register that is referenced alternates with each reference to E012. Repeatedly typing "/" re-examines the open location, and the data found there should alternate between BD or BE, and 4E. Type '/' until 4E is displayed, then type BD{return} to get 9600 baud, or BE{return} to get 19200 baud. Reexamining E012 should confirm the new contents of the register pair.

^L

CC TTY line parameters   location/default (hex)

Line	0	1	2	3	4	5
Stall code	0341/00	0351/1F	0361/1F	0371/1F	0381/1F	0391/00
Resume code	0342/00	0352/1F	0362/1F	0372/1F	0382/1F	0392/00
Stall threshold	03A3/FF	03A3/FF	03BB/C0	03CB/C0	03DB/C0	03EB/A0
Resume threshold	03A4/FF	03A4/FF	03BC/10	03CC/10	03DC/10	03EC/10
Bit rate	19200	1200	9600	9600	9600	9600
Login flag		035C				

Codes: 1F for TENEX stall/resume code
       13 for ^S (DEC stall)
       11 for ^Q (DEC resume)
       00 means don't send a code, use CTS signal instead.

Stall threshold: send stall (drop CTS) when this many characters in buffer.
Resume threshold: send resume (set CTS) when down to this many.
If threshold = FF, never stall.

Bit rate codes are

0:45.5	4:134.5	8:1200	C:4800
1:50	5:150	9:1800	D:9600
2:75	6:300	A:2000	E:19200
3:110	7:600	B:2400	F:38400

Restart addresses

F000	Cold start: simulate power-on reset. I/O reset, clear memory, initialize I/O.
F00D	Slightly warmer start: do everything cold start does EXCEPT no I/O reset.
F011	Warm start: no I/O reset or memory initialization. This leaves the interpreter's memory state unchanged.
1200	Restart interpreter, clears all function definitions.
1203	Restart interpreter's main loop, leaves definitions alone.

^L

Simulating the PF1 key on other terminals (for getting out of CTY mode)

1. Hit the BREAK key to get into CDDT.

2. Type "E9/". This prints out the saved state of TALKing (85 if you were in CTY or TALK 5 mode).

3. Zero the high two bits of loc E9 (type "05(CR)" in the above example).

4. Do ESC-P. CCL interpreter will proceed with TALK mode off.

Simulating PF1 to get out of a hung function (such as IDLE)

1. Hit BREAK.

2. If you were hung in IDLE, zero locations 56-57. This has the same effect as IDLE(CR).

3. 1203$G.