

Modified Roland PG1000

Program Listings

Modified Roland PG1000

78C10 eForth Assembly

```
;=====
;
; eForth 1.0 by Bill Muench and C. H. Ting, 1990
;
; This is an implementation for the NEC 78C10 microcomputer by
; John Talbert, 1994, Oberlin Conservatory.
;
; Register Use:      Interpreter Pointer = DE
;                   Data Stack Pointer  = SP
;                   Return Stack Pointer = HL
;
;                   Free to use: BC, EA, VA, Alternate Registers.
;
; 'doList' is accessed as a subroutine through a CALT instruction
; (Call to Jump Table). This shows up as a 'DB 80H' line
; in the $COLON and $USER Macros. When executed the
; processor jumps to an address vector located at 80H. The
; vectored 'doList' code is then located at 0F0H. The word
; 'call,' was changed to load 80H into the code area for a
; doLST assembly.
;
; A 9600 Baud serial I/O is provided. PortB/bit0 is used for serial
; output and PortC/bit3 (INT2) is used for the serial input. The
; serial input is interrupt driven with a vectored interrupt routine
; located at 0A0H. The code words ?RX, TX!, and !IO make up the
; rest of the serial I/O code. Three USER variables have been set
; up for use by these serial I/O routines: SERIN, which holds the
; received character and a flag; HAFBIT, which adjusts the software
; timing of the receiver to read in the middle of each bit frame
; (set it for 1/2 the BITIME minus 5); and BITIME, which adjusts the
; software for a specific baud rate (17H for 9600 baud assuming a 12Mhz
; processor clock).
;
; The 78C10 is an 8-bit micro, therefore cell aligning to even addresses
; is unnecessary. The $ALIGN Macro was taken out along with the NOP's
; used for cell alignment in the other Macros. All occurrences of the
; word ALGND were erased also. The word SEE no longer works because it
; relies on cell alignment.
;
; All of the system FORTH code is to be stored in ROM (up to 32K) starting
; at address 0000H. Then there is 2K of RAM starting at address C000H.
; This memory setup required the following changes:
;   1) Return and Data stacks and TIB moved to RAM.
;      (See the Memory allocation EQU assignments.)
;   2) The USER variables were moved to the micro's
;      internal RAM at FF00H to FFFFH.
;   3) PAD word was changed to move the temporary buffer
```

```

; area to RAM space.
; 4) The vocabulary pointers found in the word FORTH were
; moved to RAM space by creating two new USER variables,
; FHEAD and FLINK and changing DOVOC to read:
; DW FHEAD,CNTXT,STORE,EXIT.
; 5) NTOP and CTOP were moved to RAM space to allow dictionary
; expansion into RAM space.
;
;

```

```

; Several words were added to the ROM Dictionary. The simple operators
; 1+,1-,2+,2-,2*,2/, were defined in machine code. The words C, ,
; CCOMPILE, CODE, and ENDCODE were created to enable the creation
; of code definition.
;
;

```

```

; The NEC78C10 offers the following advantages:
;

```

- 1) Ten 16-bit internal registers and a 16-bit ALU.
Many 16-bit instructions for those FORTH stack operations.
- 2) Three 8-bit I/O ports.
- 3) Eight 8-bit Analog to Digital Converters.
- 4) Internal counters and programmable clock generators.
- 5) Internal hardware serial I/O. (can be used for MIDI I/O).
- 6) 64K address space including 256 bytes of internal RAM.

```

;=====

```

```

;; Version control

```

```

VER          EQU      01H          ;major release version
EXT          EQU      01H          ;minor extension

```

```

;; Constants

```

```

COMPO        EQU      040H         ;lexicon compile only bit
IMEDD        EQU      080H         ;lexicon immediate bit
MASKK        EQU      07F1FH       ;lexicon bit mask

```

```

CELLL        EQU      2            ;size of a cell
BASEE        EQU      10           ;default radix
VOCSS        EQU      6            ;depth of vocabulary stack

```

```

BKSP        EQU      8             ;backspace
LF          EQU      10            ;line feed
CRR         EQU      13            ;carriage return
ERR         EQU      27            ;error escape
TIC         EQU      39            ;tick

```

```

CALLL        EQU      80H          ;CALT opcodes

```

```

;; Memory allocation 0//code>--//--<name//up>--<sp//tib>--rp//em

```

```

COLDD        EQU      00100H       ;cold start

```

```

RPP          EQU      0C2F0H          ;start of return stack (RP0)
TIBB        EQU      0C200H          ;terminal input buffer (TIB)
SPP         EQU      0C1F0H          ;start of data stack (SP0)
UPP         EQU      0FF00H          ;start of user area (UP0)
NAMEE       EQU      01FFDH          ;name dictionary
CODEE       EQU      00300H          ;code dictionary
CTOP        EQU      0C390H          ;RAM code dict. expansion
NTOP        EQU      0C7FFH          ;RAM name dict. expansion
PADD        EQU      0C300H          ;PAD area

```

```
;; Initialize assembly variables
```

```

_LINK       = 0                      ;force a null link
_NAME       = NAMEE                  ;initialize name pointer
_CODE       = CODEE                  ;initialize code pointer
_USER       = 4*CELLL                ;first user variable offset

```

```
;; Define assembly macros
```

```
; Compile a code definition header.
```

```

$CODE MACRO LEX,NAME,LABEL
LABEL:                                     ;;assembly label
_CODE   = $                               ;;save code pointer
_LEN    = (LEX AND 01FH)/CELLL            ;;string cell count, round down
_NAME   = _NAME-(( _LEN+3)*CELLL)        ;;new header on cell boundary
ORG     _NAME                             ;;set name pointer
DW     _CODE,_LINK                       ;;token pointer and link
_LINK   = $                               ;;link points to a name string
DB     LEX,NAME                           ;;name string
ORG     _CODE                             ;;restore code pointer
ENDM

```

```
; Compile a colon definition header.
```

```

$COLON MACRO LEX,NAME,LABEL
$CODE LEX,NAME,LABEL
DB 80H                                     ;;include CALT doLIST
ENDM

```

```
; Compile a user variable header.
```

```

$USER MACRO LEX,NAME,LABEL
$CODE LEX,NAME,LABEL
DB 80H                                     ;;include CALT doLIST
DW DOUSE,_USER                            ;;followed by doUSER and offset
_USER = _USER+CELLL                       ;;update user area offset
ENDM

```

```

;      Compile an inline string.

D$      MACRO    FUNCT,STRNG
        DW      FUNCT                ;;function
        _LEN    = $                 ;;save address of count byte
        DB      0,STRNG             ;;count byte and string
        _CODE   = $                 ;;save code pointer
ORG     _LEN                    ;;point to count byte
        DB      _CODE-_LEN-1        ;;set count
ORG     _CODE                    ;;restore code pointer
        ENDM

;      Compile a stored string.

SD$     MACRO    STRNG
        DW      DOLIT
        _LEN    = $ + 4             ;;save address of count byte
        DW      _LEN,EXIT           ;;save cnt address on stack
        DB      0,STRNG             ;;count byte and string
        _CODE   = $                 ;;save code pointer
ORG     _LEN                    ;;point to count byte
        DB      _CODE-_LEN-1        ;;set count
ORG     _CODE                    ;;restore code pointer
        ENDM

;      Assemble inline direct threaded code ending.

$NEXT   MACRO
        DB      48H,84H             ;;EA<(DE)++,next code address into
AX
        DB      48H,28H             ;;JMP EA,jump directly to code
address
        ENDM

;; Main entry points and COLD start data

MAIN     SEGMENT
ASSUME   CS:MAIN,DS:MAIN,ES:MAIN,SS:MAIN

ORG      0000H

ORIG:    DB      54H,00,01,00        ;RESET vector, JMP 0100H
        DB      0AAH,62H,0,0        ;NMI vector, EI RETI
        DB      8 DUP(0)            ;INT T0/T1 vector
        DB      54H,0A0H,00H, 5 DUP(0) ;INT1/2 vector, JMP 00A0H
        DB      8 DUP(0)            ;INT E1/E0 vector
        DB      54H,00,02, 5 DUP(0) ;INT EIN/AD vector, JMP 0200H

```

```

DB 8 DUP(0) ;INT SR/ST vector
DB 48 DUP(0) ;FREE
DB 32 DUP(0) ;SOFTI vector at 0060H

```

```
ORG 00A0H
```

```

; Vectored INT2 routine for Serial Input from Host Computer.
; Uses address FFF0 as a counter location - do not use elsewhere!
DB 0B1H ;PUSH BC
DB 0B2H ;PUSH DE
DB 0B0H ;PUSH VA
DB 68H,0FFH ;MVI, V<FF
DB 71H,0F0H,07H ;MVIW, (V/F0)<07, number of bits to receive.
DB 70H,1FH,04CH,0FFH ;LBCD, BC<(FF4C), wait for a half bit.
DB 53H ;DCR, C<C-1 skip, LOOP1
DB 0FEH ;JR, Jump to loop1
DB 52H ;DCR, B<B-1 skip
DB 0FCH ;JR, Jump to loop1
DB 70H,1FH,4EH,0FFH ;LBCD, BC<(FF4E),wait 1 bit time, LOOP2
DB 53H ;DCR, C<C-1 skip
DB 0FEH ;JR, Jump to loop2
DB 52H ;DCR, B<B-1 skip
DB 0FCH ;JR, Jump to loop2
DB 04CH,0C2H ;MOV, A<PC, read serial input on pc3
DB 48H,31H,48H,31H ;Rotate PC3 bit into Cy
DB 48H,31H,48H,31H ;RLR, A rotate right 4xs
DB 0CH ;MOV, A<D, D collects the bits
DB 48H,31H ;RLR, shift in next bit, CY to top of D
DB 1CH ;MOV, D<A
DB 30H,0F0H ;DCRW, (V/F0)<(V/F0)-1 skip
DB 0E7H ;JR, Jump to loop2 for next bit.
DB 70H,1FH,4EH,0FFH ;LBCD, BC<(FF4E)
DB 53H,0FEH,52H,0FCH ;DCR JR DCR JR, stop bit loop time.
DB 71H,04BH,0FFH ;MVIW, (V/4B)<FF, load flag
DB 0CH,63H,04AH ;MOV STAW, A<D (V/4A)<A, load data
DB 0A0H,0A2H,0A1H ;POP, restore AV DE and BC
DB 48H,44H,0 ;SKIT,NOP
DB 0AAH,062H ;EI RETI, enable interrupts and return

```

```

;; Kernel doLST routine. Always accessed by the CALT instruction: 80H
;; which is a Call Subroutine to jump to address vector located at 0080H.

```

```

ORG 00F0H
DB 33H,33H ;HL<HL-2
DB 0A6H ;EA<DE
DB 48H,93H ;(HL)<EA
DB 0A2H ;POP DE previously pushed by CALT
DB 48H,84H ;EA<(DE)++, $NEXT

```

```

    DB 48H,28H                ;JMP EA
ORG    0080H
    DB 0F0H,0                ; set up vector to doLST

ORG    COLDD                  ;Beginning of Cold Boot
    DB 69H,0FH,4DH,0D0H     ;MM<0F, memory map (11-8)
    DB 69H,0FFH,4DH,0D2H     ;MA<FF, pa inputs (4-2)
    DB 69H,00H,4DH,0D3H     ;MB<00, pb outputs (4-6)
    DB 64H,01H,05H          ;PB<5
    DB 4DH,0D7H             ;MF<00, pf outputs (4-15)
    DB 69H,0AH,4DH,0D4H     ;MC<0A, pc1/3 inputs (4-9)
    DB 69H,0BH,4DH,0D1H     ;MCC<0B, pc mode (4-8)
    DB 64H,02H,04H          ;PC<04
    DB 64H,81H,06H          ;SMH<06, serial mode (7-7)
    DB 69H,4EH,4DH,0CAH     ;SML<4E, serial mode (7-9)
    DB 04H                  ;SP<SPP, stack pointer=data stack
    DB LOW SPP
    DB HIGH SPP
    DB 34H                  ;HL<RPP, HL=return stack pointer
    DB LOW RPP
    DB HIGH RPP
    DB 69H,00H,4DH,0E8H     ;ZCM<0, zero cross disabled (3-26)
    DB 68H,0FFH            ;V<FF
    DB 10H,68H,0FFH,69H,0   ;V'<FF, A"<0, V<FF, A<0

;; timer setups for Midi and LCD use
    DB 69H,64H,4DH,0DAH     ;TM0<64, timer0 (5-1)
    DB 69H,0FFH,4DH,0DBH     ;TM1<FF, timer1 (5-1)
    DB 64H,85H,0B3H         ;TMM<B3, timer mode (5-6)
    DB 44H,60H,0EAH,48H,0D3H ;ETM1<EA = EA60 (6-2)
    DB 64H,83H,0CCH         ;EOM<CC, timer event mode (6-14)
    DB 69H,5CH,4DH,0CCH     ;ETMM<5C, timer event mode (6-11)

    DB 54H,00,03H           ;JMP to 0300, high level cold start
                           ;COLD WORD MOVED TO THE START OF CODE AREA.
                           ;ATTEMPTED TO AUTOMATE-JMP COLD-WITH $JUMP
                           ;BUT MACRO PRODUCES ERROR CODES.

; COLD start moves the following to USER variables.
; MUST BE IN SAME ORDER AS USER VARIABLES.

UZERO:          DW      4 DUP (0)                ;reserved
                DW      SPP                    ;SP0
                DW      RPP                    ;RP0
                DW      QRX                    ;'?KEY
                DW      TXSTO                  ;'EMIT
                DW      ACCEP                  ;'EXPECT
                DW      KTAP                   ;'TAP

```



```

DW      TXSTO                ; 'ECHO
DW      DOTOK                ; 'PROMPT
DW      BASEE                ;BASE
DW      0                    ;tmp
DW      0                    ;SPAN
DW      0                    ;>IN
DW      0                    ;#TIB
DW      TIBB                 ;TIB
DW      0                    ;CSP
DW      INTER                ; 'EVAL
DW      NUMBQ                ; 'NUMBER
DW      0                    ;HLD
DW      0                    ;HANDLER
DW      0                    ;CONTEXT pointer
DW      VOCSS DUP (0)        ;vocabulary stack
DW      0                    ;CURRENT pointer
DW      0                    ;vocabulary link pointer
DW      0                    ;FORTH HEAD
DW      0                    ;FORTH LINK
DW      CTOP                 ;CP
DW      NTOP                 ;NP
DW      LASTN                ;LAST
DW      0                    ;SERIN host receive char & flag
DW      06H                  ;HAFBIT time for serial host,
                               ; (1/2 BITIME - 5)
DW      16H                  ;BITIME baud for serial host

```

ULAST:

ORG 0200H

; Interrupt routine for Analog to Digital Converters

```

DB 10H                ;EXA
DB 11H                ;EXX
; Load ADC Address and Counter into HL. Uses FFF2 and FFF3.
DB 68H,0FFH          ;V'<FF
DB 01H,0F2H          ;A<(V/F2)
DB 1EH                ;H<A
DB 01H,0F3H          ;A<(V/F3)
DB 1FH                ;L<A
; Store ADC 0.
DB 2BH                ;A<(HL)
DB 57H,80H           ;A AND 80, Skip if zero
DB 0CDH              ;Jump to EXIT if slider is disabled.
DB 1AH                ;B<A
DB 4CH,0E0H          ;A<CR0
DB 48H,21H           ;A Shift right, Midi is 7 bits, throw LSB.

```

```

    DB 60H,6AH      ;B-A, Skip if not zero
    DB 0C5H         ;Jump to EXIT if slider has not changed.
    DB 3DH          ;(HL)<A, Store slider data, 0 in top bit.
    DB 69H,0FFH    ;A<FF
    DB 0BFH,38H    ;(HL+38)<A, Store slider change flag.
    DB 32H         ;HL<HL+1, EXIT
; Store ADC 1.
    DB 2BH         ;A<(HL)
    DB 57H,80H     ;A AND 80, Skip if zero
    DB 0CDH        ;Jump to EXIT if slider is disabled.
    DB 1AH         ;B<A
    DB 4CH,0E1H    ;A<CR1
    DB 48H,21H     ;A Shift right, Midi is 7 bits, throw LSB.
    DB 60H,6AH     ;B-A, Skip if not zero
    DB 0C5H        ;Jump to EXIT if slider has not changed.
    DB 3DH         ;(HL)<A, Store slider data, 0 in top bit.
    DB 69H,0FFH    ;A<FF
    DB 0BFH,38H    ;(HL+38)<A, Store slider change flag.
    DB 32H         ;HL<HL+1, EXIT
; Store ADC 2.
    DB 2BH         ;A<(HL)
    DB 57H,80H     ;A AND 80, Skip if zero
    DB 0CDH        ;Jump to EXIT if slider is disabled.
    DB 1AH         ;B<A
    DB 4CH,0E2H    ;A<CR2
    DB 48H,21H     ;A Shift right, Midi is 7 bits, throw LSB.
    DB 60H,6AH     ;B-A, Skip if not zero
    DB 0C5H        ;Jump to EXIT if slider has not changed.
    DB 3DH         ;(HL)<A, Store slider data, 0 in top bit.
    DB 69H,0FFH    ;A<FF
    DB 0BFH,38H    ;(HL+38)<A, Store slider change flag.
    DB 32H         ;HL<HL+1, EXIT
; Store ADC 3.
    DB 2BH         ;A<(HL)
    DB 57H,80H     ;A AND 80, Skip if zero
    DB 0CDH        ;Jump to EXIT if slider is disabled.
    DB 1AH         ;B<A
    DB 4CH,0E3H    ;A<CR3
    DB 48H,21H     ;A Shift right, Midi is 7 bits, throw LSB.
    DB 60H,6AH     ;B-A, Skip if not zero
    DB 0C5H        ;Jump to EXIT if slider has not changed.
    DB 3DH         ;(HL)<A, Store slider data, 0 in top bit.
    DB 69H,0FFH    ;A<FF
    DB 0BFH,38H    ;(HL+38)<A, Store slider change flag.
    DB 32H         ;HL<HL+1, EXIT
; Update Counters
    DB 0FH         ;A<L
    DB 37H,37H     ;A-37H, Skip if borrow

```

```

DB 69H,0           ;A<0, Reset counter after 56D sliders.
DB 63H,0F3H        ;(V/F3)<A, Load counter.
DB 48H,25H,48H,25H ;A shift logical left 2xs.
DB 1AH             ;B<A
DB 74H,0AH,0E0H    ;B<B AND E0
DB 4CH,0C2H        ;A<Pc
DB 07H,1FH         ;A<A AND 1F
DB 60H,9AH         ;A<A OR B
DB 4DH,0C2H        ;Pc<A, Load high 3 bits of slider select.
DB 64H,90H,08H     ;Invert ANM bit and restart conversion.
; Return from Interrupt.
DB 10H             ;EXA
DB 11H             ;EXX
DB 0AAH           ;EI
DB 62H             ;RETI

ORG      CODEE                                ;start code dictionary

; COLD      ( -- )
;           The hilevel cold start sequence.
          CCOLD = $
          $COLON 4, 'COLD', COLD
COLD1:    DW      DOLIT,UZERO,DOLIT,UPP
          DW      DOLIT,ULAST-UZERO,CMOVE ;initialize user area
          DW      PRESE                    ;initialize stack and TIB
          DW      TBOOT,ATEXE              ;application boot
          DW      FORTH,CNTXT,AT,DUPP     ;initialize search order
          DW      CRRNT,DSTOR,OVERT
          DW      LCDINIT                  ;initialize LCD
          DW      QUIT                     ;start interpretation
          DW      BRAN,COLD1               ;just in case

;; Device dependent I/O

; BYE      ( -- )
;          Exit eForth.
          $CODE 3, 'BYE', BYE
          DB 54H,0,0                       ;JMP Reset Vector

; ?RX     ( -- c T | F )
;          Return input character and true, or a false if no input.
          $CODE 3, '?RX', QRX
          DB 68H,0FFH                       ;MVI, V<FF
          DB 01H,4BH                         ;LDAW, A<(V/4B) read serial-in flag
          DB 47H,0FFH                       ;ONI, A AND FF skip if flag not zero
          DB 0CAH                           ;JR, jump ahead1

```

```

DB 71H,04BH,0           ;MVIW, (V/4B)<0, reset flag to zero
DB 70H,1FH,4AH,0FFH    ;LBCD, BC<(FF4A), read serin data
DB 0B1H                 ;PUSH BC, push serial input data to stack
DB 69H,0FFH            ;A<FF
DB 1BH                  ;C<A, AHEAD1
DB 6AH,0                ;B<0
DB 0B1H                 ;PUSH BC, push serial input flag to stack
$NEXT

```

```

; TX!          ( c -- )
;              Send character c to the output device.

```

```

$CODE 3, 'TX!', TXSTO
DB 0BAH           ;Disable Interrupts
DB 0A1H           ;POP BC, pop char into C
DB 0B2H           ;PUSH DE, store interpreter pointer
DB 0BH,1CH        ;A<C, D<A, char in A and D
DB 68H,0FFH      ;V<FF
DB 71H,0F0H,07H  ;(V/F0)<7
DB 60H,91H       ;A<A EXOR A
DB 6DH,01H       ;E<01
DB 4DH,0C1H      ;PB<A
DB 70H,1FH,04EH,0FFH ;BC<(FF4E) set baud, LOOP1
DB 53H,0FEH,52H,0FCH ;C<C-1, JR, B<B-1, JR, jr to loop1
DB 0CH           ;A<D
DB 07H,01H       ;A<A AND 01
DB 4DH,0C1H      ;PB<A, send a bit
DB 0CH           ;A<D
DB 48H,31H       ;A rotate logical right
DB 1CH           ;D<A
DB 0,0,0,0       ;NOPs to make rec loop = transmit loop.
DB 30H,0F0H      ;(V/F0)<(V/F0)-1 skip
DB 0E8H          ;JR, jump to loop1
DB 0DH           ;A<E
DB 51H           ;A<A-1 skip
DB 0C6H          ;JR, jump to loop2
DB 0A2H          ;POP DE, restore interpreter pointer
DB 0AAH          ;Enable Interrupts
$NEXT           ;End of routine

```

```

DB 6CH,03H       ;D<03, LOOP2
DB 1DH           ;E<A
DB 71H,0F0H,01   ;(V/F0)<01
DB 4FH,0D7H      ;JRE, jump to loop1

```

```

; !IO          ( -- )
;              Initialize the serial I/O devices.

```

```

        $CODE    3, '!IO', STOIO
DB 69H,0EFH,4DH,0C7H    ;MKL<EF, enable int2 interrupt and
DB 69H,0FFH,4DH,0C6H    ;MKH<FF, disable all others with mask
DB 0AAH                ;EI, enable interrupt
$NEXT

```

```
;; The kernel
```

```

; doLIT      ( -- w )
;            Push an inline literal.

```

```

        $CODE    COMPO+5, 'doLIT', DOLIT
DB 48H,84H                ;EA<(DE)++
DB 0B4H                  ;PUSH EA
$NEXT

```

```

; EXIT      ( -- )
;          Terminate a colon definition.

```

```

        $CODE    4, 'EXIT', EXIT
DB 48H,85H                ;EA<(HL)++
DB 0B6H                  ;DE<EA
$NEXT

```

```

; EXECUTE   ( ca -- )
;          Execute the word at ca.

```

```

        $CODE    7, 'EXECUTE', EXECU
DB 0A1H                  ;POP BC
DB 21H                   ;JMP BC

```

```

; next      ( -- )
;          Run time code for the single index loop.
;          : next ( -- ) \ hilevel model
;          r> r> dup if 1 - >r @ >r exit then drop cell+ >r ;

```

```

        $CODE    COMPO+4, 'next', DONXT
DB 6AH,0                 ;B<00
DB 6BH,1                 ;C<01
DB 48H,83H              ;EA<(HL)
DB 74H,0B5H             ;EA<EA-BC Skip if no borrow
DB 0C9H                 ;JMP NEXT1
DB 48H,93H              ;(HL)<EA
DB 48H,82H              ;EA<(DE)
DB 0B6H                 ;DE<EA
$NEXT

```

```

NEXT1:          DB 22H,22H                      ;DE<DE+2
               DB 32H,32H                      ;HL<HL+2
               $NEXT

;   ?branch    ( f -- )
;               Branch if flag is zero.

               $CODE    COMPO+7,'?branch',QBRAN
               DB 6AH,0FFH                      ;B<FF
               DB 6BH,0FFH                      ;C<FF
               DB 0A4H                          ;POP EA
               DB 74H,0CDH                      ;EA AND BC Skip if not zero
               DB 0C6H                          ;JMP BRAN1
               DB 22H,22H                      ;DE<DE+2
               $NEXT

BRAN1:          DB 48H,82H                      ;EA<(DE)
               DB 0B6H                          ;DE<EA
               $NEXT

;   branch     ( -- )
;               Branch to an inline address.

               $CODE    COMPO+6,'branch',BRAN
               DB 48H,82H                      ;EA<(DE)
               DB 0B6H                          ;DE<EA
               $NEXT

;   !          ( w a -- )
;               Pop the data stack to memory.

               $CODE    1,'!',STORE
               DB 0A1H                          ;POP BC, address
               DB 0A4H                          ;POP EA, data
               DB 09H                            ;A<EAL
               DB 39H                            ;(BC)<A
               DB 12H                            ;BC<BC+1
               DB 08H                            ;A<EAH
               DB 39H                            ;(BC)<A
               $NEXT

;   @          ( a -- w )
;               Push data at memory location to the data stack.

               $CODE    1,'@',AT
               DB 0A1H                          ;POP BC
               DB 29H                            ;A<(BC)
               DB 19H                            ;EAL<A
               DB 12H                            ;BC<BC+1

```

```

    DB 29H                ;A<(BC)
    DB 18H                ;EAH<A
    DB 0B4H              ;PUSH EA
    $NEXT

; C!          ( c b -- )
;             Pop the data stack to byte memory.

    $CODE 2, 'C!', CSTOR
    DB 0A1H                ;POP BC address
    DB 0A4H                ;POP AE data
    DB 09H                ;A<EAL
    DB 39H                ;(BC)<A
    $NEXT

; C@          ( b -- c )
;             Push byte memory location to the data stack.

    $CODE 2, 'C@', CAT
    DB 0A1H                ;POP BC
    DB 29H                ;A<(BC)
    DB 6AH,0              ;B<00
    DB 1BH                ;C<A
    DB 0B1H              ;PUSH BC
    $NEXT

; RP@        ( -- a )
;             Push the current RP to the data stack.

    $CODE 3, 'RP@', RPAT
    DB 0B3H                ;PUSH HL
    $NEXT

; RP!        ( a -- )
;             Set the return stack pointer.

    $CODE COMPO+3, 'RP!', RPSTO
    DB 0A3H                ;POP HL
    $NEXT

; R>         ( -- w )
;             Pop the return stack to the data stack.

    $CODE 2, 'R>', RFROM
    DB 48H,85H            ;EA<(HL)++
    DB 0B4H              ;PUSH EA
    $NEXT

```

```
; R@      ( -- w )
;          Copy top of return stack to the data stack.
```

```
$CODE    2, 'R@', RAT
DB 48H, 83H          ;EA<(HL)
DB 0B4H              ;PUSH EA
$NEXT
```

```
; >R      ( w -- )
;          Push the data stack to the return stack.
```

```
$CODE    COMPO+2, '>R', TOR
DB 33H, 33H          ;HL<HL-2
DB 0A4H              ;POP EA
DB 48H, 93H          ;(HL)<EA
$NEXT
```

```
; SP@     ( -- a )
;          Push the current data stack pointer.
```

```
$CODE    3, 'SP@', SPAT
DB 70H, 0EH, 0FEH, 0FFH ;(FFFE)<SP
DB 70H, 1FH, 0FEH, 0FFH ;BC<(FFFE)
DB 0B1H          ;PUSH BC
$NEXT
```

```
; SP!     ( a -- )
;          Set the data stack pointer.
```

```
$CODE    3, 'SP!', SPSTO
DB 0A1H          ;POP BC
DB 70H, 1EH, 0FEH, 0FFH ;(FFFE)<BC
DB 70H, 0FH, 0FEH, 0FFH ;PC<(FFFE)
$NEXT
```

```
; DROP    ( w -- )
;          Discard top stack item.
```

```
$CODE    4, 'DROP', DROP
DB 0A4H          ;POP EA
$NEXT
```

```
; DUP     ( w -- w w )
;          Duplicate the top stack item.
```

```
$CODE    3, 'DUP', DUPP
DB 0A4H          ;POP EA
DB 0B4H          ;PUSH EA
```



```

        DB 0B4H                ;PUSH EA
        $NEXT

;  SWAP      ( w1 w2 -- w2 w1 )
;            Exchange top two stack items.

        $CODE    4,'SWAP',SWAP
        DB 0A4H                ;POP EA
        DB 0A1H                ;POP BC
        DB 0B4H                ;PUSH EA
        DB 0B1H                ;PUSH BC
        $NEXT

;  OVER      ( w1 w2 -- w1 w2 w1 )
;            Copy second stack item to top.

        $CODE    4,'OVER',OVER
        DB 0A4H                ;POP AE
        DB 0A1H                ;POP BC
        DB 0B1H                ;PUSH BC
        DB 0B4H                ;PUSH AE
        DB 0B1H                ;PUSH BC
        $NEXT

;  0<        ( n -- t )
;            Return true if n is negative.

        $CODE    2,'0<',ZLESS
        DB 0A1H                ;POP BC
        DB 69H,0FFH            ;A<FF
        DB 48H,06H            ;B Shift Left, Skip if carry
        DB 69H,0              ;A<00
        DB 1AH                ;B<A
        DB 1BH                ;C<A
        DB 0B1H                ;PUSH BC
        $NEXT

;  AND       ( w w -- w )
;            Bitwise AND.

        $CODE    3,'AND',ANDD
        DB 0A1H                ;POP BC
        DB 0A4H                ;POP AE
        DB 74H,8DH            ;EA<EA AND BC
        DB 0B4H                ;PUSH EA
        $NEXT

;  OR        ( w w -- w )

```

```
; Bitwise inclusive OR.
```

```
$CODE 2,'OR',ORR
DB 0A1H ;POP BC
DB 0A4H ;POP EA
DB 74H,9DH ;EA<EA OR BC
DB 0B4H ;PUSH EA
$NEXT
```

```
; XOR ( w w -- w )
; Bitwise exclusive OR.
```

```
$CODE 3,'XOR',XORR
DB 0A1H ;POP BC
DB 0A4H ;POP EA
DB 74H,95H ;EA<EA EX-OR BC
DB 0B4H ;PUSH EA
$NEXT
```

```
; UM+ ( w w -- w cy )
; Add two numbers, return the sum and carry flag.
```

```
$CODE 3,'UM+',UPLUS
DB 0A1H ;POP BC
DB 0A4H ;POP EA
DB 69H,0 ;A<00
DB 74H,0A5H ;EA<EA+BC Skip if no carry
DB 41H ;A<A+1
DB 1BH ;C<A
DB 6AH,0 ;B<00
DB 0B4H ;PUSH EA
DB 0B1H ;PUSH BC
$NEXT
```

```
;; System and user variables
```

```
; doVAR ( -- a )
; Run time routine for VARIABLE and CREATE.
```

```
$COLON COMPO+5,'doVAR',DOVAR
DW RFROM,EXIT
```

```
; UP ( -- a )
; Pointer to the user area.
```

```
$COLON 2,'UP',UP
DW DOVAR
DW UPP
```

```

; doUSER      ( -- a )
;              Run time routine for user variables.

          $COLON  COMPO+6,'doUSER',DOUSE
          DW      RFROM,AT,UP,AT,PLUS,EXIT

; SP0        ( -- a )
;              Pointer to bottom of the data stack.

          $USER   3,'SP0',SZERO

; RP0        ( -- a )
;              Pointer to bottom of the return stack.

          $USER   3,'RP0',RZERO

; '?KEY      ( -- a )
;              Execution vector of ?KEY.

          $USER   5,"'?KEY",TQKEY

; 'EMIT      ( -- a )
;              Execution vector of EMIT.

          $USER   5,"'EMIT",TEMIT

; 'EXPECT    ( -- a )
;              Execution vector of EXPECT.

          $USER   7,"'EXPECT",TEXPE

; 'TAP       ( -- a )
;              Execution vector of TAP.

          $USER   4,"'TAP",TTAP

; 'ECHO      ( -- a )
;              Execution vector of ECHO.

          $USER   5,"'ECHO",TECHO

; 'PROMPT    ( -- a )
;              Execution vector of PROMPT.

          $USER   7,"'PROMPT',TPROM

; BASE       ( -- a )

```

```

;           Storage of the radix base for numeric I/O.

$USER 4, 'BASE', BASE

; tmp      ( -- a )
;           A temporary storage location used in parse and find.

$USER COMPO+3, 'tmp', TEMP

; SPAN     ( -- a )
;           Hold character count received by EXPECT.

$USER 4, 'SPAN', SPAN

; >IN      ( -- a )
;           Hold the character pointer while parsing input stream.

$USER 3, '>IN', INN

; #TIB     ( -- a )
;           Hold the current count and address of the terminal input buffer.

$USER 4, '#TIB', NTIB
_USER = _USER+CELLL

; CSP      ( -- a )
;           Hold the stack pointer for error checking.

$USER 3, 'CSP', CSP

; 'EVAL    ( -- a )
;           Execution vector of EVAL.

$USER 5, "'EVAL", TEVAL

; 'NUMBER  ( -- a )
;           Execution vector of NUMBER?.

$USER 7, "'NUMBER", TNUMB

; HLD      ( -- a )
;           Hold a pointer in building a numeric output string.

$USER 3, 'HLD', HLD

; HANDLER  ( -- a )
;           Hold the return stack pointer for error handling.

```

```

    $USER 7, 'HANDLER', HANDL
; CONTEXT ( -- a )
; A area to specify vocabulary search order.
    $USER 7, 'CONTEXT', CNTXT
    _USER = _USER+VOCSS*CELLL ;vocabulary stack
; CURRENT ( -- a )
; Point to the vocabulary to be extended.
    $USER 7, 'CURRENT', CRRNT
    _USER = _USER+CELLL ;vocabulary link pointer
; FHEAD ( -- a )
; Point to the FORTH vocab head pointer.
    $USER 5, 'FHEAD', FHEAD
; FLINK ( -- a )
; Point to the FORTH vocab link pointer.
    $USER 5, 'FLINK', FLINK
; CP ( -- a )
; Point to the top of the code dictionary.
    $USER 2, 'CP', CP
; NP ( -- a )
; Point to the bottom of the name dictionary.
    $USER 2, 'NP', NP
; LAST ( -- a )
; Point to the last name in the name dictionary.
    $USER 4, 'LAST', LAST
; SERIN ( -- a )
; Point to host serial input. Flag in high, char in low byte.
    $USER 5, 'SERIN', SERIN
; HAFBIT ( -- a )
; Point to half bit time used by serial i/o routines.
    $USER 6, 'HAFBIT', HAFBIT
; BITIME ( -- a )

```

```

;           Point to bit time used to set serial i/o baud rate.

$USER    6, 'BITIME', BITIME

;; Common functions

; doVOC    ( -- )
;           Run time action of VOCABULARY's.

$COLON   COMPO+5, 'doVOC', DOVOC
DW       FHEAD, CNTXT, STORE, EXIT

; FORTH    ( -- )
;           Make FORTH the context vocabulary.

$COLON   5, 'FORTH', FORTH
DW       DOVOC, EXIT
; Head and Link pointers normally here were moved to User Ram.

; ?DUP     ( w -- w w | 0 )
;           Dup tos if its is not zero.

$CODE    4, '?DUP', QDUP
DB 6AH, 0FFH           ;B<FF
DB 6BH, 0FFH           ;C<FF
DB 0A4H                ;POP EA
DB 74H, 0DDH           ;EA AND BC, Skip if zero
DB 0B4H                ;PUSH EA
DB 0B4H                ;PUSH EA
$NEXT

; ROT      ( w1 w2 w3 -- w2 w3 w1 )
;           Rot 3rd item to top.

$COLON   3, 'ROT', ROT
DW       TOR, SWAP, RFROM, SWAP, EXIT

; 2DROP    ( w w -- )
;           Discard two items on stack.

$CODE    5, '2DROP', DDROP
DB 0A4H, 0A4H          ;POP EA, POP EA
$NEXT

; 2DUP     ( w1 w2 -- w1 w2 w1 w2 )
;           Duplicate top two items.

$CODE    4, '2DUP', DDUP

```

```

DB 0A4H,0A1H          ;POP EA, POP BC
DB 0B1H,0B4H          ;PUSH BC, PUSH EA
DB 0B1H,0B4H          ;PUSH BC, PUSH EA
$NEXT

; +          ( w w -- sum )
;           Add top two items.

$CODE 1, '+', PLUS
DB 0A1H,0A4H          ;POP BC, POP EA
DB 74H,0A5H          ;EA<EA+BC, Skip
DB 0                  ;NOP
DB 0B4H              ;PUSH EA
$NEXT

; D+        ( d d -- d )
;           Double addition, as an example using UM+.
;
;           $COLON 2, 'D+', DPLUS
;           DW      TOR, SWAP, TOR, UPLUS
;           DW      RFROM, RFROM, PLUS, PLUS, EXIT

; NOT       ( w -- w )
;           One's complement of tos.

$CODE 3, 'NOT', INVER
DB 0A1H          ;POP BC
DB 69H,0FFH      ;A<FF
DB 60H,12H       ;B<B EX-OR A
DB 60H,13H       ;C<C EX-OR A
DB 0B1H          ;PUSH BC
$NEXT

; NEGATE    ( n -- -n )
;           Two's complement of tos.

$CODE 6, 'NEGATE', NEGAT
DB 0A1H          ;POP BC
DB 69H,0FFH      ;A<FF
DB 60H,12H       ;B<B EX-OR A
DB 60H,13H       ;C<C EX-OR A
DB 12H           ;BC<BC+1
DB 0B1H          ;PUSH BC
$NEXT

; DNEGATE   ( d -- -d )
;           Two's complement of top double.

```

```

$COLON 7, 'DNEGATE', DNEGA
DW     INVER, TOR, INVER
DW     DOLIT, 1, UPLUS
DW     RFROM, PLUS, EXIT

```

```

; -      ( n1 n2 -- n1-n2 )
;      Subtraction.

```

```

$CODE 1, '-', SUBB
DB 0A1H          ;POP BC
DB 069H,0FFH    ;A<FF
DB 060H,12H     ;B<B EX-OR A
DB 060H,13H     ;C<C EX-OR A
DB 12H          ;BC<BC+1
DB 0A4H         ;POP EA
DB 74H,0A5H     ;EA<EA+BC Skip
DB 0            ;NOP
DB 0B4H        ;PUSH EA
$NEXT

```

```

; ABS      ( n -- n )
;      Return the absolute value of n.

```

```

$COLON 3, 'ABS', ABSS
DW     DUPP, ZLESS
DW     QBRAN, ABS1
DW     NEGAT

```

```

ABS1:    DW     EXIT

```

```

; =      ( w w -- t )
;      Return true if top two are equal.

```

```

$CODE 1, '=', EQUAL
DB 0A4H,0A1H    ;POP EA, POP BC
DB 69H,0FFH    ;A<FF
DB 74H,0FDH    ;EA-BC, Skip if zero
DB 69H,00H    ;A<00
DB 1AH,1BH    ;B<A, C<A
DB 0B1H      ;PUSH BC
$NEXT

```

```

; U<     ( u u -- t )
;      Unsigned compare of top two items.

```

```

$COLON 2, 'U<', ULESS
DW     DDUP, XORR, ZLESS
DW     QBRAN, ULES1

```



```

        DW      SWAP,DROP,ZLESS,EXIT
ULES1:  DW      SUBB,ZLESS,EXIT

;   <      ( n1 n2 -- t )
;          Signed compare of top two items.

$COLON  1, '<', LESS
        DW      DDUP,XORR,ZLESS
        DW      QBRAN,LESS1
        DW      DROP,ZLESS,EXIT
LESS1:  DW      SUBB,ZLESS,EXIT

;   MAX      ( n n -- n )
;          Return the greater of two top stack items.

$CODE   3, 'MAX', MAX
        DB 0A4H,0A1H      ;POP EA, POP BC
        DB 74H,0BDH      ;EA-BC, Skip if borrow
        DB 0C2H          ;Jump to Push EA
        DB 0B1H          ;PUSH BC
        DB 0C1H          ;Jump to next
        DB 0B4H          ;PUSH EA
        $NEXT

;   MIN      ( n n -- n )
;          Return the smaller of top two stack items.

$CODE   3, 'MIN', MIN
        DB 0A4H,0A1H      ;POP EA, POP BC
        DB 74H,0BDH      ;EA-BC, Skip if borrow
        DB 0C2H          ;Jump to Push EA
        DB 0B4H          ;PUSH EA
        DB 0C1H          ;Jump to next
        DB 0B1H          ;PUSH BC
        $NEXT

;   WITHIN   ( u ul uh -- t )
;          Return true if u is within the range of ul and uh.

$COLON  6, 'WITHIN', WITHI
        DW      OVER,SUBB,TOR      ;ul <= u < uh
        DW      SUBB,RFROM,ULESS,EXIT

;; Quick Operators

;

```

```

; 1+      ( n -- n+1 )
          $CODE 2,'1+',ONEP
          DB 0A1H      ;POP BC
          DB 12H      ;BC<BC+1
          DB 0B1H      ;PUSH BC
          $NEXT

; 1-      ( n -- n-1 )
          $CODE 2,'1-',ONEM
          DB 0A1H      ;POP BC
          DB 013H      ;BC<BC-1
          DB 0B1H      ;PUSH BC
          $NEXT

; 2+      ( n -- n+2 )
          $CODE 2,'2+',TWOP
          DB 0A1H      ;POP BC
          DB 12H,12H   ;BC<BC+2
          DB 0B1H      ;PUSH BC
          $NEXT

; 2-      ( n -- n-2 )
          $CODE 2,'2-',TWOM
          DB 0A1H      ;POP BC
          DB 13H,13H   ;BC<BC-2
          DB 0B1H      ;PUSH BC
          $NEXT

; 2*      ( n -- n*2 )
          $CODE 2,'2*',TWOSL
          DB 0A4H      ;POP EA
          DB 48H,0A4H   ;EA Logical Shift Left
          DB 0B4H      ;PUSH EA
          $NEXT

; 2/      ( n -- n/2 )
          $CODE 2,'2/',TWOSR
          DB 0A4H      ;POP EA
          DB 48H,0A0H   ;EA Logical Shift Right
          DB 0B4H      ;PUSH EA
          $NEXT

;; Divide

; UM/MOD  ( udl udh u -- ur uq )

```

```

;           Unsigned divide of a double by a single. Return mod and
quotient.

```

```

$COLON 6, 'UM/MOD', UMMOD
DW     DDUP, ULESS
DW     QBRAN, UMM4
DW     NEGAT, DOLIT, 15, TOR
UMM1:  DW     TOR, DUPP, UPLUS
DW     TOR, TOR, DUPP, UPLUS
DW     RFROM, PLUS, DUPP
DW     RFROM, RAT, SWAP, TOR
DW     UPLUS, RFROM, ORR
DW     QBRAN, UMM2
DW     TOR, DROP, ONEP, RFROM
DW     BRAN, UMM3
UMM2:  DW     DROP
UMM3:  DW     RFROM
DW     DONXT, UMM1
DW     DROP, SWAP, EXIT
UMM4:  DW     DROP, DDROP
DW     DOLIT, -1, DUPP, EXIT           ;overflow, return max

```

```

; M/MOD    ( d n -- r q )
;           Signed floored divide of double by single. Return mod and
quotient.

```

```

$COLON 5, 'M/MOD', MSMOD
DW     DUPP, ZLESS, DUPP, TOR
DW     QBRAN, MMOD1
DW     NEGAT, TOR, DNEGA, RFROM
MMOD1: DW     TOR, DUPP, ZLESS
DW     QBRAN, MMOD2
DW     RAT, PLUS
MMOD2: DW     RFROM, UMMOD, RFROM
DW     QBRAN, MMOD3
DW     SWAP, NEGAT, SWAP
MMOD3: DW     EXIT

```

```

; /MOD     ( n n -- r q )
;           Signed divide. Return mod and quotient.

```

```

$COLON 4, '/MOD', SLMOD
DW     OVER, ZLESS, SWAP, MSMOD, EXIT

```

```

; MOD     ( n n -- r )
;           Signed divide. Return mod only.

```

```

$COLON 3, 'MOD', MODD

```

```

        DW          SLMOD,DROP,EXIT
; /          ( n n -- q )
;           Signed divide. Return quotient only.

        $COLON    1, '/', SLASH
        DW          SLMOD,SWAP,DROP,EXIT

;; Multiply

;  UM*       ( u u -- ud )
;           Unsigned multiply. Return double product.

        $COLON    3, 'UM*', UMSTA
        DW          DOLIT,0,SWAP,DOLIT,15,TOR
UMST1:    DW          DUPP,UPLUS,TOR,TOR
        DW          DUPP,UPLUS,RFROM,PLUS,RFROM
        DW          QBRAN,UMST2
        DW          TOR,OVER,UPLUS,RFROM,PLUS
UMST2:    DW          DONXT,UMST1
        DW          ROT,DROP,EXIT

; *         ( n n -- n )
;           Signed multiply. Return single product.

        $COLON    1, '*', STAR
        DW          UMSTA,DROP,EXIT

;  M*       ( n n -- d )
;           Signed multiply. Return double product.

        $COLON    2, 'M*', MSTAR
        DW          DDUP,XORR,ZLESS,TOR
        DW          ABSS,SWAP,ABSS,UMSTA
        DW          RFROM
        DW          QBRAN,MSTA1
        DW          DNEGA
MSTA1:    DW          EXIT

; */MOD     ( n1 n2 n3 -- r q )
;           Multiply n1 and n2, then divide by n3. Return mod and quotient.

        $COLON    5, '*/MOD', SSMOD
        DW          TOR,MSTAR,RFROM,MSMOD,EXIT

; */       ( n1 n2 n3 -- q )
;           Multiply n1 by n2, then divide by n3. Return quotient only.

```

```

        $COLON 2, '*/', STASL
        DW      SSMOD, SWAP, DROP, EXIT

;; Miscellaneous

;   BL          ( -- 32 )
;               Return 32, the blank character.

        $COLON 2, 'BL', BLANK
        DW      DOLIT, ' ', EXIT

;   >CHAR      ( c -- c )
;               Filter non-printing characters.

        $COLON 5, '>CHAR', TCHAR
        DW      DOLIT, 07FH, ANDD, DUPP      ;mask msb
        DW      DOLIT, 127, BLANK, WITHI    ;check for printable
        DW      QBRAN, TCHA1
        DW      DROP, DOLIT, '_'           ;replace non-printables
TCHA1:      DW      EXIT

;   DEPTH      ( -- n )
;               Return the depth of the data stack.

        $COLON 5, 'DEPTH', DEPTH
        DW      SPAT, SZERO, AT, SWAP, SUBB
        DW      DOLIT, CELLL, SLASH, EXIT

;   PICK       ( ... +n -- ... w )
;               Copy the nth stack item to tos.

        $COLON 4, 'PICK', PICK
        DW      ONEP, TWOSL
        DW      SPAT, PLUS, AT, EXIT

;; Memory access

;   +!         ( n a -- )
;               Add n to the contents at address a.

        $COLON 2, '+!', PSTOR
        DW      SWAP, OVER, AT, PLUS
        DW      SWAP, STORE, EXIT

;   2!         ( d a -- )
;               Store the double integer to address a.

        $COLON 2, '2!', DSTOR

```

```

        DW      SWAP,OVER,STORE
        DW      TWOP,STORE,EXIT

; 2@      ( a -- d )
;          Fetch double integer from address a.

        $COLON 2,'2@',DAT
        DW      DUPP,TWOP,AT
        DW      SWAP,AT,EXIT

; COUNT   ( b -- b +n )
;          Return count byte of a string and add 1 to byte address.

        $COLON 5,'COUNT',COUNT
        DW      DUPP,ONEP
        DW      SWAP,CAT,EXIT

; HERE    ( -- a )
;          Return the top of the code dictionary.

        $COLON 4,'HERE',HERE
        DW      CP,AT,EXIT

; PAD     ( -- a )
;          Return the address of a temporary buffer.

        $COLON 3,'PAD',PAD
        DW      DOLIT,PADD,EXIT

; TIB     ( -- a )
;          Return the address of the terminal input buffer.

        $COLON 3,'TIB',TIB
        DW      NTIB,TWOP,AT,EXIT

; @EXECUTE ( a -- )
;          Execute vector stored in address a.

        $COLON 8,'@EXECUTE',ATEXE
        DW      AT,QDUP          ;?address or zero
        DW      QBRAN,EXE1
        DW      EXECU          ;execute if non-zero
EXE1:      DW      EXIT          ;do nothing if zero

; CMOVE   ( b1 b2 u -- )
;          Copy u bytes from b1 to b2.

        $COLON 5,'CMOVE',CMOVE

```

```

        DW      TOR
        DW      BRAN,CMOV2
CMOV1:   DW      TOR,DUPP,CAT
        DW      RAT,CSTOR
        DW      ONEP
        DW      RFROM,ONEP
CMOV2:   DW      DONXT,CMOV1
        DW      DDROP,EXIT

;   FILL      ( b u c -- )
;             Fill u bytes of character c to area beginning at b.

        $COLON 4,'FILL',FILL
        DW      SWAP,TOR,SWAP
        DW      BRAN,FILL2
FILL1:   DW      DDUP,CSTOR,ONEP
FILL2:   DW      DONXT,FILL1
        DW      DDROP,EXIT

;   -TRAILING ( b u -- b u )
;             Adjust the count to eliminate trailing white space.

        $COLON 9,'-TRAILING',DTRAI
        DW      TOR
        DW      BRAN,DTRA2
DTRA1:   DW      BLANK,OVER,RAT,PLUS,CAT,LESS
        DW      QBRAN,DTRA2
        DW      RFROM,ONEP,EXIT           ;adjusted count
DTRA2:   DW      DONXT,DTRA1
        DW      DOLIT,0,EXIT             ;count=0

;   PACK$     ( b u a -- a )
;             Build a counted string with u characters from b. Null fill.

        $COLON 5,'PACK$',PACKS
        DW      DUPP,TOR                 ;strings only on cell boundary
        DW      OVER,DUPP,DOLIT,0
        DW      DOLIT,CELLL,UMMOD,DROP   ;count mod cell
        DW      SUBB,OVER,PLUS
        DW      DOLIT,0,SWAP,STORE       ;null fill cell
        DW      DDUP,CSTOR,ONEP         ;save count
        DW      SWAP,CMOVE,RFROM,EXIT   ;move string

;; Numeric output, single precision

;   DIGIT     ( u -- c )
;             Convert digit u to a character.

```

```

$COLON 5, 'DIGIT', DIGIT
DW     DOLIT, 9, OVER, LESS
DW     DOLIT, 7, ANDD, PLUS
DW     DOLIT, '0', PLUS, EXIT

;   EXTRACT      ( n base -- n c )
;               Extract the least significant digit from n.

$COLON 7, 'EXTRACT', EXTRC
DW     DOLIT, 0, SWAP, UMMOD
DW     SWAP, DIGIT, EXIT

;   <#          ( -- )
;               Initiate the numeric output process.

$COLON 2, '<#', BDIGS
DW     PAD, HLD, STORE, EXIT

;   HOLD        ( c -- )
;               Insert a character into the numeric output string.

$COLON 4, 'HOLD', HOLD
DW     HLD, AT, ONEM
DW     DUPP, HLD, STORE, CSTOR, EXIT

;   #           ( u -- u )
;               Extract one digit from u and append the digit to output string.

$COLON 1, '#', DIG
DW     BASE, AT, EXTRC, HOLD, EXIT

;   #S          ( u -- 0 )
;               Convert u until all digits are added to the output string.

$COLON 2, '#S', DIGS
DIGS1:  DW     DIG, DUPP
        DW     QBRAN, DIGS2
        DW     BRAN, DIGS1
DIGS2:  DW     EXIT

;   SIGN        ( n -- )
;               Add a minus sign to the numeric output string.

$COLON 4, 'SIGN', SIGN
DW     ZLESS
DW     QBRAN, SIGN1
DW     DOLIT, '-', HOLD
SIGN1:  DW     EXIT

```



```

; #>          ( w -- b u )
;             Prepare the output string to be TYPE'd.

$COLON 2, '#>', EDIGS
DW     DROP, HLD, AT
DW     PAD, OVER, SUBB, EXIT

; str        ( n -- b u )
;             Convert a signed integer to a numeric string.

$COLON 3, 'str', STR
DW     DUPP, TOR, ABSS
DW     BDIGS, DIGS, RFROM
DW     SIGN, EDIGS, EXIT

; HEX        ( -- )
;             Use radix 16 as base for numeric conversions.

$COLON 3, 'HEX', HEX
DW     DOLIT, 16, BASE, STORE, EXIT

; DECIMAL    ( -- )
;             Use radix 10 as base for numeric conversions.

$COLON 7, 'DECIMAL', DECIM
DW     DOLIT, 10, BASE, STORE, EXIT

;; Numeric input, single precision

; DIGIT?     ( c base -- u t )
;             Convert a character to its numeric value. A flag indicates
success.

$COLON 6, 'DIGIT?', DIGTQ
DW     TOR, DOLIT, '0', SUBB
DW     DOLIT, 9, OVER, LESS
DW     QBRAN, DGTQ1
DW     DOLIT, 7, SUBB
DW     DUPP, DOLIT, 10, LESS, ORR
DGTQ1: DW     DUPP, RFROM, ULESS, EXIT

; NUMBER?    ( a -- n T | a F )
;             Convert a number string to integer. Push a flag on tos.

$COLON 7, 'NUMBER?', NUMBQ
DW     BASE, AT, TOR, DOLIT, 0, OVER, COUNT
DW     OVER, CAT, DOLIT, '$', EQUAL

```

```

        DW      QBRAN,NUMQ1
        DW      HEX,SWAP,ONEP
        DW      SWAP,ONEM
NUMQ1:   DW      OVER,CAT,DOLIT,'-',EQUAL,TOR
        DW      SWAP,RAT,SUBB,SWAP,RAT,PLUS,QDUP
        DW      QBRAN,NUMQ6
        DW      ONEM,TOR
NUMQ2:   DW      DUPP,TOR,CAT,BASE,AT,DIGTQ
        DW      QBRAN,NUMQ4
        DW      SWAP,BASE,AT,STAR,PLUS,RFROM
        DW      ONEP
        DW      DONXT,NUMQ2
        DW      RAT,SWAP,DROP
        DW      QBRAN,NUMQ3
        DW      NEGAT
NUMQ3:   DW      SWAP
        DW      BRAN,NUMQ5
NUMQ4:   DW      RFROM,RFROM,DDROP,DDROP,DOLIT,0
NUMQ5:   DW      DUPP
NUMQ6:   DW      RFROM,DDROP
        DW      RFROM,BASE,STORE,EXIT

```

```
;; Basic I/O
```

```

;   ?KEY      ( -- c T | F )
;               Return input character and true, or a false if no input.

```

```

$COLON 4,'?KEY',QKEY
DW      TQKEY,ATEXE,EXIT

```

```

;   KEY       ( -- c )
;               Wait for and return an input character.

```

```

$COLON 3,'KEY',KEY
KEY1:   DW      QKEY
        DW      QBRAN,KEY1
        DW      EXIT

```

```

;   EMIT      ( c -- )
;               Send a character to the output device.

```

```

$COLON 4,'EMIT',EMIT
DW      TEMIT,ATEXE,EXIT

```

```

;   NUF?      ( -- t )
;               Return false if no input, else pause and if CR return true.

```

```
$COLON 4,'NUF?',NUFQ
```

```

        DW      QKEY,DUPP
        DW      QBRAN,NUFQ1
        DW      DDROP,KEY,DOLIT,CRR,EQUAL
NUFQ1:   DW      EXIT

;   PACE      ( -- )
;           Send a pace character for the file downloading process.

        $COLON 4,'PACE',PACE
        DW      DOLIT,11,EMIT,EXIT

;   SPACE     ( -- )
;           Send the blank character to the output device.

        $COLON 5,'SPACE',SPACE
        DW      BLANK,EMIT,EXIT

;   SPACES   ( +n -- )
;           Send n spaces to the output device.

        $COLON 6,'SPACES',SPACS
        DW      DOLIT,0,MAX,TOR
        DW      BRAN,CHAR2
CHAR1:   DW      SPACE
CHAR2:   DW      DONXT,CHAR1
        DW      EXIT

;   TYPE     ( b u -- )
;           Output u characters from b.

        $COLON 4,'TYPE',TYPEE
        DW      TOR
        DW      BRAN,TYPE2
TYPE1:   DW      DUPP,CAT,EMIT
        DW      ONEP
TYPE2:   DW      DONXT,TYPE1
        DW      DROP,EXIT

;   CR       ( -- )
;           Output a carriage return and a line feed.

        $COLON 2,'CR',CR
        DW      DOLIT,CRR,EMIT
        DW      DOLIT,LF,EMIT,EXIT

;   do$     ( -- a )
;           Return the address of a compiled string.

```

```

$COLON  COMPO+3, 'do$', DOSTR
DW      RFROM, RAT, RFROM, COUNT, PLUS
DW      TOR, SWAP, TOR, EXIT

;  $" |      ( -- a )
;           Run time routine compiled by $". Return address of a compiled
string.

$COLON  COMPO+3, '$" | ', STRQP
DW      DOSTR, EXIT           ;force a call to do$

;  ." |      ( -- )
;           Run time routine of ." . Output a compiled string.

$COLON  COMPO+3, '." | ', DOTQP
DW      DOSTR, COUNT, TYPEE, EXIT

;  .R      ( n +n -- )
;           Display an integer in a field of n columns, right justified.

$COLON  2, '.R', DOTR
DW      TOR, STR, RFROM, OVER, SUBB
DW      SPACS, TYPEE, EXIT

;  U.R      ( u +n -- )
;           Display an unsigned integer in n column, right justified.

$COLON  3, 'U.R', UDOTR
DW      TOR, BDIGS, DIGS, EDIGS
DW      RFROM, OVER, SUBB
DW      SPACS, TYPEE, EXIT

;  U.      ( u -- )
;           Display an unsigned integer in free format.

$COLON  2, 'U.', UDOT
DW      BDIGS, DIGS, EDIGS
DW      SPACE, TYPEE, EXIT

;  .      ( w -- )
;           Display an integer in free format, preceded by a space.

$COLON  1, '.', DOT
DW      BASE, AT, DOLIT, 10, XORR   ;?decimal
DW      QBRAN, DOT1
DW      UDOT, EXIT                 ;no, display unsigned
DOT1:   DW      STR, SPACE, TYPEE, EXIT   ;yes, display signed

```

```

;   ?           ( a -- )
;               Display the contents in a memory cell.

$COLON 1, '?' ,QUEST
DW     AT, DOT, EXIT

;; Parsing

;   parse      ( b u c -- b u delta ; <string> )
;               Scan string delimited by c. Return found string and its offset.

$COLON 5, 'parse', PARS
DW     TEMP, STORE, OVER, TOR, DUPP
DW     QBRAN, PARS8
DW     ONEM, TEMP, AT, BLANK, EQUAL
DW     QBRAN, PARS3
DW     TOR
PARS1:  DW     BLANK, OVER, CAT           ;skip leading blanks ONLY
DW     SUBB, ZLESS, INVER
DW     QBRAN, PARS2
DW     ONEP
DW     DONXT, PARS1
DW     RFROM, DROP, DOLIT, 0, DUPP, EXIT
PARS2:  DW     RFROM
PARS3:  DW     OVER, SWAP
DW     TOR
PARS4:  DW     TEMP, AT, OVER, CAT, SUBB   ;scan for delimiter
DW     TEMP, AT, BLANK, EQUAL
DW     QBRAN, PARS5
DW     ZLESS
PARS5:  DW     QBRAN, PARS6
DW     ONEP
DW     DONXT, PARS4
DW     DUPP, TOR
DW     BRAN, PARS7
PARS6:  DW     RFROM, DROP, DUPP
DW     ONEP, TOR
PARS7:  DW     OVER, SUBB
DW     RFROM, RFROM, SUBB, EXIT
PARS8:  DW     OVER, RFROM, SUBB, EXIT

;   PARSE      ( c -- b u ; <string> )
;               Scan input stream and return counted string delimited by c.

$COLON 5, 'PARSE', PARSE
DW     TOR, TIB, INN, AT, PLUS           ;current input buffer pointer
DW     NTIB, AT, INN, AT, SUBB           ;remaining count
DW     RFROM, PARS, INN, PSTOR, EXIT

```

```

;   .(          ( -- )
;           Output following string up to next ) .

$COLON  IMEDD+2, '( ', DOTPR
DW      DOLIT, ')', PARSE, TYPEE, EXIT

;   (          ( -- )
;           Ignore following string up to next ) . A comment.

$COLON  IMEDD+1, '( ', PAREN
DW      DOLIT, ')', PARSE, DDROP, EXIT

;   \          ( -- )
;           Ignore following text till the end of line.

$COLON  IMEDD+1, '\ ', BKSLA
DW      NTIB, AT, INN, STORE, EXIT

;   CHAR          ( -- c )
;           Parse next word and return its first character.

$COLON  4, 'CHAR', CHAR
DW      BLANK, PARSE, DROP, CAT, EXIT

;   TOKEN          ( -- a ; <string> )
;           Parse a word from input stream and copy it to name dictionary.

$COLON  5, 'TOKEN', TOKEN
DW      BLANK, PARSE, DOLIT, 31, MIN
DW      NP, AT, OVER, SUBB, TWOM
DW      PACKS, EXIT

;   WORD          ( c -- a ; <string> )
;           Parse a word from input stream and copy it to code dictionary.

$COLON  4, 'WORD', WORDD
DW      PARSE, HERE, PACKS, EXIT

;; Dictionary search

;   NAME>          ( na -- ca )
;           Return a code address given a name address.

$COLON  5, 'NAME>', NAMET
DW      TWOM, TWOM, AT, EXIT

;   SAME?          ( a a u -- a a f \ -0+ )

```

```

;          Compare u cells in two strings. Return 0 if identical.

$COLON  5, 'SAME?', SAMEQ
DW      TOR
DW      BRAN, SAME2
SAME1:  DW      OVER, RAT, TWOSL, PLUS, AT
DW      OVER, RAT, TWOSL, PLUS, AT
DW      SUBB, QDUP
DW      QBRAN, SAME2
DW      RFROM, DROP, EXIT          ;strings not equal
SAME2:  DW      DONXT, SAME1
DW      DOLIT, 0, EXIT            ;strings equal

;  find      ( a va -- ca na | a F )
;          Search a vocabulary for a string. Return ca and na if succeeded.

$COLON  4, 'find', FIND
DW      SWAP, DUPP, CAT
DW      DOLIT, CELLL, SLASH, TEMP, STORE
DW      DUPP, AT, TOR, TWOP, SWAP
FIND1:  DW      AT, DUPP
DW      QBRAN, FIND6
DW      DUPP, AT, DOLIT, MASKK, ANDD, RAT, XORR
DW      QBRAN, FIND2
DW      TWOP, DOLIT, -1            ;true flag
DW      BRAN, FIND3
FIND2:  DW      TWOP, TEMP, AT, SAMEQ
FIND3:  DW      BRAN, FIND4
FIND6:  DW      RFROM, DROP
DW      SWAP, TWOM, SWAP, EXIT
FIND4:  DW      QBRAN, FIND5
DW      TWOM, TWOM
DW      BRAN, FIND1
FIND5:  DW      RFROM, DROP, SWAP, DROP
DW      TWOM
DW      DUPP, NAMET, SWAP, EXIT

;  NAME?    ( a -- ca na | a F )
;          Search all context vocabularies for a string.

$COLON  5, 'NAME?', NAMEQ
DW      CNTXT, DUPP, DAT, XORR    ;?context=also
DW      QBRAN, NAMQ1
DW      TWOM                      ;no, start with context
NAMQ1:  DW      TOR
NAMQ2:  DW      RFROM, TWOP, DUPP, TOR  ;next in search order
DW      AT, QDUP
DW      QBRAN, NAMQ3

```

```

        DW      FIND,QDUP          ;search vocabulary
        DW      QBRAN,NAMQ2
        DW      RFROM,DROP,EXIT    ;found name
NAMQ3:   DW      RFROM,DROP        ;name not found
        DW      DOLIT,0,EXIT      ;false flag

;; Terminal response

;   ^H          ( bot eot cur -- bot eot cur )
;              Backup the cursor by one character.

$COLON  2, '^H',BKSP
DW      TOR,OVER,RFROM,SWAP,OVER,XORR
DW      QBRAN,BACK1
DW      DOLIT,BKSPP,TECHO,ATEXE,ONEM
DW      BLANK,TECHO,ATEXE
DW      DOLIT,BKSPP,TECHO,ATEXE
BACK1:  DW      EXIT

;   TAP         ( bot eot cur c -- bot eot cur )
;              Accept and echo the key stroke and bump the cursor.

$COLON  3, 'TAP',TAP
DW      DUPP,TECHO,ATEXE
DW      OVER,CSTOR,ONEP,EXIT

;   kTAP        ( bot eot cur c -- bot eot cur )
;              Process a key stroke, CR or backspace.

$COLON  4, 'kTAP',KTAP
DW      DUPP,DOLIT,CRR,XORR
DW      QBRAN,KTAP2
DW      DOLIT,BKSPP,XORR
DW      QBRAN,KTAP1
DW      BLANK,TAP,EXIT
KTAP1:  DW      BKSP,EXIT
KTAP2:  DW      DROP,SWAP,DROP,DUPP,EXIT

;   accept      ( b u -- b u )
;              Accept characters to input buffer. Return with actual count.

$COLON  6, 'accept',ACCEP
DW      OVER,PLUS,OVER
ACCP1:  DW      DDUP,XORR
        DW      QBRAN,ACCP4
        DW      KEY,DUPP
;
        DW      BLANK,SUBB,DOLIT,95,ULESS
        DW      BLANK,DOLIT,127,WITHI

```



```

        DW      QBRAN,ACCP2
        DW      TAP
        DW      BRAN,ACCP3
ACCP2:   DW      TTAP,ATEXE
ACCP3:   DW      BRAN,ACCP1
ACCP4:   DW      DROP,OVER,SUBB,EXIT

;   EXPECT      ( b u -- )
;               Accept input stream and store count in SPAN.

        $COLON  6, 'EXPECT', EXPEC
        DW      TEXPE,ATEXE,SPAN,STORE,DROP,EXIT

;   QUERY      ( -- )
;               Accept input stream to terminal input buffer.

        $COLON  5, 'QUERY', QUERY
        DW      TIB,DOLIT,80,TEXPE,ATEXE,NTIB,STORE
        DW      DROP,DOLIT,0,INN,STORE,EXIT

;; Error handling

;   CATCH      ( ca -- 0 | err# )
;               Execute word at ca and set up an error frame for it.

        $COLON  5, 'CATCH', CATCH
        DW      SPAT,TOR,HANDL,AT,TOR      ;save error frame
        DW      RPAT,HANDL,STORE,EXECU    ;execute
        DW      RFROM,HANDL,STORE        ;restore error frame
        DW      RFROM,DROP,DOLIT,0,EXIT  ;no error

;   THROW      ( err# -- err# )
;               Reset system to current local error frame an update error flag.

        $COLON  5, 'THROW', THROW
        DW      HANDL,AT,RPSTO            ;restore return stack
        DW      RFROM,HANDL,STORE        ;restore handler frame
        DW      RFROM,SWAP,TOR,SPSTO     ;restore data stack
        DW      DROP,RFROM,EXIT

;   NULL$      ( -- a )
;               Return address of a null string with zero count.

        $COLON  5, 'NULL$', NULLS
        DW      DOVAR                    ;emulate CREATE
        DW      0
        DB      99,111,121,111,116,101

```

```

;  ABORT      ( -- )
;              Reset data stack and jump to QUIT.

          $COLON 5, 'ABORT', ABORT
          DW     NULLS, THROW

;  abort"    ( f -- )
;              Run time routine of ABORT" . Abort with a message.

          $COLON COMPO+6, 'abort"', ABORQ
          DW     QBRAN, ABOR1      ;text flag
          DW     DOSTR, THROW     ;pass error string
ABOR1:    DW     DOSTR, DROP, EXIT ;drop error

;; The text interpreter

;  $INTERPRET ( a -- )
;              Interpret a word. If failed, try to convert it to an integer.

          $COLON 10, '$INTERPRET', INTER
          DW     NAMEQ, QDUP      ;?defined
          DW     QBRAN, INTE1
          DW     AT, DOLIT, COMPO, ANDD ;?compile only lexicon bits
          D$     ABORQ, ' compile only'
          DW     EXECU, EXIT     ;execute defined word
INTE1:    DW     TNUMB, ATEXE    ;convert a number
          DW     QBRAN, INTE2
          DW     EXIT
INTE2:    DW     THROW          ;error

;  [        ( -- )
;              Start the text interpreter.

          $COLON IMEDD+1, '[', LBRAC
          DW     DOLIT, INTER, TEVAL, STORE, EXIT

;  .OK      ( -- )
;              Display 'ok' only while interpreting.

          $COLON 3, '.OK', DOTOK
          DW     DOLIT, INTER, TEVAL, AT, EQUAL
          DW     QBRAN, DOTO1
          D$     DOTQP, ' ok'
DOTO1:    DW     CR, EXIT

;  ?STACK   ( -- )
;              Abort if the data stack underflows.

```

```

$COLON 6, '?STACK', QSTAC
DW     DEPTH, ZLESS           ;check only for underflow
D$     ABORQ, ' underflow'
DW     EXIT

; EVAL      ( -- )
;           Interpret the input stream.

$COLON 4, 'EVAL', EVAL
EVAL1:  DW     TOKEN, DUPP, CAT           ;?input stream empty
DW     QBRAN, EVAL2
DW     TEVAL, ATEXE, QSTAC           ;evaluate input, check stack
DW     BRAN, EVAL1
EVAL2:  DW     DROP, TPROM, ATEXE, EXIT ;prompt

;; Shell

; PRESET   ( -- )
;           Reset data stack pointer and the terminal input buffer.

$COLON 6, 'PRESET', PRESE
DW     SZERO, AT, SPSTO
DW     DOLIT, TIBB, NTIB, TWOP, STORE, EXIT

; xio      ( a a a -- )
;           Reset the I/O vectors 'EXPECT, 'TAP, 'ECHO and 'PROMPT.

$COLON COMPO+3, 'xio', XIO
DW     DOLIT, ACCEP, TEXPE, DSTOR
DW     TECHO, DSTOR, EXIT

; FILE     ( -- )
;           Select I/O vectors for file download.

$COLON 4, 'FILE', FILE
DW     DOLIT, PACE, DOLIT, DROP
DW     DOLIT, KTAP, XIO, EXIT

; HAND     ( -- )
;           Select I/O vectors for terminal interface.

$COLON 4, 'HAND', HAND
DW     DOLIT, DOTOK, DOLIT, EMIT
DW     DOLIT, KTAP, XIO, EXIT

; I/O      ( -- a )
;           Array to store default I/O vectors.

```

```

$COLON 3, 'I/O', ISLO
DW      DOVAR                ;emulate CREATE
DW      QRX, TXSTO           ;default I/O vectors

;  CONSOLE      ( -- )
;               Initiate terminal interface.

$COLON 7, 'CONSOLE', CONSO
DW      ISLO, DAT, TQKEY, DSTOR ;restore default I/O device
DW      HAND, EXIT           ;keyboard input

;  QUIT         ( -- )
;               Reset return stack pointer and start text interpreter.

$COLON 4, 'QUIT', QUIT
DW      RZERO, AT, RPSTO     ;reset return stack pointer
QUIT1:  DW      LBRAC         ;start interpretation
QUIT2:  DW      QUERY        ;get input
        DW      DOLIT, EVAL, CATCH, QDUP ;evaluate input
        DW      QBRAN, QUIT2  ;continue till error
        DW      TPROM, AT, SWAP ;save input device
        DW      CONSO, NULLS, OVER, XORR ;?display error message
        DW      QBRAN, QUIT3
        DW      SPACE, COUNT, TYPEE   ;error message
        D$      DOTQP, ' ? '          ;error prompt
QUIT3:  DW      DOLIT, DOTOK, XORR     ;?file input
        DW      QBRAN, QUIT4
        DW      DOLIT, ERR, EMIT      ;file error, tell host
QUIT4:  DW      PRESE                ;some cleanup
        DW      BRAN, QUIT1

;; The compiler

;  '           ( -- ca )
;               Search context vocabularies for the next word in input stream.

$COLON 1, '"', TICK
DW      TOKEN, NAMEQ         ;?defined
DW      QBRAN, TICK1
DW      EXIT                 ;yes, push code address
TICK1:  DW      THROW        ;no, error

;  ALLOT      ( n -- )
;               Allocate n bytes to the code dictionary.

$COLON 5, 'ALLOT', ALLOT
DW      CP, PSTOR, EXIT     ;adjust code pointer

```

```

; ,      ( w -- )
;      Compile an integer into the code dictionary.

$COLON  1, ',', COMMA
DW      HERE, DUPP, TWOP      ;cell boundary
DW      CP, STORE, STORE, EXIT ;adjust code pointer, compile

; C,     ( b -- )
;      Compile a byte into the code dictionary

$COLON  2, 'C,', CCOMMA
DW      HERE, DUPP, ONEP
DW      CP, STORE, CSTOR, EXIT

; [COMPILE] ( -- ; <string> )
;      Compile the next immediate word into code dictionary.

$COLON  IMEDD+9, '[COMPILE]', BCOMP
DW      TICK, COMMA, EXIT

; COMPILER ( -- )
;      Compile the next address in colon list to code dictionary.

$COLON  COMPO+7, 'COMPILE', COMPI
DW      RFROM, DUPP, AT, COMMA ;compile address
DW      TWOP, TOR, EXIT      ;adjust return address

; LITERAL ( w -- )
;      Compile tos to code dictionary as an integer literal.

$COLON  IMEDD+7, 'LITERAL', LITER
DW      COMPI, DOLIT, COMMA, EXIT

; $, "    ( -- )
;      Compile a literal string up to next " .

$COLON  3, '$, "', STRCQ
DW      DOLIT, '"', WORDD      ;move string to code dictionary
DW      COUNT, PLUS          ;calculate aligned end of string
DW      CP, STORE, EXIT      ;adjust the code pointer

; RECURSE ( -- )
;      Make the current word available for compilation.

$COLON  IMEDD+7, 'RECURSE', RECUR
DW      LAST, AT, NAMET, COMMA, EXIT

```

```
;; Structures
```

```

; FOR      ( -- a )
;          Start a FOR-NEXT loop structure in a colon definition.

          $COLON  IMEDD+3, 'FOR', FOR
          DW      COMPI, TOR, HERE, EXIT

; BEGIN    ( -- a )
;          Start an infinite or indefinite loop structure.

          $COLON  IMEDD+5, 'BEGIN', BEGIN
          DW      HERE, EXIT

; NEXT     ( a -- )
;          Terminate a FOR-NEXT loop structure.

          $COLON  IMEDD+4, 'NEXT', NEXT
          DW      COMPI, DONXT, COMMA, EXIT

; UNTIL    ( a -- )
;          Terminate a BEGIN-UNTIL indefinite loop structure.

          $COLON  IMEDD+5, 'UNTIL', UNTIL
          DW      COMPI, QBRAN, COMMA, EXIT

; AGAIN    ( a -- )
;          Terminate a BEGIN-AGAIN infinite loop structure.

          $COLON  IMEDD+5, 'AGAIN', AGAIN
          DW      COMPI, BRAN, COMMA, EXIT

; IF       ( -- A )
;          Begin a conditional branch structure.

          $COLON  IMEDD+2, 'IF', IF
          DW      COMPI, QBRAN, HERE
          DW      DOLIT, 0, COMMA, EXIT

; AHEAD    ( -- A )
;          Compile a forward branch instruction.

          $COLON  IMEDD+5, 'AHEAD', AHEAD
          DW      COMPI, BRAN, HERE, DOLIT, 0, COMMA, EXIT

; REPEAT   ( A a -- )
;          Terminate a BEGIN-WHILE-REPEAT indefinite loop.

          $COLON  IMEDD+6, 'REPEAT', REPEA

```

```

        DW      AGAIN,HERE,SWAP,STORE,EXIT
; THEN      ( A -- )
;           Terminate a conditional branch structure.

$COLON     IMEDD+4,'THEN',THENN
DW        HERE,SWAP,STORE,EXIT

; AFT      ( a -- a A )
;           Jump to THEN in a FOR-AFT-THEN-NEXT loop the first time through.

$COLON     IMEDD+3,'AFT',AFT
DW        DROP,AHEAD,BEGIN,SWAP,EXIT

; ELSE     ( A -- A )
;           Start the false clause in an IF-ELSE-THEN structure.

$COLON     IMEDD+4,'ELSE',ELSEE
DW        AHEAD,SWAP,THENN,EXIT

; WHILE    ( a -- A a )
;           Conditional branch out of a BEGIN-WHILE-REPEAT loop.

$COLON     IMEDD+5,'WHILE',WHILE
DW        IFF,SWAP,EXIT

; ABORT"   ( -- ; <string> )
;           Conditional abort with an error message.

$COLON     IMEDD+6,'ABORT"',ABRTQ
DW        COMPI,ABORQ,STRCQ,EXIT

; $"      ( -- ; <string> )
;           Compile an inline string literal.

$COLON     IMEDD+2,'$',STRQ
DW        COMPI,STRQP,STRCQ,EXIT

; ."      ( -- ; <string> )
;           Compile an inline string literal to be typed out at run time.

$COLON     IMEDD+2,'."',DOTQ
DW        COMPI,DOTQP,STRCQ,EXIT

;; Name compiler

; ?UNIQUE  ( a -- a )
;           Display a warning message if the word already exists.

```

```

$COLON 7, '?UNIQUE', UNIQU
DW     DUPP, NAMEQ           ;?name exists
DW     QBRAN, UNIQ1         ;redefinitions are OK
D$     DOTQP, ' reDef '     ;but warn the user
DW     OVER, COUNT, TYPEE   ;just in case its not planned
UNIQ1: DW     DROP, EXIT

;   $,n      ( na -- )
;           Build a new dictionary name using the string at na.

$COLON 3, '$,n', SNAME
DW     DUPP, CAT            ;?null input
DW     QBRAN, PNAM1
DW     UNIQU                ;?redefinition
DW     DUPP, LAST, STORE   ;save na for vocabulary link
DW     HERE, SWAP          ;align code address
DW     TWOM                 ;link address
DW     CRRNT, AT, AT, OVER, STORE
DW     TWOM, DUPP, NP, STORE ;adjust name pointer
DW     STORE, EXIT         ;save code pointer
PNAM1: D$     STRQP, ' name' ;null input
DW     THROW

;; FORTH compiler

;   $COMPILE ( a -- )
;           Compile next word to code dictionary as a token or literal.

$COLON 8, '$COMPILE', SCOMP
DW     NAMEQ, QDUP          ;?defined
DW     QBRAN, SCOM2
DW     AT, DOLIT, IMEDD, ANDD ;?immediate
DW     QBRAN, SCOM1
DW     EXECU, EXIT         ;its immediate, execute
SCOM1: DW     COMMA, EXIT   ;its not immediate, compile
SCOM2: DW     TNUMB, ATEXE  ;try to convert to number
DW     QBRAN, SCOM3
DW     LITER, EXIT         ;compile number as integer
SCOM3: DW     THROW        ;error

;   CCOMPILE ( a -- )
;           Compile next byte to code dictionary as machine code.

$COLON 8, 'CCOMPILE', CCOMP
DW     NAMEQ, QDUP          ;?defined
DW     QBRAN, CCOM2
DW     AT, DOLIT, IMEDD, ANDD ;?immediate

```



```

        DW      QBRAN,CCOM1
        DW      EXECU,EXIT          ;its immediate, execute
CCOM1:   DW      DROP,EXIT          ;its not immediate,drop
CCOM2:   DW      TNUMB,ATEXE        ;try to convert to number
        DW      QBRAN,CCOM3
        DW      CCOMMA,EXIT        ;compile as code byte
CCOM3:   DW      THROW              ;error

;   OVERT      ( -- )
;               Link a new word into the current vocabulary.

        $COLON  5, 'OVERT', OVERT
        DW      LAST, AT, CRRNT, AT, STORE, EXIT

;   ;         ( -- )
;               Terminate a colon definition.

        $COLON  IMEDD+COMPO+1, ';', SEMIS
        DW      COMPI, EXIT, LBRAC, OVERT, EXIT

;   ]         ( -- )
;               Start compiling the words in the input stream.

        $COLON  1, ']', RBRAC
        DW      DOLIT, SCOMP, TEVAL, STORE, EXIT

;   call,     ( ca -- )
;               Assemble a call instruction to doLST.

        $COLON  5, 'call,', CALLC
        DW      DOLIT, CALLL, CCOMMA, EXIT ;Direct Threaded Code

;   :         ( -- ; <string> )
;               Start a new colon definition using next word as its name.

        $COLON  1, ':', COLON
        DW      TOKEN, SNAME
        DW      CALLC, RBRAC, EXIT

;   IMMEDIATE ( -- )
;               Make the last compiled word an immediate word.

        $COLON  9, 'IMMEDIATE', IMMED
        DW      DOLIT, IMEDD, LAST, AT, AT, ORR
        DW      LAST, AT, STORE, EXIT

```

```
;; Defining words
```

```

; USER      ( u -- ; <string> )
;           Compile a new user variable.

          $COLON 4, 'USER', USER
          DW      TOKEN, SNAME, OVERT, CALLC
          DW      COMPI, DOUSE, COMMA, EXIT

; CREATE    ( -- ; <string> )
;           Compile a new array entry without allocating code space.

          $COLON 6, 'CREATE', CREAT
          DW      TOKEN, SNAME, OVERT, CALLC
          DW      COMPI, DOVAR, EXIT

; VARIABLE  ( -- ; <string> )
;           Compile a new variable initialized to 0.

          $COLON 8, 'VARIABLE', VARIA
          DW      CREAT, DOLIT, 0, COMMA, EXIT

; CODE      ( -- )
;           Start a new code definition using next word as its name.

          $COLON 4, 'CODE', CODE
          DW      TOKEN, SNAME
          DW      DOLIT, CCOMP, TEVAL, STORE, EXIT

; ENDCODE   ( -- )
;           Terminate a code definition

          $COLON IMEDD+COMPO+7, 'ENDCODE', ENDCD
          DW      DOLIT, 48H, CCOMMA, DOLIT, 84H, CCOMMA           ; $NEXT
          DW      DOLIT, 48H, CCOMMA, DOLIT, 28H, CCOMMA
          DW      LBRAC, OVERT, EXIT

;; Tools

; _TYPE     ( b u -- )
;           Display a string. Filter non-printing characters.

          $COLON 5, '_TYPE', UTYPE
          DW      TOR                               ;start count down loop
          DW      BRAN, UTYPE2                       ;skip first pass
UTYP1:      DW      DUPP, CAT, TCHAR, EMIT          ;display only printable
          DW      ONEP                               ;increment address
UTYP2:      DW      DONXT, UTYPE1                   ;loop till done
          DW      DROP, EXIT

```

```

; dm+      ( a u -- a )
;          Dump u bytes from , leaving a+u on the stack.

$COLON 3, 'dm+', DMP
DW      OVER, DOLIT, 4, UDOTR      ;display address
DW      SPACE, TOR                ;start count down loop
DW      BRAN, PDUM2               ;skip first pass
PDUM1:   DW      DUPP, CAT, DOLIT, 3, UDOTR ;display numeric data
DW      ONEP                      ;increment address
PDUM2:   DW      DONXT, PDUM1      ;loop till done
DW      EXIT

; DUMP     ( a u -- )
;          Dump u bytes from a, in a formatted manner.

$COLON 4, 'DUMP', DUMP
DW      BASE, AT, TOR, HEX        ;save radix, set hex
DW      DOLIT, 16, SLASH         ;change count to lines
DW      TOR                      ;start count down loop
DUMP1:   DW      CR, DOLIT, 16, DDUP, DMP ;display numeric
DW      ROT, ROT
DW      SPACE, SPACE, UTYPE      ;display printable characters
DW      NUFQ, INVER             ;user control
DW      QBRAN, DUMP2
DW      DONXT, DUMP1            ;loop till done
DW      BRAN, DUMP3

DUMP2:   DW      RFROM, DROP      ;cleanup loop stack, early exit
DUMP3:   DW      DROP, RFROM, BASE, STORE ;restore radix
DW      EXIT

; .S      ( ... -- ... )
;          Display the contents of the data stack.

$COLON 2, '.S', DOTS
DW      CR, DEPTH                ;stack depth
DW      TOR                      ;start count down loop
DW      BRAN, DOTS2             ;skip first pass
DOTS1:   DW      RAT, PICK, DOT   ;index stack, display contents
DOTS2:   DW      DONXT, DOTS1    ;loop till done
D$      DOTQP, ' <sp'
DW      EXIT

; !CSP    ( -- )
;          Save stack pointer in CSP for error checking.

$COLON 4, '!CSP', STCSP
DW      SPAT, CSP, STORE, EXIT   ;save pointer

```

```

; ?CSP      ( -- )
;           Abort if stack pointer differs from that saved in CSP.

$COLON 4, '?CSP', QCSP
DW      SPAT, CSP, AT, XORR      ;compare pointers
D$      ABORQ, 'stacks'         ;abort if different
DW      EXIT

; >NAME     ( ca -- na | F )
;           Convert code address to a name address.

$COLON 5, '>NAME', TNAME
DW      CRRNT                   ;vocabulary link
TNAM1:  DW      TWOP, AT, QDUP   ;check all vocabularies
DW      QBRAN, TNAM4
DW      DDUP
TNAM2:  DW      AT, DUPP        ;?last word in a vocabulary
DW      QBRAN, TNAM3
DW      DDUP, NAMET, XORR      ;compare
DW      QBRAN, TNAM3
DW      TWOM                    ;continue with next word
DW      BRAN, TNAM2
TNAM3:  DW      SWAP, DROP, QDUP
DW      QBRAN, TNAM1
DW      SWAP, DROP, SWAP, DROP, EXIT
TNAM4:  DW      DROP, DOLIT, 0, EXIT ;false flag

; .ID      ( na -- )
;           Display the name at address.

$COLON 3, '.ID', DOTID
DW      QDUP                     ;if zero no name
DW      QBRAN, DOTI1
DW      COUNT, DOLIT, 01FH, ANDD ;mask lexicon bits
DW      UTYPE, EXIT             ;display name string
DOTI1:  D$      DOTQP, ' {noName}'
DW      EXIT

; WORDS    ( -- )
;           Display the names in the context vocabulary.

$COLON 5, 'WORDS', WORDS
DW      CR, CNTXT, AT           ;only in context
WORS1:  DW      AT, QDUP        ;?at end of list
DW      QBRAN, WORS2
DW      DUPP, SPACE, DOTID     ;display a name
DW      TWOM, NUFQ            ;user control
DW      QBRAN, WORS1

```

```

        DW      DROP
WORS2:  DW      EXIT

;; Hardware reset

;   VER      ( -- n )
;           Return the version number of this implementation.

        $COLON 3, 'VER', VERSN
        DW      DOLIT, VER*256+EXT, EXIT

;   hi      ( -- )
;           Display the sign-on message of eForth.

        $COLON 2, 'hi', HI
        DW      STOIO, CR ;initialize I/O
        D$      DOTQP, 'eForth v'
        DW      BASE, AT, HEX
        DW      VERSN, BDIGS, DIG, DIG
        DW      DOLIT, '.', HOLD
        DW      DIGS, EDIGS, TYPEE
        DW      BASE, STORE, CR, EXIT

;   'BOOT   ( -- a )
;           The application startup vector.

        $COLON 5, "'BOOT", TBOOT
        DW      DOVAR
        DW      HI ;application to boot

;   SEE     ( --word-- )
;           Decompiles word.
        $COLON 3, 'SEE', SEE
        DW      TICK
        DW      CR, ONEP
SEE1:   DW      DUPP, DUPP, SPACE, DOT, AT, DUPP
        DW      QBRAN, SEE2
        DW      TNAME
SEE2:   DW      QDUP
        DW      QBRAN, SEE3
        DW      DOTID
        DW      BRAN, SEE4
SEE3:   DW      DUPP, AT, UDOT
SEE4:   DW      TWOP, NUFQ
        DW      QBRAN, SEE1
        DW      DROP, EXIT

```

```

; ADCINIT      ( -- )
;              Init routine for starting ADC Interrupts
$CODE 7,'ADCINIT',ADCINIT
DB 68H,0FFH          ;V<FF
DB 69H,0C6H          ;A<C6
DB 63H,0F2H          ;(V/F2)<A
DB 69H,0              ;A<0
DB 63H,0F3H          ;(V/F3)<A
DB 4DH,0C8H          ;ANM <A
DB 48H,48H           ;SKIT FAD, reset INTFAD
DB 00                ;NOP
DB 64H,0EH,0FEH      ;ENABLE INTAD
$NEXT

; TMIDI        ( n -- )
;              Wait for last transmit, then send midi byte n.
$CODE 5,'TMIDI',TMIDI
DB 0A1H              ;POP BC
DB 0BH               ;A<C
DB 48H,4AH           ;SKIT FST, skip if interrupt
DB 0FDH              ;JMP TO SKIT
DB 4DH,0D8H          ;MOV TXB,A
$NEXT

; DELAY        ( n -- )
;              Wait for n loops.
$CODE 5,'DELAY',DELAY
DB 0A1H              ;POP BC
DB 53H               ;C<C-1, Skip if borrow
DB 0FEH              ;JMP
DB 52H               ;B<B-1, Skip if borrow
DB 0FCH              ;JMP
$NEXT

; LCD          ( n -- )
;              Load control n to LCD display.
$CODE 4,'LCD',LCD
DB 0A1H              ;POP BC
DB 0BH               ;A<C
DB 14H,0,0A0H        ;BC<A000
DB 39H               ;(BC)<A
$NEXT

; LLI          ( --- )
;              Sets RS=0 for LCD setup commands.
$CODE 3,'LLI',LLI
DB 64H,0AH,0EFH      ;Pc<Pc AND EF
$NEXT

```

```

; LLC          ( --- )
;              Sets RS=1 for LCD character loading
$CODE 3,'LLC',LLC
DB 64H,1AH,10H          ;Pc<Pc OR 10
$NEXT

; LI          ( n --- )
;              load LCD setup instruction n, exit ready for char loads
$COLON 2,'LI',LI
DW      LLI,LCD,LLC,EXIT

; LCDINIT    ( -- )
;              Initialize LCD display.
$COLON 7,'LCDINIT',LCDINIT
DW      DOLIT,0D7AH,DELAY
DW      DOLIT,038H,LI
DW      DOLIT,047EH,DELAY
DW      DOLIT,038H,LI
DW      DOLIT,017H,DELAY
DW      DOLIT,038H,LI
DW      DOLIT,017H,DELAY
DW      DOLIT,038H,LI
DW      DOLIT,017H,DELAY
DW      DOLIT,08H,LI
DW      DOLIT,017H,DELAY
DW      DOLIT,01H,LI
DW      DOLIT,01CCH,DELAY
DW      DOLIT,02H,LI
DW      DOLIT,01CCH,DELAY
DW      DOLIT,06H,LI
DW      DOLIT,17H,DELAY
DW      DOLIT,0EH,LI
DW      DOLIT,17H,DELAY
DW      EXIT

; #DISP      ( n,p --- )
;              Display n as a 3-digit number at LCD position p.
$COLON 5,'#DISP',NDISP
DW      DUPP,LI,SWAP
DW      BDIGS,DIG,DIG,DIG,EDIGS
DW      DROP,DUPP,CAT,LCD,ONEP
DW      DUPP,CAT,LCD,ONEP,CAT,LCD,LI,EXIT

; DISP      ( a,p --- )
;              Display packed string at a to LCD position p.
$COLON 4,'DISP',DISP
DW      LI,DUPP,CAT,TWOM,TOR

```

```

DISP1:      DW      ONEP
            DW      DUPP,CAT,LCD
            DW      DONXT,DISP1
            DW      DROP,EXIT

; CASE      ( n --- )
;           Execute one of a list of words pointed to by n.
$COLON     4,'CASE',CASE
            DW      RFROM,SWAP,TWOSL,PLUS
            DW      ATEXE,EXIT

; INCR      ( n,nmax --- n+1 )
;           Increment n mod nmax.
$COLON     4,'INCR',INCR
            DW      OVER,ONEP,LESS
            DW      QBRAN,INCR1
            DW      DROP,DOLIT,0
            DW      BRAN,INCR2
INCR1:     DW      ONEP
INCR2:     DW      EXIT

; DECR      ( n,nmax --- n-1 )
;           Decrement n mod nmax.
$COLON     4,'DECR',DECR
            DW      OVER,ONEM,ZLESS
            DW      QBRAN,DECR1
            DW      SWAP,DROP
            DW      BRAN,DECR2
DECR1:     DW      DROP
            DW      ONEM
DECR2:     DW      EXIT

; SW@      ( --- n )
;           Read Roland switches as a byte.
$CODE      3,'SW@',SWAT
            DB      4CH,0C0H      ;;A<PA
            DB      6AH,0         ;;B<0
            DB      1BH          ;;C<A
            DB      0B1H         ;;PUSH BC
$NEXT

; S@      ( --- n )
;           Return number of the lowest Roland switch on.
$CODE      2,'S@',SAT
            DB      4CH,0C0H      ;;A<PA
            DB      6BH,0         ;;C<0
            DB      74H,11H,0FFH  ;;A<A EXOR FF
            DB      74H,49H,0FFH  ;;A AND FF, SKIP IF NO ZERO

```



```

DB      0C4H          ;; JMP OUT
DB      43H           ;; C<C+1, LOOP1
DB      48H,1        ;; A SHIFT RIGHT, SKIP IF CARRY
DB      0FCH         ;; JMP LOOP1
DB      6AH,0        ;;B<0, OUT
DB      0B1H         ;;PUSH BC
$NEXT

```

```

; LED!      ( n --- )
;           Turn on/off Roland LED's.

```

```

$CODE     4, 'LED!',LEDB
DB      0A1H          ;;POP BC
DB      0BH           ;;A<C
DB      74H,9H,0FCH  ;;A<A AND FC
DB      74H,19H,1    ;;A<A OR 1
DB      4DH,0C1H     ;;PB<A
$NEXT

```

```

; eUPDAT    ( --- )
;           Move data from Slider Ram to Edit Buffer.

```

```

$CODE     6, 'eUPDAT',EUPDAT
DB      68H,0FFH     ;;V<FF
DB      6AH,0C6H     ;;B<C6
DB      1,0F0H       ;;A<(V/F0)
DB      1BH          ;;C<A
DB      29H          ;;A<(BC)
DB      63H,4        ;;(V/04)<A
DB      69H,38H     ;;A<70
DB      60H,43H     ;;C<C+A
DB      29H          ;;A<(BC)
DB      63H,6H       ;;(V/06)<A
DB      69H,38H     ;;A<38
DB      60H,43H     ;;C<C+A
DB      29H          ;;A<(BC)
DB      63H,7H       ;;(V/07)<A
$NEXT

```

```

; eLOAD     ( --- )
;           Load Edit Buffer data into Slider Memory.

```

```

$CODE     5, 'eLOAD',ELOAD
DB      68H,0FFH     ;;V<FF
DB      6AH,0C6H     ;;B<C6
DB      1,0          ;;A<(V/00)
DB      1BH          ;;C<A
DB      1,4          ;;A<(V/04)
DB      39H          ;;(BC)<A
DB      69H,38H     ;;A<38
DB      60H,43H     ;;C<C+A

```

```

DB      49H,0           ;;(BC)<0
DB      69H,70H        ;;A<38
DB      60H,43H        ;;C<C+A
DB      1,6            ;;A<(V/06)
DB      39H           ;;(BC)<A
DB      69H,38H        ;;A<38
DB      60H,43H        ;;C<C+A
DB      1,7            ;;A<(V/07)
DB      39H           ;;(BC)<A
$NEXT

```

```

; esUPDAT      ( --- )
;              Update only the Slider data of the Edit Buffer.

```

```

$CODE      7,'esUPDAT',ESUPDAT
DB      68H,0FFH       ;;V<FF
DB      6AH,0C6H       ;;B<C6
DB      1,0F0H         ;;A<(V/F0)
DB      1BH           ;;C<A
DB      29H           ;;A<(BC)
DB      63H,4         ;;(V/04)<A
$NEXT

```

```

; eSLD#        ( --- FF00 )
;              Edit Buffer Slider number.

```

```

$COLON     5,'eSLD#',ESLDN
DW         DOLIT,0FF00H,EXIT

```

```

; eFLD         ( --- FF01 )
;              Edit Buffer LCD Field.

```

```

$COLON     4,'eFLD',EFLD
DW         DOLIT,0FF01H,EXIT

```

```

; eBYTE1       ( --- FF07 )
;              Edit Buffer Midi Status/Chnl byte.

```

```

$COLON     6,'eBYTE1',EBYTE1
DW         DOLIT,0FF07H,EXIT

```

```

; eBYTE2       ( --- FF06 )
;              Edit Buffer Midi Key#, Controller#, or Program# byte.

```

```

$COLON     6,'eBYTE2',EBYTE2
DW         DOLIT,0FF06H,EXIT

```

```

; eBYTE3       ( --- FF04 )
;              Edit Buffer Slider value.

```

```

$COLON     6,'eBYTE3',EBYTE3
DW         DOLIT,0FF04H,EXIT

```

```

; eFLAG        ( --- FF05 )

```

```

;           Edit Buffer Midi Flag byte.
$COLON 5,'eFLAG',EFLAG
DW      DOLIT,0FF05H,EXIT

; FLD0      ( --- 80 )
;           LCD Field start.
$COLON 4,'FLD0',FLD0
DW      DOLIT,080H,EXIT

; FLD1      ( --- 86 )
;           LCD Field start.
$COLON 4,'FLD1',FLD01
DW      DOLIT,086H,EXIT

; FLD2      ( --- 8A )
;           LCD Field start.
$COLON 4,'FLD2',FLD2
DW      DOLIT,08AH,EXIT

; FLD3      ( --- 8D )
;           LCD Field start.
$COLON 4,'FLD3',FLD3
DW      DOLIT,08DH,EXIT

; FLD4      ( --- C0 )
;           LCD Field start.
$COLON 4,'FLD4',FLD4
DW      DOLIT,0C0H,EXIT

; FLD5      ( --- C9 )
;           LCD Field start.
$COLON 4,'FLD5',FLD5
DW      DOLIT,0C9H,EXIT

; FLD6      ( --- CD )
;           LCD Field start.
$COLON 4,'FLD6',FLD6
DW      DOLIT,0CDH,EXIT

; L0        ( --- a )
;           Packed string. 'a' is addr of count byte.
$COLON 2,'L0',L0
SD$ 'Slider'

; L1        ( --- a )
;           Packed string. 'a' is addr of count byte.
$COLON 2,'L1',L1
SD$ 'Setup#'

```

```

; L2          ( --- a )
;            Packed string. 'a' is addr of count byte.
$COLON 2,'L2',L2
SD$ '* MIDI Running *'

; L20         ( --- a )
;            Packed string. 'a' is addr of count byte.
$COLON 3,'L20',L20
SD$ 'Ch '

; L21         ( --- a )
;            Packed string. 'a' is addr of count byte.
$COLON 3,'L21',L21
SD$ 'Off'

; L40         ( --- a )
;            Packed string. 'a' is addr of count byte.
$COLON 3,'L40',L40
SD$ 'Key# '

; L41         ( --- a )
;            Packed string. 'a' is addr of count byte.
$COLON 3,'L41',L41
SD$ 'Key# A-T'

; L42         ( --- a )
;            Packed string. 'a' is addr of count byte.
$COLON 3,'L42',L42
SD$ 'Control#'

; L43         ( --- a )
;            Packed string. 'a' is addr of count byte.
$COLON 3,'L43',L43
SD$ 'Program#'

; L44         ( --- a )
;            Packed string. 'a' is addr of count byte.
$COLON 3,'L44',L44
SD$ 'Ch Press'

; L45         ( --- a )
;            Packed string. 'a' is addr of count byte.
$COLON 3,'L45',L45
SD$ 'Ptch Whl'

; L4X         ( --- a )
;            Packed string. 'a' is addr of count byte.

```

```

$COLON 3, 'L4X', L4X
SD$ '*****'

; L50      ( --- a )
;          Packed string. 'a' is addr of count byte.
$COLON 3, 'L50', L50
SD$ '***'

; LSTAT    ( n --- )
;          Choose a midi status label.
$COLON 5, 'LSTAT', LSTAT
DW      CASE, L4X, L40, L41, L42, L43, L44, L45, L4X, EXIT

; eDISP    ( --- )
;          Display the Edit buffer on the LCD
$COLON 5, 'eDISP', EDISP
DW      L0, FLD0, DISP, ESLDN, CAT, FLD01, NDISP
DW      EBYTE3, CAT, DOLIT, 80H, ANDD
DW      QBRAN, EDISP1
DW      L21, FLD2, DISP
DW      BRAN, EDISP2
EDISP1: DW      L20, FLD2, DISP
EDISP2: DW      EBYTE1, CAT, DUPP, DOLIT, 0FH, ANDD, FLD3, NDISP
DW      TWOSL, TWOSL, TWOSL, TWOSL, DOLIT, 7H, ANDD
DW      LSTAT, FLD4, DISP, EBYTE2, CAT, FLD5, NDISP
DW      EBYTE3, CAT, DOLIT, 7FH, ANDD, FLD6, NDISP
DW      DOLIT, 6H, EFLD, CSTOR, EXIT

; BL/R     ( fld --- pos )
;          Translates LCD field number to a position number.
$COLON 4, 'BL/R', BLR
DW      DUPP, EFLD, CAT, CASE
DW      FLD0, FLD01, FLD2, FLD3, FLD4, FLD5, FLD6
DW      EXIT

; BLEFT    ( --- )
;          Moves the LCD cursor to next field. Loads eFLD.
$COLON 5, 'BLEFT', BLEFT
DW      DOLIT, 40H, LEDB
DW      EFLD, CAT, DOLIT, 6, DECR, BLR, LI, EXIT

; BRIGHT   ( --- )
;          Moves the LCD cursor to next field. Loads eFLD.
$COLON 6, 'BRIGHT', BRIGHT
DW      DOLIT, 80H, LEDB
DW      EFLD, CAT, DOLIT, 6, INCR, BLR, LI, EXIT

; BLOAD    ( --- )

```

```

;           Load Buffer data shown on LCD into Slider Memory.
$COLON 5, 'BLOAD', BLOAD
DW      DOLIT, 4, LEDB
DW      ELOAD, EXIT

; BMIDI    ( --- )
;           Start the Midi program.
$COLON 5, 'BMIDI', BMIDI
DW      DOLIT, 1, LI
DW      L2, FLD0, DISP
DW      EXIT

; BUP      ( --- )
;           Increment value in LCD cursor field.
$COLON 3, 'BUP', BUP
DW      DOLIT, 10H, LEDB
DW      DOLIT, 1, EFLD, CAT
DW      DOLIT, 7, ANDD, CASE
DW      UD0, UD1, UD2, UD3, UD4, UD5, UD6, UD7, EXIT

; BDOWN    ( --- )
;           Decrement value in LCD cursor field.
$COLON 3, 'BDOWN', BDOWN
DW      DOLIT, 20H, LEDB
DW      DOLIT, 0, EFLD, CAT
DW      DOLIT, 7, ANDD, CASE
DW      UD0, UD1, UD2, UD3, UD4, UD5, UD6, UD7, EXIT

; U/D0     ( i/d --- )
;           Field increment/decrement routine.
$COLON 4, 'U/D0', UD0
DW      DROP, FLD0, LI, EXIT

; U/D7     ( i/d --- )
;           Field increment/decrement routine. (bogus field)
$COLON 4, 'U/D7', UD7
DW      DROP, EXIT

; U/D6     ( i/d --- )
;           Field increment/decrement routine.
$COLON 4, 'U/D6', UD6
DW      DROP, FLD6, LI, EXIT

; U/D1     ( i/d --- )
;           Field increment/decrement routine.
$COLON 4, 'U/D1', UD1
DW      ESLDN, CAT, DOLIT, 37H, ROT
DW      QBRAN, UD1A

```

```

        DW      INCR
        DW      BRAN,UD1B
UD1A:   DW      DECR
UD1B:   DW      CFLD1,EXIT

; CFLD1      ( sld# --- )
;           Change Slider# in field 1.  Update Edit buffer & LCD.
$COLON   5, 'CFLD1',CFLD1
        DW      ESLDN,CAT,EUPDAT,EDISP
        DW      DOLIT,1,EFLD,CSTOR,FLD01,LI,EXIT

; U/D2      ( i/d --- )
;           Field increment/decrement routine.
$COLON   4, 'U/D2',UD2
        DW      DROP,EBYTE3,CAT,DUPP,DOLIT,80H,ANDD
        DW      QBRAN,UD2A
        DW      DOLIT,7FH,ANDD,EBYTE3,CAT,L20,FLD2,DISP
        DW      BRAN,UD2B
UD2A:   DW      DOLIT,80H,ORR,EBYTE3,CSTOR,L21,FLD2,DISP
UD2B:   DW      FLD2,LI,EXIT

; U/D3      ( i/d --- )
;           Field increment/decrement routine.
$COLON   4, 'U/D3',UD3
        DW      EBYTE1,CAT,DOLIT,0FH,ANDD,DOLIT,0FH,ROT
        DW      QBRAN,UD3A
        DW      INCR
        DW      BRAN,UD3B
UD3A:   DW      DECR
UD3B:   DW      CFLD3,EXIT

; CFLD3      ( chnl --- )
;           Change midi channel in field 3.
$COLON   5, 'CFLD3',CFLD3
        DW      DUPP,EBYTE1,CAT,DOLIT,0F0H
        DW      ANDD,ORR,EBYTE1,CSTOR,FLD3,NDISP,EXIT

; U/D4      ( i/d --- )
;           Field increment/decrement routine.
$COLON   4, 'U/D4',UD4
        DW      EBYTE1,CAT,DOLIT,70H,ANDD
        DW      TWOSR,TWOSR,TWOSR,TWOSR,DOLIT,7,ROT
        DW      QBRAN,UD4A
        DW      INCR
        DW      BRAN,UD4B
UD4A:   DW      DECR
UD4B:   DW      CFLD4,EXIT

```

```

; CFLD4      ( status --- )
;           Change Midi operation label in field 4.
$COLON     5, 'CFLD4', CFLD4
DW         DUPP, TWOSL, TWOSL, TWOSL, TWOSL
DW         DOLIT, 80H, ORR, EBYTE1, CAT
DW         DOLIT, 0FH, ANDD, ORR, EBYTE1, CSTOR
DW         LSTAT, FLD4, DISP, FLD4, LI, EXIT

; U/D5      ( i/d --- )
;           Field increment/decrement routine.
$COLON     4, 'U/D5', UD5
DW         DOLIT, 0CFH, EBYTE1, CAT, DOLIT, 0F0H, ANDD, LESS
DW         QBRAN, UD5A
DW         L50, FLD5, DISP, FLD5, LI, DROP
DW         BRAN, UD5B
UD5A:      DW         DUPP, EBYTE2, CSTOR, FLD5, NDISP
UD5B:      DW         EXIT

;
;=====

LASTN      EQU      _NAME+4           ;last name address

NTOPP      EQU      _NAME-0           ;next available memory in ROM name
dictionary

CTOPP      EQU      $+0               ;next available memory in ROM code
dictionary
ROMSPC     EQU      NTOPP-CTOPP       ;UNUSED DICTIONARY ROM SPACE

MAIN      ENDS
END      ORIG

;=====

```


Modified Roland PG1000

PG1000 Assembly Listing


```

1      Page 60,132
2      ;=====
3      ;
4      ;   eForth 1.0 by Bill Muench and C. H. Ting, 1990
5      ;
6      ;   This is an implementation for the NEC 78C10 microcomputer by
7      ;   John Talbert, 1994, Oberlin Conservatory.
8      ;
9      ;   Register Use:      Interpreter Pointer = DE
10     ;                      Data Stack Pointer  = SP
11     ;                      Return Stack Pointer = HL
12     ;
13     ;                      Free to use: BC, EA, VA, Alternate Registers.
14     ;
15     ;   'doList' is accessed as a subroutine through a CALT instruction
16     ;   (Call to Jump Table). This shows up as a 'DB 80H' line
17     ;   in the $COLON and $USER Macros. When executed the
18     ;   processor jumps to an address vector located at 80H. The
19     ;   vectored 'doList' code is then located at 0F0H. The word
20     ;   'call,' was changed to load 80H into the code area for a
21     ;   doLST assembly.
22     ;
23     ;   A 9600 Baud serial I/O is provided. PortB/bit0 is used for serial
24     ;   output and PortC/bit3 (INT2) is used for the serial input. The
25     ;   serial input is interrupt driven with a vectored interrupt routine
26     ;   located at 0A0H. The code words ?RX, TX!, and !IO make up the
27     ;   rest of the serial I/O code. Three USER variables have been set
28     ;   up for use by these serial I/O routines: SERIN, which holds the
29     ;   received character and a flag; HAFBIT, which adjusts the software
30     ;   timing of the receiver to read in the middle of each bit frame
31     ;   (set it for 1/2 the BITIME minus 5); and BITIME, which adjusts the
32     ;   software for a specific baud rate (17H for 9600 baud assuming a 12Mhz
33     ;   processor clock).
34     ;
35     ;   The 78C10 is an 8-bit micro, therefore cell aligning to even addresses
36     ;   is unnecessary. The $ALIGN Macro was taken out along with the NOP's
37     ;   used for cell alignment in the other Macros. All occurrences of the
38     ;   word ALGND were erased also. The word SEE no longer works because it
39     ;   relies on cell alignment.
40     ;
41     ;   All of the system FORTH code is to be stored in ROM (up to 32K) starting
42     ;   at address 0000H. Then there is 2K of RAM starting at address C000H.
43     ;   This memory setup required the following changes:
44     ;       1) Return and Data stacks and TIB moved to RAM.
45     ;          (See the Memory allocation EQU assignments.)
46     ;       2) The USER variables were moved to the micro's
47     ;          internal RAM at FF00H to FFFFH.
48     ;       3) PAD word was changed to move the temporary buffer
49     ;          area to RAM space.
50     ;       4) The vocabulary pointers found in the word FORTH were
51     ;          moved to RAM space by creating two new USER variables,
52     ;          FHEAD and FLINK and changing DOVOC to read:
53     ;          DW FHEAD,CNTXT,STORE,EXIT.
54     ;       5) NTOP and CTOP were moved to RAM space to allow dictionary
55     ;          expansion into RAM space.
56     ;

```

```

57          ;      Several words were added to the ROM Dictionary.  The simple operators
58          ;      1+,1-,2+,2-,2*,2/, were defined in machine code.  The words C, ,
59          ;      CCOMPILE, CODE, and ENDCODE were created to enable the creation
60          ;      of code definition.
61          ;
62          ;      The NEC78C10 offers the following advantages:
63          ;      1) Ten 16-bit internal registers and a 16-bit ALU.
64          ;      Many 16-bit instructions for those FORTH stack operations.
65          ;      2) Three 8-bit I/O ports.
66          ;      3) Eight 8-bit Analog to Digital Converters.
67          ;      4) Internal counters and programmable clock generators.
68          ;      5) Internal hardware serial I/O.  (can be used for MIDI I/O).
69          ;      6) 64K address space including 256 bytes of internal RAM.
70          ;
71          ;
72          ;=====
73          ;; Version control
74 = 0001          VER          EQU          01H          ;major release version
75 = 0001          EXT          EQU          01H          ;minor extension
76
77          ;; Constants
78 = 0040          COMPO        EQU          040H        ;lexicon compile only bit
79 = 0080          IMEDD        EQU          080H        ;lexicon immediate bit
80 = 7F1F          MASKK        EQU          07F1FH      ;lexicon bit mask
81
82 = 0002          CELLL        EQU          2           ;size of a cell
83 = 000A          BASEE        EQU          10          ;default radix
84 = 0006          VOCSS        EQU          6           ;depth of vocabulary stack
85
86 = 0008          BKSPP        EQU          8           ;backspace
87 = 000A          LF           EQU          10          ;line feed
88 = 000D          CRR          EQU          13          ;carriage return
89 = 001B          ERR          EQU          27          ;error escape
90 = 0027          TIC          EQU          39          ;tick
91
92 = 0080          CALLL        EQU          80H         ;CALT opcodes
93
94          ;; Memory allocation      0//code>--//--<name//up>--<sp//tib>--rp//em
95
96 = 0100          COLDD        EQU          00100H      ;cold start
97 = C2F0          RPP          EQU          0C2F0H      ;start of return stack
(RP0)
98 = C200          TIBB        EQU          0C200H      ;terminal input buffer
(TIB)
99 = C1F0          SPP          EQU          0C1F0H      ;start of data stack (SP0)
100 = FF00          UPP          EQU          0FF00H      ;start of user area (UP0)
101 = 3FFD          NAMEE        EQU          03FFDH      ;name dictionary
102 = 0300          CODEE        EQU          00300H      ;code dictionary
103 = C390          CTOP        EQU          0C390H      ;RAM code dict. expansion
104 = C7FF          NTOP        EQU          0C7FFH      ;RAM name dict. expansion
105 = C300          PADD        EQU          0C300H      ;PAD area
106
107          ;; Initialize assembly variables
108
109 = 0000          _LINK        = 0                   ;force a null link
110 = 3FFD          _NAME        = NAMEE              ;initialize name pointer
111 = 0300          _CODE        = CODEE              ;initialize code pointer
112 = 0008          _USER        = 4*CELLL            ;first user variable offset

```

```

113
114           ;; Define assembly macros
115
116           ;       Compile a code definition header.
117
118           $CODE   MACRO   LEX,NAME,LABEL
119           LABEL:           ;;assembly label
120           _CODE   = $           ;;save code pointer
121           _LEN    = (LEX AND 01FH)/CELLL           ;;string cell count, round down
122           _NAME   = _NAME-(( _LEN+3)*CELLL)           ;;new header on cell boundary
123           ORG     _NAME           ;;set name pointer
124           DW      _CODE,_LINK           ;;token pointer and link
125           _LINK   = $           ;;link points to a name string
126           DB      LEX,NAME           ;;name string
127           ORG     _CODE           ;;restore code pointer
128           ENDM
129
130           ;       Compile a colon definition header.
131
132           $COLON  MACRO   LEX,NAME,LABEL
133           $CODE   LEX,NAME,LABEL
134           DB 80H           ;;include CALT doLIST
135           ENDM
136
137           ;       Compile a user variable header.
138
139           $USER   MACRO   LEX,NAME,LABEL
140           $CODE   LEX,NAME,LABEL
141           DB 80H           ;;include CALT doLIST
142           DW      DOUSE,_USER           ;;followed by doUSER and offset
143           _USER   = _USER+CELLL           ;;update user area offset
144           ENDM
145
146           ;       Compile an inline string.
147
148           D$      MACRO   FUNCT,STRNG
149           DW      FUNCT           ;;function
150           _LEN    = $           ;;save address of count byte
151           DB      0,STRNG           ;;count byte and string
152           _CODE   = $           ;;save code pointer
153           ORG     _LEN           ;;point to count byte
154           DB      _CODE-_LEN-1           ;;set count
155           ORG     _CODE           ;;restore code pointer
156           ENDM
157
158           ;       Compile a stored string.
159
160           SD$     MACRO   STRNG
161           DW DOLIT
162           _LEN    = $ + 4           ;;save address of count byte
163           DW      _LEN,EXIT           ;;save cnt address on stack
164           DB      0,STRNG           ;;count byte and string
165
166           ORG     _CODE   = $           ;;save code pointer
167           ORG     _LEN           ;;point to count byte
           DB      _CODE-_LEN-1           ;;set count

```

```

168          ORG      _CODE                      ;;restore code pointer
169          ENDM
170
171          ;      Assemble inline direct threaded code ending.
172
173          $NEXT   MACRO
174          DB 48H,84H                          ;;EA<(DE)++,next code address into
AX
175          DB 48H,28H                          ;;JMP EA,jump directly to code
address
176          ENDM
177
178          ;; Main entry points and COLD start data
179
180          MAIN    SEGMENT
181          ASSUME  CS:MAIN,DS:MAIN,ES:MAIN,SS:MAIN
182
183          ORG     0000H
184
185          0000   54 00 01 00          ORIG:   DB 54H,00,01,00          ;RESET vector, JMP 0100H
186          0004   AA 62 00 00          DB 0AAH,62H,0,0          ;NMI vector, EI RETI
187          0008   0008[                DB 8 DUP(0)              ;INT T0/T1 vector
188                   00
189                   ]
190
191          0010   54 A0 00                DB 54H,0A0H,00H, 5 DUP(0)          ;INT1/2 vector, JMP 00A0H
192                   0005[
193                   00
194                   ]
195
196          0018   0008[                DB 8 DUP(0)              ;INT E1/E0 vector
197                   00
198                   ]
199
200          0020   54 00 02                DB 54H,00,02, 5 DUP(0)          ;INT EIN/AD vector, JMP
0200H
201                   0005[
202                   00
203                   ]
204
205          0028   0008[                DB 8 DUP(0)              ;INT SR/ST vector
206                   00
207                   ]
208
209          0030   0030[                DB 48 DUP(0)             ;FREE
210                   00
211                   ]
212
213          0060   0020[                DB 32 DUP(0)            ;SOFTI vector at 0060H
214                   00
215                   ]
216
217
218          ORG     00A0H
219
220          ; Vectored INT2 routine for Serial Input from Host Computer.
221          ; Uses address FFF0 as a counter location - do not use elsewhere!
222          00A0   B1                      DB 0B1H                  ;PUSH BC
223          00A1   B2                      DB 0B2H                  ;PUSH DE

```

```

224 00A2 B0          DB 0B0H          ;PUSH VA
225 00A3 68 FF      DB 68H,0FFH      ;MVI, V<FF
226 00A5 71 F0 07   DB 71H,0F0H,07H ;MVIW, (V/F0)<07, number of bits to
receive.
227 00A8 70 1F 4C FF DB 70H,1FH,04CH,0FFH ;LBCD, BC<(FF4C), wait for a half bit.
228 00AC 53          DB 53H          ;DCR, C<C-1 skip, LOOP1
229 00AD FE          DB 0FEH         ;JR, Jump to loop1
230 00AE 52          DB 52H          ;DCR, B<B-1 skip
231 00AF FC          DB 0FCH         ;JR, Jump to loop1
232 00B0 70 1F 4E FF DB 70H,1FH,4EH,0FFH ;LBCD, BC<(FF4E),wait 1 bit time, LOOP2
233 00B4 53          DB 53H          ;DCR, C<C-1 skip
234 00B5 FE          DB 0FEH         ;JR, Jump to loop2
235 00B6 52          DB 52H          ;DCR, B<B-1 skip
236 00B7 FC          DB 0FCH         ;JR, Jump to loop2
237 00B8 4C C2      DB 04CH,0C2H    ;MOV, A<PC, read serial input on pc3
238 00BA 48 31 48 31 DB 48H,31H,48H,31H ;Rotate PC3 bit into Cy
239 00BE 48 31 48 31 DB 48H,31H,48H,31H ;RLR, A rotate right 4xs
240 00C2 0C          DB 0CH          ;MOV, A<D, D collects the bits
241 00C3 48 31      DB 48H,31H     ;RLR, shift in next bit, CY to top of D
242 00C5 1C          DB 1CH          ;MOV, D<A
243 00C6 30 F0      DB 30H,0F0H    ;DCRW, (V/F0)<(V/F0)-1 skip
244 00C8 E7          DB 0E7H         ;JR, Jump to loop2 for next bit.
245 00C9 70 1F 4E FF DB 70H,1FH,4EH,0FFH ;LBCD, BC<(FF4E)
246 00CD 53 FE 52 FC DB 53H,0FEH,52H,0FCH ;DCR JR DCR JR, stop bit loop time.
247 00D1 71 4B FF   DB 71H,04BH,0FFH ;MVIW, (V/4B)<FF, load flag
248 00D4 0C 63 4A   DB 0CH,63H,04AH ;MOV STAW, A<D (V/4A)<A, load data
249 00D7 A0 A2 A1     DB 0A0H,0A2H,0A1H ;POP, restore AV DE and BC
250 00DA 48 44 00   DB 48H,44H,0   ;SKIT,NOP
251 00DD AA 62      DB 0AAH,062H   ;EI RETI, enable interrupts and return
252
253 ;; Kernel doLST routine. Always accessed by the CALT instruction: 80H
254 ;; which is a Call Subroutine to jump to address vector located at 0080H.
255
256 00F0          ORG 00F0H
257 00F0 33 33      DB 33H,33H     ;HL<HL-2
258 00F2 A6          DB 0A6H        ;EA<DE
259 00F3 48 93      DB 48H,93H     ;(HL)<EA
260 00F5 A2          DB 0A2H        ;POP DE previously pushed by CALT
261 00F6 48 84      DB 48H,84H     ;EA<(DE)++, $NEXT
262 00F8 48 28      DB 48H,28H     ;JMP EA
263 0080          ORG 0080H
264 0080 F0 00      DB 0F0H,0     ; set up vector to doLST
265
266 0100          ORG COLDD
267 0100 69 0F 4D D0 DB 69H,0FH,4DH,0D0H ;Beginning of Cold Boot
268 0104 69 FF 4D D2 DB 69H,0FFH,4DH,0D2H ;MM<0F, memory map (11-8)
269 0108 69 00 4D D3 DB 69H,00H,4DH,0D3H ;MA<FF, pa inputs (4-2)
270 010C 64 01 05   DB 64H,01H,05H ;MB<00, pb outputs (4-6)
271 010F 4D D7      DB 4DH,0D7H    ;PB<5
272 0111 69 0A 4D D4 DB 69H,0AH,4DH,0D4H ;MF<00, pf outputs (4-15)
273 0115 69 0B 4D D1 DB 69H,0BH,4DH,0D1H ;MC<0A, pc1/3 inputs (4-9)
274 0119 64 02 04   DB 64H,02H,04H ;MCC<0B, pc mode (4-8)
275 011C 64 81 06   DB 64H,81H,06H ;PC<04
276 011F 69 4E 4D CA DB 69H,4EH,4DH,0CAH ;SMH<06, serial mode (7-7)
277 0123 04          DB 04H         ;SML<4E, serial mode (7-9)
278 0124 F0          DB 04H         ;SP<SPP, stack pointer=data stack
279 0125 C1          DB LOW SPP
                DB HIGH SPP

```

```

280 0126 34          DB 34H          ;HL<RPP, HL=return stack pointer
281 0127 F0          DB LOW RPP
282 0128 C2          DB HIGH RPP
283 0129 69 00 4D E8 DB 69H,00H,4DH,0E8H ;ZCM<0, zero cross disabled (3-26)
284 012D 68 FF          DB 68H,0FFH          ;V<FF
285 012F 10 68 FF 69 00 DB 10H,68H,0FFH,69H,0 ;V'<FF, A"<0, V<FF, A<0
286
287                ;; timer setups for Midi and LCD use
288 0134 69 64 4D DA          DB 69H,64H,4DH,0DAH ;TM0<64, timer0 (5-1)
289 0138 69 FF 4D DB          DB 69H,0FFH,4DH,0DBH ;TM1<FF, timer1 (5-1)
290 013C 64 85 B3          DB 64H,85H,0B3H ;TMM<B3, timer mode (5-6)
291 013F 44 60 EA 48 D3          DB 44H,60H,0EAH,48H,0D3H ;ETM1<EA = EA60 (6-2)
292 0144 64 83 CC          DB 64H,83H,0CCH ;EOM<CC, timer event mode (6-14)
293 0147 69 5C 4D CC          DB 69H,5CH,4DH,0CCH ;ETMM<5C, timer event mode (6-11)
294
295 014B 54 00 03          DB 54H,00,03H ;JMP to 0300, high level cold start
296                ;COLD WORD MOVED TO THE START OF CODE AREA.
297                ;ATTEMPTED TO AUTOMATE-JMP COLD-WITH $JUMP
298                ;BUT MACRO PRODUCES ERROR CODES.
299
300                ; COLD start moves the following to USER variables.
301                ; MUST BE IN SAME ORDER AS USER VARIABLES.
302
303 014E 0004[          UZERO:      DW      4 DUP (0)          ;reserved
304          0000
305                ]
306
307 0156 C1F0          DW      SPP          ;SP0
308 0158 C2F0          DW      RPP          ;RP0
309 015A 032E R        DW      QRX          ;'?KEY
310 015C 0347 R        DW      TXSTO         ;'EMIT
311 015E 0D34 R        DW      ACCEP         ;'EXPECT
312 0160 0D09 R        DW      KTAP          ;'TAP
313 0162 0347 R        DW      TXSTO         ;'ECHO
314 0164 0E2A R        DW      DOTOK         ;'PROMPT
315 0166 000A          DW      BASEE         ;BASE
316 0168 0000          DW      0          ;tmp
317 016A 0000          DW      0          ;SPAN
318 016C 0000          DW      0          ;>IN
319 016E 0000          DW      0          ;#TIB
320 0170 C200          DW      TIBB         ;TIB
321 0172 0000          DW      0          ;CSP
322 0174 0DEE R        DW      INTER        ;'EVAL
323 0176 0963 R        DW      NUMBQ        ;'NUMBER
324 0178 0000          DW      0          ;HLD
325 017A 0000          DW      0          ;HANDLER
326 017C 0000          DW      0          ;CONTEXT pointer
327 017E 0006[          DW      VOCSS DUP (0) ;vocabulary stack
328          0000
329                ]
330
331 018A 0000          DW      0          ;CURRENT pointer
332 018C 0000          DW      0          ;vocabulary link pointer
333 018E 0000          DW      0          ;FORTH HEAD
334 0190 0000          DW      0          ;FORTH LINK
335 0192 C390          DW      CTOP         ;CP

```



```

336 0194 C7FF          DW      NTOP          ;NP
337 0196 3527          DW      LASTN         ;LAST
338 0198 0000          DW      0              ;SERIN host receive char & flag
339 019A 0006          DW      06H           ;HAFBIT time for serial host,
340                                     ; (1/2 BITIME - 5)
341 019C 0016          DW      16H           ;BITIME baud for serial host
342
343 019E              ULAST:
344
345 0200              ORG 0200H
346
347                  ;      Interrupt routine for Analog to Digital Converters
348
349 0200 10            DB 10H              ;EXA
350 0201 11            DB 11H              ;EXX
351                  ; Load ADC Address and Counter into HL.  Uses FFF2 and FFF3.
352 0202 68 FF        DB 68H,0FFH         ;V'<FF
353 0204 01 F2        DB 01H,0F2H         ;A<(V/F2)
354 0206 1E           DB 1EH              ;H<A
355 0207 01 F3        DB 01H,0F3H         ;A<(V/F3)
356 0209 1F           DB 1FH              ;L<A
357                  ; Store ADC 0.
358 020A 2B           DB 2BH              ;A<(HL)
359 020B 57 80        DB 57H,80H         ;A AND 80, Skip if zero
360 020D CD           DB 0CDH           ;Jump to EXIT if slider is disabled.
361 020E 1A           DB 1AH              ;B<A
362 020F 4C E0        DB 4CH,0E0H        ;A<CR0
363 0211 48 21        DB 48H,21H         ;A Shift right, Midi is 7 bits, throw LSB.
364 0213 60 6A        DB 60H,6AH         ;B-A, Skip if not zero
365 0215 C5           DB 0C5H           ;Jump to EXIT if slider has not changed.
366 0216 3D           DB 3DH              ;(HL)<A, Store slider data, 0 in top bit.
367 0217 69 FF        DB 69H,0FFH         ;A<FF
368 0219 BF 38        DB 0BFH,38H        ;(HL+38)<A, Store slider change flag.
369 021B 32           DB 32H              ;HL<HL+1, EXIT
370                  ; Store ADC 1.
371 021C 2B           DB 2BH              ;A<(HL)
372 021D 57 80        DB 57H,80H         ;A AND 80, Skip if zero
373 021F CD           DB 0CDH           ;Jump to EXIT if slider is disabled.
374 0220 1A           DB 1AH              ;B<A
375 0221 4C E1        DB 4CH,0E1H        ;A<CR1
376 0223 48 21        DB 48H,21H         ;A Shift right, Midi is 7 bits, throw LSB.
377 0225 60 6A        DB 60H,6AH         ;B-A, Skip if not zero
378 0227 C5           DB 0C5H           ;Jump to EXIT if slider has not changed.
379 0228 3D           DB 3DH              ;(HL)<A, Store slider data, 0 in top bit.
380 0229 69 FF        DB 69H,0FFH         ;A<FF
381 022B BF 38        DB 0BFH,38H        ;(HL+38)<A, Store slider change flag.
382 022D 32           DB 32H              ;HL<HL+1, EXIT
383                  ; Store ADC 2.
384 022E 2B           DB 2BH              ;A<(HL)
385 022F 57 80        DB 57H,80H         ;A AND 80, Skip if zero
386 0231 CD           DB 0CDH           ;Jump to EXIT if slider is disabled.
387 0232 1A           DB 1AH              ;B<A
388 0233 4C E2        DB 4CH,0E2H        ;A<CR2
389 0235 48 21        DB 48H,21H         ;A Shift right, Midi is 7 bits, throw LSB.
390 0237 60 6A        DB 60H,6AH         ;B-A, Skip if not zero
391 0239 C5           DB 0C5H           ;Jump to EXIT if slider has not changed.

```

```

392 023A 3D          DB 3DH          ;(HL)<A, Store slider data, 0 in top bit.
393 023B 69 FF      DB 69H,0FFH    ;A<FF
394 023D BF 38      DB 0BFH,38H    ;(HL+38)<A, Store slider change flag.
395 023F 32          DB 32H          ;HL<HL+1, EXIT
396                ; Store ADC 3.
397 0240 2B          DB 2BH          ;A<(HL)
398 0241 57 80      DB 57H,80H     ;A AND 80, Skip if zero
399 0243 CD          DB 0CDH         ;Jump to EXIT if slider is disabled.
400 0244 1A          DB 1AH          ;B<A
401 0245 4C E3      DB 4CH,0E3H    ;A<CR3
402 0247 48 21      DB 48H,21H     ;A Shift right, Midi is 7 bits, throw LSB.
403 0249 60 6A      DB 60H,6AH     ;B-A, Skip if not zero
404 024B C5          DB 0C5H         ;Jump to EXIT if slider has not changed.
405 024C 3D          DB 3DH          ;(HL)<A, Store slider data, 0 in top bit.
406 024D 69 FF      DB 69H,0FFH    ;A<FF
407 024F BF 38      DB 0BFH,38H    ;(HL+38)<A, Store slider change flag.
408 0251 32          DB 32H          ;HL<HL+1, EXIT
409                ; Update Counters
410 0252 0F          DB 0FH          ;A<L
411 0253 37 37      DB 37H,37H     ;A-37H, Skip if borrow
412 0255 69 00      DB 69H,0       ;A<0, Reset counter after 56D sliders.
413 0257 63 F3      DB 63H,0F3H    ;(V/F3)<A, Load counter.
414 0259 48 25 48 25 DB 48H,25H,48H,25H ;A shift logical left 2xs.
415 025D 1A          DB 1AH          ;B<A
416 025E 74 0A E0      DB 74H,0AH,0E0H ;B<B AND E0
417 0261 4C C2      DB 4CH,0C2H    ;A<Pc
418 0263 07 1F      DB 07H,1FH     ;A<A AND 1F
419 0265 60 9A      DB 60H,9AH     ;A<A OR B
420 0267 4D C2      DB 4DH,0C2H    ;Pc<A, Load high 3 bits of slider select.
421 0269 64 90 08      DB 64H,90H,08H ;Invert ANM bit and restart conversion.
422                ; Return from Interrupt.
423 026C 10          DB 10H          ;EXA
424 026D 11          DB 11H          ;EXX
425 026E AA          DB 0AAH         ;EI
426 026F 62          DB 62H          ;RETI
427
428 0300                ORG CODEE                ;start code dictionary
429
430                ; COLD ( -- )
431                ; The hilevel cold start sequence.
432 = 0300                CCOLD = $
433                $COLON 4,'COLD',COLD
434 0300                2 COLD: ;
435 3FF3                2 ORG _NAME ;
436 3FF3 0300 R 0000      2 DW _CODE,_LINK ;
437 3FF7 04 43 4F 4C 44  2 DB 4,'COLD' ;
438 0300                2 ORG _CODE ;
439 0300 80              1 DB 80H ;
440 0301 038D R 014E R 038D R COLD1: DW DOLIT,UZERO,DOLIT,UPP
441 FF00
442 0309 038D R 0050 0803 R DW DOLIT,ULAST-UZERO,CMOVE ;initialize user area
443 030F 0E74 R DW PRESE ;initialize stack and TIB
444 0311 12DC R 07F6 R DW TBOOT,ATEXE ;application boot
445 0315 0538 R 04FD R 03DE R DW FORTH,CNTXT,AT,DUPP ;initialize search order
446 043B R
447 031D 0502 R 07BA R 10BF R DW CRRNT,DSTOR,OVERT

```

```

448 0323 1363 R          DW      LCDINIT          ;initialize LCD
449 0325 0ECC R          DW      QUIT            ;start interpretation
450 0327 03CC R 0301 R   DW      BRAN,COLD1        ;just in case
451
452
453           ;; Device dependent I/O
454
455           ;   BYE      ( -- )
456           ;           Exit eForth.
457           $CODE      3,'BYE',BYE
458 032B          1      BYE:                          ;
459 3FEB          1      ORG      _NAME                ;
460 3FEB 032B R 3FF7 R   1      DW      _CODE,_LINK    ;
461 3FEF 03 42 59 45    1      DB      3,'BYE'        ;
462 032B          1      ORG      _CODE                ;
463 032B 54 00 00      1      DB      54H,0,0          ;JMP Reset Vector
464
465           ;   ?RX     ( -- c T | F )
466           ;           Return input character and true, or a false if no input.
467
468           $CODE      3,'?RX',QRX
469 032E          1      QRX:                          ;
470 3FE3          1      ORG      _NAME                ;
471 3FE3 032E R 3FEF R   1      DW      _CODE,_LINK    ;
472 3FE7 03 3F 52 58    1      DB      3,'?RX'        ;
473 032E          1      ORG      _CODE                ;
474 032E 68 FF          DB      68H,0FFH              ;MVI, V<FF
475 0330 01 4B          DB      01H,4BH              ;LDAW, A<(V/4B) read serial-in flag
476 0332 47 FF          DB      47H,0FFH              ;ONI, A AND FF skip if flag not zero
477 0334 CA            DB      0CAH                  ;JR, jump ahead1
478 0335 71 4B 00      DB      71H,04BH,0              ;MVIW, (V/4B)<0, reset flag to zero
479 0338 70 1F 4A FF    DB      70H,1FH,4AH,0FFH          ;LBCD, BC<(FF4A), read serin data
480 033C B1            DB      0B1H                  ;PUSH BC, push serial input data to stack
481 033D 69 FF          DB      69H,0FFH              ;A<FF
482 033F 1B            DB      1BH                  ;C<A, AHEAD1
483 0340 6A 00          DB      6AH,0              ;B<0
484 0342 B1            DB      0B1H                  ;PUSH BC, push serial input flag to stack
485
486 0343 48 84          1      DB      48H,84H          ;
487 0345 48 28          1      DB      48H,28H          ;
488
489
490           ;   TX!     ( c -- )
491           ;           Send character c to the output device.
492
493           $CODE      3,'TX!',TXSTO
494 0347          1      TXSTO:                        ;
495 3FDB          1      ORG      _NAME                ;
496 3FDB 0347 R 3FE7 R   1      DW      _CODE,_LINK    ;
497 3FDF 03 54 58 21    1      DB      3,'TX!'        ;
498 0347          1      ORG      _CODE                ;
499 0347 BA            DB      0BAH                  ;Disable Interrupts
500 0348 A1            DB      0A1H                  ;POP BC, pop char into C
501 0349 B2            DB      0B2H                  ;PUSH DE, store interpreter pointer
502 034A 0B 1C          DB      0BH,1CH                  ;A<C, D<A, char in A and D
503 034C 68 FF          DB      68H,0FFH              ;V<FF

```

```

504 034E 71 F0 07          DB 71H,0F0H,07H          ;(V/F0)<7
505 0351 60 91          DB 60H,91H          ;A<A EXOR A
506 0353 6D 01          DB 6DH,01H          ;E<01
507 0355 4D C1          DB 4DH,0C1H          ;PB<A
508 0357 70 1F 4E FF      DB 70H,1FH,04EH,0FFH ;BC<(FF4E) set baud, LOOP1
509 035B 53 FE 52 FC      DB 53H,0FEH,52H,0FCH ;C<C-1, JR, B<B-1, JR, jr to loop1
510 035F 0C          DB 0CH          ;A<D
511 0360 07 01          DB 07H,01H          ;A<A AND 01
512 0362 4D C1          DB 4DH,0C1H          ;PB<A, send a bit
513 0364 0C          DB 0CH          ;A<D
514 0365 48 31          DB 48H,31H          ;A rotate logical right
515 0367 1C          DB 1CH          ;D<A
516 0368 00 00 00 00      DB 0,0,0,0          ;NOPS to make rec loop = transmit loop.
517 036C 30 F0          DB 30H,0F0H          ;(V/F0)<(V/F0)-1 skip
518 036E E8          DB 0E8H          ;JR, jump to loop1
519 036F 0D          DB 0DH          ;A<E
520 0370 51          DB 51H          ;A<A-1 skip
521 0371 C6          DB 0C6H          ;JR, jump to loop2
522 0372 A2          DB 0A2H          ;POP DE, restore interpreter pointer
523 0373 AA          DB 0AAH          ;Enable Interrupts
524          $NEXT          ;End of routine
525 0374 48 84          1          DB 48H,84H          ;
526 0376 48 28          1          DB 48H,28H          ;
527
528 0378 6C 03          DB 6CH,03H          ;D<03, LOOP2
529 037A 1D          DB 1DH          ;E<A
530 037B 71 F0 01          DB 71H,0F0H,01      ;(V/F0)<01
531 037E 4F D7          DB 4FH,0D7H          ;JRE, jump to loop1
532
533          ; !IO          ( -- )
534          ;          Initialize the serial I/O devices.
535
536          $CODE 3,'!IO',STOIO
537 0380          1          STOIO:          ;
538 3FD3          1          ORG          _NAME          ;
539 3FD3 0380 R 3FDF R      1          DW          _CODE, _LINK          ;
540 3FD7 03 21 49 4F      1          DB          3,'!IO'          ;
541 0380          1          ORG          _CODE          ;
542 0380 69 EF 4D C7      DB 69H,0EFH,4DH,0C7H ;MKL<EF, enable int2 interrupt and
543 0384 69 FF 4D C6      DB 69H,0FFH,4DH,0C6H ;MKH<FF, disable all others with mask
544 0388 AA          DB 0AAH          ;EI, enable interrupt
545          $NEXT
546 0389 48 84          1          DB 48H,84H          ;
547 038B 48 28          1          DB 48H,28H          ;
548
549
550          ;; The kernel
551
552          ; doLIT          ( -- w )
553          ;          Push an inline literal.
554
555          $CODE COMPO+5,'doLIT',DOLIT
556 038D          1          DOLIT:          ;
557 3FC9          1          ORG          _NAME          ;
558 3FC9 038D R 3FD7 R      1          DW          _CODE, _LINK          ;
559 3FCD 45 64 6F 4C 49 54 1          DB          COMPO+5,'doLIT'          ;

```

```

560 038D          1      ORG      _CODE                      ;
561 038D  48 84          DB  48H,84H                      ;EA<(DE)++
562 038F  B4              DB  0B4H                      ;PUSH EA
563              $NEXT
564 0390  48 84          1      DB  48H,84H                      ;
565 0392  48 28          1      DB  48H,28H                     ;
566
567              ; EXIT      ( -- )
568              ;          Terminate a colon definition.
569
570              $CODE  4,'EXIT',EXIT
571 0394          1      EXIT:                      ;
572 3FBF          1      ORG      _NAME                      ;
573 3FBF  0394 R 3FCD R    1      DW      _CODE,_LINK                    ;
574 3FC3  04 45 58 49 54  1      DB      4,'EXIT'                      ;
575 0394          1      ORG      _CODE                      ;
576 0394  48 85          DB  48H,85H                      ;EA<(HL)++
577 0396  B6              DB  0B6H                      ;DE<EA
578              $NEXT
579 0397  48 84          1      DB  48H,84H                      ;
580 0399  48 28          1      DB  48H,28H                     ;
581
582              ; EXECUTE  ( ca -- )
583              ;          Execute the word at ca.
584
585              $CODE  7,'EXECUTE',EXECU
586 039B          1      EXECU:                      ;
587 3FB3          1      ORG      _NAME                      ;
588 3FB3  039B R 3FC3 R    1      DW      _CODE,_LINK                    ;
589 3FB7  07 45 58 45 43 55 54  1      DB      7,'EXECUTE'                      ;
590 039B          1      ORG      _CODE                      ;
591 039B  A1              DB  0A1H                      ;POP BC
592 039C  21              DB  21H                      ;JMP BC
593
594              ; next      ( -- )
595              ;          Run time code for the single index loop.
596              ;          : next ( -- ) \ hilevel model
597              ;          r> r> dup if 1 - >r @ >r exit then drop cell+ >r ;
598
599              $CODE  COMPO+4,'next',DONXT
600 039D          1      DONXT:                      ;
601 3FA9          1      ORG      _NAME                      ;
602 3FA9  039D R 3FB7 R    1      DW      _CODE,_LINK                    ;
603 3FAD  44 6E 65 78 74  1      DB      COMPO+4,'next'                      ;
604 039D          1      ORG      _CODE                      ;
605 039D  6A 00          DB  6AH,0                      ;B<00
606 039F  6B 01          DB  6BH,1                      ;C<01
607 03A1  48 83          DB  48H,83H                     ;EA<(HL)
608 03A3  74 B5          DB  74H,0B5H                    ;EA<EA-BC Skip if no borrow
609 03A5  C9              DB  0C9H                      ;JMP NEXT1
610 03A6  48 93          DB  48H,93H                     ;(HL)<EA
611 03A8  48 82          DB  48H,82H                     ;EA<(DE)
612 03AA  B6              DB  0B6H                      ;DE<EA
613              $NEXT
614 03AB  48 84          1      DB  48H,84H                      ;
615 03AD  48 28          1      DB  48H,28H                     ;

```

```

616 03AF 22 22          NEXT1:      DB 22H,22H          ;DE<DE+2
617 03B1 32 32          DB 32H,32H          ;HL<HL+2
618                    $NEXT
619 03B3 48 84          1          DB 48H,84H          ;
620 03B5 48 28          1          DB 48H,28H          ;
621
622                    ; ?branch   ( f -- )
623                    ;          Branch if flag is zero.
624
625                    $CODE   COMPO+7,'?branch',QBRAN
626 03B7          1          QBRAN:      ;
627 3F9D          1          ORG      _NAME          ;
628 3F9D 03B7 R 3FAD R  1          DW      _CODE,_LINK          ;
629 3FA1 47 3F 62 72 61 6E 63 1          DB      COMPO+7,'?branch'          ;
630 03B7          1          ORG      _CODE          ;
631 03B7 6A FF          DB 6AH,0FFH          ;B<FF
632 03B9 6B FF          DB 6BH,0FFH          ;C<FF
633 03BB A4          DB 0A4H          ;POP EA
634 03BC 74 CD          DB 74H,0CDH          ;EA AND BC Skip if not zero
635 03BE C6          DB 0C6H          ;JMP BRAN1
636 03BF 22 22          DB 22H,22H          ;DE<DE+2
637                    $NEXT
638 03C1 48 84          1          DB 48H,84H          ;
639 03C3 48 28          1          DB 48H,28H          ;
640 03C5 48 82          BRAN1:      DB 48H,82H          ;EA<(DE)
641 03C7 B6          DB 0B6H          ;DE<EA
642                    $NEXT
643 03C8 48 84          1          DB 48H,84H          ;
644 03CA 48 28          1          DB 48H,28H          ;
645
646                    ; branch   ( -- )
647                    ;          Branch to an inline address.
648
649                    $CODE   COMPO+6,'branch',BRAN
650 03CC          1          BRAN:      ;
651 3F91          1          ORG      _NAME          ;
652 3F91 03CC R 3FA1 R  1          DW      _CODE,_LINK          ;
653 3F95 46 62 72 61 6E 63 68 1          DB      COMPO+6,'branch'          ;
654 03CC          1          ORG      _CODE          ;
655 03CC 48 82          DB 48H,82H          ;EA<(DE)
656 03CE B6          DB 0B6H          ;DE<EA
657                    $NEXT
658 03CF 48 84          1          DB 48H,84H          ;
659 03D1 48 28          1          DB 48H,28H          ;
660
661                    ; !       ( w a -- )
662                    ;          Pop the data stack to memory.
663
664                    $CODE   1,'!',STORE
665 03D3          1          STORE:     ;
666 3F8B          1          ORG      _NAME          ;
667 3F8B 03D3 R 3F95 R  1          DW      _CODE,_LINK          ;
668 3F8F 01 21          1          DB      1,'!'          ;
669 03D3          1          ORG      _CODE          ;
670 03D3 A1          DB 0A1H          ;POP BC, address
671 03D4 A4          DB 0A4H          ;POP EA, data

```

```

672 03D5 09          DB 09H          ;A<EAL
673 03D6 39          DB 39H          ; (BC)<A
674 03D7 12          DB 12H          ;BC<BC+1
675 03D8 08          DB 08H          ;A<EAH
676 03D9 39          DB 39H          ; (BC)<A
677                $NEXT
678 03DA 48 84        1          DB 48H,84H      ;
679 03DC 48 28        1          DB 48H,28H      ;
680
681                ; @      ( a -- w )
682                ;      Push data at memory location to the data stack.
683
684                $CODE 1, '@', AT
685 03DE                1          AT:                ;
686 3F85                1          ORG      _NAME          ;
687 3F85 03DE R 3F8F R 1          DW      _CODE, _LINK      ;
688 3F89 01 40         1          DB      1, '@'          ;
689 03DE                1          ORG      _CODE          ;
690 03DE A1            DB 0A1H          ;POP BC
691 03DF 29            DB 29H          ;A<(BC)
692 03E0 19            DB 19H          ;EAL<A
693 03E1 12            DB 12H          ;BC<BC+1
694 03E2 29            DB 29H          ;A<(BC)
695 03E3 18            DB 18H          ;EAH<A
696 03E4 B4            DB 0B4H          ;PUSH EA
697                $NEXT
698 03E5 48 84        1          DB 48H,84H      ;
699 03E7 48 28        1          DB 48H,28H      ;
700
701                ; C!      ( c b -- )
702                ;      Pop the data stack to byte memory.
703
704                $CODE 2, 'C!', CSTOR
705 03E9                1          CSTOR:            ;
706 3F7D                1          ORG      _NAME          ;
707 3F7D 03E9 R 3F89 R 1          DW      _CODE, _LINK      ;
708 3F81 02 43 21     1          DB      2, 'C!'          ;
709 03E9                1          ORG      _CODE          ;
710 03E9 A1            DB 0A1H          ;POP BC address
711 03EA A4            DB 0A4H          ;POP AE data
712 03EB 09            DB 09H          ;A<EAL
713 03EC 39            DB 39H          ; (BC)<A
714                $NEXT
715 03ED 48 84        1          DB 48H,84H      ;
716 03EF 48 28        1          DB 48H,28H      ;
717
718                ; C@      ( b -- c )
719                ;      Push byte memory location to the data stack.
720
721                $CODE 2, 'C@', CAT
722 03F1                1          CAT:                ;
723 3F75                1          ORG      _NAME          ;
724 3F75 03F1 R 3F81 R 1          DW      _CODE, _LINK      ;
725 3F79 02 43 40     1          DB      2, 'C@'          ;
726 03F1                1          ORG      _CODE          ;
727 03F1 A1            DB 0A1H          ;POP BC

```

```

728 03F2 29                                DB 29H                                ;A<(BC)
729 03F3 6A 00                            DB 6AH,0                              ;B<00
730 03F5 1B                                DB 1BH                                ;C<A
731 03F6 B1                                DB 0B1H                                ;PUSH BC
732                                         $NEXT
733 03F7 48 84                            1 DB 48H,84H                            ;
734 03F9 48 28                            1 DB 48H,28H                            ;
735
736                                         ; RP@      ( -- a )
737                                         ;          Push the current RP to the data stack.
738
739                                         $CODE    3,'RP@',RPAT
740 03FB                                1 RPAT:                                ;
741 3F6D                                1 ORG  _NAME                            ;
742 3F6D 03FB R 3F79 R                    1 DW  _CODE,_LINK                        ;
743 3F71 03 52 50 40                      1 DB  3,'RP@'                            ;
744 03FB                                1 ORG  _CODE                            ;
745 03FB B3                                DB 0B3H                                ;PUSH HL
746                                         $NEXT
747 03FC 48 84                            1 DB 48H,84H                            ;
748 03FE 48 28                            1 DB 48H,28H                            ;
749
750                                         ; RP!      ( a -- )
751                                         ;          Set the return stack pointer.
752
753                                         $CODE    COMPO+3,'RP!',RPSTO
754 0400                                1 RPSTO:                               ;
755 3F65                                1 ORG  _NAME                            ;
756 3F65 0400 R 3F71 R                    1 DW  _CODE,_LINK                        ;
757 3F69 43 52 50 21                      1 DB  COMPO+3,'RP!'                      ;
758 0400                                1 ORG  _CODE                            ;
759 0400 A3                                DB 0A3H                                ;POP HL
760                                         $NEXT
761 0401 48 84                            1 DB 48H,84H                            ;
762 0403 48 28                            1 DB 48H,28H                            ;
763
764                                         ; R>      ( -- w )
765                                         ;          Pop the return stack to the data stack.
766
767                                         $CODE    2,'R>',RFROM
768 0405                                1 RFROM:                               ;
769 3F5D                                1 ORG  _NAME                            ;
770 3F5D 0405 R 3F69 R                    1 DW  _CODE,_LINK                        ;
771 3F61 02 52 3E                          1 DB  2,'R>'                            ;
772 0405                                1 ORG  _CODE                            ;
773 0405 48 85                            DB 48H,85H                            ;EA<(HL)++
774 0407 B4                                DB 0B4H                                ;PUSH EA
775                                         $NEXT
776 0408 48 84                            1 DB 48H,84H                            ;
777 040A 48 28                            1 DB 48H,28H                            ;
778
779                                         ; R@      ( -- w )
780                                         ;          Copy top of return stack to the data stack.
781
782                                         $CODE    2,'R@',RAT
783 040C                                1 RAT:                                ;

```



```

784 3F55          1      ORG      _NAME          ;
785 3F55 040C R 3F61 R 1      DW      _CODE,_LINK      ;
786 3F59 02 52 40 1      DB      2,'R@'          ;
787 040C          1      ORG      _CODE          ;
788 040C 48 83          DB 48H,83H          ;EA<(HL)
789 040E B4          DB 0B4H          ;PUSH EA
790              $NEXT
791 040F 48 84          1      DB 48H,84H          ;
792 0411 48 28          1      DB 48H,28H         ;
793
794              ; >R      ( w -- )
795              ;          Push the data stack to the return stack.
796
797              $CODE  COMPO+2,'>R',TOR
798 0413          1      TOR:          ;
799 3F4D          1      ORG      _NAME          ;
800 3F4D 0413 R 3F59 R 1      DW      _CODE,_LINK      ;
801 3F51 42 3E 52          1      DB      COMPO+2,'>R'      ;
802 0413          1      ORG      _CODE          ;
803 0413 33 33          DB 33H,33H          ;HL<HL-2
804 0415 A4          DB 0A4H          ;POP EA
805 0416 48 93          DB 48H,93H         ;(HL)<EA
806              $NEXT
807 0418 48 84          1      DB 48H,84H          ;
808 041A 48 28          1      DB 48H,28H         ;
809
810              ; SP@      ( -- a )
811              ;          Push the current data stack pointer.
812
813              $CODE  3,'SP@',SPAT
814 041C          1      SPAT:          ;
815 3F45          1      ORG      _NAME          ;
816 3F45 041C R 3F51 R 1      DW      _CODE,_LINK      ;
817 3F49 03 53 50 40 1      DB      3,'SP@'          ;
818 041C          1      ORG      _CODE          ;
819 041C 70 0E FE FF          DB 70H,0EH,0FEH,0FFH      ;(FFFE)<SP
820 0420 70 1F FE FF          DB 70H,1FH,0FEH,0FFH      ;BC<(FFFE)
821 0424 B1          DB 0B1H          ;PUSH BC
822              $NEXT
823 0425 48 84          1      DB 48H,84H          ;
824 0427 48 28          1      DB 48H,28H         ;
825
826              ; SP!      ( a -- )
827              ;          Set the data stack pointer.
828
829              $CODE  3,'SP!',SPSTO
830 0429          1      SPSTO:          ;
831 3F3D          1      ORG      _NAME          ;
832 3F3D 0429 R 3F49 R 1      DW      _CODE,_LINK      ;
833 3F41 03 53 50 21 1      DB      3,'SP!'          ;
834 0429          1      ORG      _CODE          ;
835 0429 A1          DB 0A1H          ;POP BC
836 042A 70 1E FE FF          DB 70H,1EH,0FEH,0FFH      ;(FFFE)<BC
837 042E 70 0F FE FF          DB 70H,0FH,0FEH,0FFH      ;PC<(FFFE)
838              $NEXT
839 0432 48 84          1      DB 48H,84H          ;

```

```

840 0434 48 28          1          DB 48H,28H          ;
841
842          ; DROP      ( w -- )
843          ;          Discard top stack item.
844
845          $CODE      4,'DROP',DROP
846 0436          1          DROP:          ;
847 3F33          1          ORG      _NAME          ;
848 3F33 0436 R 3F41 R  1          DW      _CODE,_LINK          ;
849 3F37 04 44 52 4F 50 1          DB      4,'DROP'          ;
850 0436          1          ORG      _CODE          ;
851 0436 A4          DB 0A4H          ;POP EA
852          $NEXT
853 0437 48 84          1          DB 48H,84H          ;
854 0439 48 28          1          DB 48H,28H          ;
855
856          ; DUP      ( w -- w w )
857          ;          Duplicate the top stack item.
858
859          $CODE      3,'DUP',DUPP
860 043B          1          DUPP:          ;
861 3F2B          1          ORG      _NAME          ;
862 3F2B 043B R 3F37 R  1          DW      _CODE,_LINK          ;
863 3F2F 03 44 55 50  1          DB      3,'DUP'          ;
864 043B          1          ORG      _CODE          ;
865 043B A4          DB 0A4H          ;POP EA
866 043C B4          DB 0B4H          ;PUSH EA
867 043D B4          DB 0B4H          ;PUSH EA
868          $NEXT
869 043E 48 84          1          DB 48H,84H          ;
870 0440 48 28          1          DB 48H,28H          ;
871
872          ; SWAP     ( w1 w2 -- w2 w1 )
873          ;          Exchange top two stack items.
874
875          $CODE      4,'SWAP',SWAP
876 0442          1          SWAP:          ;
877 3F21          1          ORG      _NAME          ;
878 3F21 0442 R 3F2F R  1          DW      _CODE,_LINK          ;
879 3F25 04 53 57 41 50 1          DB      4,'SWAP'          ;
880 0442          1          ORG      _CODE          ;
881 0442 A4          DB 0A4H          ;POP EA
882 0443 A1          DB 0A1H          ;POP BC
883 0444 B4          DB 0B4H          ;PUSH EA
884 0445 B1          DB 0B1H          ;PUSH BC
885          $NEXT
886 0446 48 84          1          DB 48H,84H          ;
887 0448 48 28          1          DB 48H,28H          ;
888
889          ; OVER     ( w1 w2 -- w1 w2 w1 )
890          ;          Copy second stack item to top.
891
892          $CODE      4,'OVER',OVER
893 044A          1          OVER:          ;
894 3F17          1          ORG      _NAME          ;
895 3F17 044A R 3F25 R  1          DW      _CODE,_LINK          ;

```

```

896 3F1B 04 4F 56 45 52      1      DB      4,'OVER'      ;
897 044A                    1      ORG     _CODE      ;
898 044A A4                  DB 0A4H      ;POP AE
899 044B A1                  DB 0A1H      ;POP BC
900 044C B1                  DB 0B1H      ;PUSH BC
901 044D B4                  DB 0B4H      ;PUSH AE
902 044E B1                  DB 0B1H      ;PUSH BC
903                          $NEXT
904 044F 48 84              1      DB 48H,84H      ;
905 0451 48 28              1      DB 48H,28H      ;
906
907                          ; 0<      ( n -- t )
908                          ;      Return true if n is negative.
909
910                          $CODE  2,'0<','ZLESS
911 0453                    1      ZLESS:      ;
912 3F0F                    1      ORG     _NAME      ;
913 3F0F 0453 R 3F1B R      1      DW     _CODE,_LINK ;
914 3F13 02 30 3C          1      DB     2,'0<'      ;
915 0453                    1      ORG     _CODE      ;
916 0453 A1                  DB 0A1H      ;POP BC
917 0454 69 FF              DB 69H,0FFH ;A<FF
918 0456 48 06              DB 48H,06H   ;B Shift Left, Skip if carry
919 0458 69 00              DB 69H,0     ;A<00
920 045A 1A                  DB 1AH       ;B<A
921 045B 1B                  DB 1BH       ;C<A
922 045C B1                  DB 0B1H      ;PUSH BC
923                          $NEXT
924 045D 48 84              1      DB 48H,84H      ;
925 045F 48 28              1      DB 48H,28H      ;
926
927                          ; AND      ( w w -- w )
928                          ;      Bitwise AND.
929
930                          $CODE  3,'AND','ANDD
931 0461                    1      ANDD:      ;
932 3F07                    1      ORG     _NAME      ;
933 3F07 0461 R 3F13 R      1      DW     _CODE,_LINK ;
934 3F0B 03 41 4E 44        1      DB     3,'AND'      ;
935 0461                    1      ORG     _CODE      ;
936 0461 A1                  DB 0A1H      ;POP BC
937 0462 A4                  DB 0A4H      ;POP AE
938 0463 74 8D              DB 74H,8DH   ;EA<EA AND BC
939 0465 B4                  DB 0B4H      ;PUSH EA
940                          $NEXT
941 0466 48 84              1      DB 48H,84H      ;
942 0468 48 28              1      DB 48H,28H      ;
943
944                          ; OR      ( w w -- w )
945                          ;      Bitwise inclusive OR.
946
947                          $CODE  2,'OR','ORR
948 046A                    1      ORR:      ;
949 3EFF                    1      ORG     _NAME      ;
950 3EFF 046A R 3F0B R      1      DW     _CODE,_LINK ;
951 3F03 02 4F 52          1      DB     2,'OR'      ;

```

```

952 046A          1      ORG      _CODE                      ;
953 046A A1          DB 0A1H                          ;POP BC
954 046B A4          DB 0A4H                          ;POP EA
955 046C 74 9D       DB 74H,9DH                          ;EA<EA OR BC
956 046E B4          DB 0B4H                          ;PUSH EA
957              $NEXT
958 046F 48 84       1      DB 48H,84H                          ;
959 0471 48 28       1      DB 48H,28H                          ;
960
961              ; XOR      ( w w -- w )
962              ;          Bitwise exclusive OR.
963
964              $CODE 3,'XOR',XORR
965 0473          1      XORR:                               ;
966 3EF7          1      ORG      _NAME                      ;
967 3EF7 0473 R 3F03 R 1      DW      _CODE,_LINK                ;
968 3EFB 03 58 4F 52 1      DB      3,'XOR'                    ;
969 0473          1      ORG      _CODE                      ;
970 0473 A1          DB 0A1H                          ;POP BC
971 0474 A4          DB 0A4H                          ;POP EA
972 0475 74 95       DB 74H,95H                          ;EA<EA EX-OR BC
973 0477 B4          DB 0B4H                          ;PUSH EA
974              $NEXT
975 0478 48 84       1      DB 48H,84H                          ;
976 047A 48 28       1      DB 48H,28H                          ;
977
978              ; UM+      ( w w -- w cy )
979              ;          Add two numbers, return the sum and carry flag.
980
981              $CODE 3,'UM+',UPLUS
982 047C          1      UPLUS:                               ;
983 3EEF          1      ORG      _NAME                      ;
984 3EEF 047C R 3EFB R 1      DW      _CODE,_LINK                ;
985 3EF3 03 55 4D 2B 1      DB      3,'UM+'                    ;
986 047C          1      ORG      _CODE                      ;
987 047C A1          DB 0A1H                          ;POP BC
988 047D A4          DB 0A4H                          ;POP EA
989 047E 69 00       DB 69H,0                          ;A<00
990 0480 74 A5       DB 74H,0A5H                          ;EA<EA+BC Skip if no carry
991 0482 41          DB 41H                          ;A<A+1
992 0483 1B          DB 1BH                          ;C<A
993 0484 6A 00       DB 6AH,0                          ;B<00
994 0486 B4          DB 0B4H                          ;PUSH EA
995 0487 B1          DB 0B1H                          ;PUSH BC
996              $NEXT
997 0488 48 84       1      DB 48H,84H                          ;
998 048A 48 28       1      DB 48H,28H                          ;
999
1000             ;; System and user variables
1001
1002             ; doVAR      ( -- a )
1003             ;          Run time routine for VARIABLE and CREATE.
1004
1005             $COLON COMPO+5,'doVAR',DOVAR
1006 048C          2      DOVAR:                               ;
1007 3EE5          2      ORG      _NAME                      ;

```

```

1008 3EE5 048C R 3EF3 R      2      DW      _CODE,_LINK      ;
1009 3EE9 45 64 6F 56 41 52  2      DB      COMPO+5,'doVAR'      ;
1010 048C                    2      ORG      _CODE      ;
1011 048C 80                  1      DB      80H      ;
1012 048D 0405 R 0394 R      DW      RFROM,EXIT
1013
1014 ; UP      ( -- a )
1015 ;      Pointer to the user area.
1016
1017 $COLON 2,'UP',UP
1018 0491      2      UP:      ;
1019 3EDD      2      ORG      _NAME      ;
1020 3EDD 0491 R 3EE9 R      2      DW      _CODE,_LINK      ;
1021 3EE1 02 55 50          2      DB      2,'UP'      ;
1022 0491      2      ORG      _CODE      ;
1023 0491 80              1      DB      80H      ;
1024 0492 048C R          DW      DOVAR
1025 0494 FF00          DW      UPP
1026
1027 ; doUSER      ( -- a )
1028 ;      Run time routine for user variables.
1029
1030 $COLON COMPO+6,'doUSER',DOUSE
1031 0496      2      DOUSE:      ;
1032 3ED1      2      ORG      _NAME      ;
1033 3ED1 0496 R 3EE1 R      2      DW      _CODE,_LINK      ;
1034 3ED5 46 64 6F 55 53 45 52  2      DB      COMPO+6,'doUSER'      ;
1035 0496      2      ORG      _CODE      ;
1036 0496 80              1      DB      80H      ;
1037 0497 0405 R 03DE R 0491 R  DW      RFROM,AT,UP,AT,PLUS,EXIT
1038 03DE R 0565 R 0394 R
1039
1040 ; SP0      ( -- a )
1041 ;      Pointer to bottom of the data stack.
1042
1043 $USER 3,'SP0',SZERO
1044 04A3      2      SZERO:      ;
1045 3EC9      2      ORG      _NAME      ;
1046 3EC9 04A3 R 3ED5 R      2      DW      _CODE,_LINK      ;
1047 3ECD 03 53 50 30        2      DB      3,'SP0'      ;
1048 04A3      2      ORG      _CODE      ;
1049 04A3 80              1      DB      80H      ;
1050 04A4 0496 R 0008        1      DW      DOUSE,_USER      ;
1051
1052 ; RP0      ( -- a )
1053 ;      Pointer to bottom of the return stack.
1054
1055 $USER 3,'RP0',RZERO
1056 04A8      2      RZERO:      ;
1057 3EC1      2      ORG      _NAME      ;
1058 3EC1 04A8 R 3ECD R      2      DW      _CODE,_LINK      ;
1059 3EC5 03 52 50 30        2      DB      3,'RP0'      ;
1060 04A8      2      ORG      _CODE      ;
1061 04A8 80              1      DB      80H      ;
1062 04A9 0496 R 000A        1      DW      DOUSE,_USER      ;
1063

```

```

1064          ;   '?KEY      ( -- a )
1065          ;           Execution vector of ?KEY.
1066
1067          $USER      5,"?KEY",TQKEY
1068 04AD          2      TQKEY:          ;
1069 3EB7          2      ORG      _NAME          ;
1070 3EB7 04AD R 3EC5 R          2      DW      _CODE,_LINK          ;
1071 3EBB 05 27 3F 4B 45 59          2      DB      5,"?KEY"          ;
1072 04AD          2      ORG      _CODE          ;
1073 04AD 80          1      DB 80H          ;
1074 04AE 0496 R 000C          1      DW      DOUSE,_USER          ;
1075
1076          ;   'EMIT      ( -- a )
1077          ;           Execution vector of EMIT.
1078
1079          $USER      5,"EMIT",Temit
1080 04B2          2      Temit:          ;
1081 3EAD          2      ORG      _NAME          ;
1082 3EAD 04B2 R 3EBB R          2      DW      _CODE,_LINK          ;
1083 3EB1 05 27 45 4D 49 54          2      DB      5,"EMIT"          ;
1084 04B2          2      ORG      _CODE          ;
1085 04B2 80          1      DB 80H          ;
1086 04B3 0496 R 000E          1      DW      DOUSE,_USER          ;
1087
1088          ;   'EXPECT    ( -- a )
1089          ;           Execution vector of EXPECT.
1090
1091          $USER      7,"EXPECT",TEXPE
1092 04B7          2      TEXPE:          ;
1093 3EA1          2      ORG      _NAME          ;
1094 3EA1 04B7 R 3EB1 R          2      DW      _CODE,_LINK          ;
1095 3EA5 07 27 45 58 50 45 43          2      DB      7,"EXPECT"          ;
1096 04B7          2      ORG      _CODE          ;
1097 04B7 80          1      DB 80H          ;
1098 04B8 0496 R 0010          1      DW      DOUSE,_USER          ;
1099
1100          ;   'TAP      ( -- a )
1101          ;           Execution vector of TAP.
1102
1103          $USER      4,"TAP",TTAP
1104 04BC          2      TTAP:          ;
1105 3E97          2      ORG      _NAME          ;
1106 3E97 04BC R 3EA5 R          2      DW      _CODE,_LINK          ;
1107 3E9B 04 27 54 41 50          2      DB      4,"TAP"          ;
1108 04BC          2      ORG      _CODE          ;
1109 04BC 80          1      DB 80H          ;
1110 04BD 0496 R 0012          1      DW      DOUSE,_USER          ;
1111
1112          ;   'ECHO      ( -- a )
1113          ;           Execution vector of ECHO.
1114
1115          $USER      5,"ECHO",TECHO
1116 04C1          2      TECHO:          ;
1117 3E8D          2      ORG      _NAME          ;
1118 3E8D 04C1 R 3E9B R          2      DW      _CODE,_LINK          ;
1119 3E91 05 27 45 43 48 4F          2      DB      5,"ECHO"          ;

```

```

1120 04C1          2      ORG      _CODE          ;
1121 04C1 80      1      DB 80H          ;
1122 04C2 0496 R 0014 1      DW      DOUSE,_USER      ;
1123
1124          ;      'PROMPT      ( -- a )
1125          ;      Execution vector of PROMPT.
1126
1127          $USER      7, 'PROMPT', TPROM
1128 04C6          2      TPROM:          ;
1129 3E81          2      ORG      _NAME          ;
1130 3E81 04C6 R 3E91 R 2      DW      _CODE,_LINK      ;
1131 3E85 07 27 50 52 4F 4D 50 2      DB      7, 'PROMPT'      ;
1132 04C6          2      ORG      _CODE          ;
1133 04C6 80      1      DB 80H          ;
1134 04C7 0496 R 0016 1      DW      DOUSE,_USER      ;
1135
1136          ;      BASE      ( -- a )
1137          ;      Storage of the radix base for numeric I/O.
1138
1139          $USER      4, 'BASE', BASE
1140 04CB          2      BASE:          ;
1141 3E77          2      ORG      _NAME          ;
1142 3E77 04CB R 3E85 R 2      DW      _CODE,_LINK      ;
1143 3E7B 04 42 41 53 45 2      DB      4, 'BASE'      ;
1144 04CB          2      ORG      _CODE          ;
1145 04CB 80      1      DB 80H          ;
1146 04CC 0496 R 0018 1      DW      DOUSE,_USER      ;
1147
1148          ;      tmp      ( -- a )
1149          ;      A temporary storage location used in parse and find.
1150
1151          $USER      COMPO+3, 'tmp', TEMP
1152 04D0          2      TEMP:          ;
1153 3E6F          2      ORG      _NAME          ;
1154 3E6F 04D0 R 3E7B R 2      DW      _CODE,_LINK      ;
1155 3E73 43 74 6D 70 2      DB      COMPO+3, 'tmp'      ;
1156 04D0          2      ORG      _CODE          ;
1157 04D0 80      1      DB 80H          ;
1158 04D1 0496 R 001A 1      DW      DOUSE,_USER      ;
1159
1160          ;      SPAN      ( -- a )
1161          ;      Hold character count received by EXPECT.
1162
1163          $USER      4, 'SPAN', SPAN
1164 04D5          2      SPAN:          ;
1165 3E65          2      ORG      _NAME          ;
1166 3E65 04D5 R 3E73 R 2      DW      _CODE,_LINK      ;
1167 3E69 04 53 50 41 4E 2      DB      4, 'SPAN'      ;
1168 04D5          2      ORG      _CODE          ;
1169 04D5 80      1      DB 80H          ;
1170 04D6 0496 R 001C 1      DW      DOUSE,_USER      ;
1171
1172          ;      >IN      ( -- a )
1173          ;      Hold the character pointer while parsing input stream.
1174
1175          $USER      3, '>IN', INN

```

```

1176 04DA          2      INN:                ;
1177 3E5D          2      ORG      _NAME                ;
1178 3E5D 04DA R 3E69 R  2      DW      _CODE,_LINK            ;
1179 3E61 03 3E 49 4E    2      DB      3,'>IN'                ;
1180 04DA          2      ORG      _CODE                ;
1181 04DA 80           1      DB      80H                    ;
1182 04DB 0496 R 001E    1      DW      DOUSE,_USER            ;
1183
1184                ;      #TIB      ( -- a )
1185                ;      Hold the current count and address of the terminal input buffer.
1186
1187                $USER      4,'#TIB',NTIB
1188 04DF          2      NTIB:                ;
1189 3E53          2      ORG      _NAME                ;
1190 3E53 04DF R 3E61 R  2      DW      _CODE,_LINK            ;
1191 3E57 04 23 54 49 42  2      DB      4,'#TIB'                ;
1192 04DF          2      ORG      _CODE                ;
1193 04DF 80           1      DB      80H                    ;
1194 04E0 0496 R 0020    1      DW      DOUSE,_USER            ;
1195 = 0024                _USER = _USER+CELLL
1196
1197                ;      CSP      ( -- a )
1198                ;      Hold the stack pointer for error checking.
1199
1200                $USER      3,'CSP',CSP
1201 04E4          2      CSP:                ;
1202 3E4B          2      ORG      _NAME                ;
1203 3E4B 04E4 R 3E57 R  2      DW      _CODE,_LINK            ;
1204 3E4F 03 43 53 50    2      DB      3,'CSP'                ;
1205 04E4          2      ORG      _CODE                ;
1206 04E4 80           1      DB      80H                    ;
1207 04E5 0496 R 0024    1      DW      DOUSE,_USER            ;
1208
1209                ;      'EVAL      ( -- a )
1210                ;      Execution vector of EVAL.
1211
1212                $USER      5,"'EVAL",TEVAL
1213 04E9          2      TEVAL:                ;
1214 3E41          2      ORG      _NAME                ;
1215 3E41 04E9 R 3E4F R  2      DW      _CODE,_LINK            ;
1216 3E45 05 27 45 56 41 4C  2      DB      5,"'EVAL'"                ;
1217 04E9          2      ORG      _CODE                ;
1218 04E9 80           1      DB      80H                    ;
1219 04EA 0496 R 0026    1      DW      DOUSE,_USER            ;
1220
1221                ;      'NUMBER      ( -- a )
1222                ;      Execution vector of NUMBER?.
1223
1224                $USER      7,"'NUMBER",TNUMB
1225 04EE          2      TNUMB:                ;
1226 3E35          2      ORG      _NAME                ;
1227 3E35 04EE R 3E45 R  2      DW      _CODE,_LINK            ;
1228 3E39 07 27 4E 55 4D 42 45  2      DB      7,"'NUMBER'"                ;
1229 04EE          2      ORG      _CODE                ;
1230 04EE 80           1      DB      80H                    ;
1231 04EF 0496 R 0028    1      DW      DOUSE,_USER            ;

```



```

1232
1233 ; HLD ( -- a )
1234 ; Hold a pointer in building a numeric output string.
1235
1236 $USER 3, 'HLD', HLD
1237 04F3 2 HLD: ;
1238 3E2D 2 ORG _NAME ;
1239 3E2D 04F3 R 3E39 R 2 DW _CODE, _LINK ;
1240 3E31 03 48 4C 44 2 DB 3, 'HLD' ;
1241 04F3 2 ORG _CODE ;
1242 04F3 80 1 DB 80H ;
1243 04F4 0496 R 002A 1 DW DOUSE, _USER ;
1244
1245 ; HANDLER ( -- a )
1246 ; Hold the return stack pointer for error handling.
1247
1248 $USER 7, 'HANDLER', HANDL
1249 04F8 2 HANDL: ;
1250 3E21 2 ORG _NAME ;
1251 3E21 04F8 R 3E31 R 2 DW _CODE, _LINK ;
1252 3E25 07 48 41 4E 44 4C 45 2 DB 7, 'HANDLER' ;
1253 04F8 2 ORG _CODE ;
1254 04F8 80 1 DB 80H ;
1255 04F9 0496 R 002C 1 DW DOUSE, _USER ;
1256
1257 ; CONTEXT ( -- a )
1258 ; A area to specify vocabulary search order.
1259
1260 $USER 7, 'CONTEXT', CNTXT
1261 04FD 2 CNTXT: ;
1262 3E15 2 ORG _NAME ;
1263 3E15 04FD R 3E25 R 2 DW _CODE, _LINK ;
1264 3E19 07 43 4F 4E 54 45 58 2 DB 7, 'CONTEXT' ;
1265 04FD 2 ORG _CODE ;
1266 04FD 80 1 DB 80H ;
1267 04FE 0496 R 002E 1 DW DOUSE, _USER ;
1268 = 003C _USER = _USER+VOCSS*CELLL ;vocabulary stack
1269
1270 ; CURRENT ( -- a )
1271 ; Point to the vocabulary to be extended.
1272
1273 $USER 7, 'CURRENT', CRRT
1274 0502 2 CRRT: ;
1275 3E09 2 ORG _NAME ;
1276 3E09 0502 R 3E19 R 2 DW _CODE, _LINK ;
1277 3E0D 07 43 55 52 52 45 4E 2 DB 7, 'CURRENT' ;
1278 0502 2 ORG _CODE ;
1279 0502 80 1 DB 80H ;
1280 0503 0496 R 003C 1 DW DOUSE, _USER ;
1281 = 0040 _USER = _USER+CELLL ;vocabulary link pointer
1282
1283 ; FHEAD ( -- a )
1284 ; Point to the FORTH vocab head pointer.
1285 $USER 5, 'FHEAD', FHEAD
1286 0507 2 FHEAD: ;
1287 3DFF 2 ORG _NAME ;

```

```

1288 3DFF 0507 R 3E0D R      2      DW      _CODE,_LINK      ;
1289 3E03 05 46 48 45 41 44  2      DB      5,'FHEAD'      ;
1290 0507                    2      ORG      _CODE      ;
1291 0507 80                  1      DB      80H      ;
1292 0508 0496 R 0040        1      DW      DOUSE,_USER      ;
1293
1294 ; FLINK      ( -- a )
1295 ;           Point to the FORTH vocab link pointer.
1296 $USER      5,'FLINK',FLINK
1297 050C                    2      FLINK:      ;
1298 3DF5                    2      ORG      _NAME      ;
1299 3DF5 050C R 3E03 R      2      DW      _CODE,_LINK      ;
1300 3DF9 05 46 4C 49 4E 4B  2      DB      5,'FLINK'      ;
1301 050C                    2      ORG      _CODE      ;
1302 050C 80                  1      DB      80H      ;
1303 050D 0496 R 0042        1      DW      DOUSE,_USER      ;
1304
1305 ; CP      ( -- a )
1306 ;           Point to the top of the code dictionary.
1307 $USER      2,'CP',CP
1308
1309 0511                    2      CP:      ;
1310 3DED                    2      ORG      _NAME      ;
1311 3DED 0511 R 3DF9 R      2      DW      _CODE,_LINK      ;
1312 3DF1 02 43 50          2      DB      2,'CP'      ;
1313 0511                    2      ORG      _CODE      ;
1314 0511 80                  1      DB      80H      ;
1315 0512 0496 R 0044        1      DW      DOUSE,_USER      ;
1316
1317 ; NP      ( -- a )
1318 ;           Point to the bottom of the name dictionary.
1319 $USER      2,'NP',NP
1320
1321 0516                    2      NP:      ;
1322 3DE5                    2      ORG      _NAME      ;
1323 3DE5 0516 R 3DF1 R      2      DW      _CODE,_LINK      ;
1324 3DE9 02 4E 50          2      DB      2,'NP'      ;
1325 0516                    2      ORG      _CODE      ;
1326 0516 80                  1      DB      80H      ;
1327 0517 0496 R 0046        1      DW      DOUSE,_USER      ;
1328
1329 ; LAST      ( -- a )
1330 ;           Point to the last name in the name dictionary.
1331 $USER      4,'LAST',LAST
1332
1333 051B                    2      LAST:      ;
1334 3DDB                    2      ORG      _NAME      ;
1335 3DDB 051B R 3DE9 R      2      DW      _CODE,_LINK      ;
1336 3DDF 04 4C 41 53 54    2      DB      4,'LAST'      ;
1337 051B                    2      ORG      _CODE      ;
1338 051B 80                  1      DB      80H      ;
1339 051C 0496 R 0048        1      DW      DOUSE,_USER      ;
1340
1341 ; SERIN      ( -- a )
1342 ;           Point to host serial input. Flag in high, char in low byte.
1343

```

```

1344                                $USER 5,'SERIN',SERIN
1345 0520                2      SERIN:                ;
1346 3DD1                2      ORG      _NAME          ;
1347 3DD1 0520 R 3DDF R   2      DW      _CODE,_LINK   ;
1348 3DD5 05 53 45 52 49 4E 2      DB      5,'SERIN'     ;
1349 0520                2      ORG      _CODE          ;
1350 0520 80             1      DB      80H            ;
1351 0521 0496 R 004A    1      DW      DOUSE,_USER     ;
1352
1353 ;      HAFBIT      ( -- a )
1354 ;                      Point to half bit time used by serial i/o routines.
1355
1356                                $USER 6,'HAFBIT',HAFBIT
1357 0525                2      HAFBIT:                ;
1358 3DC5                2      ORG      _NAME          ;
1359 3DC5 0525 R 3DD5 R   2      DW      _CODE,_LINK   ;
1360 3DC9 06 48 41 46 42 49 54 2      DB      6,'HAFBIT'     ;
1361 0525                2      ORG      _CODE          ;
1362 0525 80             1      DB      80H            ;
1363 0526 0496 R 004C    1      DW      DOUSE,_USER     ;
1364
1365 ;      BITIME     ( -- a )
1366 ;                      Point to bit time used to set serial i/o baud rate.
1367
1368                                $USER 6,'BITIME',BITIME
1369 052A                2      BITIME:                ;
1370 3DB9                2      ORG      _NAME          ;
1371 3DB9 052A R 3DC9 R   2      DW      _CODE,_LINK   ;
1372 3DBD 06 42 49 54 49 4D 45 2      DB      6,'BITIME'     ;
1373 052A                2      ORG      _CODE          ;
1374 052A 80             1      DB      80H            ;
1375 052B 0496 R 004E    1      DW      DOUSE,_USER     ;
1376
1377 ;; Common functions
1378
1379 ;      doVOC      ( -- )
1380 ;                      Run time action of VOCABULARY's.
1381
1382                                $COLON COMPO+5,'doVOC',DOVOC
1383 052F                2      DOVOC:                ;
1384 3DAF                2      ORG      _NAME          ;
1385 3DAF 052F R 3DBD R   2      DW      _CODE,_LINK   ;
1386 3DB3 45 64 6F 56 4F 43 2      DB      COMPO+5,'doVOC'     ;
1387 052F                2      ORG      _CODE          ;
1388 052F 80             1      DB      80H            ;
1389 0530 0507 R 04FD R 03D3 R 2      DW      FHEAD,CNTXT,STORE,EXIT
1390 0394 R
1391
1392 ;      FORTH     ( -- )
1393 ;                      Make FORTH the context vocabulary.
1394
1395                                $COLON 5,'FORTH',FORTH
1396 0538                2      FORTH:                ;
1397 3DA5                2      ORG      _NAME          ;
1398 3DA5 0538 R 3DB3 R   2      DW      _CODE,_LINK   ;
1399 3DA9 05 46 4F 52 54 48 2      DB      5,'FORTH'     ;

```



```

1456 3D7F          1      ORG      _NAME          ;
1457 3D7F 055B R 3D8D R      1      DW      _CODE,_LINK      ;
1458 3D83 04 32 44 55 50    1      DB      4,'2DUP'        ;
1459 055B          1      ORG      _CODE          ;
1460 055B A4 A1             DB 0A4H,0A1H          ;POP EA, POP BC
1461 055D B1 B4             DB 0B1H,0B4H          ;PUSH BC, PUSH EA
1462 055F B1 B4             DB 0B1H,0B4H          ;PUSH BC, PUSH EA
1463                $NEXT
1464 0561 48 84             1      DB 48H,84H            ;
1465 0563 48 28             1      DB 48H,28H           ;
1466
1467                ; +      ( w w -- sum )
1468                ;      Add top two items.
1469
1470                $CODE 1,'+',PLUS
1471 0565          1      PLUS:          ;
1472 3D79          1      ORG      _NAME          ;
1473 3D79 0565 R 3D83 R      1      DW      _CODE,_LINK      ;
1474 3D7D 01 2B             1      DB      1,'+'          ;
1475 0565          1      ORG      _CODE          ;
1476 0565 A1 A4             DB 0A1H,0A4H          ;POP BC, POP EA
1477 0567 74 A5             DB 74H,0A5H          ;EA<EA+BC, Skip
1478 0569 00                DB 0                  ;NOP
1479 056A B4                DB 0B4H              ;PUSH EA
1480                $NEXT
1481 056B 48 84             1      DB 48H,84H           ;
1482 056D 48 28             1      DB 48H,28H           ;
1483
1484                ; D+      ( d d -- d )
1485                ;      Double addition, as an example using UM+.
1486                ;
1487                ;      $COLON 2,'D+',DPLUS
1488                ;      DW      TOR,SWAP,TOR,UPLUS
1489                ;      DW      RFROM,RFROM,PLUS,PLUS,EXIT
1490
1491                ; NOT      ( w -- w )
1492                ;      One's complement of tos.
1493
1494                $CODE 3,'NOT',INVER
1495 056F          1      INVER:         ;
1496 3D71          1      ORG      _NAME          ;
1497 3D71 056F R 3D7D R      1      DW      _CODE,_LINK      ;
1498 3D75 03 4E 4F 54        1      DB      3,'NOT'        ;
1499 056F          1      ORG      _CODE          ;
1500 056F A1                DB 0A1H              ;POP BC
1501 0570 69 FF             DB 69H,0FFH          ;A<FF
1502 0572 60 12             DB 60H,12H          ;B<B EX-OR A
1503 0574 60 13             DB 60H,13H          ;C<C EX-OR A
1504 0576 B1                DB 0B1H              ;PUSH BC
1505                $NEXT
1506 0577 48 84             1      DB 48H,84H           ;
1507 0579 48 28             1      DB 48H,28H           ;
1508
1509                ; NEGATE      ( n -- -n )
1510                ;      Two's complement of tos.
1511

```

```

1512                                $CODE 6,'NEGATE',NEGAT
1513 057B                          1      NEGAT:                                ;
1514 3D65                          1      ORG      _NAME                                ;
1515 3D65 057B R 3D75 R             1      DW      _CODE,_LINK                            ;
1516 3D69 06 4E 45 47 41 54 45    1      DB      6,'NEGATE'                                ;
1517 057B                          1      ORG      _CODE                                ;
1518 057B A1                       DB 0A1H                                ;POP BC
1519 057C 69 FF                   DB 69H,0FFH                            ;A<FF
1520 057E 60 12                   DB 60H,12H                            ;B<B EX-OR A
1521 0580 60 13                   DB 60H,13H                            ;C<C EX-OR A
1522 0582 12                      DB 12H                                ;BC<BC+1
1523 0583 B1                      DB 0B1H                                ;PUSH BC
1524                                $NEXT
1525 0584 48 84                   1      DB 48H,84H                                ;
1526 0586 48 28                   1      DB 48H,28H                                ;
1527
1528
1529                                ; DNEGATE      ( d -- -d )
1530                                ;              Two's complement of top double.
1531
1532                                $COLON 7,'DNEGATE',DNEGA
1533 0588                          2      DNEGA:                                ;
1534 3D59                          2      ORG      _NAME                                ;
1535 3D59 0588 R 3D69 R             2      DW      _CODE,_LINK                            ;
1536 3D5D 07 44 4E 45 47 41 54    2      DB      7,'DNEGATE'                                ;
1537 0588                          2      ORG      _CODE                                ;
1538 0588 80                      1      DB 80H                                ;
1539 0589 056F R 0413 R 056F R     DW      INVER,TOR,INVER
1540 058F 038D R 0001 047C R       DW      DOLIT,1,UPLUS
1541 0595 0405 R 0565 R 0394 R     DW      RFROM,PLUS,EXIT
1542
1543                                ; -      ( n1 n2 -- n1-n2 )
1544                                ;              Subtraction.
1545
1546                                $CODE 1,'-',SUBB
1547 059B                          1      SUBB:                                ;
1548 3D53                          1      ORG      _NAME                                ;
1549 3D53 059B R 3D5D R             1      DW      _CODE,_LINK                            ;
1550 3D57 01 2D                   1      DB      1,'-'                                ;
1551 059B                          1      ORG      _CODE                                ;
1552 059B A1                       DB 0A1H                                ;POP BC
1553 059C 69 FF                   DB 069H,0FFH                            ;A<FF
1554 059E 60 12                   DB 060H,12H                            ;B<B EX-OR A
1555 05A0 60 13                   DB 060H,13H                            ;C<C EX-OR A
1556 05A2 12                      DB 12H                                ;BC<BC+1
1557 05A3 A4                      DB 0A4H                                ;POP EA
1558 05A4 74 A5                   DB 74H,0A5H                            ;EA<EA+BC Skip
1559 05A6 00                      DB 0                                ;NOP
1560 05A7 B4                      DB 0B4H                                ;PUSH EA
1561                                $NEXT
1562 05A8 48 84                   1      DB 48H,84H                                ;
1563 05AA 48 28                   1      DB 48H,28H                                ;
1564
1565                                ; ABS      ( n -- n )
1566                                ;              Return the absolute value of n.
1567

```

```

1568                                $COLON 3,'ABS',ABSS
1569 05AC                            2      ABSS:                                ;
1570 3D4B                            2      ORG      _NAME                                ;
1571 3D4B 05AC R 3D57 R                2      DW      _CODE,_LINK                            ;
1572 3D4F 03 41 42 53                  2      DB      3,'ABS'                                ;
1573 05AC                            2      ORG      _CODE                                ;
1574 05AC 80                          1      DB      80H                                ;
1575 05AD 043B R 0453 R                DW      DUPP,ZLESS
1576 05B1 03B7 R 05B7 R                DW      QBRAN,ABS1
1577 05B5 057B R                       DW      NEGAT
1578 05B7 0394 R                        ABS1:   DW      EXIT
1579
1580                                ;      =      ( w w -- t )
1581                                ;      Return true if top two are equal.
1582
1583                                $CODE 1,'=',EQUAL
1584 05B9                            1      EQUAL:                                ;
1585 3D45                            1      ORG      _NAME                                ;
1586 3D45 05B9 R 3D4F R                1      DW      _CODE,_LINK                            ;
1587 3D49 01 3D                        1      DB      1,'='                                ;
1588 05B9                            1      ORG      _CODE                                ;
1589 05B9 A4 A1                          DB      0A4H,0A1H                ;POP EA, POP BC
1590 05BB 69 FF                          DB      69H,0FFH                ;A<FF
1591 05BD 74 FD                          DB      74H,0FDH                ;EA-BC, Skip if zero
1592 05BF 69 00                          DB      69H,00H                 ;A<00
1593 05C1 1A 1B                          DB      1AH,1BH                 ;B<A, C<A
1594 05C3 B1                             DB      0B1H                     ;PUSH BC
1595                                $NEXT
1596 05C4 48 84                          1      DB      48H,84H                                ;
1597 05C6 48 28                          1      DB      48H,28H                                ;
1598
1599                                ;      UC      ( u u -- t )
1600                                ;      Unsigned compare of top two items.
1601
1602                                $COLON 2,'U<',ULESS
1603 05C8                            2      ULESS:                                ;
1604 3D3D                            2      ORG      _NAME                                ;
1605 3D3D 05C8 R 3D49 R                2      DW      _CODE,_LINK                            ;
1606 3D41 02 55 3C                      2      DB      2,'U<'                                ;
1607 05C8                            2      ORG      _CODE                                ;
1608 05C8 80                          1      DB      80H                                ;
1609 05C9 055B R 0473 R 0453 R          DW      DDUP,XORR,ZLESS
1610 05CF 03B7 R 05DB R                 DW      QBRAN,ULES1
1611 05D3 0442 R 0436 R 0453 R          DW      SWAP,DROP,ZLESS,EXIT
1612 0394 R
1613 05DB 059B R 0453 R 0394 R          ULES1: DW      SUBB,ZLESS,EXIT
1614
1615                                ;      <      ( n1 n2 -- t )
1616                                ;      Signed compare of top two items.
1617
1618                                $COLON 1,'<',LESS
1619 05E1                            2      LESS:                                ;
1620 3D37                            2      ORG      _NAME                                ;
1621 3D37 05E1 R 3D41 R                2      DW      _CODE,_LINK                            ;
1622 3D3B 01 3C                          2      DB      1,'<'                                ;
1623 05E1                            2      ORG      _CODE                                ;

```

```

1624 05E1 80 1 DB 80H ;
1625 05E2 055B R 0473 R 0453 R DW DDUP,XORR,ZLESS
1626 05E8 03B7 R 05F2 R DW QBRAN,LESS1
1627 05EC 0436 R 0453 R 0394 R DW DROP,ZLESS,EXIT
1628 05F2 059B R 0453 R 0394 R LESS1: DW SUBB,ZLESS,EXIT
1629
1630 ; MAX ( n n -- n )
1631 ; Return the greater of two top stack items.
1632
1633 $CODE 3,'MAX',MAX
1634 05F8 1 MAX: ;
1635 3D2F 1 ORG _NAME ;
1636 3D2F 05F8 R 3D3B R 1 DW _CODE,_LINK ;
1637 3D33 03 4D 41 58 1 DB 3,'MAX' ;
1638 05F8 1 ORG _CODE ;
1639 05F8 A4 A1 DB 0A4H,0A1H ;POP EA, POP BC
1640 05FA 74 BD DB 74H,0BDH ;EA-BC, Skip if borrow
1641 05FC C2 DB 0C2H ;Jump to Push EA
1642 05FD B1 DB 0B1H ;PUSH BC
1643 05FE C1 DB 0C1H ;Jump to next
1644 05FF B4 DB 0B4H ;PUSH EA
1645 $NEXT
1646 0600 48 84 1 DB 48H,84H ;
1647 0602 48 28 1 DB 48H,28H ;
1648
1649 ; MIN ( n n -- n )
1650 ; Return the smaller of top two stack items.
1651
1652 $CODE 3,'MIN',MIN
1653 0604 1 MIN: ;
1654 3D27 1 ORG _NAME ;
1655 3D27 0604 R 3D33 R 1 DW _CODE,_LINK ;
1656 3D2B 03 4D 49 4E 1 DB 3,'MIN' ;
1657 0604 1 ORG _CODE ;
1658 0604 A4 A1 DB 0A4H,0A1H ;POP EA, POP BC
1659 0606 74 BD DB 74H,0BDH ;EA-BC, Skip if borrow
1660 0608 C2 DB 0C2H ;Jump to Push EA
1661 0609 B4 DB 0B4H ;PUSH EA
1662 060A C1 DB 0C1H ;Jump to next
1663 060B B1 DB 0B1H ;PUSH BC
1664 $NEXT
1665 060C 48 84 1 DB 48H,84H ;
1666 060E 48 28 1 DB 48H,28H ;
1667
1668 ; WITHIN ( u ul uh -- t )
1669 ; Return true if u is within the range of ul and uh.
1670
1671 $COLON 6,'WITHIN',WITHI
1672 0610 2 WITHI: ;
1673 3D1B 2 ORG _NAME ;
1674 3D1B 0610 R 3D2B R 2 DW _CODE,_LINK ;
1675 3D1F 06 57 49 54 48 49 4E 2 DB 6,'WITHIN' ;
1676 0610 2 ORG _CODE ;
1677 0610 80 1 DB 80H ;
1678 0611 044A R 059B R 0413 R DW OVER,SUBB,TOR ;ul <= u < uh
1679 0617 059B R 0405 R 05C8 R DW SUBB,RFROM,ULESS,EXIT

```



```

1680      0394 R
1681
1682      ;; Quick Operators
1683
1684      ;
1685
1686
1687
1688
1689      ; 1+      ( n -- n+1 )
1690      $CODE 2, '1+', ONEP
1691 061F      1      ONEP:      ;
1692 3D13      1      ORG      _NAME      ;
1693 3D13 061F R 3D1F R      1      DW      _CODE, _LINK      ;
1694 3D17 02 31 2B      1      DB      2, '1+'      ;
1695 061F      1      ORG      _CODE      ;
1696 061F A1      DB 0A1H      ;POP BC
1697 0620 12      DB 12H      ;BC<BC+1
1698 0621 B1      DB 0B1H      ;PUSH BC
1699      $NEXT
1700 0622 48 84      1      DB 48H,84H      ;
1701 0624 48 28      1      DB 48H,28H      ;
1702
1703      ; 1-      ( n -- n-1 )
1704      $CODE 2, '1-', ONEM
1705 0626      1      ONEM:      ;
1706 3D0B      1      ORG      _NAME      ;
1707 3D0B 0626 R 3D17 R      1      DW      _CODE, _LINK      ;
1708 3D0F 02 31 2D      1      DB      2, '1-'      ;
1709 0626      1      ORG      _CODE      ;
1710 0626 A1      DB 0A1H      ;POP BC
1711 0627 13      DB 013H      ;BC<BC-1
1712 0628 B1      DB 0B1H      ;PUSH BC
1713      $NEXT
1714 0629 48 84      1      DB 48H,84H      ;
1715 062B 48 28      1      DB 48H,28H      ;
1716
1717      ; 2+      ( n -- n+2 )
1718      $CODE 2, '2+', TWOP
1719 062D      1      TWOP:      ;
1720 3D03      1      ORG      _NAME      ;
1721 3D03 062D R 3D0F R      1      DW      _CODE, _LINK      ;
1722 3D07 02 32 2B      1      DB      2, '2+'      ;
1723 062D      1      ORG      _CODE      ;
1724 062D A1      DB 0A1H      ;POP BC
1725 062E 12 12      DB 12H,12H      ;BC<BC+2
1726 0630 B1      DB 0B1H      ;PUSH BC
1727      $NEXT
1728 0631 48 84      1      DB 48H,84H      ;
1729 0633 48 28      1      DB 48H,28H      ;
1730
1731      ; 2-      ( n -- n-2 )
1732      $CODE 2, '2-', TWOM
1733 0635      1      TWOM:      ;
1734 3CFB      1      ORG      _NAME      ;
1735 3CFB 0635 R 3D07 R      1      DW      _CODE, _LINK      ;

```

```

1736 3CFF 02 32 2D      1          DB      2,'2-'          ;
1737 0635              1          ORG      _CODE          ;
1738 0635 A1              DB      0A1H          ;POP BC
1739 0636 13 13          DB      13H,13H       ;BC<BC-2
1740 0638 B1              DB      0B1H          ;PUSH BC
1741                      $NEXT
1742 0639 48 84          1          DB      48H,84H       ;
1743 063B 48 28          1          DB      48H,28H      ;
1744
1745                      ; 2*          ( n -- n*2 )
1746                      $CODE 2,'2*',TWOSL
1747 063D              1          TWOSL:          ;
1748 3CF3              1          ORG      _NAME          ;
1749 3CF3 063D R 3CFF R  1          DW      _CODE,_LINK    ;
1750 3CF7 02 32 2A      1          DB      2,'2*'          ;
1751 063D              1          ORG      _CODE          ;
1752 063D A4              DB      0A4H          ;POP EA
1753 063E 48 A4          DB      48H,0A4H      ;EA Logical Shift Left
1754 0640 B4              DB      0B4H          ;PUSH EA
1755                      $NEXT
1756 0641 48 84          1          DB      48H,84H       ;
1757 0643 48 28          1          DB      48H,28H      ;
1758
1759                      ; 2/          ( n -- n/2 )
1760                      $CODE 2,'2/',TWOSR
1761 0645              1          TWOSR:          ;
1762 3CEB              1          ORG      _NAME          ;
1763 3CEB 0645 R 3CF7 R  1          DW      _CODE,_LINK    ;
1764 3CEF 02 32 2F      1          DB      2,'2/'          ;
1765 0645              1          ORG      _CODE          ;
1766 0645 A4              DB      0A4H          ;POP EA
1767 0646 48 A0          DB      48H,0A0H      ;EA Logical Shift Right
1768 0648 B4              DB      0B4H          ;PUSH EA
1769                      $NEXT
1770 0649 48 84          1          DB      48H,84H       ;
1771 064B 48 28          1          DB      48H,28H      ;
1772
1773                      ;; Divide
1774
1775                      ; UM/MOD      ( udl udh u -- ur uq )
1776                      ;          Unsigned divide of a double by a single. Return mod and quotient.
1777
1778                      $COLON 6,'UM/MOD',UMMOD
1779 064D              2          UMMOD:          ;
1780 3CDF              2          ORG      _NAME          ;
1781 3CDF 064D R 3CEF R  2          DW      _CODE,_LINK    ;
1782 3CE3 06 55 4D 2F 4D 4F 44 2          DB      6,'UM/MOD'      ;
1783 064D              2          ORG      _CODE          ;
1784 064D 80              1          DB      80H          ;
1785 064E 055B R 05C8 R  DW      DDUP,ULESS
1786 0652 03B7 R 069E R  DW      QBRAN,UMM4
1787 0656 057B R 038D R 000F DW      NEGAT,DOLIT,15,TOR
1788 0413 R
1789 065E 0413 R 043B R 047C R UMM1: DW      TOR,DUPP,UPLUS
1790 0664 0413 R 0413 R 043B R DW      TOR,TOR,DUPP,UPLUS
1791 047C R

```

```

1792 066C 0405 R 0565 R 043B R          DW      RFROM, PLUS, DUPP
1793 0672 0405 R 040C R 0442 R          DW      RFROM, RAT, SWAP, TOR
1794          0413 R
1795 067A 047C R 0405 R 046A R          DW      UPLUS, RFROM, ORR
1796 0680 03B7 R 0690 R          DW      QBRAN, UMM2
1797 0684 0413 R 0436 R 061F R          DW      TOR, DROP, ONEP, RFROM
1798          0405 R
1799 068C 03CC R 0692 R          DW      BRAN, UMM3
1800 0690 0436 R          UMM2:      DW      DROP
1801 0692 0405 R          UMM3:      DW      RFROM
1802 0694 039D R 065E R          DW      DONXT, UMM1
1803 0698 0436 R 0442 R 0394 R          DW      DROP, SWAP, EXIT
1804 069E 0436 R 0555 R          UMM4:      DW      DROP, DDROP
1805 06A2 038D R FFFF 043B R          DW      DOLIT, -1, DUPP, EXIT      ;overflow, return max
1806          0394 R
1807
1808          ;      M/MOD      ( d n -- r q )
1809          ;      Signed floored divide of double by single. Return mod and
quotient.
1810
1811          $COLON 5, 'M/MOD', MSMOD
1812 06AA          2      MSMOD:      ;
1813 3CD5          2      ORG      _NAME      ;
1814 3CD5 06AA R 3CE3 R          2      DW      _CODE, _LINK      ;
1815 3CD9 05 4D 2F 4D 4F 44          2      DB      5, 'M/MOD'      ;
1816 06AA          2      ORG      _CODE      ;
1817 06AA 80          1      DB      80H      ;
1818 06AB 043B R 0453 R 043B R          DW      DUPP, ZLESS, DUPP, TOR
1819          0413 R
1820 06B3 03B7 R 06BF R          DW      QBRAN, MMOD1
1821 06B7 057B R 0413 R 0588 R          DW      NEGAT, TOR, DNEGA, RFROM
1822          0405 R
1823 06BF 0413 R 043B R 0453 R          MMOD1:      DW      TOR, DUPP, ZLESS
1824 06C5 03B7 R 06CD R          DW      QBRAN, MMOD2
1825 06C9 040C R 0565 R          DW      RAT, PLUS
1826 06CD 0405 R 064D R 0405 R          MMOD2:      DW      RFROM, UMMOD, RFROM
1827 06D3 03B7 R 06DD R          DW      QBRAN, MMOD3
1828 06D7 0442 R 057B R 0442 R          DW      SWAP, NEGAT, SWAP
1829 06DD 0394 R          MMOD3:      DW      EXIT
1830
1831          ;      /MOD      ( n n -- r q )
1832          ;      Signed divide. Return mod and quotient.
1833
1834          $COLON 4, '/MOD', SLMOD
1835 06DF          2      SLMOD:      ;
1836 3CCB          2      ORG      _NAME      ;
1837 3CCB 06DF R 3CD9 R          2      DW      _CODE, _LINK      ;
1838 3CCF 04 2F 4D 4F 44          2      DB      4, '/MOD'      ;
1839 06DF          2      ORG      _CODE      ;
1840 06DF 80          1      DB      80H      ;
1841 06E0 044A R 0453 R 0442 R          DW      OVER, ZLESS, SWAP, MSMOD, EXIT
1842          06AA R 0394 R
1843
1844          ;      MOD      ( n n -- r )
1845          ;      Signed divide. Return mod only.
1846
1847          $COLON 3, 'MOD', MODD

```

```

1848 06EA          2      MODD:          ;
1849 3CC3          2      ORG      _NAME          ;
1850 3CC3 06EA R 3CCF R  2      DW      _CODE,_LINK      ;
1851 3CC7 03 4D 4F 44  2      DB      3,'MOD'          ;
1852 06EA          2      ORG      _CODE          ;
1853 06EA 80          1      DB      80H          ;
1854 06EB 06DF R 0436 R 0394 R  2      DW      SLMOD,DROP,EXIT
1855
1856          ; /      ( n n -- q )
1857          ;      Signed divide. Return quotient only.
1858
1859          $COLON 1, '/', SLASH
1860 06F1          2      SLASH:          ;
1861 3CBD          2      ORG      _NAME          ;
1862 3CBD 06F1 R 3CC7 R  2      DW      _CODE,_LINK      ;
1863 3CC1 01 2F          2      DB      1, '/'          ;
1864 06F1          2      ORG      _CODE          ;
1865 06F1 80          1      DB      80H          ;
1866 06F2 06DF R 0442 R 0436 R  2      DW      SLMOD,SWAP,DROP,EXIT
1867 0394 R
1868
1869          ;; Multiply
1870
1871          ; UM*      ( u u -- ud )
1872          ;      Unsigned multiply. Return double product.
1873
1874          $COLON 3, 'UM*', UMSTA
1875 06FA          2      UMSTA:          ;
1876 3CB5          2      ORG      _NAME          ;
1877 3CB5 06FA R 3CC1 R  2      DW      _CODE,_LINK      ;
1878 3CB9 03 55 4D 2A  2      DB      3,'UM*'          ;
1879 06FA          2      ORG      _CODE          ;
1880 06FA 80          1      DB      80H          ;
1881 06FB 038D R 0000 0442 R  2      DW      DOLIT,0,SWAP,DOLIT,15,TOR
1882 038D R 000F 0413 R
1883 0707 043B R 047C R 0413 R  UMST1:  DW      DUPP,UPLUS,TOR,TOR
1884 0413 R
1885 070F 043B R 047C R 0405 R  DW      DUPP,UPLUS,RFROM,PLUS,RFROM
1886 0565 R 0405 R
1887 0719 03B7 R 0727 R  DW      QBRAN,UMST2
1888 071D 0413 R 044A R 047C R  DW      TOR,OVER,UPLUS,RFROM,PLUS
1889 0405 R 0565 R
1890 0727 039D R 0707 R  UMST2:  DW      DONXT,UMST1
1891 072B 054A R 0436 R 0394 R  DW      ROT,DROP,EXIT
1892
1893          ; *      ( n n -- n )
1894          ;      Signed multiply. Return single product.
1895
1896          $COLON 1, '*', STAR
1897 0731          2      STAR:          ;
1898 3CAF          2      ORG      _NAME          ;
1899 3CAF 0731 R 3CB9 R  2      DW      _CODE,_LINK      ;
1900 3CB3 01 2A          2      DB      1, '*'          ;
1901 0731          2      ORG      _CODE          ;
1902 0731 80          1      DB      80H          ;
1903 0732 06FA R 0436 R 0394 R  2      DW      UMSTA,DROP,EXIT

```

```

1904
1905 ; M* ( n n -- d )
1906 ; Signed multiply. Return double product.
1907
1908 $COLON 2, 'M*', MSTAR
1909 0738 2 MSTAR: ;
1910 3CA7 2 ORG _NAME ;
1911 3CA7 0738 R 3CB3 R 2 DW _CODE, _LINK ;
1912 3CAB 02 4D 2A 2 DB 2, 'M*' ;
1913 0738 2 ORG _CODE ;
1914 0738 80 1 DB 80H ;
1915 0739 055B R 0473 R 0453 R DW DDUP, XORR, ZLESS, TOR
1916 0413 R
1917 0741 05AC R 0442 R 05AC R DW ABSS, SWAP, ABSS, UMSTA
1918 06FA R
1919 0749 0405 R DW RFROM
1920 074B 03B7 R 0751 R DW QBRAN, MSTA1
1921 074F 0588 R DW DNEGA
1922 0751 0394 R MSTA1: DW EXIT
1923
1924 ; */MOD ( n1 n2 n3 -- r q )
1925 ; Multiply n1 and n2, then divide by n3. Return mod and quotient.
1926
1927 $COLON 5, '*/MOD', SSMOD
1928 0753 2 SSMOD: ;
1929 3C9D 2 ORG _NAME ;
1930 3C9D 0753 R 3CAB R 2 DW _CODE, _LINK ;
1931 3CA1 05 2A 2F 4D 4F 44 2 DB 5, '*/MOD' ;
1932 0753 2 ORG _CODE ;
1933 0753 80 1 DB 80H ;
1934 0754 0413 R 0738 R 0405 R DW TOR, MSTAR, RFROM, MSMOD, EXIT
1935 06AA R 0394 R
1936
1937 ; */ ( n1 n2 n3 -- q )
1938 ; Multiply n1 by n2, then divide by n3. Return quotient only.
1939
1940 $COLON 2, '*/', STASL
1941 075E 2 STASL: ;
1942 3C95 2 ORG _NAME ;
1943 3C95 075E R 3CA1 R 2 DW _CODE, _LINK ;
1944 3C99 02 2A 2F 2 DB 2, '*/' ;
1945 075E 2 ORG _CODE ;
1946 075E 80 1 DB 80H ;
1947 075F 0753 R 0442 R 0436 R DW SSMOD, SWAP, DROP, EXIT
1948 0394 R
1949
1950 ;; Miscellaneous
1951
1952 ; BL ( -- 32 )
1953 ; Return 32, the blank character.
1954
1955 $COLON 2, 'BL', BLANK
1956 0767 2 BLANK: ;
1957 3C8D 2 ORG _NAME ;
1958 3C8D 0767 R 3C99 R 2 DW _CODE, _LINK ;
1959 3C91 02 42 4C 2 DB 2, 'BL' ;

```

```

1960 0767          2      ORG      _CODE          ;
1961 0767 80          1      DB 80H          ;
1962 0768 038D R 0020 0394 R      DW      DOLIT,' ',EXIT
1963
1964          ; >CHAR      ( c -- c )
1965          ;          Filter non-printing characters.
1966
1967          $COLON 5,'>CHAR',TCHAR
1968 076E          2      TCHAR:          ;
1969 3C83          2      ORG      _NAME          ;
1970 3C83 076E R 3C91 R      DW      _CODE,_LINK          ;
1971 3C87 05 3E 43 48 41 52      2      DB      5,'>CHAR'          ;
1972 076E          2      ORG      _CODE          ;
1973 076E 80          1      DB 80H          ;
1974 076F 038D R 007F 0461 R      DW      DOLIT,07FH,ANDD,DUPP      ;mask msb
1975 043B R
1976 0777 038D R 007F 0767 R      DW      DOLIT,127,BLANK,WITHI      ;check for printable
1977 0610 R
1978 077F 03B7 R 0789 R      DW      QBRAN,TCHA1
1979 0783 0436 R 038D R 005F      DW      DROP,DOLIT,'_'          ;replace non-printables
1980 0789 0394 R      TCHA1:      DW      EXIT
1981
1982          ; DEPTH      ( -- n )
1983          ;          Return the depth of the data stack.
1984
1985          $COLON 5,'DEPTH',DEPTH
1986 078B          2      DEPTH:          ;
1987 3C79          2      ORG      _NAME          ;
1988 3C79 078B R 3C87 R      DW      _CODE,_LINK          ;
1989 3C7D 05 44 45 50 54 48      2      DB      5,'DEPTH'          ;
1990 078B          2      ORG      _CODE          ;
1991 078B 80          1      DB 80H          ;
1992 078C 041C R 04A3 R 03DE R      DW      SPAT,SZERO,AT,SWAP,SUBB
1993 0442 R 059B R
1994 0796 038D R 0002 06F1 R      DW      DOLIT,CELLL,SLASH,EXIT
1995 0394 R
1996
1997          ; PICK      ( ... +n -- ... w )
1998          ;          Copy the nth stack item to tos.
1999
2000          $COLON 4,'PICK',PICK
2001 079E          2      PICK:          ;
2002 3C6F          2      ORG      _NAME          ;
2003 3C6F 079E R 3C7D R      DW      _CODE,_LINK          ;
2004 3C73 04 50 49 43 4B          2      DB      4,'PICK'          ;
2005 079E          2      ORG      _CODE          ;
2006 079E 80          1      DB 80H          ;
2007 079F 061F R 063D R      DW      ONEP,TWOSL
2008 07A3 041C R 0565 R 03DE R      DW      SPAT,PLUS,AT,EXIT
2009 0394 R
2010
2011          ;; Memory access
2012
2013          ; +!      ( n a -- )
2014          ;          Add n to the contents at address a.
2015

```

```

2016                                $COLON 2, '+!', PSTOR
2017 07AB                            2      PSTOR:                                ;
2018 3C67                            2      ORG      _NAME                                ;
2019 3C67 07AB R 3C73 R              2      DW      _CODE, _LINK                            ;
2020 3C6B 02 2B 21                   2      DB      2, '+!'                                ;
2021 07AB                            2      ORG      _CODE                                ;
2022 07AB 80                          1      DB      80H                                ;
2023 07AC 0442 R 044A R 03DE R       2      DW      SWAP, OVER, AT, PLUS
2024 0565 R                           2
2025 07B4 0442 R 03D3 R 0394 R       2      DW      SWAP, STORE, EXIT
2026
2027                                ; 2!      ( d a -- )
2028                                ;          Store the double integer to address a.
2029
2030                                $COLON 2, '2!', DSTOR
2031 07BA                            2      DSTOR:                                ;
2032 3C5F                            2      ORG      _NAME                                ;
2033 3C5F 07BA R 3C6B R              2      DW      _CODE, _LINK                            ;
2034 3C63 02 32 21                   2      DB      2, '2!'                                ;
2035 07BA                            2      ORG      _CODE                                ;
2036 07BA 80                          1      DB      80H                                ;
2037 07BB 0442 R 044A R 03D3 R       2      DW      SWAP, OVER, STORE
2038 07C1 062D R 03D3 R 0394 R       2      DW      TWOP, STORE, EXIT
2039
2040                                ; 2@      ( a -- d )
2041                                ;          Fetch double integer from address a.
2042
2043                                $COLON 2, '2@', DAT
2044 07C7                            2      DAT:                                ;
2045 3C57                            2      ORG      _NAME                                ;
2046 3C57 07C7 R 3C63 R              2      DW      _CODE, _LINK                            ;
2047 3C5B 02 32 40                   2      DB      2, '2@'                                ;
2048 07C7                            2      ORG      _CODE                                ;
2049 07C7 80                          1      DB      80H                                ;
2050 07C8 043B R 062D R 03DE R       2      DW      DUPP, TWOP, AT
2051 07CE 0442 R 03DE R 0394 R       2      DW      SWAP, AT, EXIT
2052
2053                                ; COUNT      ( b -- b + n )
2054                                ;          Return count byte of a string and add 1 to byte address.
2055
2056                                $COLON 5, 'COUNT', COUNT
2057 07D4                            2      COUNT:                                ;
2058 3C4D                            2      ORG      _NAME                                ;
2059 3C4D 07D4 R 3C5B R              2      DW      _CODE, _LINK                            ;
2060 3C51 05 43 4F 55 4E 54          2      DB      5, 'COUNT'                                ;
2061 07D4                            2      ORG      _CODE                                ;
2062 07D4 80                          1      DB      80H                                ;
2063 07D5 043B R 061F R              2      DW      DUPP, ONEP
2064 07D9 0442 R 03F1 R 0394 R       2      DW      SWAP, CAT, EXIT
2065
2066                                ; HERE      ( -- a )
2067                                ;          Return the top of the code dictionary.
2068
2069                                $COLON 4, 'HERE', HERE
2070 07DF                            2      HERE:                                ;
2071 3C43                            2      ORG      _NAME                                ;

```

```

2072 3C43 07DF R 3C51 R      2      DW      _CODE,_LINK      ;
2073 3C47 04 48 45 52 45    2      DB      4,'HERE'        ;
2074 07DF                    2      ORG      _CODE          ;
2075 07DF 80                  1      DB      80H             ;
2076 07E0 0511 R 03DE R 0394 R  DW      CP,AT,EXIT      ;
2077
2078 ; PAD      ( -- a )
2079 ;          Return the address of a temporary buffer.
2080
2081 $COLON 3,'PAD',PAD
2082 07E6      2      PAD:
2083 3C3B      2      ORG      _NAME          ;
2084 3C3B 07E6 R 3C47 R      2      DW      _CODE,_LINK      ;
2085 3C3F 03 50 41 44        2      DB      3,'PAD'        ;
2086 07E6      2      ORG      _CODE          ;
2087 07E6 80                  1      DB      80H             ;
2088 07E7 038D R C300 0394 R  DW      DOLIT,PADD,EXIT ;
2089
2090 ; TIB      ( -- a )
2091 ;          Return the address of the terminal input buffer.
2092
2093 $COLON 3,'TIB',TIB
2094 07ED      2      TIB:
2095 3C33      2      ORG      _NAME          ;
2096 3C33 07ED R 3C3F R      2      DW      _CODE,_LINK      ;
2097 3C37 03 54 49 42        2      DB      3,'TIB'        ;
2098 07ED      2      ORG      _CODE          ;
2099 07ED 80                  1      DB      80H             ;
2100 07EE 04DF R 062D R 03DE R  DW      NTIB,TWOP,AT,EXIT ;
2101 0394 R
2102
2103 ; @EXECUTE ( a -- )
2104 ;          Execute vector stored in address a.
2105
2106 $COLON 8,'@EXECUTE',ATEXE
2107 07F6      2      ATEXE:
2108 3C25      2      ORG      _NAME          ;
2109 3C25 07F6 R 3C37 R      2      DW      _CODE,_LINK      ;
2110 3C29 08 40 45 58 45 43 55 2      DB      8,'@EXECUTE'    ;
2111 07F6      2      ORG      _CODE          ;
2112 07F6 80                  1      DB      80H             ;
2113 07F7 03DE R 053D R      DW      AT,QDUP          ;?address or zero
2114 07FB 03B7 R 0801 R      DW      QBRAN,EXE1
2115 07FF 039B R      DW      EXECU          ;execute if non-zero
2116 0801 0394 R      EXE1:  DW      EXIT          ;do nothing if zero
2117
2118 ; CMOVE      ( b1 b2 u -- )
2119 ;          Copy u bytes from b1 to b2.
2120
2121 $COLON 5,'CMOVE',CMOVE
2122 0803      2      CMOVE:
2123 3C1B      2      ORG      _NAME          ;
2124 3C1B 0803 R 3C29 R      2      DW      _CODE,_LINK      ;
2125 3C1F 05 43 4D 4F 56 45    2      DB      5,'CMOVE'    ;
2126 0803      2      ORG      _CODE          ;
2127 0803 80                  1      DB      80H             ;

```



```

2128 0804 0413 R DW TOR
2129 0806 03CC R 081A R DW BRAN,CMOV2
2130 080A 0413 R 043B R 03F1 R CMOV1: DW TOR,DUPP,CAT
2131 0810 040C R 03E9 R DW RAT,CSTOR
2132 0814 061F R DW ONEP
2133 0816 0405 R 061F R DW RFROM,ONEP
2134 081A 039D R 080A R CMOV2: DW DONXT,CMOV1
2135 081E 0555 R 0394 R DW DDROP,EXIT
2136
2137 ; FILL ( b u c -- )
2138 ; Fill u bytes of character c to area beginning at b.
2139
2140 $COLON 4,'FILL',FILL
2141 0822 2 FILL: ;
2142 3C11 2 ORG _NAME ;
2143 3C11 0822 R 3C1F R 2 DW _CODE,_LINK ;
2144 3C15 04 46 49 4C 4C 2 DB 4,'FILL' ;
2145 0822 2 ORG _CODE ;
2146 0822 80 1 DB 80H ;
2147 0823 0442 R 0413 R 0442 R DW SWAP,TOR,SWAP
2148 0829 03CC R 0833 R DW BRAN,FILL2
2149 082D 055B R 03E9 R 061F R FILL1: DW DDUP,CSTOR,ONEP
2150 0833 039D R 082D R FILL2: DW DONXT,FILL1
2151 0837 0555 R 0394 R DW DDROP,EXIT
2152
2153 ; -TRAILING ( b u -- b u )
2154 ; Adjust the count to eliminate trailing white space.
2155
2156 $COLON 9,'-TRAILING',DTRAI
2157 083B 2 DTRAI: ;
2158 3C03 2 ORG _NAME ;
2159 3C03 083B R 3C15 R 2 DW _CODE,_LINK ;
2160 3C07 09 2D 54 52 41 49 4C 2 DB 9,'-TRAILING' ;
2161 083B 2 ORG _CODE ;
2162 083B 80 1 DB 80H ;
2163 083C 0413 R DW TOR
2164 083E 03CC R 0858 R DW BRAN,DTRA2
2165 0842 0767 R 044A R 040C R DTRA1: DW BLANK,OVER,RAT,PLUS,CAT,LESS
2166 0565 R 03F1 R 05E1 R
2167 084E 03B7 R 0858 R DW QBRAN,DTRA2
2168 0852 0405 R 061F R 0394 R DW RFROM,ONEP,EXIT ;adjusted count
2169 0858 039D R 0842 R DTRA2: DW DONXT,DTRA1
2170 085C 038D R 0000 0394 R DW DOLIT,0,EXIT ;count=0
2171
2172 ; PACK$ ( b u a -- a )
2173 ; Build a counted string with u characters from b. Null fill.
2174
2175 $COLON 5,'PACK$',PACKS
2176 0862 2 PACKS: ;
2177 3BF9 2 ORG _NAME ;
2178 3BF9 0862 R 3C07 R 2 DW _CODE,_LINK ;
2179 3BFD 05 50 41 43 4B 24 2 DB 5,'PACK$' ;
2180 0862 2 ORG _CODE ;
2181 0862 80 1 DB 80H ;
2182 0863 043B R 0413 R DW DUPP,TOR ;strings only on cell boundary
2183 0867 044A R 043B R 038D R DW OVER,DUPP,DOLIT,0

```

```

2184      0000
2185 086F 038D R 0002 064D R      DW      DOLIT,CELLL,UMMOD,DROP ;count mod cell
2186      0436 R
2187 0877 059B R 044A R 0565 R      DW      SUBB,OVER,PLUS
2188 087D 038D R 0000 0442 R      DW      DOLIT,0,SWAP,STORE ;null fill cell
2189      03D3 R
2190 0885 055B R 03E9 R 061F R      DW      DDUP,CSTOR,ONEP ;save count
2191 088B 0442 R 0803 R 0405 R      DW      SWAP,CMOVE,RFROM,EXIT ;move string
2192      0394 R
2193
2194      ;; Numeric output, single precision
2195
2196      ;   DIGIT      ( u -- c )
2197      ;           Convert digit u to a character.
2198
2199      $COLON 5,'DIGIT',DIGIT
2200 0893      2   DIGIT:      ;
2201 3BEF      2   ORG      _NAME      ;
2202 3BEF 0893 R 3BFD R      2   DW      _CODE,_LINK      ;
2203 3BF3 05 44 49 47 49 54      2   DB      5,'DIGIT'      ;
2204 0893      2   ORG      _CODE      ;
2205 0893 80      1   DB 80H      ;
2206 0894 038D R 0009 044A R      DW      DOLIT,9,OVER,LESS
2207      05E1 R
2208 089C 038D R 0007 0461 R      DW      DOLIT,7,ANDD,PLUS
2209      0565 R
2210 08A4 038D R 0030 0565 R      DW      DOLIT,'0',PLUS,EXIT
2211      0394 R
2212
2213      ;   EXTRACT      ( n base -- n c )
2214      ;           Extract the least significant digit from n.
2215
2216      $COLON 7,'EXTRACT',EXTRC
2217 08AC      2   EXTRC:      ;
2218 3BE3      2   ORG      _NAME      ;
2219 3BE3 08AC R 3BF3 R      2   DW      _CODE,_LINK      ;
2220 3BE7 07 45 58 54 52 41 43      2   DB      7,'EXTRACT'      ;
2221 08AC      2   ORG      _CODE      ;
2222 08AC 80      1   DB 80H      ;
2223 08AD 038D R 0000 0442 R      DW      DOLIT,0,SWAP,UMMOD
2224      064D R
2225 08B5 0442 R 0893 R 0394 R      DW      SWAP,DIGIT,EXIT
2226
2227      ;   <#      ( -- )
2228      ;           Initiate the numeric output process.
2229
2230      $COLON 2,'<#',BDIGS
2231 08BB      2   BDIGS:      ;
2232 3BDB      2   ORG      _NAME      ;
2233 3BDB 08BB R 3BE7 R      2   DW      _CODE,_LINK      ;
2234 3BDF 02 3C 23      2   DB      2,'<#'      ;
2235 08BB      2   ORG      _CODE      ;
2236 08BB 80      1   DB 80H      ;
2237 08BC 07E6 R 04F3 R 03D3 R      DW      PAD,HLD,STORE,EXIT
2238      0394 R
2239

```

```

2240                ;   HOLD      ( c -- )
2241                ;               Insert a character into the numeric output string.
2242
2243                $COLON 4, 'HOLD', HOLD
2244 08C4            2   HOLD:      ;
2245 3BD1            2   ORG        _NAME      ;
2246 3BD1 08C4 R 3BDF R 2   DW      _CODE, _LINK ;
2247 3BD5 04 48 4F 4C 44 2   DB      4, 'HOLD' ;
2248 08C4            2   ORG        _CODE      ;
2249 08C4 80         1   DB 80H      ;
2250 08C5 04F3 R 03DE R 0626 R 2   DW      HLD, AT, ONEM
2251 08CB 043B R 04F3 R 03D3 R 2   DW      DUPP, HLD, STORE, CSTOR, EXIT
2252      03E9 R 0394 R
2253
2254                ;   #         ( u -- u )
2255                ;               Extract one digit from u and append the digit to output string.
2256
2257                $COLON 1, '#', DIG
2258 08D5            2   DIG:      ;
2259 3BCB            2   ORG        _NAME      ;
2260 3BCB 08D5 R 3BD5 R 2   DW      _CODE, _LINK ;
2261 3BCF 01 23      2   DB      1, '#' ;
2262 08D5            2   ORG        _CODE      ;
2263 08D5 80         1   DB 80H      ;
2264 08D6 04CB R 03DE R 08AC R 2   DW      BASE, AT, EXTRC, HOLD, EXIT
2265      08C4 R 0394 R
2266
2267                ;   #S        ( u -- 0 )
2268                ;               Convert u until all digits are added to the output string.
2269
2270                $COLON 2, '#S', DIGS
2271 08E0            2   DIGS:     ;
2272 3BC3            2   ORG        _NAME      ;
2273 3BC3 08E0 R 3BCF R 2   DW      _CODE, _LINK ;
2274 3BC7 02 23 53  2   DB      2, '#S' ;
2275 08E0            2   ORG        _CODE      ;
2276 08E0 80         1   DB 80H      ;
2277 08E1 08D5 R 043B R 2   DIGS1:  DW      DIG, DUPP
2278 08E5 03B7 R 08ED R 2   DW      QBRAN, DIGS2
2279 08E9 03CC R 08E1 R 2   DW      BRAN, DIGS1
2280 08ED 0394 R    2   DIGS2:  DW      EXIT
2281
2282                ;   SIGN      ( n -- )
2283                ;               Add a minus sign to the numeric output string.
2284
2285                $COLON 4, 'SIGN', SIGN
2286 08EF            2   SIGN:     ;
2287 3BB9            2   ORG        _NAME      ;
2288 3BB9 08EF R 3BC7 R 2   DW      _CODE, _LINK ;
2289 3BBD 04 53 49 47 4E 2   DB      4, 'SIGN' ;
2290 08EF            2   ORG        _CODE      ;
2291 08EF 80         1   DB 80H      ;
2292 08F0 0453 R    2   DW      ZLESS
2293 08F2 03B7 R 08FC R 2   DW      QBRAN, SIGN1
2294 08F6 038D R 002D 08C4 R 2   DW      DOLIT, '-', HOLD
2295 08FC 0394 R    2   SIGN1:  DW      EXIT

```

```

2296
2297 ; #> ( w -- b u )
2298 ; Prepare the output string to be TYPE'd.
2299
2300 $COLON 2, '#>', EDIGS
2301 08FE 2 EDIGS: ;
2302 3BB1 2 ORG _NAME ;
2303 3BB1 08FE R 3BBD R 2 DW _CODE, _LINK ;
2304 3BB5 02 23 3E 2 DB 2, '#>' ;
2305 08FE 2 ORG _CODE ;
2306 08FE 80 1 DB 80H ;
2307 08FF 0436 R 04F3 R 03DE R DW DROP, HLD, AT
2308 0905 07E6 R 044A R 059B R DW PAD, OVER, SUBB, EXIT
2309 0394 R
2310
2311 ; str ( n -- b u )
2312 ; Convert a signed integer to a numeric string.
2313
2314 $COLON 3, 'str', STR
2315 090D 2 STR: ;
2316 3BA9 2 ORG _NAME ;
2317 3BA9 090D R 3BB5 R 2 DW _CODE, _LINK ;
2318 3BAD 03 73 74 72 2 DB 3, 'str' ;
2319 090D 2 ORG _CODE ;
2320 090D 80 1 DB 80H ;
2321 090E 043B R 0413 R 05AC R DW DUPP, TOR, ABSS
2322 0914 08BB R 08E0 R 0405 R DW BDIGS, DIGS, RFROM
2323 091A 08EF R 08FE R 0394 R DW SIGN, EDIGS, EXIT
2324
2325 ; HEX ( -- )
2326 ; Use radix 16 as base for numeric conversions.
2327
2328 $COLON 3, 'HEX', HEX
2329 0920 2 HEX: ;
2330 3BA1 2 ORG _NAME ;
2331 3BA1 0920 R 3BAD R 2 DW _CODE, _LINK ;
2332 3BA5 03 48 45 58 2 DB 3, 'HEX' ;
2333 0920 2 ORG _CODE ;
2334 0920 80 1 DB 80H ;
2335 0921 038D R 0010 04CB R DW DOLIT, 16, BASE, STORE, EXIT
2336 03D3 R 0394 R
2337
2338 ; DECIMAL ( -- )
2339 ; Use radix 10 as base for numeric conversions.
2340
2341 $COLON 7, 'DECIMAL', DECIM
2342 092B 2 DECIM: ;
2343 3B95 2 ORG _NAME ;
2344 3B95 092B R 3BA5 R 2 DW _CODE, _LINK ;
2345 3B99 07 44 45 43 49 4D 41 2 DB 7, 'DECIMAL' ;
2346 092B 2 ORG _CODE ;
2347 092B 80 1 DB 80H ;
2348 092C 038D R 000A 04CB R DW DOLIT, 10, BASE, STORE, EXIT
2349 03D3 R 0394 R
2350
2351 ;; Numeric input, single precision

```

```

2352
2353 ; DIGIT? ( c base -- u t )
2354 ; Convert a character to its numeric value. A flag indicates
success.
2355
2356 $COLON 6, 'DIGIT?', DIGTQ
2357 0936 2 DIGTQ: ;
2358 3B89 2 ORG _NAME ;
2359 3B89 0936 R 3B99 R 2 DW _CODE, _LINK ;
2360 3B8D 06 44 49 47 49 54 3F 2 DB 6, 'DIGIT?' ;
2361 0936 2 ORG _CODE ;
2362 0936 80 1 DB 80H ;
2363 0937 0413 R 038D R 0030 DW TOR, DOLIT, '0', SUBB
2364 059B R
2365 093F 038D R 0009 044A R DW DOLIT, 9, OVER, LESS
2366 05E1 R
2367 0947 03B7 R 095B R DW QBRAN, DGTQ1
2368 094B 038D R 0007 059B R DW DOLIT, 7, SUBB
2369 0951 043B R 038D R 000A DW DUPP, DOLIT, 10, LESS, ORR
2370 05E1 R 046A R
2371 095B 043B R 0405 R 05C8 R DGTQ1: DW DUPP, RFROM, ULESS, EXIT
2372 0394 R
2373
2374 ; NUMBER? ( a -- n T | a F )
2375 ; Convert a number string to integer. Push a flag on tos.
2376
2377 $COLON 7, 'NUMBER?', NUMBQ
2378 0963 2 NUMBQ: ;
2379 3B7D 2 ORG _NAME ;
2380 3B7D 0963 R 3B8D R 2 DW _CODE, _LINK ;
2381 3B81 07 4E 55 4D 42 45 52 2 DB 7, 'NUMBER?' ;
2382 0963 2 ORG _CODE ;
2383 0963 80 1 DB 80H ;
2384 0964 04CB R 03DE R 0413 R DW BASE, AT, TOR, DOLIT, 0, OVER, COUNT
2385 038D R 0000 044A R
2386 07D4 R
2387 0972 044A R 03F1 R 038D R DW OVER, CAT, DOLIT, '$', EQUAL
2388 0024 05B9 R
2389 097C 03B7 R 098A R DW QBRAN, NUMQ1
2390 0980 0920 R 0442 R 061F R DW HEX, SWAP, ONEP
2391 0986 0442 R 0626 R DW SWAP, ONEM
2392 098A 044A R 03F1 R 038D R NUMQ1: DW OVER, CAT, DOLIT, '-', EQUAL, TOR
2393 002D 05B9 R 0413 R
2394 0996 0442 R 040C R 059B R DW SWAP, RAT, SUBB, SWAP, RAT, PLUS, QDUP
2395 0442 R 040C R 0565 R
2396 053D R
2397 09A4 03B7 R 09EE R DW QBRAN, NUMQ6
2398 09A8 0626 R 0413 R DW ONEM, TOR
2399 09AC 043B R 0413 R 03F1 R NUMQ2: DW DUPP, TOR, CAT, BASE, AT, DIGTQ
2400 04CB R 03DE R 0936 R
2401 09B8 03B7 R 09E0 R DW QBRAN, NUMQ4
2402 09BC 0442 R 04CB R 03DE R DW SWAP, BASE, AT, STAR, PLUS, RFROM
2403 0731 R 0565 R 0405 R
2404 09C8 061F R DW ONEP
2405 09CA 039D R 09AC R DW DONXT, NUMQ2
2406 09CE 040C R 0442 R 0436 R DW RAT, SWAP, DROP
2407 09D4 03B7 R 09DA R DW QBRAN, NUMQ3

```

```

2408 09D8 057B R          DW      NEGAT
2409 09DA 0442 R          NUMQ3:   DW      SWAP
2410 09DC 03CC R 09EC R          DW      BRAN,NUMQ5
2411 09E0 0405 R 0405 R 0555 R  NUMQ4:   DW      RFROM,RFROM,DDROP,DDROP,DOLIT,0
2412      0555 R 038D R 0000
2413 09EC 043B R          NUMQ5:   DW      DUPP
2414 09EE 0405 R 0555 R          NUMQ6:   DW      RFROM,DDROP
2415 09F2 0405 R 04CB R 03D3 R  DW      RFROM,BASE,STORE,EXIT
2416      0394 R
2417
2418      ;; Basic I/O
2419
2420      ;   ?KEY      ( -- c T | F )
2421      ;           Return input character and true, or a false if no input.
2422
2423      $COLON 4,'?KEY',QKEY
2424 09FA      2      QKEY:
2425 3B73      2      ORG      _NAME
2426 3B73 09FA R 3B81 R          DW      _CODE,_LINK
2427 3B77 04 3F 4B 45 59      2      DB      4,'?KEY'
2428 09FA      2      ORG      _CODE
2429 09FA 80      1      DB 80H
2430 09FB 04AD R 07F6 R 0394 R  DW      TQKEY,ATEXE,EXIT
2431
2432      ;   KEY      ( -- c )
2433      ;           Wait for and return an input character.
2434
2435      $COLON 3,'KEY',KEY
2436 0A01      2      KEY:
2437 3B6B      2      ORG      _NAME
2438 3B6B 0A01 R 3B77 R          DW      _CODE,_LINK
2439 3B6F 03 4B 45 59      2      DB      3,'KEY'
2440 0A01      2      ORG      _CODE
2441 0A01 80      1      DB 80H
2442 0A02 09FA R          KEY1:   DW      QKEY
2443 0A04 03B7 R 0A02 R          DW      QBRAN,KEY1
2444 0A08 0394 R          DW      EXIT
2445
2446      ;   EMIT      ( c -- )
2447      ;           Send a character to the output device.
2448
2449      $COLON 4,'EMIT',EMIT
2450 0A0A      2      EMIT:
2451 3B61      2      ORG      _NAME
2452 3B61 0A0A R 3B6F R          DW      _CODE,_LINK
2453 3B65 04 45 4D 49 54      2      DB      4,'EMIT'
2454 0A0A      2      ORG      _CODE
2455 0A0A 80      1      DB 80H
2456 0A0B 04B2 R 07F6 R 0394 R  DW      TEMIT,ATEXE,EXIT
2457
2458      ;   NUF?      ( -- t )
2459      ;           Return false if no input, else pause and if CR return true.
2460
2461      $COLON 4,'NUF?',NUFQ
2462 0A11      2      NUFQ:
2463 3B57      2      ORG      _NAME

```

```

2464 3B57 0A11 R 3B65 R      2      DW      _CODE,_LINK      ;
2465 3B5B 04 4E 55 46 3F    2      DB      4,'NUF?'      ;
2466 0A11                2      ORG      _CODE      ;
2467 0A11 80                1      DB      80H      ;
2468 0A12 09FA R 043B R      DW      QKEY,DUPP      ;
2469 0A16 03B7 R 0A24 R      DW      QBRAN,NUFQ1    ;
2470 0A1A 0555 R 0A01 R 038D R  DW      DDROP,KEY,DOLIT,CRR,EQUAL
2471 000D 05B9 R
2472 0A24 0394 R      NUFQ1:      DW      EXIT
2473
2474      ;   PACE      ( -- )
2475      ;           Send a pace character for the file downloading process.
2476
2477      $COLON 4,'PACE',PACE
2478 0A26      2      PACE:      ;
2479 3B4D      2      ORG      _NAME      ;
2480 3B4D 0A26 R 3B5B R      2      DW      _CODE,_LINK      ;
2481 3B51 04 50 41 43 45    2      DB      4,'PACE'      ;
2482 0A26      2      ORG      _CODE      ;
2483 0A26 80                1      DB      80H      ;
2484 0A27 038D R 000B 0A0A R  DW      DOLIT,11,EMIT,EXIT
2485 0394 R
2486
2487      ;   SPACE      ( -- )
2488      ;           Send the blank character to the output device.
2489
2490      $COLON 5,'SPACE',SPACE
2491 0A2F      2      SPACE:      ;
2492 3B43      2      ORG      _NAME      ;
2493 3B43 0A2F R 3B51 R      2      DW      _CODE,_LINK      ;
2494 3B47 05 53 50 41 43 45  2      DB      5,'SPACE'      ;
2495 0A2F      2      ORG      _CODE      ;
2496 0A2F 80                1      DB      80H      ;
2497 0A30 0767 R 0A0A R 0394 R  DW      BLANK,EMIT,EXIT
2498
2499      ;   SPACES      ( +n -- )
2500      ;           Send n spaces to the output device.
2501
2502      $COLON 6,'SPACES',SPACS
2503 0A36      2      SPACS:      ;
2504 3B37      2      ORG      _NAME      ;
2505 3B37 0A36 R 3B47 R      2      DW      _CODE,_LINK      ;
2506 3B3B 06 53 50 41 43 45 53  2      DB      6,'SPACES'      ;
2507 0A36      2      ORG      _CODE      ;
2508 0A36 80                1      DB      80H      ;
2509 0A37 038D R 0000 05F8 R  DW      DOLIT,0,MAX,TOR
2510 0413 R
2511 0A3F 03CC R 0A45 R      DW      BRAN,CHAR2
2512 0A43 0A2F R      CHAR1:      DW      SPACE
2513 0A45 039D R 0A43 R      CHAR2:      DW      DONXT,CHAR1
2514 0A49 0394 R      DW      EXIT
2515
2516      ;   TYPE      ( b u -- )
2517      ;           Output u characters from b.
2518
2519      $COLON 4,'TYPE',TYPEE

```

```

2520 0A4B          2      TYPEE:          ;
2521 3B2D          2      ORG      _NAME          ;
2522 3B2D 0A4B R 3B3B R      2      DW      _CODE,_LINK      ;
2523 3B31 04 54 59 50 45    2      DB      4,'TYPE'        ;
2524 0A4B          2      ORG      _CODE          ;
2525 0A4B 80        1      DB      80H          ;
2526 0A4C 0413 R      DW      TOR          ;
2527 0A4E 03CC R 0A5A R      DW      BRAN,TYPE2      ;
2528 0A52 043B R 03F1 R 0A0A R  TYPE1: DW      DUPP,CAT,EMIT  ;
2529 0A58 061F R      DW      ONEP          ;
2530 0A5A 039D R 0A52 R      TYPE2: DW      DONXT,TYPE1  ;
2531 0A5E 0436 R 0394 R      DW      DROP,EXIT      ;
2532
2533          ; CR      ( -- )
2534          ;      Output a carriage return and a line feed.
2535
2536          $COLON 2,'CR',CR
2537 0A62          2      CR:          ;
2538 3B25          2      ORG      _NAME          ;
2539 3B25 0A62 R 3B31 R      2      DW      _CODE,_LINK      ;
2540 3B29 02 43 52        2      DB      2,'CR'        ;
2541 0A62          2      ORG      _CODE          ;
2542 0A62 80        1      DB      80H          ;
2543 0A63 038D R 000D 0A0A R  DW      DOLIT,CRR,EMIT  ;
2544 0A69 038D R 000A 0A0A R  DW      DOLIT,LF,EMIT,EXIT ;
2545 0394 R
2546
2547          ; do$      ( -- a )
2548          ;      Return the address of a compiled string.
2549
2550          $COLON COMPO+3,'do$',DOSTR
2551 0A71          2      DOSTR:         ;
2552 3B1D          2      ORG      _NAME          ;
2553 3B1D 0A71 R 3B29 R      2      DW      _CODE,_LINK      ;
2554 3B21 43 64 6F 24        2      DB      COMPO+3,'do$'  ;
2555 0A71          2      ORG      _CODE          ;
2556 0A71 80        1      DB      80H          ;
2557 0A72 0405 R 040C R 0405 R  DW      RFROM,RAT,RFROM,COUNT,PLUS ;
2558 07D4 R 0565 R
2559 0A7C 0413 R 0442 R 0413 R  DW      TOR,SWAP,TOR,EXIT
2560 0394 R
2561
2562          ; $"|      ( -- a )
2563          ;      Run time routine compiled by $" . Return address of a compiled
string.
2564
2565          $COLON COMPO+3,'$"|',STRQP
2566 0A84          2      STRQP:         ;
2567 3B15          2      ORG      _NAME          ;
2568 3B15 0A84 R 3B21 R      2      DW      _CODE,_LINK      ;
2569 3B19 43 24 22 7C        2      DB      COMPO+3,'$"|'  ;
2570 0A84          2      ORG      _CODE          ;
2571 0A84 80        1      DB      80H          ;
2572 0A85 0A71 R 0394 R      DW      DOSTR,EXIT      ;force a call to do$
2573
2574          ; ."|      ( -- )
2575          ;      Run time routine of ." . Output a compiled string.

```



```

2576
2577                                $COLON  COMPO+3, '.'|',DOTQP
2578 0A89                          2      DOTQP:                                ;
2579 3B0D                          2      ORG      _NAME                                ;
2580 3B0D 0A89 R 3B19 R            2      DW      _CODE,_LINK                            ;
2581 3B11 43 2E 22 7C            2      DB      COMPO+3, '.'|'                                ;
2582 0A89                          2      ORG      _CODE                                ;
2583 0A89 80                      1      DB      80H                                ;
2584 0A8A 0A71 R 07D4 R 0A4B R    DW      DOSTR,COUNT,TYPEE,EXIT
2585 0394 R
2586
2587                                ; .R      ( n +n -- )
2588                                ;      Display an integer in a field of n columns, right justified.
2589
2590                                $COLON  2, '.R',DOTR
2591 0A92                          2      DOTR:                                ;
2592 3B05                          2      ORG      _NAME                                ;
2593 3B05 0A92 R 3B11 R            2      DW      _CODE,_LINK                            ;
2594 3B09 02 2E 52                2      DB      2, '.R'                                ;
2595 0A92                          2      ORG      _CODE                                ;
2596 0A92 80                      1      DB      80H                                ;
2597 0A93 0413 R 090D R 0405 R    DW      TOR,STR,RFROM,OVER,SUBB
2598 044A R 059B R
2599 0A9D 0A36 R 0A4B R 0394 R    DW      SPACS,TYPEE,EXIT
2600
2601                                ; U.R      ( u +n -- )
2602                                ;      Display an unsigned integer in n column, right justified.
2603
2604                                $COLON  3, 'U.R',UDOTR
2605 0AA3                          2      UDOTR:                               ;
2606 3AFD                          2      ORG      _NAME                                ;
2607 3AFD 0AA3 R 3B09 R            2      DW      _CODE,_LINK                            ;
2608 3B01 03 55 2E 52            2      DB      3, 'U.R'                                ;
2609 0AA3                          2      ORG      _CODE                                ;
2610 0AA3 80                      1      DB      80H                                ;
2611 0AA4 0413 R 08BB R 08E0 R    DW      TOR,BDIGS,DIGS,EDIGS
2612 08FE R
2613 0AAC 0405 R 044A R 059B R    DW      RFROM,OVER,SUBB
2614 0AB2 0A36 R 0A4B R 0394 R    DW      SPACS,TYPEE,EXIT
2615
2616                                ; U.      ( u -- )
2617                                ;      Display an unsigned integer in free format.
2618
2619                                $COLON  2, 'U.',UDOT
2620 0AB8                          2      UDOT:                                ;
2621 3AF5                          2      ORG      _NAME                                ;
2622 3AF5 0AB8 R 3B01 R            2      DW      _CODE,_LINK                            ;
2623 3AF9 02 55 2E                2      DB      2, 'U.'                                ;
2624 0AB8                          2      ORG      _CODE                                ;
2625 0AB8 80                      1      DB      80H                                ;
2626 0AB9 08BB R 08E0 R 08FE R    DW      BDIGS,DIGS,EDIGS
2627 0ABF 0A2F R 0A4B R 0394 R    DW      SPACE,TYPEE,EXIT
2628
2629                                ; .      ( w -- )
2630                                ;      Display an integer in free format, preceded by a space.
2631

```

```

2632                                $COLON 1, '.', DOT
2633 0AC5                            2      DOT:                                ;
2634 3AEF                            2      ORG      _NAME                                ;
2635 3AEF 0AC5 R 3AF9 R                2      DW      _CODE, _LINK                            ;
2636 3AF3 01 2E                        2      DB      1, '.'                                ;
2637 0AC5                            2      ORG      _CODE                                ;
2638 0AC5 80                           1      DB      80H                                ;
2639 0AC6 04CB R 03DE R 038D R          2      DW      BASE, AT, DOLIT, 10, XORR    ;?decimal
2640 000A 0473 R                        2
2641 0AD0 03B7 R 0AD8 R                2      DW      QBRAN, DOT1
2642 0AD4 0AB8 R 0394 R                2      DW      UDOT, EXIT    ;no, display unsigned
2643 0AD8 090D R 0A2F R 0A4B R  DOT1:  2      DW      STR, SPACE, TYPEE, EXIT    ;yes, display signed
2644 0394 R
2645
2646                                ; ?      ( a -- )
2647                                ;      Display the contents in a memory cell.
2648
2649                                $COLON 1, '?', QUEST
2650 0AE0                            2      QUEST:                                ;
2651 3AE9                            2      ORG      _NAME                                ;
2652 3AE9 0AE0 R 3AF3 R                2      DW      _CODE, _LINK                            ;
2653 3AED 01 3F                          2      DB      1, '?'                                ;
2654 0AE0                            2      ORG      _CODE                                ;
2655 0AE0 80                             1      DB      80H                                ;
2656 0AE1 03DE R 0AC5 R 0394 R          2      DW      AT, DOT, EXIT
2657
2658                                ;; Parsing
2659
2660                                ; parse      ( b u c -- b u delta ; <string> )
2661                                ;      Scan string delimited by c. Return found string and its offset.
2662
2663                                $COLON 5, 'parse', PARS
2664 0AE7                            2      PARS:                                ;
2665 3ADF                            2      ORG      _NAME                                ;
2666 3ADF 0AE7 R 3AED R                2      DW      _CODE, _LINK                            ;
2667 3AE3 05 70 61 72 73 65            2      DB      5, 'parse'                            ;
2668 0AE7                            2      ORG      _CODE                                ;
2669 0AE7 80                             1      DB      80H                                ;
2670 0AE8 04D0 R 03D3 R 044A R          2      DW      TEMP, STORE, OVER, TOR, DUPP
2671 0413 R 043B R
2672 0AF2 03B7 R 0B70 R                2      DW      QBRAN, PARS8
2673 0AF6 0626 R 04D0 R 03DE R          2      DW      ONEM, TEMP, AT, BLANK, EQUAL
2674 0767 R 05B9 R
2675 0B00 03B7 R 0B2A R                2      DW      QBRAN, PARS3
2676 0B04 0413 R                        2      DW      TOR
2677 0B06 0767 R 044A R 03F1 R  PARS1:  2      DW      BLANK, OVER, CAT    ;skip leading blanks ONLY
2678 0B0C 059B R 0453 R 056F R          2      DW      SUBB, ZLESS, INVER
2679 0B12 03B7 R 0B28 R                2      DW      QBRAN, PARS2
2680 0B16 061F R                        2      DW      ONEP
2681 0B18 039D R 0B06 R                2      DW      DONXT, PARS1
2682 0B1C 0405 R 0436 R 038D R          2      DW      RFROM, DROP, DOLIT, 0, DUPP, EXIT
2683 0000 043B R 0394 R
2684 0B28 0405 R                        2      PARS2:  DW      RFROM
2685 0B2A 044A R 0442 R                2      PARS3:  DW      OVER, SWAP
2686 0B2E 0413 R                        2      DW      TOR
2687 0B30 04D0 R 03DE R 044A R  PARS4:  2      DW      TEMP, AT, OVER, CAT, SUBB    ;scan for delimiter

```

```

2688      03F1 R 059B R
2689 0B3A 04D0 R 03DE R 0767 R      DW      TEMP,AT,BLANK,EQUAL
2690      05B9 R
2691 0B42 03B7 R 0B48 R      DW      QBRAN,PARS5
2692 0B46 0453 R      DW      ZLESS
2693 0B48 03B7 R 0B5A R      PARS5:   DW      QBRAN,PARS6
2694 0B4C 061F R      DW      ONEP
2695 0B4E 039D R 0B30 R      DW      DONXT,PARS4
2696 0B52 043B R 0413 R      DW      DUPP,TOR
2697 0B56 03CC R 0B64 R      DW      BRAN,PARS7
2698 0B5A 0405 R 0436 R 043B R    PARS6:   DW      RFROM,DROP,DUPP
2699 0B60 061F R 0413 R      DW      ONEP,TOR
2700 0B64 044A R 059B R      PARS7:   DW      OVER,SUBB
2701 0B68 0405 R 0405 R 059B R    DW      RFROM,RFROM,SUBB,EXIT
2702      0394 R
2703 0B70 044A R 0405 R 059B R    PARS8:   DW      OVER,RFROM,SUBB,EXIT
2704      0394 R
2705
2706      ; PARSE      ( c -- b u ; <string> )
2707      ; Scan input stream and return counted string delimited by c.
2708
2709      $COLON 5,'PARSE',PARSE
2710 0B78      2 PARSE:
2711 3AD5      2 ORG      _NAME
2712 3AD5 0B78 R 3AE3 R      2 DW      _CODE,_LINK
2713 3AD9 05 50 41 52 53 45  2 DB      5,'PARSE'
2714 0B78      2 ORG      _CODE
2715 0B78 80      1 DB      80H
2716 0B79 0413 R 07ED R 04DA R    DW      TOR,TIB,INN,AT,PLUS ;current input buffer pointer
2717      03DE R 0565 R
2718 0B83 04DF R 03DE R 04DA R    DW      NTIB,AT,INN,AT,SUBB ;remaining count
2719      03DE R 059B R
2720 0B8D 0405 R 0AE7 R 04DA R    DW      RFROM,PARS,INN,PSTOR,EXIT
2721      07AB R 0394 R
2722
2723      ; .(      ( -- )
2724      ; Output following string up to next ) .
2725
2726      $COLON IMEDD+2,'(.DOTPR
2727 0B97      2 DOTPR:
2728 3ACD      2 ORG      _NAME
2729 3ACD 0B97 R 3AD9 R      2 DW      _CODE,_LINK
2730 3AD1 82 2E 28      2 DB      IMEDD+2,'(' ;
2731 0B97      2 ORG      _CODE
2732 0B97 80      1 DB      80H
2733 0B98 038D R 0029 0B78 R      DW      DOLIT,')',PARSE,TYPEE,EXIT
2734      0A4B R 0394 R
2735
2736      ; (      ( -- )
2737      ; Ignore following string up to next ) . A comment.
2738
2739      $COLON IMEDD+1,'(,PAREN
2740 0BA2      2 PAREN:
2741 3AC7      2 ORG      _NAME
2742 3AC7 0BA2 R 3AD1 R      2 DW      _CODE,_LINK
2743 3ACB 81 28      2 DB      IMEDD+1,'(' ;

```

```

2744 OBA2          2          ORG      _CODE          ;
2745 OBA2          1          DB      80H          ;
2746 OBA3          038D R 0029 0B78 R
2747              0555 R 0394 R
2748
2749              ; \      ( -- )
2750              ;      Ignore following text till the end of line.
2751
2752              $COLON  IMEDD+1, '\',BKSLA
2753 OBAD          2          BKSLA:          ;
2754 3AC1          2          ORG      _NAME          ;
2755 3AC1          0BAD R 3ACB R      2          DW      _CODE, _LINK      ;
2756 3AC5          81 5C          2          DB      IMEDD+1, '\ '      ;
2757 OBAD          2          ORG      _CODE          ;
2758 OBAD          1          DB      80H          ;
2759 OBAE          04DF R 03DE R 04DA R
2760              03D3 R 0394 R
2761
2762              ; CHAR      ( -- c )
2763              ;      Parse next word and return its first character.
2764
2765              $COLON  4, 'CHAR', CHAR
2766 OBB8          2          CHAR:          ;
2767 3AB7          2          ORG      _NAME          ;
2768 3AB7          0BB8 R 3AC5 R      2          DW      _CODE, _LINK      ;
2769 3ABB          04 43 48 41 52      2          DB      4, 'CHAR'      ;
2770 OBB8          2          ORG      _CODE          ;
2771 OBB8          1          DB      80H          ;
2772 OBB9          0767 R 0B78 R 0436 R
2773              03F1 R 0394 R
2774
2775              ; TOKEN      ( -- a ; <string> )
2776              ;      Parse a word from input stream and copy it to name dictionary.
2777
2778              $COLON  5, 'TOKEN', TOKEN
2779 OBC3          2          TOKEN:          ;
2780 3AAD          2          ORG      _NAME          ;
2781 3AAD          0BC3 R 3ABB R      2          DW      _CODE, _LINK      ;
2782 3AB1          05 54 4F 4B 45 4E      2          DB      5, 'TOKEN'      ;
2783 OBC3          2          ORG      _CODE          ;
2784 OBC3          1          DB      80H          ;
2785 OBC4          0767 R 0B78 R 038D R
2786              001F 0604 R
2787 OBCE          0516 R 03DE R 044A R
2788              059B R 0635 R
2789 OBD8          0862 R 0394 R
2790              DW      PACKS, EXIT
2791
2792              ; WORD      ( c -- a ; <string> )
2793              ;      Parse a word from input stream and copy it to code dictionary.
2794
2795              $COLON  4, 'WORD', WORDD
2796 OBD8          2          WORDD:          ;
2797 3AA3          2          ORG      _NAME          ;
2798 3AA3          0BDC R 3AB1 R      2          DW      _CODE, _LINK      ;
2799 3AA7          04 57 4F 52 44      2          DB      4, 'WORD'      ;
2800 OBD8          2          ORG      _CODE          ;

```

```

2800 0BDC 80 1 DB 80H ;
2801 0BDD 0B78 R 07DF R 0862 R DW PARSE,HERE,PACKS,EXIT ;
2802 0394 R
2803
2804 ;; Dictionary search
2805
2806 ; NAME> ( na -- ca )
2807 ; Return a code address given a name address.
2808
2809 $COLON 5,'NAME> ',NAMET
2810 0BE5 2 NAMET: ;
2811 3A99 2 ORG _NAME ;
2812 3A99 0BE5 R 3AA7 R 2 DW _CODE,_LINK ;
2813 3A9D 05 4E 41 4D 45 3E 2 DB 5,'NAME>' ;
2814 0BE5 2 ORG _CODE ;
2815 0BE5 80 1 DB 80H ;
2816 0BE6 0635 R 0635 R 03DE R DW TWOM,TWOM,AT,EXIT ;
2817 0394 R
2818
2819 ; SAME? ( a a u -- a a f \ -0+ )
2820 ; Compare u cells in two strings. Return 0 if identical.
2821
2822 $COLON 5,'SAME?' ,SAMEQ
2823 0BEE 2 SAMEQ: ;
2824 3A8F 2 ORG _NAME ;
2825 3A8F 0BEE R 3A9D R 2 DW _CODE,_LINK ;
2826 3A93 05 53 41 4D 45 3F 2 DB 5,'SAME?' ;
2827 0BEE 2 ORG _CODE ;
2828 0BEE 80 1 DB 80H ;
2829 0BEF 0413 R DW TOR
2830 0BF1 03CC R 0C17 R DW BRAN,SAME2
2831 0BF5 044A R 040C R 063D R SAME1: DW OVER,RAT,TWOSL,PLUS,AT
2832 0565 R 03DE R
2833 0BFF 044A R 040C R 063D R DW OVER,RAT,TWOSL,PLUS,AT
2834 0565 R 03DE R
2835 0C09 059B R 053D R DW SUBB,QDUP
2836 0C0D 03B7 R 0C17 R DW QBRAN,SAME2
2837 0C11 0405 R 0436 R 0394 R DW RFROM,DROP,EXIT ;strings not equal
2838 0C17 039D R 0BF5 R SAME2: DW DONXT,SAME1
2839 0C1B 038D R 0000 0394 R DW DOLIT,0,EXIT ;strings equal
2840
2841 ; find ( a va -- ca na | a F )
2842 ; Search a vocabulary for a string. Return ca and na if succeeded.
2843
2844 $COLON 4,'find',FIND
2845 0C21 2 FIND: ;
2846 3A85 2 ORG _NAME ;
2847 3A85 0C21 R 3A93 R 2 DW _CODE,_LINK ;
2848 3A89 04 66 69 6E 64 2 DB 4,'find' ;
2849 0C21 2 ORG _CODE ;
2850 0C21 80 1 DB 80H ;
2851 0C22 0442 R 043B R 03F1 R DW SWAP,DUPP,CAT
2852 0C28 038D R 0002 06F1 R DW DOLIT,CELLL,SLASH,TEMP,STORE
2853 04D0 R 03D3 R
2854 0C32 043B R 03DE R 0413 R DW DUPP,AT,TOR,TWOP,SWAP
2855 062D R 0442 R

```

```

2856 0C3C 03DE R 043B R      FIND1:      DW      AT,DUPP
2857 0C40 03B7 R 0C6C R      DW      QBRAN,FIND6
2858 0C44 043B R 03DE R 038D R      DW      DUPP,AT,DOLIT,MASKK,ANDD,RAT,XORR
2859      7F1F 0461 R 040C R
2860      0473 R
2861 0C52 03B7 R 0C60 R      DW      QBRAN,FIND2
2862 0C56 062D R 038D R FFFF      DW      TWOP,DOLIT,-1          ;true flag
2863 0C5C 03CC R 0C68 R      DW      BRAN,FIND3
2864 0C60 062D R 04D0 R 03DE R      FIND2:      DW      TWOP,TEMP,AT,SAMEQ
2865      0BEE R
2866 0C68 03CC R 0C78 R      FIND3:      DW      BRAN,FIND4
2867 0C6C 0405 R 0436 R      FIND6:      DW      RFROM,DROP
2868 0C70 0442 R 0635 R 0442 R      DW      SWAP,TWOM,SWAP,EXIT
2869      0394 R
2870 0C78 03B7 R 0C84 R      FIND4:      DW      QBRAN,FIND5
2871 0C7C 0635 R 0635 R      DW      TWOM,TWOM
2872 0C80 03CC R 0C3C R      DW      BRAN,FIND1
2873 0C84 0405 R 0436 R 0442 R      FIND5:      DW      RFROM,DROP,SWAP,DROP
2874      0436 R
2875 0C8C 0635 R      DW      TWOM
2876 0C8E 043B R 0BE5 R 0442 R      DW      DUPP,NAMET,SWAP,EXIT
2877      0394 R
2878
2879      ; NAME?      ( a -- ca na | a F )
2880      ; Search all context vocabularies for a string.
2881
2882      $COLON 5,'NAME?',NAMEQ
2883 0C96      2      NAMEQ:
2884 3A7B      2      ORG      _NAME      ;
2885 3A7B 0C96 R 3A89 R      2      DW      _CODE,_LINK      ;
2886 3A7F 05 4E 41 4D 45 3F      2      DB      5,'NAME?'      ;
2887 0C96      2      ORG      _CODE      ;
2888 0C96 80      1      DB 80H      ;
2889 0C97 04FD R 043B R 07C7 R      DW      CNTXT,DUPP,DAT,XORR      ;?context=also
2890      0473 R
2891 0C9F 03B7 R 0CA5 R      DW      QBRAN,NAMQ1
2892 0CA3 0635 R      DW      TWOM      ;no, start with context
2893 0CA5 0413 R      NAMQ1:      DW      TOR
2894 0CA7 0405 R 062D R 043B R      NAMQ2:      DW      RFROM,TWOP,DUPP,TOR      ;next in search order
2895      0413 R
2896 0CAF 03DE R 053D R      DW      AT,QDUP
2897 0CB3 03B7 R 0CC5 R      DW      QBRAN,NAMQ3
2898 0CB7 0C21 R 053D R      DW      FIND,QDUP      ;search vocabulary
2899 0CBB 03B7 R 0CA7 R      DW      QBRAN,NAMQ2
2900 0CBF 0405 R 0436 R 0394 R      DW      RFROM,DROP,EXIT      ;found name
2901 0CC5 0405 R 0436 R      NAMQ3:      DW      RFROM,DROP      ;name not found
2902 0CC9 038D R 0000 0394 R      DW      DOLIT,0,EXIT      ;false flag
2903
2904      ;; Terminal response
2905
2906      ; ^H      ( bot eot cur -- bot eot cur )
2907      ; Backup the cursor by one character.
2908
2909      $COLON 2,'^H',BKSP
2910 0CCF      2      BKSP:
2911 3A73      2      ORG      _NAME      ;

```

```

2912 3A73 0CCF R 3A7F R      2      DW      _CODE,_LINK      ;
2913 3A77 02 5E 48          2      DB      2,'^H'          ;
2914 0CCF                    2      ORG      _CODE          ;
2915 0CCF 80                  1      DB      80H           ;
2916 0CD0 0413 R 044A R 0405 R  DW      TOR,OVER,RFROM,SWAP,OVER,XORR
2917          0442 R 044A R 0473 R
2918 0CDC 03B7 R 0CF8 R      DW      QBRAN,BACK1
2919 0CEO 038D R 0008 04C1 R  DW      DOLIT,BKSPP,TECHO,ATEXE,ONEM
2920          07F6 R 0626 R
2921 0CEA 0767 R 04C1 R 07F6 R  DW      BLANK,TECHO,ATEXE
2922 0CF0 038D R 0008 04C1 R  DW      DOLIT,BKSPP,TECHO,ATEXE
2923          07F6 R
2924 0CF8 0394 R              BACK1:   DW      EXIT
2925
2926          ; TAP      ( bot eot cur c -- bot eot cur )
2927          ;          Accept and echo the key stroke and bump the cursor.
2928
2929          $COLON 3,'TAP',TAP
2930 0CFA                    2      TAP:          ;
2931 3A6B                    2      ORG      _NAME          ;
2932 3A6B 0CFA R 3A77 R      2      DW      _CODE,_LINK      ;
2933 3A6F 03 54 41 50        2      DB      3,'TAP'        ;
2934 0CFA                    2      ORG      _CODE          ;
2935 0CFA 80                  1      DB      80H           ;
2936 0CFB 043B R 04C1 R 07F6 R  DW      DUPP,TECHO,ATEXE
2937 0D01 044A R 03E9 R 061F R  DW      OVER,CSTOR,ONEP,EXIT
2938          0394 R
2939
2940          ; kTAP     ( bot eot cur c -- bot eot cur )
2941          ;          Process a key stroke, CR or backspace.
2942
2943          $COLON 4,'kTAP',KTAP
2944 0D09                    2      KTAP:         ;
2945 3A61                    2      ORG      _NAME          ;
2946 3A61 0D09 R 3A6F R      2      DW      _CODE,_LINK      ;
2947 3A65 04 6B 54 41 50    2      DB      4,'kTAP'      ;
2948 0D09                    2      ORG      _CODE          ;
2949 0D09 80                  1      DB      80H           ;
2950 0D0A 043B R 038D R 000D  DW      DUPP,DOLIT,CRR,XORR
2951          0473 R
2952 0D12 03B7 R 0D2A R      DW      QBRAN,KTAP2
2953 0D16 038D R 0008 0473 R  DW      DOLIT,BKSPP,XORR
2954 0D1C 03B7 R 0D26 R      DW      QBRAN,KTAP1
2955 0D20 0767 R 0CFA R 0394 R  DW      BLANK,TAP,EXIT
2956 0D26 0CCF R 0394 R      KTAP1:   DW      BKSP,EXIT
2957 0D2A 0436 R 0442 R 0436 R  KTAP2:   DW      DROP,SWAP,DROP,DUPP,EXIT
2958          043B R 0394 R
2959
2960          ; accept   ( b u -- b u )
2961          ;          Accept characters to input buffer. Return with actual count.
2962
2963          $COLON 6,'accept',ACCEP
2964 0D34                    2      ACCEP:        ;
2965 3A55                    2      ORG      _NAME          ;
2966 3A55 0D34 R 3A65 R      2      DW      _CODE,_LINK      ;
2967 3A59 06 61 63 63 65 70 74 2  DB      6,'accept'      ;

```

```

2968 0D34          2      ORG      _CODE          ;
2969 0D34 80      1      DB      80H          ;
2970 0D35 044A R 0565 R 044A R      DW      OVER, PLUS, OVER
2971 0D3B 055B R 0473 R      ACCP1:  DW      DDUP, XORR
2972 0D3F 03B7 R 0D61 R      DW      QBRAN, ACCP4
2973 0D43 0A01 R 043B R      DW      KEY, DUPP
2974          ;      DW      BLANK, SUBB, DOLIT, 95, ULESS
2975 0D47 0767 R 038D R 007F      DW      BLANK, DOLIT, 127, WITHI
2976          0610 R
2977 0D4F 03B7 R 0D59 R      DW      QBRAN, ACCP2
2978 0D53 0CFA R      DW      TAP
2979 0D55 03CC R 0D5D R      DW      BRAN, ACCP3
2980 0D59 04BC R 07F6 R      ACCP2:  DW      TTAP, ATEXE
2981 0D5D 03CC R 0D3B R      ACCP3:  DW      BRAN, ACCP1
2982 0D61 0436 R 044A R 059B R      ACCP4:  DW      DROP, OVER, SUBB, EXIT
2983          0394 R
2984
2985          ;      EXPECT      ( b u -- )
2986          ;      Accept input stream and store count in SPAN.
2987
2988          $COLON 6, 'EXPECT', EXPEC
2989 0D69          2      EXPEC:          ;
2990 3A49          2      ORG      _NAME          ;
2991 3A49 0D69 R 3A59 R      2      DW      _CODE, _LINK      ;
2992 3A4D 06 45 58 50 45 43 54      2      DB      6, 'EXPECT'      ;
2993 0D69          2      ORG      _CODE          ;
2994 0D69 80      1      DB      80H          ;
2995 0D6A 04B7 R 07F6 R 04D5 R      DW      TEXPE, ATEXE, SPAN, STORE, DROP, EXIT
2996          03D3 R 0436 R 0394 R
2997
2998          ;      QUERY      ( -- )
2999          ;      Accept input stream to terminal input buffer.
3000
3001          $COLON 5, 'QUERY', QUERY
3002 0D76          2      QUERY:          ;
3003 3A3F          2      ORG      _NAME          ;
3004 3A3F 0D76 R 3A4D R      2      DW      _CODE, _LINK      ;
3005 3A43 05 51 55 45 52 59      2      DB      5, 'QUERY'      ;
3006 0D76          2      ORG      _CODE          ;
3007 0D76 80      1      DB      80H          ;
3008 0D77 07ED R 038D R 0050      DW      TIB, DOLIT, 80, TEXPE, ATEXE, NTIB, STORE
3009          04B7 R 07F6 R 04DF R
3010          03D3 R
3011 0D85          DW      DROP, DOLIT, 0, INN, STORE, EXIT
3012          0436 R 038D R 0000
3013          04DA R 03D3 R 0394 R
3014          ;; Error handling
3015
3016          ;      CATCH      ( ca -- 0 | err# )
3017          ;      Execute word at ca and set up an error frame for it.
3018
3019          $COLON 5, 'CATCH', CATCH
3020 0D91          2      CATCH:          ;
3021 3A35          2      ORG      _NAME          ;
3022 3A35 0D91 R 3A43 R      2      DW      _CODE, _LINK      ;
3023 3A39 05 43 41 54 43 48      2      DB      5, 'CATCH'      ;

```



```

3024 0D91          2      ORG      _CODE          ;
3025 0D91 80      1      DB 80H          ;
3026 0D92 041C R 0413 R 04F8 R      DW      SPAT,TOR,HANDL,AT,TOR ;save error frame
3027          03DE R 0413 R
3028 0D9C 03FB R 04F8 R 03D3 R      DW      RPAT,HANDL,STORE,EXECU ;execute
3029          039B R
3030 0DA4 0405 R 04F8 R 03D3 R      DW      RFROM,HANDL,STORE      ;restore error frame
3031 0DAA 0405 R 0436 R 038D R      DW      RFROM,DROP,DOLIT,0,EXIT ;no error
3032          0000 0394 R
3033
3034          ;      THROW      ( err# -- err# )
3035          ;      Reset system to current local error frame an update error flag.
3036
3037          $COLON 5,'THROW',THROW
3038 0DB4          2      THROW:          ;
3039 3A2B          2      ORG      _NAME          ;
3040 3A2B 0DB4 R 3A39 R      2      DW      _CODE,_LINK      ;
3041 3A2F 05 54 48 52 4F 57      2      DB      5,'THROW'      ;
3042 0DB4          2      ORG      _CODE          ;
3043 0DB4 80      1      DB 80H          ;
3044 0DB5 04F8 R 03DE R 0400 R      DW      HANDL,AT,RPSTO      ;restore return stack
3045 0DBB 0405 R 04F8 R 03D3 R      DW      RFROM,HANDL,STORE      ;restore handler frame
3046 0DC1 0405 R 0442 R 0413 R      DW      RFROM,SWAP,TOR,SPSTO ;restore data stack
3047          0429 R
3048 0DC9 0436 R 0405 R 0394 R      DW      DROP,RFROM,EXIT
3049
3050          ;      NULL$      ( -- a )
3051          ;      Return address of a null string with zero count.
3052
3053          $COLON 5,'NULL$',NULLS
3054 0DCF          2      NULLS:          ;
3055 3A21          2      ORG      _NAME          ;
3056 3A21 0DCF R 3A2F R      2      DW      _CODE,_LINK      ;
3057 3A25 05 4E 55 4C 4C 24      2      DB      5,'NULL$'      ;
3058 0DCF          2      ORG      _CODE          ;
3059 0DCF 80      1      DB 80H          ;
3060 0DD0 048C R      DW      DOVAR          ;emulate CREATE
3061 0DD2 0000      DW      0
3062 0DD4 63 6F 79 6F 74 65      DB      99,111,121,111,116,101
3063
3064          ;      ABORT      ( -- )
3065          ;      Reset data stack and jump to QUIT.
3066
3067          $COLON 5,'ABORT',ABORT
3068 0DDA          2      ABORT:          ;
3069 3A17          2      ORG      _NAME          ;
3070 3A17 0DDA R 3A25 R      2      DW      _CODE,_LINK      ;
3071 3A1B 05 41 42 4F 52 54      2      DB      5,'ABORT'      ;
3072 0DDA          2      ORG      _CODE          ;
3073 0DDA 80      1      DB 80H          ;
3074 0ddb 0DCF R 0DB4 R      DW      NULLS,THROW
3075
3076          ;      abort"      ( f -- )
3077          ;      Run time routine of ABORT" . Abort with a message.
3078
3079          $COLON COMPO+6,'abort"',ABORQ

```

```

3080 0DDF          2      ABORQ:          ;
3081 3A0B          2      ORG      _NAME          ;
3082 3A0B 0DDF R 3A1B R 2      DW      _CODE,_LINK      ;
3083 3A0F 46 61 62 6F 72 74 22 2      DB      COMPO+6,'abort" ' ;
3084 0DDF          2      ORG      _CODE          ;
3085 0DDF 80        1      DB 80H          ;
3086 0DE0 03B7 R 0DE8 R      DW      QBRAN,ABOR1      ;text flag
3087 0DE4 0A71 R 0DB4 R      DW      DOSTR,THROW      ;pass error string
3088 0DE8 0A71 R 0436 R 0394 R ABOR1:      DW      DOSTR,DROP,EXIT  ;drop error
3089
3090                ;; The text interpreter
3091
3092                ;  $INTERPRET ( a -- )
3093                ;      Interpret a word. If failed, try to convert it to an integer.
3094
3095                $COLON 10,$INTERPRET',INTER
3096 0DEE          2      INTER:          ;
3097 39FB          2      ORG      _NAME          ;
3098 39FB 0DEE R 3A0F R 2      DW      _CODE,_LINK      ;
3099 39FF 0A 24 49 4E 54 45 52 2      DB      10,$INTERPRET' ;
3100 0DEE          2      ORG      _CODE          ;
3101 0DEE 80        1      DB 80H          ;
3102 0DEF 0C96 R 053D R      DW      NAMEQ,QDUP        ;?defined
3103 0DF3 03B7 R 0E13 R      DW      QBRAN,INTE1      ;
3104 0DF7 03DE R 038D R 0040 2      DW      AT,DOLIT,COMPO,ANDD ;?compile only lexicon bits
3105 0461 R
3106                D$      ABORQ,' compile only'
3107 0DFF 0DDF R          1      DW      ABORQ          ;
3108 0E01 00 20 63 6F 6D 70 69 1      DB      0,' compile only' ;
3109 0E01          1      ORG      _LEN          ;
3110 0E01 0D        1      DB      _CODE-_LEN-1      ;
3111 0E0F          1      ORG      _CODE          ;
3112 0E0F 039B R 0394 R      DW      EXECU,EXIT        ;execute defined word
3113 0E13 04EE R 07F6 R      INTE1:      DW      TNUMB,ATEXE      ;convert a number
3114 0E17 03B7 R 0E1D R      DW      QBRAN,INTE2      ;
3115 0E1B 0394 R          DW      EXIT          ;
3116 0E1D 0DB4 R      INTE2:      DW      THROW          ;error
3117
3118                ;  [ ( -- )
3119                ;      Start the text interpreter.
3120
3121                $COLON IMEDD+1,[' ,LBRAC
3122 0E1F          2      LBRAC:          ;
3123 39F5          2      ORG      _NAME          ;
3124 39F5 0E1F R 39FF R 2      DW      _CODE,_LINK      ;
3125 39F9 81 5B        2      DB      IMEDD+1,[' ;
3126 0E1F          2      ORG      _CODE          ;
3127 0E1F 80        1      DB 80H          ;
3128 0E20 038D R 0DEE R 04E9 R      DW      DOLIT,INTER,TEVAL,STORE,EXIT
3129 03D3 R 0394 R
3130
3131                ;  .OK ( -- )
3132                ;      Display 'ok' only while interpreting.
3133
3134                $COLON 3,'.OK',DOTOK
3135 0E2A          2      DOTOK:          ;

```

```

3136 39ED                2      ORG      _NAME                ;
3137 39ED 0E2A R 39F9 R  2      DW      _CODE,_LINK            ;
3138 39F1 03 2E 4F 4B    2      DB      3,'.OK'                ;
3139 0E2A                2      ORG      _CODE                ;
3140 0E2A 80              1      DB      80H                  ;
3141 0E2B 038D R 0DEE R 04E9 R  DW      DOLIT,INTER,TEVAL,AT,EQUAL
3142                03DE R 05B9 R
3143 0E35 03B7 R 0E3F R    DW      QBRAN,DOTO1
3144                D$      DOTQP,' ok'
3145 0E39 0A89 R          1      DW      DOTQP                ;
3146 0E3B 00 20 6F 6B    1      DB      0,' ok'                ;
3147 0E3B                1      ORG      _LEN                ;
3148 0E3B 03              1      DB      _CODE-_LEN-1          ;
3149 0E3F                1      ORG      _CODE                ;
3150 0E3F 0A62 R 0394 R  DOTO1:  DW      CR,EXIT
3151
3152                ;   ?STACK      ( -- )
3153                ;   Abort if the data stack underflows.
3154
3155                $COLON 6,'?STACK',QSTAC
3156 0E43                2      QSTAC:                ;
3157 39E1                2      ORG      _NAME                ;
3158 39E1 0E43 R 39F1 R    2      DW      _CODE,_LINK            ;
3159 39E5 06 3F 53 54 41 43 4B 2      DB      6,'?STACK'            ;
3160 0E43                2      ORG      _CODE                ;
3161 0E43 80              1      DB      80H                  ;
3162 0E44 078B R 0453 R    DW      DEPTH,ZLESS            ;check only for underflow
3163                D$      ABORQ,' underflow'
3164 0E48 0DDF R          1      DW      ABORQ                ;
3165 0E4A 00 20 75 6E 64 65 72 1      DB      0,' underflow'            ;
3166 0E4A                1      ORG      _LEN                ;
3167 0E4A 0A              1      DB      _CODE-_LEN-1          ;
3168 0E55                1      ORG      _CODE                ;
3169 0E55 0394 R          DW      EXIT
3170
3171                ;   EVAL      ( -- )
3172                ;   Interpret the input stream.
3173
3174                $COLON 4,'EVAL',EVAL
3175 0E57                2      EVAL:                ;
3176 39D7                2      ORG      _NAME                ;
3177 39D7 0E57 R 39E5 R    2      DW      _CODE,_LINK            ;
3178 39DB 04 45 56 41 4C    2      DB      4,'EVAL'            ;
3179 0E57                2      ORG      _CODE                ;
3180 0E57 80              1      DB      80H                  ;
3181 0E58 0BC3 R 043B R 03F1 R  EVAL1:  DW      TOKEN,DUPP,CAT        ;?input stream empty
3182 0E5E 03B7 R 0E6C R    DW      QBRAN,EVAL2
3183 0E62 04E9 R 07F6 R 0E43 R  DW      TEVAL,ATEXE,QSTAC    ;evaluate input, check stack
3184 0E68 03CC R 0E58 R    DW      BRAN,EVAL1
3185 0E6C 0436 R 04C6 R 07F6 R  EVAL2:  DW      DROP,TPROM,ATEXE,EXIT ;prompt
3186 0394 R
3187
3188                ;; Shell
3189
3190                ;   PRESET      ( -- )
3191                ;   Reset data stack pointer and the terminal input buffer.

```

```

3192
3193                                $COLON 6, 'PRESET', PRESE
3194 0E74                2      PRESE:                                ;
3195 39CB                2      ORG      _NAME                        ;
3196 39CB 0E74 R 39DB R  2      DW      _CODE, _LINK                ;
3197 39CF 06 50 52 45 53 45 54 2      DB      6, 'PRESET'                ;
3198 0E74                2      ORG      _CODE                        ;
3199 0E74 80                1      DB      80H                        ;
3200 0E75 04A3 R 03DE R 0429 R  DW      SZERO, AT, SPSTO
3201 0E7B 038D R C200 04DF R  DW      DOLIT, TIBB, NTIB, TWOP, STORE, EXIT
3202      062D R 03D3 R 0394 R
3203
3204                ;      xio      ( a a a -- )
3205                ;      Reset the I/O vectors 'EXPECT, 'TAP, 'ECHO and 'PROMPT.
3206
3207                                $COLON COMPO+3, 'xio', XIO
3208 0E87                2      XIO:                                ;
3209 39C3                2      ORG      _NAME                        ;
3210 39C3 0E87 R 39CF R  2      DW      _CODE, _LINK                ;
3211 39C7 43 78 69 6F  2      DB      COMPO+3, 'xio'                ;
3212 0E87                2      ORG      _CODE                        ;
3213 0E87 80                1      DB      80H                        ;
3214 0E88 038D R 0D34 R 04B7 R  DW      DOLIT, ACCEP, TEXPE, DSTOR
3215      07BA R
3216 0E90 04C1 R 07BA R 0394 R  DW      TECHO, DSTOR, EXIT
3217
3218                ;      FILE      ( -- )
3219                ;      Select I/O vectors for file download.
3220
3221                                $COLON 4, 'FILE', FILE
3222 0E96                2      FILE:                                ;
3223 39B9                2      ORG      _NAME                        ;
3224 39B9 0E96 R 39C7 R  2      DW      _CODE, _LINK                ;
3225 39BD 04 46 49 4C 45  2      DB      4, 'FILE'                ;
3226 0E96                2      ORG      _CODE                        ;
3227 0E96 80                1      DB      80H                        ;
3228 0E97 038D R 0A26 R 038D R  DW      DOLIT, PACE, DOLIT, DROP
3229      0436 R
3230 0E9F 038D R 0D09 R 0E87 R  DW      DOLIT, KTAP, XIO, EXIT
3231      0394 R
3232
3233                ;      HAND      ( -- )
3234                ;      Select I/O vectors for terminal interface.
3235
3236                                $COLON 4, 'HAND', HAND
3237 0EA7                2      HAND:                                ;
3238 39AF                2      ORG      _NAME                        ;
3239 39AF 0EA7 R 39BD R  2      DW      _CODE, _LINK                ;
3240 39B3 04 48 41 4E 44  2      DB      4, 'HAND'                ;
3241 0EA7                2      ORG      _CODE                        ;
3242 0EA7 80                1      DB      80H                        ;
3243 0EA8 038D R 0E2A R 038D R  DW      DOLIT, DOTOK, DOLIT, EMIT
3244      0A0A R
3245 0EB0 038D R 0D09 R 0E87 R  DW      DOLIT, KTAP, XIO, EXIT
3246      0394 R
3247

```

```

3248          ; I/O      ( -- a )
3249          ;          Array to store default I/O vectors.
3250
3251          $COLON 3, 'I/O', ISLO
3252 0EB8      2          ISLO:          ;
3253 39A7      2          ORG          _NAME          ;
3254 39A7 0EB8 R 39B3 R      2          DW          _CODE, _LINK          ;
3255 39AB 03 49 2F 4F      2          DB          3, 'I/O'          ;
3256 0EB8      2          ORG          _CODE          ;
3257 0EB8 80      1          DB 80H          ;
3258 0EB9      2          DW          DOVAR          ;emulate CREATE
3259 0EBB 032E R 0347 R      2          DW          QRX, TXSTO          ;default I/O vectors
3260
3261          ; CONSOLE  ( -- )
3262          ;          Initiate terminal interface.
3263
3264          $COLON 7, 'CONSOLE', CONSO
3265 0EBF      2          CONSO:          ;
3266 399B      2          ORG          _NAME          ;
3267 399B 0EBF R 39AB R      2          DW          _CODE, _LINK          ;
3268 399F 07 43 4F 4E 53 4F 4C 2          DB          7, 'CONSOLE'          ;
3269 0EBF      2          ORG          _CODE          ;
3270 0EBF 80      1          DB 80H          ;
3271 0EC0 0EB8 R 07C7 R 04AD R      2          DW          ISLO, DAT, TQKEY, DSTOR          ;restore default I/O device
3272 07BA R
3273 0EC8 0EA7 R 0394 R      2          DW          HAND, EXIT          ;keyboard input
3274
3275          ; QUIT      ( -- )
3276          ;          Reset return stack pointer and start text interpreter.
3277
3278          $COLON 4, 'QUIT', QUIT
3279 0ECC      2          QUIT:          ;
3280 3991      2          ORG          _NAME          ;
3281 3991 0ECC R 399F R      2          DW          _CODE, _LINK          ;
3282 3995 04 51 55 49 54      2          DB          4, 'QUIT'          ;
3283 0ECC      2          ORG          _CODE          ;
3284 0ECC 80      1          DB 80H          ;
3285 0ECD 04A8 R 03DE R 0400 R      2          DW          RZERO, AT, RPSTO          ;reset return stack pointer
3286 0ED3 0E1F R          QUIT1:      DW          LBRAC          ;start interpretation
3287 0ED5 0D76 R          QUIT2:      DW          QUERY          ;get input
3288 0ED7 038D R 0E57 R 0D91 R      2          DW          DOLIT, EVAL, CATCH, QDUP          ;evaluate input
3289 053D R
3290 0EDF 03B7 R 0ED5 R          DW          QBRAN, QUIT2          ;continue till error
3291 0EE3 04C6 R 03DE R 0442 R      2          DW          TPROM, AT, SWAP          ;save input device
3292 0EE9 0EBF R 0DCF R 044A R      2          DW          CONSO, NULLS, OVER, XORR          ;?display error message
3293 0473 R
3294 0EF1 03B7 R 0F01 R          DW          QBRAN, QUIT3
3295 0EF5 0A2F R 07D4 R 0A4B R      2          DW          SPACE, COUNT, TYPEE          ;error message
3296          D$          DOTQP, ' ? '          ;error prompt
3297 0EFB 0A89 R          1          DW          DOTQP          ;
3298 0EFD 00 20 3F 20          1          DB          0, ' ? '          ;
3299 0EFD          1          ORG          _LEN          ;
3300 0EFD 03          1          DB          _CODE- _LEN-1          ;
3301 0F01          1          ORG          _CODE          ;
3302 0F01 038D R 0E2A R 0473 R      2          QUIT3:      DW          DOLIT, DOTOK, XORR          ;?file input
3303 0F07 03B7 R 0F11 R          DW          QBRAN, QUIT4

```

```

3304 0F0B 038D R 001B 0A0A R      DW      DOLIT,ERR,EMIT      ;file error, tell host
3305 0F11 0E74 R                  QUIT4:  DW      PRESE           ;some cleanup
3306 0F13 03CC R 0ED3 R          DW      BRAN,QUIT1
3307
3308      ;; The compiler
3309
3310      ;      '      ( -- ca )
3311      ;      Search context vocabularies for the next word in input stream.
3312
3313      $COLON 1,'"','TICK
3314 0F17      2      TICK:      ;
3315 398B      2      ORG      _NAME      ;
3316 398B 0F17 R 3995 R          2      DW      _CODE,_LINK      ;
3317 398F 01 27      2      DB      1,'"''      ;
3318 0F17      2      ORG      _CODE      ;
3319 0F17 80      1      DB      80H      ;
3320 0F18 0BC3 R 0C96 R          DW      TOKEN,NAMEQ      ;?defined
3321 0F1C 03B7 R 0F22 R          DW      QBRAN,TICK1
3322 0F20 0394 R          DW      EXIT      ;yes, push code address
3323 0F22 0DB4 R      TICK1:  DW      THROW      ;no, error
3324
3325      ;      ALLOT      ( n -- )
3326      ;      Allocate n bytes to the code dictionary.
3327
3328      $COLON 5,'ALLOT',ALLOT
3329 0F24      2      ALLOT:      ;
3330 3981      2      ORG      _NAME      ;
3331 3981 0F24 R 398F R          2      DW      _CODE,_LINK      ;
3332 3985 05 41 4C 4C 4F 54      2      DB      5,'ALLOT'      ;
3333 0F24      2      ORG      _CODE      ;
3334 0F24 80      1      DB      80H      ;
3335 0F25 0511 R 07AB R 0394 R    DW      CP,PSTOR,EXIT      ;adjust code pointer
3336
3337      ;      ,      ( w -- )
3338      ;      Compile an integer into the code dictionary.
3339
3340      $COLON 1,',','COMMA
3341 0F2B      2      COMMA:      ;
3342 397B      2      ORG      _NAME      ;
3343 397B 0F2B R 3985 R          2      DW      _CODE,_LINK      ;
3344 397F 01 2C      2      DB      1,',','      ;
3345 0F2B      2      ORG      _CODE      ;
3346 0F2B 80      1      DB      80H      ;
3347 0F2C 07DF R 043B R 062D R    DW      HERE,DUPP,TWOP      ;cell boundary
3348 0F32 0511 R 03D3 R 03D3 R    DW      CP,STORE,STORE,EXIT ;adjust code pointer, compile
3349 0394 R
3350
3351      ;      C,      ( b -- )
3352      ;      Compile a byte into the code dictionary
3353
3354      $COLON 2,'C','CCOMMA
3355 0F3A      2      CCOMMA:      ;
3356 3973      2      ORG      _NAME      ;
3357 3973 0F3A R 397F R          2      DW      _CODE,_LINK      ;
3358 3977 02 43 2C      2      DB      2,'C','      ;
3359 0F3A      2      ORG      _CODE      ;

```

```

3360 0F3A 80 1 DB 80H ;
3361 0F3B 07DF R 043B R 061F R DW HERE,DUPP,ONEP
3362 0F41 0511 R 03D3 R 03E9 R DW CP,STORE,CSTOR,EXIT
3363 0394 R
3364
3365 ; [COMPILE] ( -- ; <string> )
3366 ; Compile the next immediate word into code dictionary.
3367
3368 $COLON IMEDD+9,'[COMPILE]',BCOMP
3369 0F49 2 BCOMP: ;
3370 3965 2 ORG _NAME ;
3371 3965 0F49 R 3977 R 2 DW _CODE,_LINK ;
3372 3969 89 5B 43 4F 4D 50 49 2 DB IMEDD+9,'[COMPILE]' ;
3373 0F49 2 ORG _CODE ;
3374 0F49 80 1 DB 80H ;
3375 0F4A 0F17 R 0F2B R 0394 R DW TICK,COMMA,EXIT
3376
3377 ; COMPILE ( -- )
3378 ; Compile the next address in colon list to code dictionary.
3379
3380 $COLON COMPO+7,'COMPILE',COMPI
3381 0F50 2 COMPI: ;
3382 3959 2 ORG _NAME ;
3383 3959 0F50 R 3969 R 2 DW _CODE,_LINK ;
3384 395D 47 43 4F 4D 50 49 4C 2 DB COMPO+7,'COMPILE' ;
3385 0F50 2 ORG _CODE ;
3386 0F50 80 1 DB 80H ;
3387 0F51 0405 R 043B R 03DE R DW RFROM,DUPP,AT,COMMA ;compile address
3388 0F2B R
3389 0F59 062D R 0413 R 0394 R DW TWOP,TOR,EXIT ;adjust return address
3390
3391 ; LITERAL ( w -- )
3392 ; Compile tos to code dictionary as an integer literal.
3393
3394 $COLON IMEDD+7,'LITERAL',LITER
3395 0F5F 2 LITER: ;
3396 394D 2 ORG _NAME ;
3397 394D 0F5F R 395D R 2 DW _CODE,_LINK ;
3398 3951 87 4C 49 54 45 52 41 2 DB IMEDD+7,'LITERAL' ;
3399 0F5F 2 ORG _CODE ;
3400 0F5F 80 1 DB 80H ;
3401 0F60 0F50 R 038D R 0F2B R DW COMPI,DOLIT,COMMA,EXIT
3402 0394 R
3403
3404 ; $," ( -- )
3405 ; Compile a literal string up to next " .
3406
3407 $COLON 3,'$','',STRCQ
3408 0F68 2 STRCQ: ;
3409 3945 2 ORG _NAME ;
3410 3945 0F68 R 3951 R 2 DW _CODE,_LINK ;
3411 3949 03 24 2C 22 2 DB 3,'$','' ;
3412 0F68 2 ORG _CODE ;
3413 0F68 80 1 DB 80H ;
3414 0F69 038D R 0022 0BDC R DW DOLIT,'','',WORDD ;move string to code
dictionary
3415 0F6F 07D4 R 0565 R DW COUNT,PLUS ;calculate aligned end of string

```

```

3416 0F73 0511 R 03D3 R 0394 R          DW      CP,STORE,EXIT          ;adjust the code pointer
3417
3418          ; RECURSE      ( -- )
3419          ;              Make the current word available for compilation.
3420
3421          $COLON  IMEDD+7,'RECURSE',RECUR
3422 0F79          2          RECUR:          ;
3423 3939          2          ORG      _NAME          ;
3424 3939 0F79 R 3949 R          2          DW      _CODE,_LINK          ;
3425 393D 87 52 45 43 55 52 53  2          DB      IMEDD+7,'RECURSE'          ;
3426 0F79          2          ORG      _CODE          ;
3427 0F79 80          1          DB      80H          ;
3428 0F7A 051B R 03DE R 0BE5 R          DW      LAST,AT,NAMET,COMMA,EXIT
3429 0F2B R 0394 R
3430
3431          ;; Structures
3432
3433          ; FOR      ( -- a )
3434          ;              Start a FOR-NEXT loop structure in a colon definition.
3435
3436          $COLON  IMEDD+3,'FOR',FOR
3437 0F84          2          FOR:          ;
3438 3931          2          ORG      _NAME          ;
3439 3931 0F84 R 393D R          2          DW      _CODE,_LINK          ;
3440 3935 83 46 4F 52          2          DB      IMEDD+3,'FOR'          ;
3441 0F84          2          ORG      _CODE          ;
3442 0F84 80          1          DB      80H          ;
3443 0F85 0F50 R 0413 R 07DF R          DW      COMPI,TOR,HERE,EXIT
3444 0394 R
3445
3446          ; BEGIN      ( -- a )
3447          ;              Start an infinite or indefinite loop structure.
3448
3449          $COLON  IMEDD+5,'BEGIN',BEGIN
3450 0F8D          2          BEGIN:          ;
3451 3927          2          ORG      _NAME          ;
3452 3927 0F8D R 3935 R          2          DW      _CODE,_LINK          ;
3453 392B 85 42 45 47 49 4E          2          DB      IMEDD+5,'BEGIN'          ;
3454 0F8D          2          ORG      _CODE          ;
3455 0F8D 80          1          DB      80H          ;
3456 0F8E 07DF R 0394 R          DW      HERE,EXIT
3457
3458          ; NEXT      ( a -- )
3459          ;              Terminate a FOR-NEXT loop structure.
3460
3461          $COLON  IMEDD+4,'NEXT',NEXT
3462 0F92          2          NEXT:          ;
3463 391D          2          ORG      _NAME          ;
3464 391D 0F92 R 392B R          2          DW      _CODE,_LINK          ;
3465 3921 84 4E 45 58 54          2          DB      IMEDD+4,'NEXT'          ;
3466 0F92          2          ORG      _CODE          ;
3467 0F92 80          1          DB      80H          ;
3468 0F93 0F50 R 039D R 0F2B R          DW      COMPI,DONXT,COMMA,EXIT
3469 0394 R
3470
3471          ; UNTIL      ( a -- )

```



```

3472          ;          Terminate a BEGIN-UNTIL indefinite loop structure.
3473
3474          $COLON  IMEDD+5,'UNTIL',UNTIL
3475 0F9B          2      UNTIL:          ;
3476 3913          2      ORG          _NAME          ;
3477 3913 0F9B R 3921 R          2      DW          _CODE,_LINK          ;
3478 3917 85 55 4E 54 49 4C          2      DB          IMEDD+5,'UNTIL'          ;
3479 0F9B          2      ORG          _CODE          ;
3480 0F9B 80          1      DB 80H          ;
3481 0F9C 0F50 R 03B7 R 0F2B R          DW          COMPI,QBRAN,COMMA,EXIT
3482          0394 R
3483
3484          ;  AGAIN          ( a -- )
3485          ;          Terminate a BEGIN-AGAIN infinite loop structure.
3486
3487          $COLON  IMEDD+5,'AGAIN',AGAIN
3488 0FA4          2      AGAIN:          ;
3489 3909          2      ORG          _NAME          ;
3490 3909 0FA4 R 3917 R          2      DW          _CODE,_LINK          ;
3491 390D 85 41 47 41 49 4E          2      DB          IMEDD+5,'AGAIN'          ;
3492 0FA4          2      ORG          _CODE          ;
3493 0FA4 80          1      DB 80H          ;
3494 0FA5 0F50 R 03CC R 0F2B R          DW          COMPI,BRAN,COMMA,EXIT
3495          0394 R
3496
3497          ;  IF          ( -- A )
3498          ;          Begin a conditional branch structure.
3499
3500          $COLON  IMEDD+2,'IF',IFF
3501 0FAD          2      IFF:          ;
3502 3901          2      ORG          _NAME          ;
3503 3901 0FAD R 390D R          2      DW          _CODE,_LINK          ;
3504 3905 82 49 46          2      DB          IMEDD+2,'IF'          ;
3505 0FAD          2      ORG          _CODE          ;
3506 0FAD 80          1      DB 80H          ;
3507 0FAE 0F50 R 03B7 R 07DF R          DW          COMPI,QBRAN,HERE
3508 0FB4 038D R 0000 0F2B R          DW          DOLIT,0,COMMA,EXIT
3509          0394 R
3510
3511          ;  AHEAD          ( -- A )
3512          ;          Compile a forward branch instruction.
3513
3514          $COLON  IMEDD+5,'AHEAD',AHEAD
3515 0FBC          2      AHEAD:          ;
3516 38F7          2      ORG          _NAME          ;
3517 38F7 0FBC R 3905 R          2      DW          _CODE,_LINK          ;
3518 38FB 85 41 48 45 41 44          2      DB          IMEDD+5,'AHEAD'          ;
3519 0FBC          2      ORG          _CODE          ;
3520 0FBC 80          1      DB 80H          ;
3521 0FBD 0F50 R 03CC R 07DF R          DW          COMPI,BRAN,HERE,DOLIT,0,COMMA,EXIT
3522          038D R 0000 0F2B R
3523          0394 R
3524
3525          ;  REPEAT          ( A a -- )
3526          ;          Terminate a BEGIN-WHILE-REPEAT indefinite loop.
3527

```

```

3528                                $COLON  IMEDD+6,'REPEAT',REPEA
3529 0FCB                            2      REPEA:                                ;
3530 38EB                            2      ORG      _NAME                          ;
3531 38EB 0FCB R 38FB R              2      DW      _CODE,_LINK                       ;
3532 38EF 86 52 45 50 45 41 54      2      DB      IMEDD+6,'REPEAT'                    ;
3533 0FCB                            2      ORG      _CODE                          ;
3534 0FCB 80                          1      DB      80H                               ;
3535 0FCC 0FA4 R 07DF R 0442 R      DW      AGAIN,HERE,SWAP,STORE,EXIT
3536      03D3 R 0394 R
3537
3538                                ; THEN      ( A -- )
3539                                ;          Terminate a conditional branch structure.
3540
3541                                $COLON  IMEDD+4,'THEN',THENN
3542 0FD6                            2      THENN:                                ;
3543 38E1                            2      ORG      _NAME                          ;
3544 38E1 0FD6 R 38EF R              2      DW      _CODE,_LINK                       ;
3545 38E5 84 54 48 45 4E            2      DB      IMEDD+4,'THEN'                    ;
3546 0FD6                            2      ORG      _CODE                          ;
3547 0FD6 80                          1      DB      80H                               ;
3548 0FD7 07DF R 0442 R 03D3 R      DW      HERE,SWAP,STORE,EXIT
3549      0394 R
3550
3551                                ; AFT      ( a -- a A )
3552                                ;          Jump to THEN in a FOR-AFT-THEN-NEXT loop the first time through.
3553
3554                                $COLON  IMEDD+3,'AFT',AFT
3555 0FDF                            2      AFT:                                ;
3556 38D9                            2      ORG      _NAME                          ;
3557 38D9 0FDF R 38E5 R              2      DW      _CODE,_LINK                       ;
3558 38DD 83 41 46 54                2      DB      IMEDD+3,'AFT'                    ;
3559 0FDF                            2      ORG      _CODE                          ;
3560 0FDF 80                          1      DB      80H                               ;
3561 0FE0 0436 R 0FBC R 0F8D R      DW      DROP,AHEAD,BEGIN,SWAP,EXIT
3562      0442 R 0394 R
3563
3564                                ; ELSE      ( A -- A )
3565                                ;          Start the false clause in an IF-ELSE-THEN structure.
3566
3567                                $COLON  IMEDD+4,'ELSE',ELSEE
3568 0FEA                            2      ELSEE:                                ;
3569 38CF                            2      ORG      _NAME                          ;
3570 38CF 0FEA R 38DD R              2      DW      _CODE,_LINK                       ;
3571 38D3 84 45 4C 53 45            2      DB      IMEDD+4,'ELSE'                    ;
3572 0FEA                            2      ORG      _CODE                          ;
3573 0FEA 80                          1      DB      80H                               ;
3574 0FEB 0FBC R 0442 R 0FD6 R      DW      AHEAD,SWAP,THENN,EXIT
3575      0394 R
3576
3577                                ; WHILE     ( a -- A a )
3578                                ;          Conditional branch out of a BEGIN-WHILE-REPEAT loop.
3579
3580                                $COLON  IMEDD+5,'WHILE',WHILE
3581 0FF3                            2      WHILE:                                ;
3582 38C5                            2      ORG      _NAME                          ;
3583 38C5 0FF3 R 38D3 R              2      DW      _CODE,_LINK                       ;

```

```

3584 38C9 85 57 48 49 4C 45      2          DB      IMEDD+5,'WHILE'          ;
3585 0FF3                                2          ORG      _CODE                      ;
3586 0FF3 80                          1          DB      80H                          ;
3587 0FF4 0FAD R 0442 R 0394 R      2          DW      IFF,SWAP,EXIT
3588
3589 ; ABORT" ( -- ; <string> )
3590 ; Conditional abort with an error message.
3591
3592 $COLON IMEDD+6,'ABORT"',ABRTQ
3593 0FFA                                2          ABRTQ:                                ;
3594 38B9                                2          ORG      _NAME                          ;
3595 38B9 0FFA R 38C9 R                2          DW      _CODE,_LINK                      ;
3596 38BD 86 41 42 4F 52 54 22      2          DB      IMEDD+6,'ABORT"'          ;
3597 0FFA                                2          ORG      _CODE                      ;
3598 0FFA 80                          1          DB      80H                          ;
3599 0FFB 0F50 R 0DDF R 0F68 R      2          DW      COMPI,ABORQ,STRCQ,EXIT
3600 0394 R
3601
3602 ; $" ( -- ; <string> )
3603 ; Compile an inline string literal.
3604
3605 $COLON IMEDD+2,'$',STRQ
3606 1003                                2          STRQ:                                ;
3607 38B1                                2          ORG      _NAME                          ;
3608 38B1 1003 R 38BD R                2          DW      _CODE,_LINK                      ;
3609 38B5 82 24 22                    2          DB      IMEDD+2,'$"'          ;
3610 1003                                2          ORG      _CODE                      ;
3611 1003 80                          1          DB      80H                          ;
3612 1004 0F50 R 0A84 R 0F68 R      2          DW      COMPI,STRQP,STRCQ,EXIT
3613 0394 R
3614
3615 ; ." ( -- ; <string> )
3616 ; Compile an inline string literal to be typed out at run time.
3617
3618 $COLON IMEDD+2,'.',DOTQ
3619 100C                                2          DOTQ:                                ;
3620 38A9                                2          ORG      _NAME                          ;
3621 38A9 100C R 38B5 R                2          DW      _CODE,_LINK                      ;
3622 38AD 82 2E 22                    2          DB      IMEDD+2,'.'"          ;
3623 100C                                2          ORG      _CODE                      ;
3624 100C 80                          1          DB      80H                          ;
3625 100D 0F50 R 0A89 R 0F68 R      2          DW      COMPI,DOTQP,STRCQ,EXIT
3626 0394 R
3627
3628 ;; Name compiler
3629
3630 ; ?UNIQUE ( a -- a )
3631 ; Display a warning message if the word already exists.
3632
3633 $COLON 7,'?UNIQUE',UNIQU
3634 1015                                2          UNIQU:                                ;
3635 389D                                2          ORG      _NAME                          ;
3636 389D 1015 R 38AD R                2          DW      _CODE,_LINK                      ;
3637 38A1 07 3F 55 4E 49 51 55      2          DB      7,'?UNIQUE'          ;
3638 1015                                2          ORG      _CODE                      ;
3639 1015 80                          1          DB      80H                          ;

```

```

3640 1016 043B R 0C96 R          DW      DUPP,NAMEQ          ;?name exists
3641 101A 03B7 R 102E R          DW      QBRAN,UNIQL1       ;redefinitions are OK
3642                                D$      DOTQP,' reDef '    ;but warn the user
3643 101E 0A89 R                  1          DW      DOTQP              ;
3644 1020 00 20 72 65 44 65 66  1          DB      0,' reDef '      ;
3645 1020                                1          ORG      _LEN              ;
3646 1020 07                                1          DB      _CODE-_LEN-1    ;
3647 1028                                1          ORG      _CODE              ;
3648 1028 044A R 07D4 R 0A4B R    DW      OVER,COUNT,TYPEE   ;just in case its not planned
3649 102E 0436 R 0394 R          UNIQL1:  DW      DROP,EXIT
3650
3651                                ; $,n      ( na -- )
3652                                ;          Build a new dictionary name using the string at na.
3653
3654                                $COLON  3,'$,n',SNAME
3655 1032                                2          SNAME:              ;
3656 3895                                2          ORG      _NAME              ;
3657 3895 1032 R 38A1 R            2          DW      _CODE,_LINK          ;
3658 3899 03 24 2C 6E            2          DB      3,'$,n'          ;
3659 1032                                2          ORG      _CODE              ;
3660 1032 80                                1          DB      80H              ;
3661 1033 043B R 03F1 R          DW      DUPP,CAT           ;?null input
3662 1037 03B7 R 105F R          DW      QBRAN,PNAM1
3663 103B 1015 R                  DW      UNIQU              ;?redefinition
3664 103D 043B R 051B R 03D3 R    DW      DUPP,LAST,STORE   ;save na for vocabulary link
3665 1043 07DF R 0442 R          DW      HERE,SWAP         ;align code address
3666 1047 0635 R                  DW      TWOM               ;link address
3667 1049 0502 R 03DE R 03DE R    DW      CRRNT,AT,AT,OVER,STORE
3668 044A R 03D3 R
3669 1053 0635 R 043B R 0516 R    DW      TWOM,DUPP,NP,STORE ;adjust name pointer
3670 03D3 R
3671 105B 03D3 R 0394 R          DW      STORE,EXIT        ;save code pointer
3672 105F                                PNAM1:  D$      STRQP,' name'  ;null input
3673 105F 0A84 R                  1          DW      STRQP              ;
3674 1061 00 20 6E 61 6D 65      1          DB      0,' name'      ;
3675 1061                                1          ORG      _LEN              ;
3676 1061 05                                1          DB      _CODE-_LEN-1    ;
3677 1067                                1          ORG      _CODE              ;
3678 1067 0DB4 R                  DW      THROW
3679
3680                                ;; FORTH compiler
3681
3682                                ; $COMPILE ( a -- )
3683                                ;          Compile next word to code dictionary as a token or literal.
3684
3685                                $COLON  8,'$COMPILE',SCOMP
3686 1069                                2          SCOMP:              ;
3687 3887                                2          ORG      _NAME              ;
3688 3887 1069 R 3899 R            2          DW      _CODE,_LINK          ;
3689 388B 08 24 43 4F 4D 50 49    2          DB      8,'$COMPILE'      ;
3690 1069                                2          ORG      _CODE              ;
3691 1069 80                                1          DB      80H              ;
3692 106A 0C96 R 053D R            DW      NAMEQ,QDUP         ;?defined
3693 106E 03B7 R 1086 R            DW      QBRAN,SCOM2
3694 1072 03DE R 038D R 0080      DW      AT,DOLIT,IMEDD,ANDD ;?immediate
3695 0461 R

```

```

3696 107A 03B7 R 1082 R          DW      QBRAN,SCOM1
3697 107E 039B R 0394 R          DW      EXECU,EXIT          ;its immediate, execute
3698 1082 0F2B R 0394 R          SCOM1:   DW      COMMA,EXIT      ;its not immediate, compile
3699 1086 04EE R 07F6 R          SCOM2:   DW      TNUMB,ATEXE   ;try to convert to number
3700 108A 03B7 R 1092 R          DW      QBRAN,SCOM3
3701 108E 0F5F R 0394 R          DW      LITER,EXIT        ;compile number as integer
3702 1092 0DB4 R          SCOM3:   DW      THROW          ;error
3703
3704          ;   CCOMPILE   ( a -- )
3705          ;           Compile next byte to code dictionary as machine code.
3706
3707          $COLON 8,'CCOMPILE',CCOMP
3708 1094          2   CCOMP:   ;
3709 3879          2   ORG      _NAME      ;
3710 3879          2   DW      _CODE,_LINK ;
3711 387D 08 43 43 4F 4D 50 49 2   DB      8,'CCOMPILE' ;
3712 1094          2   ORG      _CODE      ;
3713 1094 80          1   DB      80H      ;
3714 1095 0C96 R 053D R          DW      NAMEQ,QDUP        ;?defined
3715 1099 03B7 R 10B1 R          DW      QBRAN,CCOM2
3716 109D 03DE R 038D R 0080    DW      AT,DOLIT,IMEDD,ANDD ;?immediate
3717 0461 R
3718 10A5 03B7 R 10AD R          DW      QBRAN,CCOM1
3719 10A9 039B R 0394 R          DW      EXECU,EXIT        ;its immediate, execute
3720 10AD 0436 R 0394 R          CCOM1:   DW      DROP,EXIT      ;its not immediate,drop
3721 10B1 04EE R 07F6 R          CCOM2:   DW      TNUMB,ATEXE   ;try to convert to number
3722 10B5 03B7 R 10BD R          DW      QBRAN,CCOM3
3723 10B9 0F3A R 0394 R          DW      CCOMMA,EXIT      ;compile as code byte
3724 10BD 0DB4 R          CCOM3:   DW      THROW          ;error
3725
3726          ;   OVERT     ( -- )
3727          ;           Link a new word into the current vocabulary.
3728
3729          $COLON 5,'OVERT',OVERT
3730 10BF          2   OVERT:   ;
3731 386F          2   ORG      _NAME      ;
3732 386F 10BF R 387D R          2   DW      _CODE,_LINK ;
3733 3873 05 4F 56 45 52 54 2   DB      5,'OVERT' ;
3734 10BF          2   ORG      _CODE      ;
3735 10BF 80          1   DB      80H      ;
3736 10C0 051B R 03DE R 0502 R    DW      LAST,AT,CRRNT,AT,STORE,EXIT
3737 03DE R 03D3 R 0394 R
3738
3739          ; ;           ( -- )
3740          ;           Terminate a colon definition.
3741
3742          $COLON IMEDD+COMPO+1,',' ,SEMIS
3743 10CC          2   SEMIS:   ;
3744 3869          2   ORG      _NAME      ;
3745 3869 10CC R 3873 R          2   DW      _CODE,_LINK ;
3746 386D C1 3B          2   DB      IMEDD+COMPO+1,',' ;
3747 10CC          2   ORG      _CODE      ;
3748 10CC 80          1   DB      80H      ;
3749 10CD 0F50 R 0394 R 0E1F R    DW      COMPI,EXIT,LBRAC,OVERT,EXIT
3750 10BF R 0394 R
3751

```

```

3752          ; ]          ( -- )
3753          ;          Start compiling the words in the input stream.
3754
3755          $COLON 1,']',RBRAC
3756 10D7      2          RBRAC:          ;
3757 3863      2          ORG          _NAME          ;
3758 3863      2          DW          _CODE,_LINK          ;
3759 3867      2          DB          1,']'          ;
3760 10D7      2          ORG          _CODE          ;
3761 10D7      1          DB          80H          ;
3762 10D8      038D R 1069 R 04E9 R
3763          03D3 R 0394 R          DW          DOLIT,SCOMP,TEVAL,STORE,EXIT
3764
3765          ; call,      ( ca -- )
3766          ;          Assemble a call instruction to doLST.
3767
3768          $COLON 5,'call,',CALLC
3769 10E2      2          CALLC:          ;
3770 3859      2          ORG          _NAME          ;
3771 3859      2          DW          _CODE,_LINK          ;
3772 385D      2          DB          5,'call,'          ;
3773 10E2      2          ORG          _CODE          ;
3774 10E2      1          DB          80H          ;
3775 10E3      038D R 0080 0F3A R
3776          0394 R          DW          DOLIT,CALLL,CCOMMA,EXIT ;Direct Threaded Code
3777
3778          ; :          ( -- ; <string> )
3779          ;          Start a new colon definition using next word as its name.
3780
3781          $COLON 1,':',COLON
3782 10EB      2          COLON:          ;
3783 3853      2          ORG          _NAME          ;
3784 3853      2          DW          _CODE,_LINK          ;
3785 3857      2          DB          1,':'          ;
3786 10EB      2          ORG          _CODE          ;
3787 10EB      1          DB          80H          ;
3788 10EC      0BC3 R 1032 R          DW          TOKEN,SNAME
3789 10F0      10E2 R 10D7 R 0394 R
3790          DW          CALLC,RBRAC,EXIT
3791          ; IMMEDIATE ( -- )
3792          ;          Make the last compiled word an immediate word.
3793
3794          $COLON 9,'IMMEDIATE',IMMED
3795 10F6      2          IMMED:          ;
3796 3845      2          ORG          _NAME          ;
3797 3845      2          DW          _CODE,_LINK          ;
3798 3849      2          DB          9,'IMMEDIATE'          ;
3799 10F6      2          ORG          _CODE          ;
3800 10F6      1          DB          80H          ;
3801 10F7      038D R 0080 051B R
3802          03DE R 03DE R 046A R          DW          DOLIT,IMEDD,LAST,AT,AT,ORR
3803 1103      051B R 03DE R 03D3 R
3804          0394 R          DW          LAST,AT,STORE,EXIT
3805
3806          ;; Defining words
3807

```

```

3808          ;  USER      ( u -- ; <string> )
3809          ;              Compile a new user variable.
3810
3811          $COLON  4, 'USER', USER
3812 110B      2          USER:      ;
3813 383B      2          ORG      _NAME      ;
3814 383B      2          DW      _CODE, _LINK      ;
3815 383F      2          DB      4, 'USER'      ;
3816 110B      2          ORG      _CODE      ;
3817 110B      1          DB      80H      ;
3818 110C      0BC3 R 1032 R 10BF R      DW      TOKEN, SNAME, OVERT, CALLC
3819          10E2 R
3820 1114      0F50 R 0496 R 0F2B R      DW      COMPI, DOUSE, COMMA, EXIT
3821          0394 R
3822
3823          ;  CREATE      ( -- ; <string> )
3824          ;              Compile a new array entry without allocating code space.
3825
3826          $COLON  6, 'CREATE', CREAT
3827 111C      2          CREAT:     ;
3828 382F      2          ORG      _NAME      ;
3829 382F      2          DW      _CODE, _LINK      ;
3830 3833      2          DB      6, 'CREATE'      ;
3831 111C      2          ORG      _CODE      ;
3832 111C      1          DB      80H      ;
3833 111D      0BC3 R 1032 R 10BF R      DW      TOKEN, SNAME, OVERT, CALLC
3834          10E2 R
3835 1125      0F50 R 048C R 0394 R      DW      COMPI, DOVAR, EXIT
3836
3837          ;  VARIABLE   ( -- ; <string> )
3838          ;              Compile a new variable initialized to 0.
3839
3840          $COLON  8, 'VARIABLE', VARIA
3841 112B      2          VARIA:    ;
3842 3821      2          ORG      _NAME      ;
3843 3821      2          DW      _CODE, _LINK      ;
3844 3825      2          DB      8, 'VARIABLE'      ;
3845 112B      2          ORG      _CODE      ;
3846 112B      1          DB      80H      ;
3847 112C      111C R 038D R 0000      DW      CREAT, DOLIT, 0, COMMA, EXIT
3848          0F2B R 0394 R
3849
3850          ;  CODE      ( -- )
3851          ;              Start a new code definition using next word as its name.
3852
3853          $COLON  4, 'CODE', CODE
3854 1136      2          CODE:    ;
3855 3817      2          ORG      _NAME      ;
3856 3817      2          DW      _CODE, _LINK      ;
3857 381B      2          DB      4, 'CODE'      ;
3858 1136      2          ORG      _CODE      ;
3859 1136      1          DB      80H      ;
3860 1137      0BC3 R 1032 R      DW      TOKEN, SNAME
3861 113B      038D R 1094 R 04E9 R      DW      DOLIT, CCOMP, TEVAL, STORE, EXIT
3862          03D3 R 0394 R
3863

```

```

3864          ; ENDCODE      ( -- )
3865          ;              Terminate a code definition
3866
3867          $COLON  IMEDD+COMPO+7,'ENDCODE',ENDCD
3868 1145      2          ENDCD:          ;
3869 380B      2          ORG      _NAME          ;
3870 380B      1145 R 381B R      2          DW      _CODE,_LINK          ;
3871 380F      C7 45 4E 44 43 4F 44 2          DB      IMEDD+COMPO+7,'ENDCODE'          ;
3872 1145      2          ORG      _CODE          ;
3873 1145      80              1          DB 80H          ;
3874 1146      038D R 0048 0F3A R      1          DW      DOLIT,48H,CCOMMA,DOLIT,84H,CCOMMA          ;$NEXT
3875          038D R 0084 0F3A R
3876 1152      038D R 0048 0F3A R      DW      DOLIT,48H,CCOMMA,DOLIT,28H,CCOMMA
3877          038D R 0028 0F3A R
3878 115E      0E1F R 10BF R 0394 R      DW      LBRAC,OVERT,EXIT
3879
3880          ;; Tools
3881
3882          ; _TYPE      ( b u -- )
3883          ;              Display a string. Filter non-printing characters.
3884
3885          $COLON  5,'_TYPE',UTYPE
3886 1164      2          UTYPE:          ;
3887 3801      2          ORG      _NAME          ;
3888 3801      1164 R 380F R      2          DW      _CODE,_LINK          ;
3889 3805      05 5F 54 59 50 45      2          DB      5,'_TYPE'          ;
3890 1164      2          ORG      _CODE          ;
3891 1164      80              1          DB 80H          ;
3892 1165      0413 R      DW      TOR          ;start count down loop
3893 1167      03CC R 1175 R      DW      BRAN,UTYP2          ;skip first pass
3894 116B      043B R 03F1 R 076E R  UTYP1:  DW      DUPP,CAT,TCHAR,EMIT          ;display only printable
3895          0A0A R
3896 1173      061F R      DW      ONEP          ;increment address
3897 1175      039D R 116B R      UTYP2:  DW      DONXT,UTYP1          ;loop till done
3898 1179      0436 R 0394 R      DW      DROP,EXIT
3899
3900          ; dm+      ( a u -- a )
3901          ;              Dump u bytes from , leaving a+u on the stack.
3902
3903          $COLON  3,'dm+',DMP
3904 117D      2          DMP:          ;
3905 37F9      2          ORG      _NAME          ;
3906 37F9      117D R 3805 R      2          DW      _CODE,_LINK          ;
3907 37FD      03 64 6D 2B      2          DB      3,'dm+'          ;
3908 117D      2          ORG      _CODE          ;
3909 117D      80              1          DB 80H          ;
3910 117E      044A R 038D R 0004      DW      OVER,DOLIT,4,UDOTR          ;display address
3911          0AA3 R
3912 1186      0A2F R 0413 R      DW      SPACE,TOR          ;start count down loop
3913 118A      03CC R 119A R      DW      BRAN,PDUM2          ;skip first pass
3914 118E      043B R 03F1 R 038D R  PDUM1:  DW      DUPP,CAT,DOLIT,3,UDOTR          ;display numeric data
3915          0003 0AA3 R
3916 1198      061F R      DW      ONEP          ;increment address
3917 119A      039D R 118E R      PDUM2:  DW      DONXT,PDUM1          ;loop till done
3918 119E      0394 R      DW      EXIT
3919

```



```

3920          ;   DUMP      ( a u -- )
3921          ;           Dump u bytes from a, in a formatted manner.
3922
3923          $COLON 4, 'DUMP', DUMP
3924 11A0          2       DUMP:          ;
3925 37EF          2       ORG      _NAME          ;
3926 37EF 11A0 R 37FD R      2       DW      _CODE, _LINK          ;
3927 37F3 04 44 55 4D 50      2       DB      4, 'DUMP'          ;
3928 11A0          2       ORG      _CODE          ;
3929 11A0 80          1       DB 80H          ;
3930 11A1 04CB R 03DE R 0413 R      DW      BASE, AT, TOR, HEX      ;save radix, set hex
3931 0920 R
3932 11A9 038D R 0010 06F1 R      DW      DOLIT, 16, SLASH      ;change count to lines
3933 11AF 0413 R      DW      TOR      ;start count down loop
3934 11B1 0A62 R 038D R 0010      DUMP1:   DW      CR, DOLIT, 16, DDUP, DMP      ;display numeric
3935 055B R 117D R
3936 11BB 054A R 054A R      DW      ROT, ROT
3937 11BF 0A2F R 0A2F R 1164 R      DW      SPACE, SPACE, UTYPE      ;display printable characters
3938 11C5 0A11 R 056F R      DW      NUFQ, INVER      ;user control
3939 11C9 03B7 R 11D5 R      DW      QBRAN, DUMP2
3940 11CD 039D R 11B1 R      DW      DONXT, DUMP1      ;loop till done
3941 11D1 03CC R 11D9 R      DW      BRAN, DUMP3
3942 11D5 0405 R 0436 R      DUMP2:   DW      RFROM, DROP      ;cleanup loop stack, early exit
3943 11D9 0436 R 0405 R 04CB R      DUMP3:   DW      DROP, RFROM, BASE, STORE      ;restore radix
3944 03D3 R
3945 11E1 0394 R      DW      EXIT
3946
3947          ;   .S      ( ... -- ... )
3948          ;           Display the contents of the data stack.
3949
3950          $COLON 2, '.S', DOTS
3951 11E3          2       DOTS:          ;
3952 37E7          2       ORG      _NAME          ;
3953 37E7 11E3 R 37F3 R      2       DW      _CODE, _LINK          ;
3954 37EB 02 2E 53          2       DB      2, '.S'          ;
3955 11E3          2       ORG      _CODE          ;
3956 11E3 80          1       DB 80H          ;
3957 11E4 0A62 R 078B R      DW      CR, DEPTH      ;stack depth
3958 11E8 0413 R      DW      TOR      ;start count down loop
3959 11EA 03CC R 11F4 R      DW      BRAN, DOTS2      ;skip first pass
3960 11EE 040C R 079E R 0AC5 R      DOTS1:   DW      RAT, PICK, DOT      ;index stack, display contents
3961 11F4 039D R 11EE R      DOTS2:   DW      DONXT, DOTS1      ;loop till done
3962      D$      DOTQP, ' <sp'
3963 11F8 0A89 R      1       DW      DOTQP          ;
3964 11FA 00 20 3C 73 70      1       DB      0, ' <sp'          ;
3965 11FA          1       ORG      _LEN          ;
3966 11FA 04          1       DB      _CODE- _LEN-1          ;
3967 11FF          1       ORG      _CODE          ;
3968 11FF 0394 R      DW      EXIT
3969
3970          ;   !CSP      ( -- )
3971          ;           Save stack pointer in CSP for error checking.
3972
3973          $COLON 4, '!CSP', STCSP
3974 1201          2       STCSP:          ;
3975 37DD          2       ORG      _NAME          ;

```

```

3976 37DD 1201 R 37EB R          2          DW      _CODE,_LINK          ;
3977 37E1 04 21 43 53 50        2          DB      4,'!CSP'              ;
3978 1201                                2          ORG      _CODE              ;
3979 1201 80                      1          DB      80H                  ;
3980 1202 041C R 04E4 R 03D3 R   2          DW      SPAT,CSP,STORE,EXIT   ;save pointer
3981 0394 R
3982
3983 ;      ?CSP      ( -- )
3984 ;      Abort if stack pointer differs from that saved in CSP.
3985
3986 $COLON 4,'?CSP',QCSP
3987 120A          2          QCSP:
3988 37D3          2          ORG      _NAME              ;
3989 37D3 120A R 37E1 R          2          DW      _CODE,_LINK          ;
3990 37D7 04 3F 43 53 50        2          DB      4,'?CSP'              ;
3991 120A          2          ORG      _CODE              ;
3992 120A 80                      1          DB      80H                  ;
3993 120B 041C R 04E4 R 03DE R   2          DW      SPAT,CSP,AT,XORR      ;compare pointers
3994 0473 R
3995 D$      ABORQ,'stacks'        ;abort if different
3996 1213 0DDF R          1          DW      ABORQ              ;
3997 1215 00 73 74 61 63 6B 73  1          DB      0,'stacks'          ;
3998 1215          1          ORG      _LEN              ;
3999 1215 06                      1          DB      _CODE-_LEN-1        ;
4000 121C          1          ORG      _CODE              ;
4001 121C 0394 R          1          DW      EXIT
4002
4003 ;      >NAME      ( ca -- na | F )
4004 ;      Convert code address to a name address.
4005
4006 $COLON 5,'>NAME',TNAME
4007 121E          2          TNAME:
4008 37C9          2          ORG      _NAME              ;
4009 37C9 121E R 37D7 R          2          DW      _CODE,_LINK          ;
4010 37CD 05 3E 4E 41 4D 45        2          DB      5,'>NAME'          ;
4011 121E          2          ORG      _CODE              ;
4012 121E 80                      1          DB      80H                  ;
4013 121F 0502 R          1          DW      CRRNT              ;vocabulary link
4014 1221 062D R 03DE R 053D R   2          DW      TWOP,AT,QDUP        ;check all vocabularies
4015 1227 03B7 R 1259 R          2          DW      QBRAN,TNAM4
4016 122B 055B R          2          DW      DDUP
4017 122D 03DE R 043B R          2          DW      AT,DUPP            ;?last word in a vocabulary
4018 1231 03B7 R 1245 R          2          DW      QBRAN,TNAM3
4019 1235 055B R 0BE5 R 0473 R   2          DW      DDUP,NAME,T,XORR    ;compare
4020 123B 03B7 R 1245 R          2          DW      QBRAN,TNAM3
4021 123F 0635 R          2          DW      TWOM              ;continue with next word
4022 1241 03CC R 122D R          2          DW      BRAN,TNAM2
4023 1245 0442 R 0436 R 053D R   2          DW      SWAP,DROP,QDUP
4024 124B 03B7 R 1221 R          2          DW      QBRAN,TNAM1
4025 124F 0442 R 0436 R 0442 R   2          DW      SWAP,DROP,SWAP,DROP,EXIT
4026 0436 R 0394 R
4027 1259 0436 R 038D R 0000     2          TNAM4:      DW      DROP,DOLIT,0,EXIT    ;false flag
4028 0394 R
4029
4030 ;      .ID      ( na -- )
4031 ;      Display the name at address.

```

```

4032
4033                                $COLON 3, '.ID', DOTID
4034 1261                2      DOTID:
4035 37C1                2      ORG      _NAME
4036 37C1 1261 R 37CD R 2      DW      _CODE, _LINK
4037 37C5 03 2E 49 44 2      DB      3, '.ID'
4038 1261                2      ORG      _CODE
4039 1261 80             1      DB      80H
4040 1262 053D R        DW      QDUP
4041 1264 03B7 R 1274 R DW      QBRAN, DOTI1
4042 1268 07D4 R 038D R 001F DW      COUNT, DOLIT, 01FH, ANDD
4043 0461 R
4044 1270 1164 R 0394 R DW      UTYPE, EXIT
4045 1274                DOTI1:
4046 1274 0A89 R        1      DW      DOTQP, '{noName}'
4047 1276 00 20 7B 6E 6F 4E 61 1      DB      0, '{noName}'
4048 1276                1      ORG      _LEN
4049 1276 09           1      DB      _CODE- _LEN-1
4050 1280                1      ORG      _CODE
4051 1280 0394 R        DW      EXIT
4052
4053                ;   WORDS      ( -- )
4054                ;   Display the names in the context vocabulary.
4055
4056                                $COLON 5, 'WORDS', WORDS
4057 1282                2      WORDS:
4058 37B7                2      ORG      _NAME
4059 37B7 1282 R 37C5 R 2      DW      _CODE, _LINK
4060 37BB 05 57 4F 52 44 53 2      DB      5, 'WORDS'
4061 1282                2      ORG      _CODE
4062 1282 80           1      DB      80H
4063 1283 0A62 R 04FD R 03DE R DW      CR, CNTXT, AT
4064 1289 03DE R 053D R        WORS1: DW      AT, QDUP
4065 128D 03B7 R 12A1 R        DW      QBRAN, WORS2
4066 1291 043B R 0A2F R 1261 R DW      DUPP, SPACE, DOTID
4067 1297 0635 R 0A11 R        DW      TWOM, NUFQ
4068 129B 03B7 R 1289 R        DW      QBRAN, WORS1
4069 129F 0436 R        DW      DROP
4070 12A1 0394 R        WORS2:   DW      EXIT
4071
4072                ;; Hardware reset
4073
4074                ;   VER      ( -- n )
4075                ;   Return the version number of this implementation.
4076
4077                                $COLON 3, 'VER', VERSN
4078 12A3                2      VERSN:
4079 37AF                2      ORG      _NAME
4080 37AF 12A3 R 37BB R 2      DW      _CODE, _LINK
4081 37B3 03 56 45 52 2      DB      3, 'VER'
4082 12A3                2      ORG      _CODE
4083 12A3 80           1      DB      80H
4084 12A4 038D R 0101 0394 R DW      DOLIT, VER*256+EXT, EXIT
4085
4086                ;   hi      ( -- )
4087                ;   Display the sign-on message of eForth.

```

```

4088
4089
4090 12AA          2      HI:          $COLON  2,'hi',HI
4091 37A7          2      ORG          _NAME          ;
4092 37A7 12AA R 37B3 R 2      DW          _CODE,_LINK      ;
4093 37AB 02 68 69 2      DB          2,'hi'          ;
4094 12AA          2      ORG          _CODE          ;
4095 12AA 80       1      DB 80H          ;
4096 12AB 0380 R 0A62 R 1      DW          STOIO,CR          ;initialize I/O
4097
4098 12AF 0A89 R   1      D$          DOTQP,'eForth v'
4099 12B1 00 65 46 6F 72 74 68 1      DW          DOTQP          ;
4100 12B1          1      ORG          _LEN          ;
4101 12B1 08       1      DB          _CODE-_LEN-1      ;
4102 12BA          1      ORG          _CODE          ;
4103 12BA 04CB R 03DE R 0920 R 1      DW          BASE,AT,HEX
4104 12C0 12A3 R 08BB R 08D5 R 1      DW          VERSN,BDIGS,DIG,DIG
4105 08D5 R
4106 12C8 038D R 002E 08C4 R 1      DW          DOLIT,'.',HOLD
4107 12CE 08E0 R 08FE R 0A4B R 1      DW          DIGS,EDIGS,TYPEE
4108 12D4 04CB R 03D3 R 0A62 R 1      DW          BASE,STORE,CR,EXIT
4109 0394 R
4110
4111          ; 'BOOT      ( -- a )
4112          ;          The application startup vector.
4113
4114          $COLON  5,'"BOOT",TBOOT
4115 12DC          2      TBOOT:          ;
4116 379D          2      ORG          _NAME          ;
4117 379D 12DC R 37AB R 2      DW          _CODE,_LINK      ;
4118 37A1 05 27 42 4F 4F 54 2      DB          5,'"BOOT"      ;
4119 12DC          2      ORG          _CODE          ;
4120 12DC 80       1      DB 80H          ;
4121 12DD 048C R   1      DW          DOVAR
4122 12DF 12AA R   1      DW          HI          ;application to boot
4123
4124          ; SEE      ( --word-- )
4125          ;          Decompiles word.
4126          $COLON  3,'SEE',SEE
4127 12E1          2      SEE:          ;
4128 3795          2      ORG          _NAME          ;
4129 3795 12E1 R 37A1 R 2      DW          _CODE,_LINK      ;
4130 3799 03 53 45 45 2      DB          3,'SEE'          ;
4131 12E1          2      ORG          _CODE          ;
4132 12E1 80       1      DB 80H          ;
4133 12E2 0F17 R   1      DW          TICK
4134 12E4 0A62 R 061F R 1      DW          CR,ONEP
4135 12E8 043B R 043B R 0A2F R 1      SEE1:  DW          DUPP,DUPP,SPACE,DOT,AT,DUPP
4136 0AC5 R 03DE R 043B R
4137 12F4 03B7 R 12FA R 1      DW          QBRAN,SEE2
4138 12F8 121E R   1      DW          TNAME
4139 12FA 053D R   1      SEE2:  DW          QDUP
4140 12FC 03B7 R 1306 R 1      DW          QBRAN,SEE3
4141 1300 1261 R   1      DW          DOTID
4142 1302 03CC R 130C R 1      DW          BRAN,SEE4
4143 1306 043B R 03DE R 0AB8 R 1      SEE3:  DW          DUPP,AT,UDOT

```

```

4144 130C 062D R 0A11 R          SEE4:    DW      TWOP,NUFQ
4145 1310 03B7 R 12E8 R          DW      QBRAN,SEE1
4146 1314 0436 R 0394 R          DW      DROP,EXIT
4147
4148
4149          ;  ADCINIT      ( -- )
4150          ;              Init routine for starting ADC Interrupts
4151          $CODE 7,'ADCINIT',ADCINIT
4152 1318          1      ADCINIT:          ;
4153 3789          1      ORG      _NAME          ;
4154 3789 1318 R 3799 R          1      DW      _CODE,_LINK          ;
4155 378D 07 41 44 43 49 4E 49 1 1      DB      7,'ADCINIT'          ;
4156 1318          1      ORG      _CODE          ;
4157 1318 68 FF          DB 68H,0FFH          ;V<FF
4158 131A 69 C6          DB 69H,0C6H          ;A<C6
4159 131C 63 F2          DB 63H,0F2H          ;(V/F2)<A
4160 131E 69 00          DB 69H,0          ;A<0
4161 1320 63 F3          DB 63H,0F3H          ;(V/F3)<A
4162 1322 4D C8          DB 4DH,0C8H          ;ANM <A
4163 1324 48 48          DB 48H,48H          ;SKIT FAD, reset INTFAD
4164 1326 00          DB 00          ;NOP
4165 1327 64 0E FE          DB 64H,0EH,0FEH          ;ENABLE INTAD
4166          $NEXT
4167 132A 48 84          1      DB 48H,84H          ;
4168 132C 48 28          1      DB 48H,28H          ;
4169
4170          ;  TMIDI      ( n -- )
4171          ;              Wait for last transmit, then send midi byte n.
4172          $CODE 5,'TMIDI',TMIDI
4173 132E          1      TMIDI:          ;
4174 377F          1      ORG      _NAME          ;
4175 377F 132E R 378D R          1      DW      _CODE,_LINK          ;
4176 3783 05 54 4D 49 44 49 1 1      DB      5,'TMIDI'          ;
4177 132E          1      ORG      _CODE          ;
4178 132E A1          DB 0A1H          ;POP BC
4179 132F 0B          DB 0BH          ;A<C
4180 1330 48 4A          DB 48H,4AH          ;SKIT FST, skip if interrupt
4181 1332 FD          DB 0FDH          ;JMP TO SKIT
4182 1333 4D D8          DB 4DH,0D8H          ;MOV TXB,A
4183          $NEXT
4184 1335 48 84          1      DB 48H,84H          ;
4185 1337 48 28          1      DB 48H,28H          ;
4186
4187          ;  DELAY      ( n -- )
4188          ;              Wait for n loops.
4189          $CODE 5,'DELAY',DELAY
4190 1339          1      DELAY:          ;
4191 3775          1      ORG      _NAME          ;
4192 3775 1339 R 3783 R          1      DW      _CODE,_LINK          ;
4193 3779 05 44 45 4C 41 59 1 1      DB      5,'DELAY'          ;
4194 1339          1      ORG      _CODE          ;
4195 1339 A1          DB 0A1H          ;POP BC
4196 133A 53          DB 53H          ;C<C-1, Skip if borrow
4197 133B FE          DB 0FEH          ;JMP
4198 133C 52          DB 52H          ;B<B-1, Skip if borrow
4199 133D FC          DB 0FCH          ;JMP

```

```

4200                                $NEXT
4201 133E 48 84                      1      DB 48H,84H                      ;
4202 1340 48 28                      1      DB 48H,28H                      ;
4203
4204                                ; LCD      ( n -- )
4205                                ;          Load control n to LCD display.
4206                                ;          $CODE 4,'LCD',LCD
4207 1342                      1      LCD:                                ;
4208 376B                      1      ORG      _NAME                                ;
4209 376B 1342 R 3779 R          1      DW      _CODE,_LINK                            ;
4210 376F 04 4C 43 44          1      DB      4,'LCD'                                ;
4211 1342                      1      ORG      _CODE                                ;
4212 1342 A1                    1      DB 0A1H                                ;POP BC
4213 1343 0B                    1      DB 0BH                                ;A<C
4214 1344 14 00 A0              1      DB 14H,0,0A0H                            ;BC<A000
4215 1347 39                    1      DB 39H                                ;(BC)<A
4216                                $NEXT
4217 1348 48 84                      1      DB 48H,84H                      ;
4218 134A 48 28                      1      DB 48H,28H                      ;
4219
4220                                ; LLI      ( --- )
4221                                ;          Sets RS=0 for LCD setup commands.
4222                                ;          $CODE 3,'LLI',LLI
4223 134C                      1      LLI:                                ;
4224 3763                      1      ORG      _NAME                                ;
4225 3763 134C R 376F R          1      DW      _CODE,_LINK                            ;
4226 3767 03 4C 4C 49          1      DB      3,'LLI'                                ;
4227 134C                      1      ORG      _CODE                                ;
4228 134C 64 0A EF              1      DB 64H,0AH,0EFH                            ;Pc<Pc AND EF
4229                                $NEXT
4230 134F 48 84                      1      DB 48H,84H                      ;
4231 1351 48 28                      1      DB 48H,28H                      ;
4232
4233                                ; LLC      ( --- )
4234                                ;          Sets RS=1 for LCD character loading
4235                                ;          $CODE 3,'LLC',LLC
4236 1353                      1      LLC:                                ;
4237 375B                      1      ORG      _NAME                                ;
4238 375B 1353 R 3767 R          1      DW      _CODE,_LINK                            ;
4239 375F 03 4C 4C 43          1      DB      3,'LLC'                                ;
4240 1353                      1      ORG      _CODE                                ;
4241 1353 64 1A 10              1      DB 64H,1AH,10H                            ;Pc<Pc OR 10
4242                                $NEXT
4243 1356 48 84                      1      DB 48H,84H                      ;
4244 1358 48 28                      1      DB 48H,28H                      ;
4245
4246                                ; LI       ( n --- )
4247                                ;          load LCD setup instruction n, exit ready for char loads
4248                                ;          $COLON 2,'LI',LI
4249 135A                      2      LI:                                ;
4250 3753                      2      ORG      _NAME                                ;
4251 3753 135A R 375F R          2      DW      _CODE,_LINK                            ;
4252 3757 02 4C 49              2      DB      2,'LI'                                ;
4253 135A                      2      ORG      _CODE                                ;
4254 135A 80                    1      DB 80H                                ;
4255 135B 134C R 1342 R 1353 R    1      DW      LLI,LCD,LLC,EXIT

```

```

4256      0394 R
4257
4258      ; LCDINIT      ( -- )
4259      ;              Initialize LCD display.
4260      ;              $COLON 7,'LCDINIT',LCDINIT
4261 1363      2      LCDINIT:
4262 3747      2      ORG      _NAME      ;
4263 3747 1363 R 3757 R      2      DW      _CODE,_LINK      ;
4264 374B 07 4C 43 44 49 4E 49      2      DB      7,'LCDINIT'      ;
4265 1363      2      ORG      _CODE      ;
4266 1363 80      1      DB      80H      ;
4267 1364 038D R 0D7A 1339 R      DW      DOLIT,0D7AH,DELAY
4268 136A 038D R 0038 135A R      DW      DOLIT,038H,LI
4269 1370 038D R 047E 1339 R      DW      DOLIT,047EH,DELAY
4270 1376 038D R 0038 135A R      DW      DOLIT,038H,LI
4271 137C 038D R 0017 1339 R      DW      DOLIT,017H,DELAY
4272 1382 038D R 0038 135A R      DW      DOLIT,038H,LI
4273 1388 038D R 0017 1339 R      DW      DOLIT,017H,DELAY
4274 138E 038D R 0038 135A R      DW      DOLIT,038H,LI
4275 1394 038D R 0017 1339 R      DW      DOLIT,017H,DELAY
4276 139A 038D R 0008 135A R      DW      DOLIT,08H,LI
4277 13A0 038D R 0017 1339 R      DW      DOLIT,017H,DELAY
4278 13A6 038D R 0001 135A R      DW      DOLIT,01H,LI
4279 13AC 038D R 01CC 1339 R      DW      DOLIT,01CCH,DELAY
4280 13B2 038D R 0002 135A R      DW      DOLIT,02H,LI
4281 13B8 038D R 01CC 1339 R      DW      DOLIT,01CCH,DELAY
4282 13BE 038D R 0006 135A R      DW      DOLIT,06H,LI
4283 13C4 038D R 0017 1339 R      DW      DOLIT,17H,DELAY
4284 13CA 038D R 000E 135A R      DW      DOLIT,0EH,LI
4285 13D0 038D R 0017 1339 R      DW      DOLIT,17H,DELAY
4286 13D6 0394 R      DW      EXIT
4287
4288      ; #DISP      ( n,p --- )
4289      ;              Display n as a 3-digit number at LCD position p.
4290      ;              $COLON 5,'#DISP',NDISP
4291 13D8      2      NDISP:
4292 373D      2      ORG      _NAME      ;
4293 373D 13D8 R 374B R      2      DW      _CODE,_LINK      ;
4294 3741 05 23 44 49 53 50      2      DB      5,'#DISP'      ;
4295 13D8      2      ORG      _CODE      ;
4296 13D8 80      1      DB      80H      ;
4297 13D9 043B R 135A R 0442 R      DW      DUPP,LI,SWAP
4298 13DF 08BB R 08D5 R 08D5 R      DW      BDIGS,DIG,DIG,DIG,EDIGS
4299      08D5 R 08FE R
4300 13E9 0436 R 043B R 03F1 R      DW      DROP,DUPP,CAT,LCD,ONEP
4301      1342 R 061F R
4302 13F3 043B R 03F1 R 1342 R      DW      DUPP,CAT,LCD,ONEP,CAT,LCD,LI,EXIT
4303      061F R 03F1 R 1342 R
4304      135A R 0394 R
4305
4306      ; DISP      ( a,p --- )
4307      ;              Display packed string at a to LCD position p.
4308      ;              $COLON 4,'DISP',DISP
4309 1403      2      DISP:
4310 3733      2      ORG      _NAME      ;
4311 3733 1403 R 3741 R      2      DW      _CODE,_LINK      ;

```

```

4312 3737 04 44 49 53 50      2      DB      4, 'DISP'      ;
4313 1403                      2      ORG      _CODE      ;
4314 1403 80                    1      DB      80H      ;
4315 1404 135A R 043B R 03F1 R  DW      LI, DUPP, CAT, TWOM, TOR
4316      0635 R 0413 R
4317 140E 061F R              DISPl:  DW      ONEP
4318 1410 043B R 03F1 R 1342 R  DW      DUPP, CAT, LCD
4319 1416 039D R 140E R      DW      DONXT, DISPl
4320 141A 0436 R 0394 R      DW      DROP, EXIT
4321
4322      ; CASE      ( n --- )
4323      ;      Execute one of a list of words pointed to by n.
4324      $COLON 4, 'CASE', CASE
4325 141E                      2      CASE:      ;
4326 3729                      2      ORG      _NAME      ;
4327 3729 141E R 3737 R      2      DW      _CODE, _LINK      ;
4328 372D 04 43 41 53 45      2      DB      4, 'CASE'      ;
4329 141E                      2      ORG      _CODE      ;
4330 141E 80                    1      DB      80H      ;
4331 141F 0405 R 0442 R 063D R  DW      RFROM, SWAP, TWOSL, PLUS
4332      0565 R
4333 1427 07F6 R 0394 R      DW      ATEXE, EXIT
4334
4335      ; INCR      ( n, nmax --- n+1 )
4336      ;      Increment n mod nmax.
4337      $COLON 4, 'INCR', INCR
4338 142B                      2      INCR:      ;
4339 371F                      2      ORG      _NAME      ;
4340 371F 142B R 372D R      2      DW      _CODE, _LINK      ;
4341 3723 04 49 4E 43 52      2      DB      4, 'INCR'      ;
4342 142B                      2      ORG      _CODE      ;
4343 142B 80                    1      DB      80H      ;
4344 142C 044A R 061F R 05E1 R  DW      OVER, ONEP, LESS
4345 1432 03B7 R 1440 R      DW      QBRAN, INCR1
4346 1436 0436 R 038D R 0000  DW      DROP, DOLIT, 0
4347 143C 03CC R 1442 R      DW      BRAN, INCR2
4348 1440 061F R              INCR1:  DW      ONEP
4349 1442 0394 R              INCR2:  DW      EXIT
4350
4351      ; DECR      ( n, nmax --- n-1 )
4352      ;      Decrement n mod nmax.
4353      $COLON 4, 'DECR', DECR
4354 1444                      2      DECR:      ;
4355 3715                      2      ORG      _NAME      ;
4356 3715 1444 R 3723 R      2      DW      _CODE, _LINK      ;
4357 3719 04 44 45 43 52      2      DB      4, 'DECR'      ;
4358 1444                      2      ORG      _CODE      ;
4359 1444 80                    1      DB      80H      ;
4360 1445 044A R 0626 R 0453 R  DW      OVER, ONEM, ZLESS
4361 144B 03B7 R 1457 R      DW      QBRAN, DECR1
4362 144F 0442 R 0436 R      DW      SWAP, DROP
4363 1453 03CC R 145B R      DW      BRAN, DECR2
4364 1457 0436 R              DECR1:  DW      DROP
4365 1459 0626 R              DW      ONEM
4366 145B 0394 R              DECR2:  DW      EXIT
4367

```



```

4368          ; SW@          ( --- n )
4369          ;          Read Roland switches as a byte.
4370          $CODE 3,'SW@',SWAT
4371 145D          1          SWAT:          ;
4372 370D          1          ORG          _NAME          ;
4373 370D 145D R 3719 R          1          DW          _CODE,_LINK          ;
4374 3711 03 53 57 40          1          DB          3,'SW@'          ;
4375 145D          1          ORG          _CODE          ;
4376 145D 4C C0          DB          4CH,0C0H          ;;A<PA
4377 145F 6A 00          DB          6AH,0          ;;B<0
4378 1461 1B          DB          1BH          ;;C<A
4379 1462 B1          DB          0B1H          ;;PUSH BC
4380          $NEXT
4381 1463 48 84          1          DB 48H,84H          ;
4382 1465 48 28          1          DB 48H,28H          ;
4383
4384          ; S@          ( --- n )
4385          ;          Return number of the lowest Roland switch on.
4386          $CODE 2,'S@',SAT
4387 1467          1          SAT:          ;
4388 3705          1          ORG          _NAME          ;
4389 3705 1467 R 3711 R          1          DW          _CODE,_LINK          ;
4390 3709 02 53 40          1          DB          2,'S@'          ;
4391 1467          1          ORG          _CODE          ;
4392 1467 4C C0          DB          4CH,0C0H          ;;A<PA
4393 1469 6B 00          DB          6BH,0          ;;C<0
4394 146B 74 11 FF          DB          74H,11H,0FFH          ;;A<A EXOR FF
4395 146E 74 49 FF          DB          74H,49H,0FFH          ;;A AND FF, SKIP IF NO ZERO
4396 1471 C4          DB          0C4H          ;; JMP OUT
4397 1472 43          DB          43H          ;; C<C+1, LOOP1
4398 1473 48 01          DB          48H,1          ;; A SHIFT RIGHT, SKIP IF CARRY
4399 1475 FC          DB          0FCH          ;; JMP LOOP1
4400 1476 6A 00          DB          6AH,0          ;;B<0, OUT
4401 1478 B1          DB          0B1H          ;;PUSH BC
4402          $NEXT
4403 1479 48 84          1          DB 48H,84H          ;
4404 147B 48 28          1          DB 48H,28H          ;
4405
4406          ; LED!          ( n --- )
4407          ;          Turn on/off Roland LED's.
4408          $CODE 4,'LED!',LEDB
4409 147D          1          LEDB:          ;
4410 36FB          1          ORG          _NAME          ;
4411 36FB 147D R 3709 R          1          DW          _CODE,_LINK          ;
4412 36FF 04 4C 45 44 21          1          DB          4,'LED!'          ;
4413 147D          1          ORG          _CODE          ;
4414 147D A1          DB          0A1H          ;;POP BC
4415 147E 0B          DB          0BH          ;;A<C
4416 147F 74 09 FC          DB          74H,9H,0FCH          ;;A<A AND FC
4417 1482 74 19 01          DB          74H,19H,1          ;;A<A OR 1
4418 1485 4D C1          DB          4DH,0C1H          ;;PB<A
4419          $NEXT
4420 1487 48 84          1          DB 48H,84H          ;
4421 1489 48 28          1          DB 48H,28H          ;
4422
4423          ; eUPDAT          ( --- )

```

```

4424          ;                Move data from Slider Ram to Edit Buffer.
4425          $CODE 6,'eUPDAT',EUPDAT
4426 148B      1      EUPDAT:
4427 36EF      1      ORG      _NAME
4428 36EF      148B R 36FF R      1      DW      _CODE,_LINK
4429 36F3      06 65 55 50 44 41 54 1      DB      6,'eUPDAT'
4430 148B      1      ORG      _CODE
4431 148B      68 FF      DB      68H,0FFH      ;;V<FF
4432 148D      6A C6      DB      6AH,0C6H      ;;B<C6
4433 148F      01 F0      DB      1,0F0H      ;;A<(V/F0)
4434 1491      1B        DB      1BH      ;;C<A
4435 1492      29        DB      29H      ;;A<(BC)
4436 1493      63 04      DB      63H,4      ;;(V/04)<A
4437 1495      69 38      DB      69H,38H      ;;A<70
4438 1497      60 43      DB      60H,43H      ;;C<C+A
4439 1499      29        DB      29H      ;;A<(BC)
4440 149A      63 06      DB      63H,6H      ;;(V/06)<A
4441 149C      69 38      DB      69H,38H      ;;A<38
4442 149E      60 43      DB      60H,43H      ;;C<C+A
4443 14A0      29        DB      29H      ;;A<(BC)
4444 14A1      63 07      DB      63H,7H      ;;(V/07)<A
4445          $NEXT
4446 14A3      48 84      1      DB      48H,84H
4447 14A5      48 28      1      DB      48H,28H
4448
4449          ; eLOAD      ( --- )
4450          ;                Load Edit Buffer data into Slider Memory.
4451          $CODE 5,'eLOAD',ELOAD
4452 14A7      1      ELOAD:
4453 36E5      1      ORG      _NAME
4454 36E5      14A7 R 36F3 R      1      DW      _CODE,_LINK
4455 36E9      05 65 4C 4F 41 44 1      DB      5,'eLOAD'
4456 14A7      1      ORG      _CODE
4457 14A7      68 FF      DB      68H,0FFH      ;;V<FF
4458 14A9      6A C6      DB      6AH,0C6H      ;;B<C6
4459 14AB      01 00      DB      1,0      ;;A<(V/00)
4460 14AD      1B        DB      1BH      ;;C<A
4461 14AE      01 04      DB      1,4      ;;A<(V/04)
4462 14B0      39        DB      39H      ;;(BC)<A
4463 14B1      69 38      DB      69H,38H      ;;A<38
4464 14B3      60 43      DB      60H,43H      ;;C<C+A
4465 14B5      49 00      DB      49H,0      ;;(BC)<0
4466 14B7      69 70      DB      69H,70H      ;;A<38
4467 14B9      60 43      DB      60H,43H      ;;C<C+A
4468 14BB      01 06      DB      1,6      ;;A<(V/06)
4469 14BD      39        DB      39H      ;;(BC)<A
4470 14BE      69 38      DB      69H,38H      ;;A<38
4471 14C0      60 43      DB      60H,43H      ;;C<C+A
4472 14C2      01 07      DB      1,7      ;;A<(V/07)
4473 14C4      39        DB      39H      ;;(BC)<A
4474          $NEXT
4475 14C5      48 84      1      DB      48H,84H
4476 14C7      48 28      1      DB      48H,28H
4477
4478          ; esUPDAT      ( --- )
4479          ;                Update only the Slider data of the Edit Buffer.

```

```

4480                                     $CODE 7,'esUPDAT',ESUPDAT
4481 14C9                               1     ESUPDAT:
4482 36D9                               1     ORG      _NAME
4483 36D9 14C9 R 36E9 R                 1     DW      _CODE,_LINK
4484 36DD 07 65 73 55 50 44 41         1     DB      7,'esUPDAT'
4485 14C9                               1     ORG      _CODE
4486 14C9 68 FF                         DB      68H,0FFH      ;;V<FF
4487 14CB 6A C6                         DB      6AH,0C6H     ;;B<C6
4488 14CD 01 F0                         DB      1,0F0H      ;;A<(V/F0)
4489 14CF 1B                             DB      1BH         ;;C<A
4490 14D0 29                             DB      29H         ;;A<(BC)
4491 14D1 63 04                         DB      63H,4       ;;(V/04)<A
4492                                     $NEXT
4493 14D3 48 84                           1     DB 48H,84H
4494 14D5 48 28                           1     DB 48H,28H
4495
4496                                     ; eSLD# ( --- FF00 )
4497                                     ; Edit Buffer Slider number.
4498                                     $COLON 5,'eSLD#',ESLDN
4499 14D7                               2     ESLDN:
4500 36CF                               2     ORG      _NAME
4501 36CF 14D7 R 36DD R                 2     DW      _CODE,_LINK
4502 36D3 05 65 53 4C 44 23             2     DB      5,'eSLD#'
4503 14D7                               2     ORG      _CODE
4504 14D7 80                             1     DB 80H
4505 14D8 038D R FF00 0394 R             DW      DOLIT,0FF00H,EXIT
4506
4507                                     ; eFLD ( --- FF01 )
4508                                     ; Edit Buffer LCD Field.
4509                                     $COLON 4,'eFLD',EFLD
4510 14DE                               2     EFLD:
4511 36C5                               2     ORG      _NAME
4512 36C5 14DE R 36D3 R                 2     DW      _CODE,_LINK
4513 36C9 04 65 46 4C 44                 2     DB      4,'eFLD'
4514 14DE                               2     ORG      _CODE
4515 14DE 80                             1     DB 80H
4516 14DF 038D R FF01 0394 R             DW      DOLIT,0FF01H,EXIT
4517
4518                                     ; eBYTE1 ( --- FF07 )
4519                                     ; Edit Buffer Midi Status/Chnl byte.
4520                                     $COLON 6,'eBYTE1',EByte1
4521 14E5                               2     EByte1:
4522 36B9                               2     ORG      _NAME
4523 36B9 14E5 R 36C9 R                 2     DW      _CODE,_LINK
4524 36BD 06 65 42 59 54 45 31           2     DB      6,'eByte1'
4525 14E5                               2     ORG      _CODE
4526 14E5 80                             1     DB 80H
4527 14E6 038D R FF07 0394 R             DW      DOLIT,0FF07H,EXIT
4528
4529                                     ; eBYTE2 ( --- FF06 )
4530                                     ; Edit Buffer Midi Key#, Controller#, or Program# byte.
4531                                     $COLON 6,'eByte2',EByte2
4532 14EC                               2     EByte2:
4533 36AD                               2     ORG      _NAME
4534 36AD 14EC R 36BD R                 2     DW      _CODE,_LINK
4535 36B1 06 65 42 59 54 45 32           2     DB      6,'eByte2'

```

```

4536 14EC          2      ORG      _CODE          ;
4537 14EC 80      1      DB 80H          ;
4538 14ED 038D R FF06 0394 R      DW      DOLIT,0FF06H,EXIT
4539
4540          ; eBYTE3      ( --- FF04 )
4541          ;                      Edit Buffer Slider value.
4542          ;                      $COLON 6,'eBYTE3',EBYTE3
4543 14F3          2      EBYTE3:          ;
4544 36A1          2      ORG      _NAME          ;
4545 36A1 14F3 R 36B1 R      2      DW      _CODE,_LINK      ;
4546 36A5 06 65 42 59 54 45 33  2      DB      6,'eBYTE3'      ;
4547 14F3          2      ORG      _CODE          ;
4548 14F3 80      1      DB 80H          ;
4549 14F4 038D R FF04 0394 R      DW      DOLIT,0FF04H,EXIT
4550
4551          ; eFLAG      ( --- FF05 )
4552          ;                      Edit Buffer Midi Flag byte.
4553          ;                      $COLON 5,'eFLAG',EFLAG
4554 14FA          2      EFLAG:          ;
4555 3697          2      ORG      _NAME          ;
4556 3697 14FA R 36A5 R      2      DW      _CODE,_LINK      ;
4557 369B 05 65 46 4C 41 47      2      DB      5,'eFLAG'      ;
4558 14FA          2      ORG      _CODE          ;
4559 14FA 80      1      DB 80H          ;
4560 14FB 038D R FF05 0394 R      DW      DOLIT,0FF05H,EXIT
4561
4562          ; FLD0      ( --- 80 )
4563          ;                      LCD Field start.
4564          ;                      $COLON 4,'FLD0',FLD0
4565 1501          2      FLD0:          ;
4566 368D          2      ORG      _NAME          ;
4567 368D 1501 R 369B R      2      DW      _CODE,_LINK      ;
4568 3691 04 46 4C 44 30      2      DB      4,'FLD0'      ;
4569 1501          2      ORG      _CODE          ;
4570 1501 80      1      DB 80H          ;
4571 1502 038D R 0080 0394 R      DW      DOLIT,080H,EXIT
4572
4573          ; FLD1      ( --- 86 )
4574          ;                      LCD Field start.
4575          ;                      $COLON 4,'FLD1',FLD01
4576 1508          2      FLD01:          ;
4577 3683          2      ORG      _NAME          ;
4578 3683 1508 R 3691 R      2      DW      _CODE,_LINK      ;
4579 3687 04 46 4C 44 31      2      DB      4,'FLD1'      ;
4580 1508          2      ORG      _CODE          ;
4581 1508 80      1      DB 80H          ;
4582 1509 038D R 0086 0394 R      DW      DOLIT,086H,EXIT
4583
4584          ; FLD2      ( --- 8A )
4585          ;                      LCD Field start.
4586          ;                      $COLON 4,'FLD2',FLD2
4587 150F          2      FLD2:          ;
4588 3679          2      ORG      _NAME          ;
4589 3679 150F R 3687 R      2      DW      _CODE,_LINK      ;
4590 367D 04 46 4C 44 32      2      DB      4,'FLD2'      ;
4591 150F          2      ORG      _CODE          ;

```

```

4592 150F 80 1 DB 80H ;
4593 1510 038D R 008A 0394 R DW DOLIT,08AH,EXIT ;
4594
4595 ; FLD3 ( --- 8D ) ;
4596 ; LCD Field start. ;
4597 $COLON 4,'FLD3',FLD3 ;
4598 1516 2 FLD3: ;
4599 366F 2 ORG _NAME ;
4600 366F 1516 R 367D R 2 DW _CODE,_LINK ;
4601 3673 04 46 4C 44 33 2 DB 4,'FLD3' ;
4602 1516 2 ORG _CODE ;
4603 1516 80 1 DB 80H ;
4604 1517 038D R 008D 0394 R DW DOLIT,08DH,EXIT ;
4605
4606 ; FLD4 ( --- C0 ) ;
4607 ; LCD Field start. ;
4608 $COLON 4,'FLD4',FLD4 ;
4609 151D 2 FLD4: ;
4610 3665 2 ORG _NAME ;
4611 3665 151D R 3673 R 2 DW _CODE,_LINK ;
4612 3669 04 46 4C 44 34 2 DB 4,'FLD4' ;
4613 151D 2 ORG _CODE ;
4614 151D 80 1 DB 80H ;
4615 151E 038D R 00C0 0394 R DW DOLIT,0C0H,EXIT ;
4616
4617 ; FLD5 ( --- C9 ) ;
4618 ; LCD Field start. ;
4619 $COLON 4,'FLD5',FLD5 ;
4620 1524 2 FLD5: ;
4621 365B 2 ORG _NAME ;
4622 365B 1524 R 3669 R 2 DW _CODE,_LINK ;
4623 365F 04 46 4C 44 35 2 DB 4,'FLD5' ;
4624 1524 2 ORG _CODE ;
4625 1524 80 1 DB 80H ;
4626 1525 038D R 00C9 0394 R DW DOLIT,0C9H,EXIT ;
4627
4628 ; FLD6 ( --- CD ) ;
4629 ; LCD Field start. ;
4630 $COLON 4,'FLD6',FLD6 ;
4631 152B 2 FLD6: ;
4632 3651 2 ORG _NAME ;
4633 3651 152B R 365F R 2 DW _CODE,_LINK ;
4634 3655 04 46 4C 44 36 2 DB 4,'FLD6' ;
4635 152B 2 ORG _CODE ;
4636 152B 80 1 DB 80H ;
4637 152C 038D R 00CD 0394 R DW DOLIT,0CDH,EXIT ;
4638
4639 ; L0 ( --- a ) ;
4640 ; Packed string. 'a' is addr of count byte. ;
4641 $COLON 2,'L0',L0 ;
4642 1532 2 L0: ;
4643 3649 2 ORG _NAME ;
4644 3649 1532 R 3655 R 2 DW _CODE,_LINK ;
4645 364D 02 4C 30 2 DB 2,'L0' ;
4646 1532 2 ORG _CODE ;
4647 1532 80 1 DB 80H ;

```

```

4648          SD$ 'Slider'
4649 1533 038D R          1          DW DOLIT
4650 1535 1539 R 0394 R  1          DW      _LEN,EXIT          ;
4651 1539 00 53 6C 69 64 65 72 1          DB      0,'Slider'          ;
4652 1539          1          ORG      _LEN          ;
4653 1539 06          1          DB      _CODE-_LEN-1          ;
4654 1540          1          ORG      _CODE          ;
4655
4656          ; L1          ( --- a )
4657          ;          Packed string. 'a' is addr of count byte.
4658          $COLON 2,'L1',L1
4659 1540          2          L1:          ;
4660 3641          2          ORG      _NAME          ;
4661 3641 1540 R 364D R  2          DW      _CODE,_LINK          ;
4662 3645 02 4C 31          2          DB      2,'L1'          ;
4663 1540          2          ORG      _CODE          ;
4664 1540 80          1          DB      80H          ;
4665          SD$ 'Setup#'
4666 1541 038D R          1          DW DOLIT
4667 1543 1547 R 0394 R  1          DW      _LEN,EXIT          ;
4668 1547 00 53 65 74 75 70 23 1          DB      0,'Setup#'          ;
4669 1547          1          ORG      _LEN          ;
4670 1547 06          1          DB      _CODE-_LEN-1          ;
4671 154E          1          ORG      _CODE          ;
4672
4673          ; L2          ( --- a )
4674          ;          Packed string. 'a' is addr of count byte.
4675          $COLON 2,'L2',L2
4676 154E          2          L2:          ;
4677 3639          2          ORG      _NAME          ;
4678 3639 154E R 3645 R  2          DW      _CODE,_LINK          ;
4679 363D 02 4C 32          2          DB      2,'L2'          ;
4680 154E          2          ORG      _CODE          ;
4681 154E 80          1          DB      80H          ;
4682          SD$ '* MIDI Running *'
4683 154F 038D R          1          DW DOLIT
4684 1551 1555 R 0394 R  1          DW      _LEN,EXIT          ;
4685 1555 00 2A 20 4D 49 44 49 1          DB      0,'* MIDI Running *'          ;
4686 1555          1          ORG      _LEN          ;
4687 1555 10          1          DB      _CODE-_LEN-1          ;
4688 1566          1          ORG      _CODE          ;
4689
4690          ; L20          ( --- a )
4691          ;          Packed string. 'a' is addr of count byte.
4692          $COLON 3,'L20',L20
4693 1566          2          L20:          ;
4694 3631          2          ORG      _NAME          ;
4695 3631 1566 R 363D R  2          DW      _CODE,_LINK          ;
4696 3635 03 4C 32 30          2          DB      3,'L20'          ;
4697 1566          2          ORG      _CODE          ;
4698 1566 80          1          DB      80H          ;
4699          SD$ 'Ch '
4700 1567 038D R          1          DW DOLIT
4701 1569 156D R 0394 R  1          DW      _LEN,EXIT          ;
4702 156D 00 43 68 20          1          DB      0,'Ch '          ;
4703 156D          1          ORG      _LEN          ;

```

```

4704 156D 03          1          DB      _CODE-_LEN-1          ;
4705 1571          1          ORG      _CODE          ;
4706
4707          ; L21      ( --- a )
4708          ;          Packed string. 'a' is addr of count byte.
4709          $COLON 3,'L21',L21
4710 1571          2          L21:
4711 3629          2          ORG      _NAME          ;
4712 3629 1571 R 3635 R 2          DW      _CODE,_LINK          ;
4713 362D 03 4C 32 31 2          DB      3,'L21'          ;
4714 1571          2          ORG      _CODE          ;
4715 1571 80          1          DB      80H          ;
4716          SD$ 'Off'
4717 1572 038D R      1          DW      DOLIT
4718 1574 1578 R 0394 R 1          DW      _LEN,EXIT          ;
4719 1578 00 4F 66 66 1          DB      0,'Off'          ;
4720 1578          1          ORG      _LEN          ;
4721 1578 03          1          DB      _CODE-_LEN-1          ;
4722 157C          1          ORG      _CODE          ;
4723
4724          ; L40      ( --- a )
4725          ;          Packed string. 'a' is addr of count byte.
4726          $COLON 3,'L40',L40
4727 157C          2          L40:
4728 3621          2          ORG      _NAME          ;
4729 3621 157C R 362D R 2          DW      _CODE,_LINK          ;
4730 3625 03 4C 34 30 2          DB      3,'L40'          ;
4731 157C          2          ORG      _CODE          ;
4732 157C 80          1          DB      80H          ;
4733          SD$ 'Key# '
4734 157D 038D R      1          DW      DOLIT
4735 157F 1583 R 0394 R 1          DW      _LEN,EXIT          ;
4736 1583 00 4B 65 79 23 20 20 1          DB      0,'Key# '          ;
4737 1583          1          ORG      _LEN          ;
4738 1583 08          1          DB      _CODE-_LEN-1          ;
4739 158C          1          ORG      _CODE          ;
4740
4741          ; L41      ( --- a )
4742          ;          Packed string. 'a' is addr of count byte.
4743          $COLON 3,'L41',L41
4744 158C          2          L41:
4745 3619          2          ORG      _NAME          ;
4746 3619 158C R 3625 R 2          DW      _CODE,_LINK          ;
4747 361D 03 4C 34 31 2          DB      3,'L41'          ;
4748 158C          2          ORG      _CODE          ;
4749 158C 80          1          DB      80H          ;
4750          SD$ 'Key# A-T'
4751 158D 038D R      1          DW      DOLIT
4752 158F 1593 R 0394 R 1          DW      _LEN,EXIT          ;
4753 1593 00 4B 65 79 23 20 41 1          DB      0,'Key# A-T'          ;
4754 1593          1          ORG      _LEN          ;
4755 1593 08          1          DB      _CODE-_LEN-1          ;
4756 159C          1          ORG      _CODE          ;
4757
4758          ; L42      ( --- a )
4759          ;          Packed string. 'a' is addr of count byte.

```

```

4760                                $COLON 3,'L42',L42
4761 159C                            2      L42:                                ;
4762 3611                            2      ORG                                _NAME                                ;
4763 3611 159C R 361D R                2      DW                                _CODE,_LINK                                ;
4764 3615 03 4C 34 32                 2      DB                                3,'L42'                                ;
4765 159C                            2      ORG                                _CODE                                ;
4766 159C 80                          1      DB 80H                                ;
4767                                SD$ 'Control#'
4768 159D 038D R                      1      DW DOLIT
4769 159F 15A3 R 0394 R                1      DW                                _LEN,EXIT                                ;
4770 15A3 00 43 6F 6E 74 72 6F       1      DB                                0,'Control#'                                ;
4771 15A3                            1      ORG                                _LEN                                ;
4772 15A3 08                          1      DB                                _CODE-_LEN-1                                ;
4773 15AC                            1      ORG                                _CODE                                ;
4774
4775                                ; L43      ( --- a )
4776                                ;      Packed string. 'a' is addr of count byte.
4777                                $COLON 3,'L43',L43
4778 15AC                            2      L43:                                ;
4779 3609                            2      ORG                                _NAME                                ;
4780 3609 15AC R 3615 R                2      DW                                _CODE,_LINK                                ;
4781 360D 03 4C 34 33                 2      DB                                3,'L43'                                ;
4782 15AC                            2      ORG                                _CODE                                ;
4783 15AC 80                          1      DB 80H                                ;
4784                                SD$ 'Program#'
4785 15AD 038D R                      1      DW DOLIT
4786 15AF 15B3 R 0394 R                1      DW                                _LEN,EXIT                                ;
4787 15B3 00 50 72 6F 67 72 61       1      DB                                0,'Program#'                                ;
4788 15B3                            1      ORG                                _LEN                                ;
4789 15B3 08                          1      DB                                _CODE-_LEN-1                                ;
4790 15BC                            1      ORG                                _CODE                                ;
4791
4792                                ; L44      ( --- a )
4793                                ;      Packed string. 'a' is addr of count byte.
4794                                $COLON 3,'L44',L44
4795 15BC                            2      L44:                                ;
4796 3601                            2      ORG                                _NAME                                ;
4797 3601 15BC R 360D R                2      DW                                _CODE,_LINK                                ;
4798 3605 03 4C 34 34                 2      DB                                3,'L44'                                ;
4799 15BC                            2      ORG                                _CODE                                ;
4800 15BC 80                          1      DB 80H                                ;
4801                                SD$ 'Ch Press'
4802 15BD 038D R                      1      DW DOLIT
4803 15BF 15C3 R 0394 R                1      DW                                _LEN,EXIT                                ;
4804 15C3 00 43 68 20 50 72 65       1      DB                                0,'Ch Press'                                ;
4805 15C3                            1      ORG                                _LEN                                ;
4806 15C3 08                          1      DB                                _CODE-_LEN-1                                ;
4807 15CC                            1      ORG                                _CODE                                ;
4808
4809                                ; L45      ( --- a )
4810                                ;      Packed string. 'a' is addr of count byte.
4811                                $COLON 3,'L45',L45
4812 15CC                            2      L45:                                ;
4813 35F9                            2      ORG                                _NAME                                ;
4814 35F9 15CC R 3605 R                2      DW                                _CODE,_LINK                                ;
4815 35FD 03 4C 34 35                 2      DB                                3,'L45'                                ;

```



```

4816 15CC          2      ORG      _CODE          ;
4817 15CC  80      1      DB      80H          ;
4818              ;
4819 15CD  038D R   1      DW      DOLIT         ;
4820 15CF  15D3 R 0394 R 1      DW      _LEN,EXIT       ;
4821 15D3  00 50 74 63 68 20 57 1      DB      0,'Ptch Whl'     ;
4822 15D3          1      ORG      _LEN          ;
4823 15D3  08      1      DB      _CODE-_LEN-1      ;
4824 15DC          1      ORG      _CODE          ;
4825              ;
4826              ; L4X      ( --- a )
4827              ;      Packed string. 'a' is addr of count byte.
4828              ;      $COLON 3,'L4X',L4X
4829 15DC          2      L4X:          ;
4830 35F1          2      ORG      _NAME          ;
4831 35F1  15DC R 35FD R 2      DW      _CODE,_LINK     ;
4832 35F5  03 4C 34 58 2      DB      3,'L4X'         ;
4833 15DC          2      ORG      _CODE          ;
4834 15DC  80      1      DB      80H          ;
4835              ;      SD$ '*****'
4836 15DD  038D R   1      DW      DOLIT         ;
4837 15DF  15E3 R 0394 R 1      DW      _LEN,EXIT       ;
4838 15E3  00 2A 2A 2A 2A 2A 2A 1      DB      0,'*****'     ;
4839 15E3          1      ORG      _LEN          ;
4840 15E3  08      1      DB      _CODE-_LEN-1      ;
4841 15EC          1      ORG      _CODE          ;
4842              ;
4843              ; L50      ( --- a )
4844              ;      Packed string. 'a' is addr of count byte.
4845              ;      $COLON 3,'L50',L50
4846 15EC          2      L50:          ;
4847 35E9          2      ORG      _NAME          ;
4848 35E9  15EC R 35F5 R 2      DW      _CODE,_LINK     ;
4849 35ED  03 4C 35 30 2      DB      3,'L50'         ;
4850 15EC          2      ORG      _CODE          ;
4851 15EC  80      1      DB      80H          ;
4852              ;      SD$ '****'
4853 15ED  038D R   1      DW      DOLIT         ;
4854 15EF  15F3 R 0394 R 1      DW      _LEN,EXIT       ;
4855 15F3  00 2A 2A 2A 1      DB      0,'****'     ;
4856 15F3          1      ORG      _LEN          ;
4857 15F3  03      1      DB      _CODE-_LEN-1      ;
4858 15F7          1      ORG      _CODE          ;
4859              ;
4860              ; LSTAT     ( n --- )
4861              ;      Choose a midi status label.
4862              ;      $COLON 5,'LSTAT',LSTAT
4863 15F7          2      LSTAT:         ;
4864 35DF          2      ORG      _NAME          ;
4865 35DF  15F7 R 35ED R 2      DW      _CODE,_LINK     ;
4866 35E3  05 4C 53 54 41 54 2      DB      5,'LSTAT'     ;
4867 15F7          2      ORG      _CODE          ;
4868 15F7  80      1      DB      80H          ;
4869 15F8  141E R 15DC R 157C R 1      DW      CASE,L4X,L40,L41,L42,L43,L44,L45,L4X,EXIT
4870      158C R 159C R 15AC R
4871      15BC R 15CC R 15DC R

```

```

4872      0394 R
4873
4874      ; eDISP      ( --- )
4875      ;          Display the Edit buffer on the LCD
4876      $COLON 5,'eDISP',EDISP
4877 160C      2      EDISP:      ;
4878 35D5      2      ORG      _NAME      ;
4879 35D5 160C R 35E3 R      2      DW      _CODE,_LINK      ;
4880 35D9 05 65 44 49 53 50      2      DB      5,'eDISP'      ;
4881 160C      2      ORG      _CODE      ;
4882 160C 80      1      DB 80H      ;
4883 160D 1532 R 1501 R 1403 R      DW      L0,FLD0,DISP,ESLDN,CAT,FLD01,NDISP
4884      14D7 R 03F1 R 1508 R
4885      13D8 R
4886 161B 14F3 R 03F1 R 038D R      DW      EBYTE3,CAT,DOLIT,80H,ANDD
4887      0080 0461 R
4888 1625 03B7 R 1633 R      DW      QBRAN,EDISP1
4889 1629 1571 R 150F R 1403 R      DW      L21,FLD2,DISP
4890 162F 03CC R 1639 R      DW      BRAN,EDISP2
4891 1633 1566 R 150F R 1403 R      EDISP1:  DW      L20,FLD2,DISP
4892 1639 14E5 R 03F1 R 043B R      EDISP2:  DW      EBYTE1,CAT,DUPP,DOLIT,0FH,ANDD,FLD3,NDISP
4893      038D R 000F 0461 R
4894      1516 R 13D8 R
4895 1649 063D R 063D R 063D R      DW      TWOSL,TWOSL,TWOSL,TWOSL,DOLIT,7H,ANDD
4896      063D R 038D R 0007
4897      0461 R
4898 1657 15F7 R 151D R 1403 R      DW      LSTAT,FLD4,DISP,EBYTE2,CAT,FLD5,NDISP
4899      14EC R 03F1 R 1524 R
4900      13D8 R
4901 1665 14F3 R 03F1 R 038D R      DW      EBYTE3,CAT,DOLIT,7FH,ANDD,FLD6,NDISP
4902      007F 0461 R 152B R
4903      13D8 R
4904 1673 038D R 0006 14DE R      DW      DOLIT,6H,EFLD,CSTOR,EXIT
4905      03E9 R 0394 R
4906
4907      ; BL/R      ( fld --- pos )
4908      ;          Translates LCD field number to a position number.
4909      $COLON 4,'BL/R',BLR
4910 167D      2      BLR:      ;
4911 35CB      2      ORG      _NAME      ;
4912 35CB 167D R 35D9 R      2      DW      _CODE,_LINK      ;
4913 35CF 04 42 4C 2F 52      2      DB      4,'BL/R'      ;
4914 167D      2      ORG      _CODE      ;
4915 167D 80      1      DB 80H      ;
4916 167E 043B R 14DE R 03F1 R      DW      DUPP,EFLD,CAT,CASE
4917      141E R
4918 1686 1501 R 1508 R 150F R      DW      FLD0,FLD01,FLD2,FLD3,FLD4,FLD5,FLD6
4919      1516 R 151D R 1524 R
4920      152B R
4921 1694 0394 R      DW      EXIT
4922
4923      ; BLEFT      ( --- )
4924      ;          Moves the LCD cursor to next field. Loads eFLD.
4925      $COLON 5,'BLEFT',BLEFT
4926 1696      2      BLEFT:      ;
4927 35C1      2      ORG      _NAME      ;

```

```

4928 35C1 1696 R 35CF R      2      DW      _CODE,_LINK      ;
4929 35C5 05 42 4C 45 46 54  2      DB      5,'BLEFT'      ;
4930 1696                      2      ORG      _CODE      ;
4931 1696 80                      1      DB      80H      ;
4932 1697 038D R 0040 147D R      DW      DOLIT,40H,LEDB
4933 169D 14DE R 03F1 R 038D R      DW      EFLD,CAT,DOLIT,6,DECR,BLR,LI,EXIT
4934      0006 1444 R 167D R
4935      135A R 0394 R
4936
4937      ; BRIGHT      ( --- )
4938      ;      Moves the LCD cursor to next field. Loads eFLD.
4939      $COLON 6,'BRIGHT',BRIGHT
4940 16AD                      2      BRIGHT:      ;
4941 35B5                      2      ORG      _NAME      ;
4942 35B5 16AD R 35C5 R      2      DW      _CODE,_LINK      ;
4943 35B9 06 42 52 49 47 48 54  2      DB      6,'BRIGHT'      ;
4944 16AD                      2      ORG      _CODE      ;
4945 16AD 80                      1      DB      80H      ;
4946 16AE 038D R 0080 147D R      DW      DOLIT,80H,LEDB
4947 16B4 14DE R 03F1 R 038D R      DW      EFLD,CAT,DOLIT,6,INCR,BLR,LI,EXIT
4948      0006 142B R 167D R
4949      135A R 0394 R
4950
4951      ; BLOAD      ( --- )
4952      ;      Load Buffer data shown on LCD into Slider Memory.
4953      $COLON 5,'BLOAD',BLOAD
4954 16C4                      2      BLOAD:      ;
4955 35AB                      2      ORG      _NAME      ;
4956 35AB 16C4 R 35B9 R      2      DW      _CODE,_LINK      ;
4957 35AF 05 42 4C 4F 41 44      2      DB      5,'BLOAD'      ;
4958 16C4                      2      ORG      _CODE      ;
4959 16C4 80                      1      DB      80H      ;
4960 16C5 038D R 0004 147D R      DW      DOLIT,4,LEDB
4961 16CB 14A7 R 0394 R      DW      ELOAD,EXIT
4962
4963      ; BMIDI      ( --- )
4964      ;      Start the Midi program.
4965      $COLON 5,'BMIDI',BMIDI
4966 16CF                      2      BMIDI:      ;
4967 35A1                      2      ORG      _NAME      ;
4968 35A1 16CF R 35AF R      2      DW      _CODE,_LINK      ;
4969 35A5 05 42 4D 49 44 49      2      DB      5,'BMIDI'      ;
4970 16CF                      2      ORG      _CODE      ;
4971 16CF 80                      1      DB      80H      ;
4972 16D0 038D R 0001 135A R      DW      DOLIT,1,LI
4973 16D6 154E R 1501 R 1403 R      DW      L2,FLD0,DISP
4974 16DC 0394 R      DW      EXIT
4975
4976      ; BUP      ( --- )
4977      ;      Increment value in LCD cursor field.
4978      $COLON 3,'BUP',BUP
4979 16DE                      2      BUP:      ;
4980 3599                      2      ORG      _NAME      ;
4981 3599 16DE R 35A5 R      2      DW      _CODE,_LINK      ;
4982 359D 03 42 55 50      2      DB      3,'BUP'      ;
4983 16DE                      2      ORG      _CODE      ;

```

```

4984 16DE 80          1          DB 80H          ;
4985 16DF 038D R 0010 147D R      DW      DOLIT,10H,LEDB
4986 16E5 038D R 0001 14DE R      DW      DOLIT,1,EFLD,CAT
4987          03F1 R
4988 16ED 038D R 0007 0461 R      DW      DOLIT,7,ANDD,CASE
4989          141E R
4990 16F5 1730 R 1747 R 1779 R      DW      UD0,UD1,UD2,UD3,UD4,UD5,UD6,UD7,EXIT
4991          17B6 R 17F0 R 1846 R
4992          173E R 1739 R 0394 R
4993
4994          ; BDOWN      ( --- )
4995          ;          Decrement value in LCD cursor field.
4996          $COLON 3,'BDOWN',BDOWN
4997 1707          2          BDOWN:          ;
4998 3591          2          ORG          _NAME          ;
4999 3591 1707 R 359D R      2          DW          _CODE,_LINK          ;
5000 3595 03 42 44 4F 57 4E      2          DB          3,'BDOWN'          ;
5001 1707          2          ORG          _CODE          ;
5002 1707 80          1          DB 80H          ;
5003 1708 038D R 0020 147D R      DW      DOLIT,20H,LEDB
5004 170E 038D R 0000 14DE R      DW      DOLIT,0,EFLD,CAT
5005          03F1 R
5006 1716 038D R 0007 0461 R      DW      DOLIT,7,ANDD,CASE
5007          141E R
5008 171E 1730 R 1747 R 1779 R      DW      UD0,UD1,UD2,UD3,UD4,UD5,UD6,UD7,EXIT
5009          17B6 R 17F0 R 1846 R
5010          173E R 1739 R 0394 R
5011
5012          ; U/D0      ( i/d --- )
5013          ;          Field increment/decrement routine.
5014          $COLON 4,'U/D0',UD0
5015 1730          2          UD0:          ;
5016 3587          2          ORG          _NAME          ;
5017 3587 1730 R 3595 R      2          DW          _CODE,_LINK          ;
5018 358B 04 55 2F 44 30      2          DB          4,'U/D0'          ;
5019 1730          2          ORG          _CODE          ;
5020 1730 80          1          DB 80H          ;
5021 1731 0436 R 1501 R 135A R      DW      DROP,FLD0,LI,EXIT
5022          0394 R
5023
5024          ; U/D7      ( i/d --- )
5025          ;          Field increment/decrement routine. (bogus field)
5026          $COLON 4,'U/D7',UD7
5027 1739          2          UD7:          ;
5028 357D          2          ORG          _NAME          ;
5029 357D 1739 R 358B R      2          DW          _CODE,_LINK          ;
5030 3581 04 55 2F 44 37      2          DB          4,'U/D7'          ;
5031 1739          2          ORG          _CODE          ;
5032 1739 80          1          DB 80H          ;
5033 173A 0436 R 0394 R      DW      DROP,EXIT
5034
5035          ; U/D6      ( i/d --- )
5036          ;          Field increment/decrement routine.
5037          $COLON 4,'U/D6',UD6
5038 173E          2          UD6:          ;
5039 3573          2          ORG          _NAME          ;

```

```

5040 3573 173E R 3581 R      2      DW      _CODE,_LINK      ;
5041 3577 04 55 2F 44 36    2      DB      4,'U/D6'      ;
5042 173E      2      ORG      _CODE      ;
5043 173E 80      1      DB      80H      ;
5044 173F 0436 R 152B R 135A R  DW      DROP,FLD6,LI,EXIT
5045      0394 R
5046
5047      ; U/D1      ( i/d --- )
5048      ;      Field increment/decrement routine.
5049      $COLON 4,'U/D1',UD1
5050 1747      2      UD1:      ;
5051 3569      2      ORG      _NAME      ;
5052 3569 1747 R 3577 R      2      DW      _CODE,_LINK      ;
5053 356D 04 55 2F 44 31    2      DB      4,'U/D1'      ;
5054 1747      2      ORG      _CODE      ;
5055 1747 80      1      DB      80H      ;
5056 1748 14D7 R 03F1 R 038D R  DW      ESLDN,CAT,DOLIT,37H,ROT
5057      0037 054A R
5058 1752 03B7 R 175C R      DW      QBRAN,UD1A
5059 1756 142B R      DW      INCR
5060 1758 03CC R 175E R      DW      BRAN,UD1B
5061 175C 1444 R      UD1A:      DW      DECR
5062 175E 1762 R 0394 R      UD1B:      DW      CFLD1,EXIT
5063
5064      ; CFLD1      ( sld# --- )
5065      ;      Change Slider# in field 1. Update Edit buffer & LCD.
5066      $COLON 5,'CFLD1',CFLD1
5067 1762      2      CFLD1:      ;
5068 355F      2      ORG      _NAME      ;
5069 355F 1762 R 356D R      2      DW      _CODE,_LINK      ;
5070 3563 05 43 46 4C 44 31  2      DB      5,'CFLD1'      ;
5071 1762      2      ORG      _CODE      ;
5072 1762 80      1      DB      80H      ;
5073 1763 14D7 R 03F1 R 148B R  DW      ESLDN,CAT,EUPDAT,EDISP
5074      160C R
5075 176B 038D R 0001 14DE R  DW      DOLIT,1,EFLD,CSTOR,FLD01,LI,EXIT
5076      03E9 R 1508 R 135A R
5077      0394 R
5078
5079      ; U/D2      ( i/d --- )
5080      ;      Field increment/decrement routine.
5081      $COLON 4,'U/D2',UD2
5082 1779      2      UD2:      ;
5083 3555      2      ORG      _NAME      ;
5084 3555 1779 R 3563 R      2      DW      _CODE,_LINK      ;
5085 3559 04 55 2F 44 32    2      DB      4,'U/D2'      ;
5086 1779      2      ORG      _CODE      ;
5087 1779 80      1      DB      80H      ;
5088 177A 0436 R 14F3 R 03F1 R  DW      DROP,EBYTE3,CAT,DUPP,DOLIT,80H,ANDD
5089      043B R 038D R 0080
5090      0461 R
5091 1788 03B7 R 17A0 R      DW      QBRAN,UD2A
5092 178C 038D R 007F 0461 R  DW      DOLIT,7FH,ANDD,EBYTE3,CAT,L20,FLD2,DISP
5093      14F3 R 03F1 R 1566 R
5094      150F R 1403 R
5095 179C 03CC R 17B0 R      DW      BRAN,UD2B

```

```

5096 17A0 038D R 0080 046A R      UD2A:      DW      DOLIT,80H,ORR,EBYTE3,CSTOR,L21,FLD2,DISP
5097      14F3 R 03E9 R 1571 R
5098      150F R 1403 R
5099 17B0 150F R 135A R 0394 R      UD2B:      DW      FLD2,LI,EXIT
5100
5101      ; U/D3      ( i/d --- )
5102      ;      Field increment/decrement routine.
5103      $COLON 4,'U/D3',UD3
5104 17B6      2      UD3:      ;
5105 354B      2      ORG      _NAME      ;
5106 354B 17B6 R 3559 R      2      DW      _CODE,_LINK      ;
5107 354F 04 55 2F 44 33      2      DB      4,'U/D3'      ;
5108 17B6      2      ORG      _CODE      ;
5109 17B6 80      1      DB 80H      ;
5110 17B7 14E5 R 03F1 R 038D R      DW      EBYTE1,CAT,DOLIT,0FH,ANDD,DOLIT,0FH,ROT
5111      000F 0461 R 038D R
5112      000F 054A R
5113 17C7 03B7 R 17D1 R      DW      QBRAN,UD3A
5114 17CB 142B R      DW      INCR
5115 17CD 03CC R 17D3 R      DW      BRAN,UD3B
5116 17D1 1444 R      UD3A:      DW      DECR
5117 17D3 17D7 R 0394 R      UD3B:      DW      CFLD3,EXIT
5118
5119      ; CFLD3      ( chnl --- )
5120      ;      Change midi channel in field 3.
5121      $COLON 5,'CFLD3',CFLD3
5122 17D7      2      CFLD3:      ;
5123 3541      2      ORG      _NAME      ;
5124 3541 17D7 R 354F R      2      DW      _CODE,_LINK      ;
5125 3545 05 43 46 4C 44 33      2      DB      5,'CFLD3'      ;
5126 17D7      2      ORG      _CODE      ;
5127 17D7 80      1      DB 80H      ;
5128 17D8 043B R 14E5 R 03F1 R      DW      DUPP,EBYTE1,CAT,DOLIT,0F0H
5129      038D R 00F0
5130 17E2 0461 R 046A R 14E5 R      DW      ANDD,ORR,EBYTE1,CSTOR,FLD3,NDISP,EXIT
5131      03E9 R 1516 R 13D8 R
5132      0394 R
5133
5134      ; U/D4      ( i/d --- )
5135      ;      Field increment/decrement routine.
5136      $COLON 4,'U/D4',UD4
5137 17F0      2      UD4:      ;
5138 3537      2      ORG      _NAME      ;
5139 3537 17F0 R 3545 R      2      DW      _CODE,_LINK      ;
5140 353B 04 55 2F 44 34      2      DB      4,'U/D4'      ;
5141 17F0      2      ORG      _CODE      ;
5142 17F0 80      1      DB 80H      ;
5143 17F1 14E5 R 03F1 R 038D R      DW      EBYTE1,CAT,DOLIT,70H,ANDD
5144      0070 0461 R
5145 17FB 0645 R 0645 R 0645 R      DW      TWOSR,TWOSR,TWOSR,TWOSR,DOLIT,7,ROT
5146      0645 R 038D R 0007
5147      054A R
5148 1809 03B7 R 1813 R      DW      QBRAN,UD4A
5149 180D 142B R      DW      INCR
5150 180F 03CC R 1815 R      DW      BRAN,UD4B
5151 1813 1444 R      UD4A:      DW      DECR

```

```

5152 1815 1819 R 0394 R          UD4B:      DW      CFLD4,EXIT
5153
5154          ; CFLD4      ( status --- )
5155          ;          Change Midi operation label in field 4.
5156          $COLON 5,'CFLD4',CFLD4
5157 1819          2          CFLD4:          ;
5158 352D          2          ORG      _NAME          ;
5159 352D 1819 R 353B R          2          DW      _CODE,_LINK          ;
5160 3531 05 43 46 4C 44 34      2          DB      5,'CFLD4'          ;
5161 1819          2          ORG      _CODE          ;
5162 1819 80          1          DB      80H          ;
5163 181A 043B R 063D R 063D R          DW      DUPP,TWOSL,TWOSL,TWOSL,TWOSL
5164          063D R 063D R
5165 1824 038D R 0080 046A R          DW      DOLIT,80H,ORR,EBYTE1,CAT
5166          14E5 R 03F1 R
5167 182E 038D R 000F 0461 R          DW      DOLIT,0FH,ANDD,ORR,EBYTE1,CSTOR
5168          046A R 14E5 R 03E9 R
5169 183A 15F7 R 151D R 1403 R          DW      LSTAT,FLD4,DISP,FLD4,LI,EXIT
5170          151D R 135A R 0394 R
5171
5172          ; U/D5      ( i/d --- )
5173          ;          Field increment/decrement routine.
5174          $COLON 4,'U/D5',UD5
5175 1846          2          UD5:          ;
5176 3523          2          ORG      _NAME          ;
5177 3523 1846 R 3531 R          2          DW      _CODE,_LINK          ;
5178 3527 04 55 2F 44 35          2          DB      4,'U/D5'          ;
5179 1846          2          ORG      _CODE          ;
5180 1846 80          1          DB      80H          ;
5181 1847 038D R 00CF 14E5 R          DW      DOLIT,0CFH,EBYTE1,CAT,DOLIT,0F0H,ANDD,LESS
5182          03F1 R 038D R 00F0
5183          0461 R 05E1 R
5184 1857 03B7 R 186B R          DW      QBRAN,UD5A
5185 185B 15EC R 1524 R 1403 R          DW      L50,FLD5,DISP,FLD5,LI,DROP
5186          1524 R 135A R 0436 R
5187 1867 03CC R 1875 R          DW      BRAN,UD5B
5188 186B 043B R 14EC R 03E9 R          UD5A:      DW      DUPP,EBYTE2,CSTOR,FLD5,NDISP
5189          1524 R 13D8 R
5190 1875 0394 R          UD5B:      DW      EXIT
5191
5192
5193
5194
5195          ;
5196          ;=====
5197
5198 = 3527          LASTN      EQU      _NAME+4          ;last name address
5199
5200 = 3523          NTOPP      EQU      _NAME-0          ;next available memory in ROM name
dictionary
5201 = 1877          CTOPP      EQU      $+0          ;next available memory in ROM code
dictionary
5202 =          ROMSPC      EQU      NTOPP-CTOPP ;UNUSED DICTIONARY ROM SPACE
5203
5204 1877          MAIN      ENDS
5205          END      ORIG

```

Macros:

N a m e	Lines
\$CODE	9
\$COLON	2
\$NEXT	2
\$USER	4
D\$	7
SD\$	8

Segments and Groups:

N a m e	Length	Align	Combine	Class
MAIN	3FFC	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr
ABOR1	L NEAR	0DE8	MAIN
ABORQ	L NEAR	0DDF	MAIN
ABORT	L NEAR	0DDA	MAIN
ABRTQ	L NEAR	0FFA	MAIN
ABS1	L NEAR	05B7	MAIN
ABSS	L NEAR	05AC	MAIN
ACCEP	L NEAR	0D34	MAIN
ACCP1	L NEAR	0D3B	MAIN
ACCP2	L NEAR	0D59	MAIN
ACCP3	L NEAR	0D5D	MAIN
ACCP4	L NEAR	0D61	MAIN
ADCINIT	L NEAR	1318	MAIN
AFT	L NEAR	0FDF	MAIN
AGAIN	L NEAR	0FA4	MAIN
AHEAD	L NEAR	0FBC	MAIN
ALLOT	L NEAR	0F24	MAIN
ANDD	L NEAR	0461	MAIN
AT	L NEAR	03DE	MAIN
ATEXE	L NEAR	07F6	MAIN
BACK1	L NEAR	0CF8	MAIN
BASE	L NEAR	04CB	MAIN
BASEE	NUMBER	000A	
BCOMP	L NEAR	0F49	MAIN
BDIGS	L NEAR	08BB	MAIN
BDOWN	L NEAR	1707	MAIN
BEGIN	L NEAR	0F8D	MAIN
BITIME	L NEAR	052A	MAIN
BKSLA	L NEAR	0BAD	MAIN
BKSP	L NEAR	0CCF	MAIN
BKSPP	NUMBER	0008	
BLANK	L NEAR	0767	MAIN
BLEFT	L NEAR	1696	MAIN
BLOAD	L NEAR	16C4	MAIN
BLR	L NEAR	167D	MAIN

BMIDI	L NEAR 16CF	MAIN
BRAN	L NEAR 03CC	MAIN
BRAN1	L NEAR 03C5	MAIN
BRIGHT	L NEAR 16AD	MAIN
BUP	L NEAR 16DE	MAIN
BYE	L NEAR 032B	MAIN
CALLC	L NEAR 10E2	MAIN
CALLL	NUMBER 0080	
CASE	L NEAR 141E	MAIN
CAT	L NEAR 03F1	MAIN
CATCH	L NEAR 0D91	MAIN
CCOLD	NEAR 0300	MAIN
CCOM1	L NEAR 10AD	MAIN
CCOM2	L NEAR 10B1	MAIN
CCOM3	L NEAR 10BD	MAIN
CCOMMA	L NEAR 0F3A	MAIN
CCOMP	L NEAR 1094	MAIN
CELLL	NUMBER 0002	
CFLD1	L NEAR 1762	MAIN
CFLD3	L NEAR 17D7	MAIN
CFLD4	L NEAR 1819	MAIN
CHAR	L NEAR 0BB8	MAIN
CHAR1	L NEAR 0A43	MAIN
CHAR2	L NEAR 0A45	MAIN
CMOV1	L NEAR 080A	MAIN
CMOV2	L NEAR 081A	MAIN
CMOVE	L NEAR 0803	MAIN
CNTXT	L NEAR 04FD	MAIN
CODE	L NEAR 1136	MAIN
CODEE	NUMBER 0300	
COLD	L NEAR 0300	MAIN
COLD1	L NEAR 0301	MAIN
COLDD	NUMBER 0100	
COLON	L NEAR 10EB	MAIN
COMMA	L NEAR 0F2B	MAIN
COMPI	L NEAR 0F50	MAIN
COMPO	NUMBER 0040	
CONSO	L NEAR 0EBF	MAIN
COUNT	L NEAR 07D4	MAIN
CP	L NEAR 0511	MAIN
CR	L NEAR 0A62	MAIN
CREAT	L NEAR 111C	MAIN
CRR	NUMBER 000D	
CRRNT	L NEAR 0502	MAIN
CSP	L NEAR 04E4	MAIN
CSTOR	L NEAR 03E9	MAIN
CTOP	NUMBER C390	
CTOPP	NEAR 1877	MAIN
DAT	L NEAR 07C7	MAIN
DDRDP	L NEAR 0555	MAIN
DDUP	L NEAR 055B	MAIN
DECIM	L NEAR 092B	MAIN
DECR	L NEAR 1444	MAIN
DECR1	L NEAR 1457	MAIN

DECR2	L NEAR 145B	MAIN
DELAY	L NEAR 1339	MAIN
DEPTH	L NEAR 078B	MAIN
DGTQ1	L NEAR 095B	MAIN
DIG	L NEAR 08D5	MAIN
DIGIT	L NEAR 0893	MAIN
DIGS	L NEAR 08E0	MAIN
DIGS1	L NEAR 08E1	MAIN
DIGS2	L NEAR 08ED	MAIN
DIGTQ	L NEAR 0936	MAIN
DISP	L NEAR 1403	MAIN
DISP1	L NEAR 140E	MAIN
DMP	L NEAR 117D	MAIN
DNEGA	L NEAR 0588	MAIN
DOLIT	L NEAR 038D	MAIN
DONXT	L NEAR 039D	MAIN
DOSTR	L NEAR 0A71	MAIN
DOT	L NEAR 0AC5	MAIN
DOT1	L NEAR 0AD8	MAIN
DOTI1	L NEAR 1274	MAIN
DOTID	L NEAR 1261	MAIN
DOTO1	L NEAR 0E3F	MAIN
DOTOK	L NEAR 0E2A	MAIN
DOTPR	L NEAR 0B97	MAIN
DOTQ	L NEAR 100C	MAIN
DOTQP	L NEAR 0A89	MAIN
DOTR	L NEAR 0A92	MAIN
DOTS	L NEAR 11E3	MAIN
DOTS1	L NEAR 11EE	MAIN
DOTS2	L NEAR 11F4	MAIN
DOUSE	L NEAR 0496	MAIN
DOVAR	L NEAR 048C	MAIN
DOVOC	L NEAR 052F	MAIN
DROP	L NEAR 0436	MAIN
DSTOR	L NEAR 07BA	MAIN
DTRA1	L NEAR 0842	MAIN
DTRA2	L NEAR 0858	MAIN
DTRAI	L NEAR 083B	MAIN
DUMP	L NEAR 11A0	MAIN
DUMP1	L NEAR 11B1	MAIN
DUMP2	L NEAR 11D5	MAIN
DUMP3	L NEAR 11D9	MAIN
DUPP	L NEAR 043B	MAIN
EBYTE1	L NEAR 14E5	MAIN
EBYTE2	L NEAR 14EC	MAIN
EBYTE3	L NEAR 14F3	MAIN
EDIGS	L NEAR 08FE	MAIN
EDISP	L NEAR 160C	MAIN
EDISP1	L NEAR 1633	MAIN
EDISP2	L NEAR 1639	MAIN
EFLAG	L NEAR 14FA	MAIN
EFLD	L NEAR 14DE	MAIN
ELOAD	L NEAR 14A7	MAIN
ELSEE	L NEAR 0FEA	MAIN
EMIT	L NEAR 0A0A	MAIN

ENDCD	L NEAR 1145	MAIN
EQUAL	L NEAR 05B9	MAIN
ERR	NUMBER 001B	
ESLDN	L NEAR 14D7	MAIN
ESUPDAT	L NEAR 14C9	MAIN
EUPDAT	L NEAR 148B	MAIN
EVAL	L NEAR 0E57	MAIN
EVAL1	L NEAR 0E58	MAIN
EVAL2	L NEAR 0E6C	MAIN
EXE1	L NEAR 0801	MAIN
EXECU	L NEAR 039B	MAIN
EXIT	L NEAR 0394	MAIN
EXPEC	L NEAR 0D69	MAIN
EXT	NUMBER 0001	
EXTRC	L NEAR 08AC	MAIN
FHEAD	L NEAR 0507	MAIN
FILE	L NEAR 0E96	MAIN
FILL	L NEAR 0822	MAIN
FILL1	L NEAR 082D	MAIN
FILL2	L NEAR 0833	MAIN
FIND	L NEAR 0C21	MAIN
FIND1	L NEAR 0C3C	MAIN
FIND2	L NEAR 0C60	MAIN
FIND3	L NEAR 0C68	MAIN
FIND4	L NEAR 0C78	MAIN
FIND5	L NEAR 0C84	MAIN
FIND6	L NEAR 0C6C	MAIN
FLD0	L NEAR 1501	MAIN
FLD01	L NEAR 1508	MAIN
FLD2	L NEAR 150F	MAIN
FLD3	L NEAR 1516	MAIN
FLD4	L NEAR 151D	MAIN
FLD5	L NEAR 1524	MAIN
FLD6	L NEAR 152B	MAIN
FLINK	L NEAR 050C	MAIN
FOR	L NEAR 0F84	MAIN
FORTH	L NEAR 0538	MAIN
HAFBIT	L NEAR 0525	MAIN
HAND	L NEAR 0EA7	MAIN
HANDL	L NEAR 04F8	MAIN
HERE	L NEAR 07DF	MAIN
HEX	L NEAR 0920	MAIN
HI	L NEAR 12AA	MAIN
HLD	L NEAR 04F3	MAIN
HOLD	L NEAR 08C4	MAIN
IFF	L NEAR 0FAD	MAIN
IMEDD	NUMBER 0080	
IMMED	L NEAR 10F6	MAIN
INCR	L NEAR 142B	MAIN
INCR1	L NEAR 1440	MAIN
INCR2	L NEAR 1442	MAIN
INN	L NEAR 04DA	MAIN
INTE1	L NEAR 0E13	MAIN

INTE2	L NEAR 0E1D	MAIN
INTER	L NEAR 0DEE	MAIN
INVER	L NEAR 056F	MAIN
ISLO	L NEAR 0EB8	MAIN
KEY	L NEAR 0A01	MAIN
KEY1	L NEAR 0A02	MAIN
KTAP	L NEAR 0D09	MAIN
KTAP1	L NEAR 0D26	MAIN
KTAP2	L NEAR 0D2A	MAIN
L0	L NEAR 1532	MAIN
L1	L NEAR 1540	MAIN
L2	L NEAR 154E	MAIN
L20	L NEAR 1566	MAIN
L21	L NEAR 1571	MAIN
L40	L NEAR 157C	MAIN
L41	L NEAR 158C	MAIN
L42	L NEAR 159C	MAIN
L43	L NEAR 15AC	MAIN
L44	L NEAR 15BC	MAIN
L45	L NEAR 15CC	MAIN
L4X	L NEAR 15DC	MAIN
L50	L NEAR 15EC	MAIN
LAST	L NEAR 051B	MAIN
LASTN	NUMBER 3527	
LBRAC	L NEAR 0E1F	MAIN
LCD	L NEAR 1342	MAIN
LCDINIT	L NEAR 1363	MAIN
LEDB	L NEAR 147D	MAIN
LESS	L NEAR 05E1	MAIN
LESS1	L NEAR 05F2	MAIN
LF	NUMBER 000A	
LI	L NEAR 135A	MAIN
LITER	L NEAR 0F5F	MAIN
LLC	L NEAR 1353	MAIN
LLI	L NEAR 134C	MAIN
LSTAT	L NEAR 15F7	MAIN
MASKK	NUMBER 7F1F	
MAX	L NEAR 05F8	MAIN
MIN	L NEAR 0604	MAIN
MMOD1	L NEAR 06BF	MAIN
MMOD2	L NEAR 06CD	MAIN
MMOD3	L NEAR 06DD	MAIN
MODD	L NEAR 06EA	MAIN
MSMOD	L NEAR 06AA	MAIN
MSTA1	L NEAR 0751	MAIN
MSTAR	L NEAR 0738	MAIN
NAMEE	NUMBER 3FFD	
NAMEQ	L NEAR 0C96	MAIN
NAMET	L NEAR 0BE5	MAIN
NAMQ1	L NEAR 0CA5	MAIN
NAMQ2	L NEAR 0CA7	MAIN
NAMQ3	L NEAR 0CC5	MAIN

NDISP	L NEAR 13D8	MAIN
NEGAT	L NEAR 057B	MAIN
NEXT	L NEAR 0F92	MAIN
NEXT1	L NEAR 03AF	MAIN
NP	L NEAR 0516	MAIN
NTIB	L NEAR 04DF	MAIN
NTOP	NUMBER C7FF	
NTOPP	NUMBER 3523	
NUFQ	L NEAR 0A11	MAIN
NUFQ1	L NEAR 0A24	MAIN
NULLS	L NEAR 0DCF	MAIN
NUMBQ	L NEAR 0963	MAIN
NUMQ1	L NEAR 098A	MAIN
NUMQ2	L NEAR 09AC	MAIN
NUMQ3	L NEAR 09DA	MAIN
NUMQ4	L NEAR 09E0	MAIN
NUMQ5	L NEAR 09EC	MAIN
NUMQ6	L NEAR 09EE	MAIN
ONEM	L NEAR 0626	MAIN
ONEP	L NEAR 061F	MAIN
ORIG	L NEAR 0000	MAIN
ORR	L NEAR 046A	MAIN
OVER	L NEAR 044A	MAIN
OVERT	L NEAR 10BF	MAIN
PACE	L NEAR 0A26	MAIN
PACKS	L NEAR 0862	MAIN
PAD	L NEAR 07E6	MAIN
PADD	NUMBER C300	
PAREN	L NEAR 0BA2	MAIN
PARS	L NEAR 0AE7	MAIN
PARS1	L NEAR 0B06	MAIN
PARS2	L NEAR 0B28	MAIN
PARS3	L NEAR 0B2A	MAIN
PARS4	L NEAR 0B30	MAIN
PARS5	L NEAR 0B48	MAIN
PARS6	L NEAR 0B5A	MAIN
PARS7	L NEAR 0B64	MAIN
PARS8	L NEAR 0B70	MAIN
PARSE	L NEAR 0B78	MAIN
PDUM1	L NEAR 118E	MAIN
PDUM2	L NEAR 119A	MAIN
PICK	L NEAR 079E	MAIN
PLUS	L NEAR 0565	MAIN
PNAM1	L NEAR 105F	MAIN
PRESE	L NEAR 0E74	MAIN
PSTOR	L NEAR 07AB	MAIN
QBRAN	L NEAR 03B7	MAIN
QCSP	L NEAR 120A	MAIN
QDUP	L NEAR 053D	MAIN
QKEY	L NEAR 09FA	MAIN
QRX	L NEAR 032E	MAIN
QSTAC	L NEAR 0E43	MAIN
QUERY	L NEAR 0D76	MAIN

QUEST	L NEAR 0AE0	MAIN
QUIT	L NEAR 0ECC	MAIN
QUIT1	L NEAR 0ED3	MAIN
QUIT2	L NEAR 0ED5	MAIN
QUIT3	L NEAR 0F01	MAIN
QUIT4	L NEAR 0F11	MAIN
RAT	L NEAR 040C	MAIN
RBRAC	L NEAR 10D7	MAIN
RECUR	L NEAR 0F79	MAIN
REPEA	L NEAR 0FCB	MAIN
RFROM	L NEAR 0405	MAIN
ROMSPC	TEXT NTOPP-CTOPP	
ROT	L NEAR 054A	MAIN
RPAT	L NEAR 03FB	MAIN
RPP	NUMBER C2F0	
RPSTO	L NEAR 0400	MAIN
RZERO	L NEAR 04A8	MAIN
SAME1	L NEAR 0BF5	MAIN
SAME2	L NEAR 0C17	MAIN
SAMEQ	L NEAR 0BEE	MAIN
SAT	L NEAR 1467	MAIN
SCOM1	L NEAR 1082	MAIN
SCOM2	L NEAR 1086	MAIN
SCOM3	L NEAR 1092	MAIN
SCOMP	L NEAR 1069	MAIN
SEE	L NEAR 12E1	MAIN
SEE1	L NEAR 12E8	MAIN
SEE2	L NEAR 12FA	MAIN
SEE3	L NEAR 1306	MAIN
SEE4	L NEAR 130C	MAIN
SEMIS	L NEAR 10CC	MAIN
SERIN	L NEAR 0520	MAIN
SIGN	L NEAR 08EF	MAIN
SIGN1	L NEAR 08FC	MAIN
SLASH	L NEAR 06F1	MAIN
SLMOD	L NEAR 06DF	MAIN
SNAME	L NEAR 1032	MAIN
SPACE	L NEAR 0A2F	MAIN
SPACS	L NEAR 0A36	MAIN
SPAN	L NEAR 04D5	MAIN
SPAT	L NEAR 041C	MAIN
SPP	NUMBER C1F0	
SPSTO	L NEAR 0429	MAIN
SSMOD	L NEAR 0753	MAIN
STAR	L NEAR 0731	MAIN
STASL	L NEAR 075E	MAIN
STCSP	L NEAR 1201	MAIN
STOIO	L NEAR 0380	MAIN
STORE	L NEAR 03D3	MAIN
STR	L NEAR 090D	MAIN
STRCQ	L NEAR 0F68	MAIN
STRQ	L NEAR 1003	MAIN
STRQP	L NEAR 0A84	MAIN
SUBB	L NEAR 059B	MAIN

SWAP	L NEAR 0442	MAIN
SWAT	L NEAR 145D	MAIN
SZERO	L NEAR 04A3	MAIN
TAP	L NEAR 0CFA	MAIN
TBOOT	L NEAR 12DC	MAIN
TCHA1	L NEAR 0789	MAIN
TCHAR	L NEAR 076E	MAIN
TECHO	L NEAR 04C1	MAIN
TEMIT	L NEAR 04B2	MAIN
TEMP	L NEAR 04D0	MAIN
TEVAL	L NEAR 04E9	MAIN
TEXPE	L NEAR 04B7	MAIN
THENN	L NEAR 0FD6	MAIN
THROW	L NEAR 0DB4	MAIN
TIB	L NEAR 07ED	MAIN
TIBB	NUMBER C200	
TIC	NUMBER 0027	
TICK	L NEAR 0F17	MAIN
TICK1	L NEAR 0F22	MAIN
TMIDI	L NEAR 132E	MAIN
TNAM1	L NEAR 1221	MAIN
TNAM2	L NEAR 122D	MAIN
TNAM3	L NEAR 1245	MAIN
TNAM4	L NEAR 1259	MAIN
TNAME	L NEAR 121E	MAIN
TNUMB	L NEAR 04EE	MAIN
TOKEN	L NEAR 0BC3	MAIN
TOR	L NEAR 0413	MAIN
TPROM	L NEAR 04C6	MAIN
TQKEY	L NEAR 04AD	MAIN
TTAP	L NEAR 04BC	MAIN
TWOM	L NEAR 0635	MAIN
TWOP	L NEAR 062D	MAIN
TWOSL	L NEAR 063D	MAIN
TWOSR	L NEAR 0645	MAIN
TXSTO	L NEAR 0347	MAIN
TYPE1	L NEAR 0A52	MAIN
TYPE2	L NEAR 0A5A	MAIN
TYPEE	L NEAR 0A4B	MAIN
UD0	L NEAR 1730	MAIN
UD1	L NEAR 1747	MAIN
UD1A	L NEAR 175C	MAIN
UD1B	L NEAR 175E	MAIN
UD2	L NEAR 1779	MAIN
UD2A	L NEAR 17A0	MAIN
UD2B	L NEAR 17B0	MAIN
UD3	L NEAR 17B6	MAIN
UD3A	L NEAR 17D1	MAIN
UD3B	L NEAR 17D3	MAIN
UD4	L NEAR 17F0	MAIN
UD4A	L NEAR 1813	MAIN
UD4B	L NEAR 1815	MAIN
UD5	L NEAR 1846	MAIN
UD5A	L NEAR 186B	MAIN

UD5B	L NEAR 1875	MAIN
UD6	L NEAR 173E	MAIN
UD7	L NEAR 1739	MAIN
UDOT	L NEAR 0AB8	MAIN
UDOTR	L NEAR 0AA3	MAIN
ULAST	L NEAR 019E	MAIN
ULES1	L NEAR 05DB	MAIN
ULESS	L NEAR 05C8	MAIN
UMM1	L NEAR 065E	MAIN
UMM2	L NEAR 0690	MAIN
UMM3	L NEAR 0692	MAIN
UMM4	L NEAR 069E	MAIN
UMMOD	L NEAR 064D	MAIN
UMST1	L NEAR 0707	MAIN
UMST2	L NEAR 0727	MAIN
UMSTA	L NEAR 06FA	MAIN
UNIQ1	L NEAR 102E	MAIN
UNIQUE	L NEAR 1015	MAIN
UNTIL	L NEAR 0F9B	MAIN
UP	L NEAR 0491	MAIN
UPLUS	L NEAR 047C	MAIN
UPP	NUMBER FF00	
USER	L NEAR 110B	MAIN
UTYP1	L NEAR 116B	MAIN
UTYP2	L NEAR 1175	MAIN
UTYPE	L NEAR 1164	MAIN
UZERO	L NEAR 014E	MAIN
VARIA	L NEAR 112B	MAIN
VER	NUMBER 0001	
VERSN	L NEAR 12A3	MAIN
VOCSS	NUMBER 0006	
WHILE	L NEAR 0FF3	MAIN
WITHI	L NEAR 0610	MAIN
WORDD	L NEAR 0BDC	MAIN
WORDS	L NEAR 1282	MAIN
WORS1	L NEAR 1289	MAIN
WORS2	L NEAR 12A1	MAIN
XIO	L NEAR 0E87	MAIN
XORR	L NEAR 0473	MAIN
ZLESS	L NEAR 0453	MAIN
@FILENAME	TEXT TST	
_CODE	NEAR 1846	MAIN
_LEN	NUMBER 0002	
_LINK	NEAR 3527	MAIN
_NAME	NUMBER 3523	
_USER	NUMBER 0050	

2953 Source Lines
6417 Total Lines
463 Symbols

50298 + 348662 Bytes symbol space free

0 Warning Errors
0 Severe Errors