```
; From http://www.jupiter-ace.co.uk/romlisting.html
; [ thanks to Geoff Wearmouth ]
; Disassembly of the file "C:\ACE\JupiterAce.rom"
;
; CPU Type: Z80
;
; Created with dZ80 1.50
;
; on Monday, 21 of January 2002 at 07:11 PM
;
; last updated 02-NOV-2002
;
; Cross-assembles to an 8K ROM file.
;
; Note. A Low-level Assembly Listing only.
;

; With two minor corrections, and Vdrive support;
; floating point words have been moved to a .tap;
; and this version can no longer be assembled to give the original ROM.

; Word 'VER' was added to display the version of the ROM.

;;; some code was copied back from a .tap source,
;;; with word length fields removed

INVERSE:    .EQU    $80

; System variables (needs to be completed)

EXWRCH:     .EQU    $3C29
CONTEXT:    .EQU    $3C33

#define DEFB    .BYTE
#define DEFW    .WORD
#define DEFM    .TEXT
#define EQU     .EQU
#define ORG     .ORG

        ORG     $0000

; -------------------
; THE 'START' RESTART
; -------------------

L0000:  DI                              ; disable interrupts.
        LD      HL,$3C00                ; start of 'User' RAM
        LD      A,$FC                   ; a test byte and 1K masking byte.
        JR      L0028                   ; forward to continue at Part 2.

; -------------------
; THE 'PRINT' RESTART
; -------------------

L0008:  EXX                             ; preserve main registers.
        BIT     3,(IX+$3E)              ; test FLAGS for print destination.
        JP      L03EE                   ; forward to

; ---------------------------
; THE 'STACK WORD DE' RESTART
; ---------------------------

L0010:  LD      HL,($3C3B)              ; SPARE
        LD      (HL),E
        INC     HL
        JP      L085F                   ;

; -------------------------
; THE 'POP WORD DE' RESTART
; -------------------------


L0018:  LD      HL,($3C3B)              ; SPARE
```

```
74          DEC     HL
75          LD      D,(HL)
76          JP      L0859                   ;
77
78  ; ------------------
79  ; THE 'ERROR' RESTART
80  ; ------------------
81
82  L0020:  POP     HL
83          LD      A,(HL)
84          LD      ($3C3D),A               ; ERR_NO
85          JP      L00AD                   ;
86
87  ; ----------------------------------
88  ; THE 'INITIALIZATION ROUTINE' Part 2.
89  ; ----------------------------------
90
91  L0028:  INC     H                       ; increase high byte
92          LD      (HL),A                  ; insert A value
93          CP      (HL)                    ; compare to expected
94          JR      Z,L0028                 ; loop back while RAM is populated.
95
96          AND     H                       ; limit to nearest 1K segment.
97          LD      H,A                     ; place back in H.
98          LD      ($3C18),HL              ; set system variable RAMTOP.
99          LD      SP,HL                   ; initialize the stack pointer.
100
101 ; the Z80 instructions CALL, PUSH and POP can now be used.
102
103         LD      HL,L010D                ; prepare to copy the system variables
104                                         ; initial state from ROM.
105         JR      L003B                   ; skip past the fixed-position restart.
106
107 ; ----------------------
108 ; THE 'INTERRUPT' RESTART
109 ; ----------------------
110
111 L0038:  JP      L013A                   ; jump to somewhere more convenient.
112
113 ;-------------------------------------------------------------------------------
114 ;
115 ; MEMORY MAP
116 ;
117 ; $0000 +=====================================================+
118 ;       |                                                     |
119 ;       |                   ROM 8K                            |
120 ;       |                                     v $2300         |
121 ; $2000 +=====================================================+ - - - - - -
122 ;       |       copy of $2400                 |0|<  cassette  >|
123 ; $2400 +-------------------------------------+-+-------------+
124 ;       |       VIDEO MEMORY 768 bytes        |0| PAD 254 bytes| 1K RAM
125 ; $2800 +-------------------------------------+-+-------------+
126 ;       |       copy of $2c00                 ^ $2700         |
127 ; $2C00 +-----------------------------------------------------+
128 ;       |       CHARACTER SET - Write-Only                    | 1K RAM
129 ; $3000 +-----------------------------------------------------+
130 ;       |       copy of $3c00                                 |
131 ; $3400 +-----------------------------------------------------+
132 ;       |       copy of $3c00                                 |
133 ; $3800 +-----------------------------------------------------+
134 ;       |       copy of $3c00                                 |
135 ; $3C00 +-------+-------------------------------------------+
136 ;       |SYSVARS| DICT {12} DATA STACK ->         <- RET STACK | 1K RAM
137 ; $4000 +=======+=============================================+ - - - - - -
138 ;       |                                                     |
139 ;                       48K AVAILABLE FOR EXPANSION.
140 ;       |                                                     |
141 ; $FFFF +=====================================================+
142 ;
143 ; The Ace had an 8K ROM and was sold with 3K of RAM each byte of which had
144 ; at least two addresses and sometimes four addresses so the mapping of the
145 ; 3K of RAM was as above.
146 ; The 768 bytes of video memory is accessed by the ROM using addresses
```

```
147   ; $2400 - $26FF. This gives priority to the video circuitry which also needs
148   ; this information to build the TV picture. The byte at $2700 is set to zero
149   ; so that it is easy for the ROM to detect when it is at the end of the screen.
150   ; The 254 bytes remaining are the PAD - the workspace used by FORTH.
151   ; This same area is used by the tape recorder routines to assemble the tape
152   ; header information but since, for accurate tape timing, the FORTH ROM needs
153   ; priority over the video circuitry, then the ROM uses addresses $2301 - $23FF.
154   ;
155   ; Similarly the Character Set is written to by the ROM (and User) at the 1K
156   ; section starting at $2C00. The video circuitry accesses this using addresses
157   ; $2800 - $2BFF to build the TV picture. It is not possible for the ROM or User
158   ; to read back the information from either address so this precludes the saving
159   ; of character sets and writing a driver for a device like the ZX Printer.
160   ;
161   ; The final 1K or RAM has four addresses although it is normal to use addresses
162   ; $3C00 - $3FFF. The first sixty three bytes are the System Variables which
163   ; hold information like the number BASE and CONTEXT, and even the plotting
164   ; coordinates should the user wish to develop a word like DRAW to draw lines.
165   ;
166   ; Then comes the User Dictionary, the first word of which is "FORTH" which links
167   ; to the Dictionary in ROM. Next a gap of 12 bytes to allow for Data Stack
168   ; underflow and then the Data Stack itself which grows upwards.
169   ; At the opposite end of free memory is the Return Stack (machine stack) which
170   ; grows downwards.
171
172   ; -------------------------------------
173   ; THE 'INITIALIZATION ROUTINE' Part 3.
174   ; -------------------------------------
175
176   L003B:  LD      DE,$3C24                ; destination system variable L_HALF
177           LD      BC,$002D                ; number of bytes.
178           LDIR                            ; copy initial state from ROM to RAM.
179
180           LD      IX,$3C00                ; set IX to index the system variables.
181           LD      IY,L04C8                ; set IY to the SLOW return address.
182
183   L004B:  CALL    L0A24                   ; routine CLS.
184
185           XOR     A                       ; clear accumulator.
186
187           LD      ($2700),A               ; make location after screen zero.
188
189   ; There are 128 bit-mapped 8x8 characters.
190   ; Define the 8 Battenberg graphics ($10 to $17) from low byte of address.
191   ; This routine also sets the other characters $00 to $0F and $18 to $1F
192   ; to copies of this range. The inverse form of character $17 is used as the
193   ; normal cursor - character $97.
194
195   L0052:  LD      HL,$2C00                ; point to the start of the 1K write-
196                                           ; only Character Set RAM.
197
198   L0055:  LD      A,L                     ; set A to low byte of address
199           AND     $BF                     ; AND %10111111
200           RRCA                            ; rotate
201           RRCA                            ; three times
202           RRCA                            ; to test bit 2
203           JR      NC,L005F                ; forward if not set.
204
205           RRCA                            ; else rotate
206           RRCA                            ; twice more.
207
208   L005F:  RRCA                            ; set carry from bit (3) or (6)
209
210           LD      B,A
211
212           SBC     A,A                     ; $00 or $FF
213           RR      B
214           LD      B,A
215           SBC     A,A
216           XOR     B
217           AND     $F0
218           XOR     B
219           LD      (HL),A                  ; insert the byte.
```

```
220             INC     L                       ; increment low byte of address
221             JR      NZ,L0055                ; loop back until the first 256 bytes
222                                             ; have been filled with 32 repeating
223                                             ; characters.
224
225     ; Now copy the bit patterns at the end of this ROM to the last 768 bytes of
226     ; the Character RAM, filling in some blank bytes omitted to save ROM space.
227     ; This process starts at high memory and works downwards.
228
229     L006E:  LD      DE,$2FFF                ; top of destination.
230             LD      HL,L1FFB                ; end of copyright character.
231             LD      BC,$0008                ; 8 characters
232
233             LDDR                            ; copy the  Â©  character
234
235             EX      DE,HL                   ; switch pointers.
236
237             LD      A,$5F                   ; set character counter to ninety five.
238                                             ; i.e. %0101 1111
239                                             ; bit 5 shows which 32-character sector
240                                             ; we are in.
241
242     ; enter a loop for the remaining characters supplying zero bytes as required.
243
244     L007C:  LD      C,$07                   ; set byte counter to seven.
245
246             BIT     5,A                     ; test bit 5 of the counter.
247             JR      Z,L0085                 ; forward if not in middle section
248                                             ; which includes "[A-Z]"
249
250             LD      (HL),B                  ; else insert a zero byte.
251             DEC     HL                      ; decrement the destination address.
252             DEC     C                       ; and the byte counter.
253
254     L0085:  EX      DE,HL                   ; switch pointers.
255
256             LDDR                            ; copy the 5 or 6 characters.
257
258             EX      DE,HL                   ; switch pointers.
259
260             LD      (HL),B                  ; always insert the blank top byte.
261             DEC     HL                      ; decrement the address.
262
263             DEC     A                       ; decrement the character counter.
264
265             JR      NZ,L007C                ; back for all 95 characters.
266
267         JP  X009B
268         NOP
269
270     ; ---------------
271     ; THE 'QUIT' WORD
272     ; ---------------
273     ; (  --  )
274     ; Clears return stack, empties input buffer and returns control to the
275     ; keyboard.
276
277     L0092:  DEFM    "QUI"                   ; 'name field'
278             DEFB    'T' + $80
279
280     L0096:  DEFW    $0000                   ; 'link field' - end of linked list.
281
282     L0098:  DEFB    $04                     ; 'name length field'
283
284     L0099:  DEFW    L009B                   ; 'code field'
285                                             ; address of machine code for routine.
286
287     ; ---
288
289     L009B:  LD      SP,($3C18)              ; set stack-pointer to RAMTOP.
290
291             EI                              ; Enable Interrupts.
292
```

```
293           JP      L04F2                   ; jump forward to the main execution
294                                           ; loop.
295
296   ; ----------------
297   ; THE 'ABORT' WORD
298   ; ----------------
299   ; Clears the data and return stacks, deletes any incomplete definition
300   ; left in the dictionary, prints 'ERROR' and the byte from address $3C3D
301   ; if the byte is non-negative, empties the input buffer, and returns
302   ; control to the keyboard.
303
304
305   L00A3:  DEFM    "ABOR"                  ; 'name field'
306           DEFB    'T' + $80
307
308           DEFW    L0098                   ; 'link field' to previous word QUIT.
309
310   L00AA:  DEFB    $05                     ; 'name length field'
311
312   L00AB:  DEFW    L00AD                   ; 'code field'
313
314   ; ---
315
316   ; -> also continuation of the error restart.
317
318   L00AD:  PUSH    IY                      ; preserve current IY value slow/fast.
319
320           LD      IY,L04B9                ; set IY to FAST
321                                           ; now empty the data stack
322           LD      HL,($3C37)              ; STKBOT
323           LD      ($3C3B),HL              ; SPARE
324           LD      HL,$3C3E                ; address FLAGS
325           LD      A,(HL)                  ; fetch status from FLAGS.
326           AND     $B3                     ; AND %10110011
327                                           ; reset bit 2 - show definition complete
328                                           ; reset bit 3 - output to screen.
329                                           ; reset bit 6 - show in interpreter mode
330           BIT     2,(HL)                  ; was there an incomplete definition ?
331           LD      (HL),A                  ; update FLAGS
332           JR      Z,L00DE                 ; forward if no incomplete word.
333
334   L00C4:  CALL    L04B9                   ; do forth
335
336           DEFW    L0490                   ; dict          address of sv DICT
337           DEFW    L08B3                   ; @             value of sv DICT (d).
338           DEFW    L104B                   ; stk_data      d.          length field
339           DEFB    $05                     ; five          d, 5.
340           DEFW    L0DD2                   ; +             d+5.        code field
341           DEFW    L086B                   ; dup           d+5, d+5.
342           DEFW    L1610                   ; prvcur        d+5.
343           DEFW    L15B5                   ; namefield     n.
344           DEFW    L1011                   ; stackwrd      n.
345           DEFW    $3C37                   ; (stkbot)      n, stkbot.
346           DEFW    L08C1                   ; !             .
347           DEFW    L1A0E                   ; end-forth.    .
348
349   ; at this stage the system variable STKBOT holds the address of the
350   ; obsolete name field and the system variable CURRENT points to the
351   ; address of the previous complete word - obtained from the old link field.
352
353   L00DE:  BIT     7,(IX+$3D)              ; test ERR_NO for normal value 255.
354           JR      NZ,L00FF                ; set-min then main-loop if OK.
355
356           CALL    L1808                   ; else pr-inline
357
358   ; ---
359
360   L00E7:  DEFM    "ERRO"                  ; the message "ERROR" with the last
361           DEFB    'R' + $80               ; character inverted.
362
363   ; ---
364
365   L00EC:  CALL    L04B9                   ; forth
```

```
366
367             DEFW    L1011                   ; stack next word
368             DEFW    $3C3D                   ; -> system variable ERR_NO
369             DEFW    L0896                   ; C@            - fetch content byte
370             DEFW    L09B3                   ; .             - print it
371             DEFW    L0A95                   ; CR
372             DEFW    L1A0E                   ; end-forth.
373
374             LD      (IX+$3D),$FF            ; set ERR_NO to 'No Error'
375
376     L00FF:  LD      HL,($3C37)              ; fetch STKBOT
377             LD      BC,$000C                ; allow twelve bytes for stack underflow
378             ADD     HL,BC                   ; add the extra
379             LD      ($3C3B),HL              ; set SPARE
380             POP     IY                      ; restore previous state of IY
381
382             JR      L009B                   ; rejoin main loop
383
384     ; -------------------------
385     ; THE 'DEFAULT ENVIRONMENT'
386     ; -------------------------
387     ; This is the default environment that is copied from ROM to RAM as part of
388     ; the initialization process. This also contains the FORTH word FORTH definition
389
390     L010D:  DEFW    $26E0                   ; L_HALF
391
392             DEFB    $00                     ; KEYCOD
393             DEFB    $00                     ; KEYCNT copy the 32 bytes.
394             DEFB    $00                     ; STATIN
395             DEFW    $0000                   ; EXWRCH
396             DEFB    $00                     ; FRAMES
397             DEFB    $00                     ; FRAMES
398             DEFB    $00                     ; FRAMES
399             DEFB    $00                     ; FRAMES
400             DEFB    $00                     ; XCOORD
401             DEFB    $00                     ; YCOORD
402             DEFW    $3C4C                   ; CURRENT
403             DEFW    $3C4C                   ; CONTEXT
404             DEFW    $3C4F                   ; VOCLNK
405             DEFW    $3C51                   ; STKBOT
406             DEFW    $3C45                   ; DICT
407             DEFW    $3C5D                   ; SPARE
408             DEFB    $FF                     ; ERR_NO
409             DEFB    $00                     ; FLAGS
410             DEFB    $0A                     ; BASE
411
412     ; FORTH
413
414             DEFM    "FORT"                  ; The 'name field'
415             DEFB    'H' + $80               ; FORTH
416
417
418             DEFW    $0000                   ; length field - filled when next word
419                                             ; is defined.
420             DEFW    L1FFF                   ; link field copied to $3C49.
421             DEFB    $05                     ; name length field
422             DEFW    L11B5                   ; code field
423             DEFW    $3C49                   ; address of parameters
424             DEFB    $00                     ; VOCLNK                        [$3C4F]
425             DEFB    $00                     ; - link to next vocabulary.
426             DEFB    $00                     ; last byte to be copied.    to [$3C51]
427
428     ; ----------------------------------------------
429     ; THE 'CONTINUATION OF THE Z80 INTERRUPT' ROUTINE
430     ; ----------------------------------------------
431     ; The destination of the jump at $0038.
432     ; Begin by saving both accumulators and the 3 main registers.
433
434     L013A:  PUSH    AF                      ; preserve both accumulators
435             EX      AF,AF'                  ;
436             PUSH    AF                      ;
437
438             PUSH    BC                      ; and main registers.
```

```
439           PUSH    DE                      ;
440           PUSH    HL                      ;
441
442   ; Now wait for 62 * 12 clock cycles. ( To avoid flicker perhaps? ).
443
444           LD      B,$3E                   ; delay counter.
445
446   L0142:  DJNZ    L0142                   ; self loop for delay
447
448   ; Increment the 4-byte frames counter for use as a system clock.
449
450           LD      HL,$3C2B                ; FRAMES1
451
452   L0147:  INC     (HL)                    ; increment timer.
453           INC     HL                      ; next significant byte of four.
454           JR      Z,L0147                 ; loop back if the value wrapped back
455                                           ; to zero.
456
457   ; Note. as manual points out, there is no actual check on this and if
458   ; you leave your Ace switched on for 2.75 years it will advance to the
459   ; following system variables although it takes several millennia to advance
460   ; through the screen coordinates.
461
462   ; Now read the keyboard and if no new key then exit after restoring the
463   ; preserved registers.
464
465           CALL    L0310                   ; routine KEYBOARD.
466
467           LD      HL,$3C28                ; address system variable STATIN
468
469           BIT     0,(HL)                  ; new key?
470           JR      Z,L0176                 ; forward if not to RESTORE/EXIT
471
472           AND     A                       ; zero key code ?
473           JR      Z,L0176                 ; forward if so to EXIT.
474
475           CP      $20                     ; compare to SPACE
476           JR      C,L0170                 ; forward if less as an Editing Key.
477
478           BIT     1,(HL)                  ; CAPS shift?
479           CALL    NZ,L0807                ; routine TO_UPPER
480
481           BIT     2,(HL)                  ; GRAPHICS mode?
482           JR      Z,L0167                 ; skip forward if not
483
484           AND     $9F                     ; convert to one of 8 mosaic characters
485
486   L0167:  BIT     3,(HL)                  ; INVERSE mode?
487           JR      Z,L016D                 ; forward if not.
488
489           OR      $80                     ; set bit 7 to make character inverse.
490
491   L016D:  CALL    L0196                   ; routine pr_buffer
492
493   L0170:  CALL    L01E6                   ; routine EDIT_KEY
494           CALL    L0282                   ; routine pr_cursor
495
496   ; Before exiting restore the preserved registers.
497
498   L0176:  POP     HL                      ;
499           POP     DE                      ;
500           POP     BC                      ;
501           POP     AF                      ;
502           EX      AF,AF'                  ;
503           POP     AF                      ;
504
505           EI                              ; Enable Interrupts
506
507           RET                             ; return.
508
509   ; ----------------------------------
510   ; THE 'PRINT to LOWER SCREEN' ROUTINE
511   ; ----------------------------------
```

```
512
513   L017E:  CP      $0D                     ; carriage return?
514           JR      NZ,L0196                ; forward if not
515
516   ; a carriage return to input buffer i.e. lower screen memory.
517
518           LD      HL,$2700                ; set pointer to location after the
519                                           ; input buffer.
520
521           LD      ($3C22),HL              ; set ENDBUF - end of logical line
522           LD      ($3C20),HL              ; set the CURSOR
523
524           XOR     A                       ; clear A
525
526           CALL    L0198                   ; print character zero.
527
528           LD      HL,$26E0                ; left hand position of bottom line.
529           LD      ($3C1E),HL              ; set INSCRN to this position.
530           RET                             ; return.
531
532   ; ---------------------------------------
533   ; THE 'PRINT CHARACTER TO BUFFER' ROUTINE
534   ; ---------------------------------------
535
536   L0196:  AND     A                       ; check for zero character
537           RET     Z                       ; return if so.
538
539   ; => also called from previous routine only to print a zero skipping above test.
540
541   L0198:  EX      AF,AF'                  ; preserve the output character.
542
543           LD      HL,($3C22)              ; fetch ENDBUF end of logical line
544           LD      A,(HL)                  ; fetch character from position
545           AND     A                       ; is it zero ?
546           JR      Z,L01A6                 ; skip forward if so.
547
548   ; else lower screen scrolling is required.
549
550           LD      DE,$D900                ; $0000 - $2700
551           ADD     HL,DE                   ; test if position is within video RAM
552           JR      NC,L01CE                ; forward if < $26FF
553
554   ; now check that the limit of 22 lines in lower screen is not exceeded.
555
556   L01A6:  LD      DE,($3C24)              ; fetch start of buffer from L_HALF
557           LD      HL,$DBA0                ; $0000 - $2460
558           ADD     HL,DE                   ;
559           JR      NC,L01E4                ; forward to exit if buffer full.
560
561
562           LD      HL,($3C1C)              ; fetch position SCRPOS for upper screen
563           LD      BC,$0020                ; allow an extra 32 characters - 1 line.
564           ADD     HL,BC                   ;
565           SBC     HL,DE                   ; subtract the start of input buffer
566           PUSH    DE                      ; and save the L_HALF value
567
568           CALL    NC,L0421                ; routine to scroll upper display.
569
570           CALL    L02B0                   ; find zerobyte loc in HL
571
572           POP     DE                      ; retrieve the L_HALF value
573
574           CALL    L042F                   ; routine scroll and blank
575
576   ; The four system variables INSCRN, CURSOR, ENDBUF and L_HALF are each
577   ; reduced by 32 bytes a screen line.
578
579           LD      HL,$3C1E                ; address INSCRN the left-hand location
580                                           ; of the current input line.
581
582           LD      B,$04                   ; four system variables to update
583
584   L01C9:  CALL    L0443                   ; routine SCR-PTRS
```

```
585
586             DJNZ    L01C9                   ; repeat for all four pointers.
587
588     ; ok to print
589
590     L01CE:  CALL    L0302                   ; routine find characters to EOL.
591
592             LD      D,H                     ; HL is end of line
593             LD      E,L                     ; transfer to DE register.
594             INC     HL                      ; increment
595             LD      ($3C22),HL              ; update ENDBUF
596             DEC     HL                      ; decrement
597             DEC     HL                      ; so HL = DE -1
598
599             JR      Z,L01DD                 ; skip if BC zero.
600
601             LDDR                            ; else move the characters.
602
603     L01DD:  EX      AF,AF'                  ; restore the output character.
604             LD      (DE),A                  ; insert at screen position.
605                                             ; (a zero if CR lower)
606             INC     DE                      ; next character position
607             LD      ($3C20),DE              ; update CURSOR
608
609     L01E4:  XOR     A                       ; ?
610             RET                             ; return.
611
612     ; -------------------------
613     ; THE 'EDIT KEY' SUBROUTINE
614     ; -------------------------
615
616     L01E6:  LD      HL,L01F0                ; address the EDIT KEYS table.
617
618             LD      D,$00                   ; prepare to index by one byte.
619             LD      E,A                     ; character code to E.
620             ADD     HL,DE                   ; index into the table.
621
622             LD      E,(HL)                  ; pick up required offset to the
623                                             ; handling routine.
624
625             ADD     HL,DE                   ; add to the current address.
626             JP      (HL)                    ; exit via the routine.
627
628     ; --------------------
629     ; THE 'EDIT KEYS' TABLE
630     ; --------------------
631
632     L01F0:  DEFB    $20             ; L0210         $00     - RET
633     L01F1:  DEFB    $13             ; L0204         $01     - LEFT
634     L01F2:  DEFB    $0C             ; L01FE         $02     - CAPS
635     L01F3:  DEFB    $1E             ; L0211         $03     - RIGHT
636     L01F4:  DEFB    $0A             ; L01FE         $04     - GRAPH
637     L01F5:  DEFB    $37             ; L022C         $05     - DEL
638     L01F6:  DEFB    $1A             ; L0210         $06     - RET
639     L01F7:  DEFB    $50             ; L0247         $07     - UP
640     L01F8:  DEFB    $06             ; L01FE         $08     - INV
641     L01F9:  DEFB    $9C             ; L0295         $09     - DOWN
642     L01FA:  DEFB    $C9             ; L02C3         $0A     - DEL LINE
643     L01FB:  DEFB    $15             ; L0210         $0B     - RET
644     L01FC:  DEFB    $14             ; L0210         $0C     - RET
645     L01FD:  DEFB    $D3             ; L02D0         $0D     - KEY-ENTER
646
647     ; ------------------------------
648     ; THE 'TOGGLE STATUS BIT' ROUTINE
649     ; ------------------------------
650     ; The keycodes have been cleverly mapped to individual bits of the STATIN
651     ; system variable so this simple routine maintains all three status bits.
652     ; KEY '2' - CAPS SHIFT, '4' - GRAPHICS, '8' - INVERSE VIDEO.
653
654     L01FE:  LD      HL,$3C28                ; system variable STATIN
655             XOR     (HL)                    ; toggle the single relevant bit.
656             LD      (HL),A                  ; put back.
657             RET                             ; return.
```

```
658
659   ; ---------------------------
660   ; THE 'CURSOR LEFT' SUBROUTINE
661   ; ---------------------------
662   ; this subroutine moves the cursor to the left unless the character at that
663   ; position is zero.
664
665   L0204:  LD      HL,($3C20)              ; fetch CURSOR.
666           DEC     HL                      ; decrement value.
667           LD      A,(HL)                  ; fetch character at new position.
668           AND     A                       ; test for zero. (cr)
669           RET     Z                       ; return if so.                  >>
670
671           LD      ($3C20),HL              ; else update CURSOR
672           INC     HL                      ; step back
673           LD      (HL),A                  ; and put character that was at new
674                                           ; cursor position where cursor is now.
675
676   L0210:  RET                             ; return.
677
678   ; Note. various unallocated keys in the EDIT KEYS table point to the
679   ; above RET instruction.
680
681   ; ----------------------------
682   ; THE 'CURSOR RIGHT' SUBROUTINE
683   ; ----------------------------
684
685   L0211:  LD      HL,($3C20)              ; fetch CURSOR position
686           INC     HL                      ; and increment it.
687
688           LD      DE,($3C22)              ; fetch ENDBUF - end of current line.
689           AND     A                       ; prepare to subtract.
690           SBC     HL,DE                   ; test
691           RET     Z                       ; return if zero - CURSOR is at ENDBUF
692
693           ADD     HL,DE                   ; else reform the pointers.
694           LD      ($3C20),HL              ; update CURSOR
695           LD      A,(HL)                  ; fetch character at new position.
696           DEC     HL                      ; decrement
697           LD      (HL),A                  ; and insert where cursor was.
698           RET                             ; ret.
699
700   ; ---------------------------
701   ; THE 'DELETE CURSOR' ROUTINE
702   ; ---------------------------
703   ; Moves cursor position to right and then continues into DEL-CHAR
704
705   L0225:  LD      HL,($3C20)              ; fetch CURSOR
706           INC     HL                      ; increment position.
707           LD      ($3C20),HL              ; update CURSOR
708
709
710   ; ----------------------------
711   ; THE 'DELETE CHARACTER' ROUTINE
712   ; ----------------------------
713
714   L022C:  CALL    L0302                   ; routine finds characters to EOL.
715
716           LD      H,D                     ; transfer CURSOR position DE to HL.
717           LD      L,E                     ;
718           DEC     DE                      ; decrement DE
719           LD      A,(DE)                  ; fetch character to left of original
720                                           ; cursor.
721           AND     A                       ; test for zero.
722           RET     Z                       ; return if so.                >>
723
724           LD      ($3C20),DE              ; else update CURSOR
725           LD      A,B                     ; check for count of characters
726           OR      C                       ; being zero
727           JR      Z,L023F                 ; skip if so.
728
729   L023D:  LDIR                            ; else shift characters to left.
730
```

```
L023F:  DEC     HL                      ; decrement HL so that points to end -
                                        ; last position on the logical line.
        LD      (HL),$20                ; insert a space.
        LD      ($3C22),HL              ; set ENDBUF
        INC     C                       ; reset zero flag??
        RET                             ; return.

; ----------------------
; THE 'CURSOR UP' ROUTINE
; ----------------------
; When the cursor is moved up while editing a multi-line word definition,
; then the cursor is first moved to the left of the screen abutting the
; character zeros at the leftmost position.
; These zero characters appear as spaces but mark the beginning of each logical
; line. A logical line may, for instance if it contains a text item, extend over
; several physical screen lines.

L0247:  CALL    L0204                   ; routine CURSOR-LEFT
        JR      Z,L0254                 ; skip forward if not possible.

; else move left by thirty two positions. This may achieve a vertical move if
; attempted when a word is first being entered. Alternatively if one of the
; calls to cursor left fails having encountered a zero, then all subsequent
; calls will fail. The routine will return with the cursor adjacent to the zero.

        LD      B,$1F                   ; count 31 decimal
L024E:  CALL    L0204                   ; move cursor left thirty one times.
        DJNZ    L024E                   ; makes thirty two moves counting first

        RET                             ; return.

; ---

L0254:  LD      HL,($3C1E)              ; fetch INSCRN start of current line.
        LD      DE,($3C24)              ; fetch L_HALF start of buffer.
        AND     A                       ; reset carry for
        SBC     HL,DE                   ; true subtraction.
        RET     Z                       ; return if at beginning of input buffer

        CALL    L0225                   ; routine DEL-CURSOR

        LD      HL,($3C1E)              ; fetch INSCRN leftmost location of
                                        ; current line.
        LD      DE,$FFE0                ; make DE minus thirty two.
        XOR     A                       ; clear accumulator to zero.

L0269:  ADD     HL,DE                   ; subtract 32
        CP      (HL)                    ; compare contents to zero
                                        ; ( i.e. prev (cr) or buffer start?)
        JR      NZ,L0269                ; loop back until HL holds zero.

        LD      ($3C1E),HL              ; update INSCRN

        CALL    L02F4                   ; find endbuf

        LD      ($3C20),HL              ; set CURSOR

; ----------
; PR_CURSOR
; ----------

L0276:  LD      A,$A0                   ; inverse space - so solid square

        CALL    L017E                   ; routine PR_LOWER

        LD      HL,($3C20)              ; CURSOR
        DEC     HL
        LD      ($3C20),HL              ; CURSOR

; -> from interrupt
L0282:  LD      HL,($3C20)              ; CURSOR

        LD      A,($3C28)               ; STATIN
```

```
804             RRA                             ; ignore bit 0
805             LD      (HL),$97                ; pixel cursor.
806             RRA                             ; test bit 1 - CAPS
807             JR      NC,L0290                ; forward if no CAPS SHIFT
808
809             LD      (HL),$C3                ; inverse [C] cursor.
810
811     L0290:  RRA                             ; test bit 2 - GRAPHICS.
812             RET     NC                      ; return if not
813
814     L0292:  LD      (HL),$C7                ; inverse [G] cursor.
815             RET                             ; return
816
817     ; -------------------------
818     ; THE 'CURSOR DOWN' ROUTINE
819     ; -------------------------
820
821
822     L0295:  CALL    L0211                   ; routine CURSOR RIGHT
823             JR      Z,L02A2                 ; forward if not possible.
824
825             LD      B,$1F                   ; set counter to thirty one.
826
827     L029C:  CALL    L0211                   ; routine CURSOR RIGHT
828             DJNZ    L029C                   ; thirty two moves altogether.
829             RET                             ; return.
830
831     ; ---
832
833     L02A2:  CALL    L02B0                   ; find zerobyte
834             RET     PO                      ; return if    found
835
836             PUSH    HL                      ; save position
837             CALL    L0225                   ; routine DEL-CURSOR
838             POP     HL                      ; retrieve position.
839             CALL    L02ED                   ; set logical line
840             JR      L0276                   ; back to exit via pr_cursor.
841
842     ; ---
843     ; find zerobyte
844     ; ---
845     ; -> called 5 times
846
847     L02B0:  LD      HL,$2700                ; this location is always zero.
848                                             ; the byte following video RAM.
849             LD      DE,($3C1E)              ; INSCRN        e.g. $26E0
850
851             AND     A                       ; prepare for true subtraction
852
853             SBC     HL,DE                   ; subtract to give number of chars
854
855             LD      B,H                     ; transfer count to
856             LD      C,L                     ; the BC register pair.
857
858             EX      DE,HL                   ; transfer INSCR value to HL.
859
860             INC     HL                      ; start next location
861             XOR     A                       ; search for a zero character.
862
863             CPIR                            ; at most BC locations.
864                                             ; sets P/O flag if BC!=0
865
866             DEC     HL                      ; step back to last non-zero
867             RET                             ; return.
868
869     ; -------------------------
870     ; THE 'DELETE LINE' ROUTINE
871     ; -------------------------
872     ; CHR$ 10
873
874     L02C3:  LD      HL,($3C22)              ; ENDBUF
875             DEC     HL                      ;
876             LD      ($3C20),HL              ; CURSOR
```

```
877
878   L02CA:  CALL    L022C                   ; KEY-DEL
879           JR      NZ,L02CA                ; repeat
880
881           RET                             ; return.
882
883   ; -------------------------
884   ; THE 'KEY-ENTER' SUBROUTINE
885   ; -------------------------
886
887   L02D0:  LD      HL,$3C28                ; STATIN
888           SET     5,(HL)                  ; signal new key.
889           RES     0,(HL)                  ; reset new key flag
890           RET                             ; return.
891
892
893   ; ------------------------
894   ; THE 'SET BUFFER' ROUTINE
895   ; ------------------------
896   ; called by LIST, QUERY
897
898   L02D8:  LD      HL,$2700                ; one past end of screen.
899           LD      DE,($3C24)              ; fetch start of buffer from L_HALF
900
901           CALL    L07FA                   ; routine SPACE_FILL
902
903           LD      HL,$26E0                ; first location of bottom line.
904           LD      ($3C24),HL              ; set L_HALF
905
906           LD      (HL),$00                ; insert a ZERO.
907
908   ; -> called by retype
909   L02EA:  LD      HL,($3C24)              ; fetch L_HALF
910
911   ; -> from cursor down
912   L02ED:  LD      ($3C1E),HL              ; set INSCRN
913           INC     HL                      ; step past the zero
914           LD      ($3C20),HL              ; set CURSOR
915
916   ; => from cursor up.
917   L02F4:  CALL    L02B0                   ; find zerobyte
918
919           LD      A,$20                   ; prepare a space
920
921   L02F9:  DEC     HL                      ; move to the left.
922           CP      (HL)                    ; compare to space.
923           JR      Z,L02F9                 ; back while spaces exist.
924
925           INC     HL                      ; point to last space encountered.
926           LD      ($3C22),HL              ; set ENDBUF - end of logical line.
927           RET                             ; return.
928
929   ; ---------------------------------
930   ; THE 'COUNT TO END OF LINE' ROUTINE
931   ; ---------------------------------
932   ; Find the number of characters to the end of the logical line.
933
934   L0302:  LD      HL,($3C22)              ; system variable ENDBUF
935           LD      DE,($3C20)              ; system variable CURSOR
936           AND     A                       ; prepare to subtract.
937           SBC     HL,DE                   ; subtract to give character places
938           LD      B,H                     ; transfer result
939           LD      C,L                     ; to the BC register pair.
940           ADD     HL,DE                   ; reform the pointers.
941
942           RET                             ; return with zero flag set if cursor
943                                           ; at EOL.
944
945   ; ----------------------
946   ; THE 'KEYBOARD' ROUTINE
947   ; ----------------------
948
949   L0310:  CALL    L0336                   ; routine KEY_SCAN
```

```
950
951             LD      B,A                     ; save key in B
952
953             LD      HL,($3C26)              ; load L with KEYCOD - last key pressed
954                                             ; load H with KEYCNT - debounce counter
955
956             XOR     L                       ; compare to previous key.
957             JR      Z,L0325                 ; forward if a match.
958
959             XOR     L                       ; reform original
960             JR      Z,L0320                 ; forward if zero - no key.
961
962             XOR     A                       ; else clear accumulator.
963
964             CP      L                       ; compare with last.
965             RET     NZ                      ; return if not zero.
966
967     L0320:  LD      L,B                     ; set L to original keycode
968             LD      H,$20                   ; set counter to thirty two.
969             JR      L0332                   ; forward to store values and exit
970                                             ; returning zero.
971
972     ; ---
973
974     ; Key is same as previously accepted key.
975     ; It repeats after two interrupts
976
977     L0325:  DEC     H                       ; decrement the counter.
978             LD      A,H                     ; fetch counter to A.
979             CP      $1E                     ; compare to thirty.
980             JR      Z,L0331                 ; forward if so to return key in A.
981
982             XOR     A                       ; clear accumulator.
983             CP      H                       ; is counter zero?
984             JR      NZ,L0332                ; forward if not to keep counting.
985
986             LD      H,$04                   ; else set counter to four.
987
988     L0331:  LD      A,L                     ; pick up previous key.
989
990     L0332:  LD      ($3C26),HL              ;  update KEYCOD/KEYCNT
991
992             RET                             ; return.
993
994     ;-------------------------------------------------------------------------
995     ;                           LOGICAL VIEW OF KEYBOARD
996     ;
997     ;         0     1     2     3     4 -Bits-  4     3     2     1     0
998     ; PORT                                                                PORT
999     ;
1000    ; F7FE  [ 1 ] [ 2 ] [ 3 ] [ 4 ] [ 5 ]  |  [ 6 ] [ 7 ] [ 8 ] [ 9 ] [ 0 ]   EFFE
1001    ;  ^                                    |                                   v
1002    ; FBFE  [ Q ] [ W ] [ E ] [ R ] [ T ]  |  [ Y ] [ U ] [ I ] [ O ] [ P ]   DFFE
1003    ;  ^                                    |                                   v
1004    ; FDFE  [ A ] [ S ] [ D ] [ F ] [ G ]  |  [ H ] [ J ] [ K ] [ L ] [ ENT ] BFFE
1005    ;  ^                                    |                                   v
1006    ; FEFE  [SHI] [SYM] [ Z ] [ X ] [ C ]  |  [ V ] [ B ] [ N ] [ M ] [ SPC ] 7FFE
1007    ;  ^            v                                                  ^        v
1008    ; Start         +------------>--------------------->-------------+        End
1009    ;
1010    ;
1011    ;-------------------------------------------------------------------------
1012
1013
1014    ; ---------------------------------
1015    ; THE 'KEYBOARD SCANNING' SUBROUTINE
1016    ; ---------------------------------
1017    ; This routine is called by the KEYBOARD routine 50 times a second and
1018    ; by the ACE FORTH 'INKEY' WORD.
1019    ; The above diagram shows the logical view of the Keyboard and PORTS.
1020    ; The physical view is similar except that the symbol shift key is to the
1021    ; left of the space key.
1022
```

```
1023
1024  L0336:  LD      BC,$FEFE                ; port address - B is also an 8 counter
1025
1026          IN      D,(C)                   ; read from port to D.
1027                                          ; when a key is pressed, the
1028                                          ; corresponding bit is reset.
1029
1030          LD      E,D                     ; save in E
1031
1032          SRL     D                       ; read the outer SHIFT key.
1033
1034          SBC     A,A                     ; $00 if SHIFT else $FF.
1035          AND     $D8                     ; $00 if SHIFT else $D8.
1036
1037          SRL     D                       ; read the symbol shift bit
1038          JR      C,L0347                 ; skip if not pressed.
1039
1040          LD      A,$28                   ; load A with 40 decimal.
1041
1042  L0347:  ADD     A,$57                   ; gives $7F SYM, $57 SHIFT, or $2F
1043
1044  ; Since 8 will be subtracted from the initial key value there are three
1045  ; distinct ranges 0 - 39, 40 - 79, 80 - 119.
1046
1047          LD      L,A                     ; save key range value in L
1048          LD      A,E                     ; fetch the original port reading.
1049          OR      $03                     ; cancel the two shift bits.
1050
1051          LD      E,$FF                   ; set a flag to detect multiple keys.
1052
1053  ; KEY_LINE the half-row loop.
1054
1055  L034F:  CPL                             ; complement bits
1056
1057          AND     $1F                     ; mask off the rightmost five key bits.
1058          LD      D,A                     ; save a copy in D.
1059          JR      Z,L0362                 ; forward if no keys pressed to do the
1060                                          ; next row.
1061
1062          LD      A,L                     ; else fetch the key value
1063          INC     E                       ; test E for $FF
1064          JR      NZ,L036B                ; forward if not now zero to quit
1065
1066  L0359:  SUB     $08                     ; subtract 8 from key value
1067
1068          SRL     D                       ; test next bit affecting zero and carry
1069
1070          JR      NC,L0359                ; loop back until the set bit is found.
1071
1072          LD      E,A                     ; transfer key value to E.
1073          JR      NZ,L036B                ; forward to abort if more than one key
1074                                          ; is pressed in the row.
1075
1076  L0362:  DEC     L                       ; decrement the key value for next row.
1077
1078          RLC     B                       ; rotate the 8 counter and port address
1079
1080          JR      NC,L036D                ; skip forward when all 8 rows have
1081                                          ; been read.
1082
1083          IN      A,(C)                   ; else read the next half-row.
1084          JR      L034F                   ; and back to KEY_LINE.
1085
1086  ; ---
1087  ; ABORTKEY
1088
1089  L036B:  LD      E,$FF                   ; signal invalid key.
1090
1091  ; the normal exit checks if E holds a key and not $FF.
1092
1093  L036D:  LD      A,E                     ; fetch possible key value.
1094          INC     A                       ; increment
1095          RET     Z                       ; return if was $FF as original.
```

```
1096
1097            LD      HL,L0376                ; else address KEY TABLE
1098            ADD     HL,DE                   ; index into table.
1099                                            ; (D is zero)
1100
1101            LD      A,(HL)                  ; pick up character.
1102
1103            RET                             ; return with translated character.
1104
1105
1106
1107    ; ---------------
1108    ; THE 'KEY TABLE'
1109    ; ---------------
1110
1111    ; ----------------------
1112    ; THE '40 UNSHIFTED KEYS'
1113    ; ----------------------
1114
1115    L0376:  DEFB    $76                     ; V - v
1116            DEFB    $68                     ; H - h
1117            DEFB    $79                     ; Y - y
1118            DEFB    $36                     ; 6 - 6
1119            DEFB    $35                     ; 5 - 5
1120            DEFB    $74                     ; T - t
1121            DEFB    $67                     ; G - g
1122            DEFB    $63                     ; C - c
1123            DEFB    $62                     ; B - b
1124            DEFB    $6A                     ; J - j
1125            DEFB    $75                     ; U - u
1126            DEFB    $37                     ; 7 - 7
1127            DEFB    $34                     ; 4 - 4
1128            DEFB    $72                     ; R - r
1129            DEFB    $66                     ; F - f
1130            DEFB    $78                     ; X - x
1131            DEFB    $6E                     ; N - n
1132            DEFB    $6B                     ; K - k
1133            DEFB    $69                     ; I - i
1134            DEFB    $38                     ; 8 - 8
1135            DEFB    $33                     ; 3 - 3
1136            DEFB    $65                     ; E - e
1137            DEFB    $64                     ; D - d
1138            DEFB    $7A                     ; Z - z
1139            DEFB    $6D                     ; M - m
1140            DEFB    $6C                     ; L - l
1141            DEFB    $6F                     ; O - o
1142            DEFB    $39                     ; 9 - 9
1143            DEFB    $32                     ; 2 - 2
1144            DEFB    $77                     ; W - w
1145            DEFB    $73                     ; S - s
1146            DEFB    $00                     ; SYMBOL
1147            DEFB    $20                     ; SPACE
1148            DEFB    $0D                     ; ENTER
1149            DEFB    $70                     ; P - p
1150            DEFB    $30                     ; 0 - 0
1151            DEFB    $31                     ; 1 - 1
1152            DEFB    $71                     ; Q - q
1153            DEFB    $61                     ; A - a
1154            DEFB    $00                     ; SHIFT
1155
1156    ; ---------------------
1157    ; THE '40 SHIFTED KEYS'
1158    ; ---------------------
1159
1160            DEFB    $56                     ; V - V
1161            DEFB    $48                     ; H - H
1162            DEFB    $59                     ; Y - Y
1163            DEFB    $07                     ; 6 - 7 KEY-UP
1164            DEFB    $01                     ; 5 - 1 KEY-LEFT
1165            DEFB    $54                     ;
1166            DEFB    $47
1167            DEFB    $43
1168            DEFB    $42
```

```
1169            DEFB    $4A
1170            DEFB    $55
1171            DEFB    $09                     ; 7 - 9 KEY-DOWN
1172            DEFB    $08                     ; 4 - 8 INV-VIDEO
1173            DEFB    $52
1174            DEFB    $46
1175            DEFB    $58
1176            DEFB    $4E
1177            DEFB    $4B
1178            DEFB    $49
1179            DEFB    $03                     ; 8 - 3 KEY-RIGHT
1180            DEFB    $33                     ; 3 - 3
1181            DEFB    $45
1182            DEFB    $44
1183            DEFB    $5A
1184            DEFB    $4D
1185            DEFB    $4C
1186            DEFB    $4F
1187            DEFB    $04                     ; 9 - 4 GRAPH
1188            DEFB    $02                     ; 2 - 2 CAPS LOCK
1189            DEFB    $57                     ; W - W
1190            DEFB    $53                     ; S - S
1191            DEFB    $00                     ; SYMB
1192            DEFB    $20                     ; SPACE
1193            DEFB    $0D                     ; ENTER
1194            DEFB    $50                     ; P - P
1195            DEFB    $05                     ; 0 - 5   DEL
1196            DEFB    $0A                     ; 1 - 0A  DEL_LINE
1197            DEFB    $51                     ; Q - Q
1198            DEFB    $41                     ; A - A
1199            DEFB    $00                     ; SHIFT
1200
1201    ; --------------------------
1202    ; THE '40 SYMBOL SHIFT KEYS'
1203    ; --------------------------
1204
1205            DEFB    $2F                     ; V - /
1206            DEFB    $5E                     ; H - ^
1207            DEFB    $5B                     ; Y - [
1208            DEFB    $26                     ; 6 - &
1209            DEFB    $25                     ; 5 - %
1210            DEFB    $3E                     ; T - >
1211            DEFB    $7D                     ;
1212            DEFB    $3F
1213            DEFB    $2A
1214            DEFB    $2D
1215            DEFB    $5D
1216            DEFB    $27
1217            DEFB    $24
1218            DEFB    $3C
1219            DEFB    $7B
1220            DEFB    $60
1221            DEFB    $2C
1222            DEFB    $2B
1223            DEFB    $7F
1224            DEFB    $28
1225            DEFB    $23
1226            DEFB    $1A         ; modified from E -E
1227            DEFB    $5C
1228            DEFB    $3A
1229            DEFB    $2E
1230            DEFB    $3D
1231            DEFB    $3B
1232            DEFB    $29
1233            DEFB    $40                     ; 2 - @
1234            DEFB    $19                     ; modified from W - W
1235            DEFB    $7C                     ; S
1236            DEFB    $00                     ; SYMB
1237            DEFB    $20                     ; SPACE
1238            DEFB    $0D                     ; ENTER
1239            DEFB    $22                     ; P - "
1240            DEFB    $5F                     ; 0 - _
1241            DEFB    $21                     ; 1 - !
```

```
1242            DEFB    $18                     ; modified from Q - Q
1243            DEFB    $7E                     ; A - ~
1244            DEFB    $00                     ; SHIFT
1245
1246    ; end of key tables
1247
1248
1249    ; --------------------------
1250    ; THE 'PRINT ROUTINE' Part 2.
1251    ; --------------------------
1252    ; If output is not directed into the input buffer then jump forward else
1253    ; call the routine to output to lower screen.
1254
1255    L03EE:  JR      Z,L03F5                 ; forward to main screen print.
1256
1257            CALL    L017E                   ; PR_LOWER
1258
1259            EXX                             ; restore main set
1260            RET                             ; return.               >>
1261
1262    ; the print output is not directed to the input buffer but first check that
1263    ; the user has not set up a vector to their own routine to print characters
1264    ; for instance to a printer.
1265
1266    L03F5:  LD      B,A                     ; save the character in the B register.
1267
1268            LD      HL,(EXWRCH)             ; fetch possible vector from EXWRCH
1269                                            ; (normally 0)
1270            LD      A,H                     ; test for
1271            OR      L                       ; the value zero.
1272            LD      A,B                     ; fetch the character back to A.
1273
1274            JR      Z,L03FF                 ; skip forward if no user-supplied
1275                                            ; routine.
1276
1277    L03FE:  JP      (HL)                    ; else jump to user-supplied routine
1278                                            ; which should finish with a JP (IY)
1279
1280    ; ---
1281    ; PRINTING TO UPPER SCREEN
1282    ; ---
1283
1284    L03FF:  LD      HL,($3C1C)              ; SCRPOS
1285            LD      DE,($3C24)              ; L_HALF
1286
1287            EX      DE,HL                   ; ??
1288
1289            SCF                             ; inclusive byte.
1290            SBC     HL,DE                   ; subtract screen position+1 from
1291                                            ; the start of input buffer.
1292            EX      DE,HL                   ; hl=scrpos
1293
1294            CALL    C,L0421                 ; if no room then scroll upper display
1295
1296            CP      $0D                     ; carriage return?
1297
1298            JR      Z,L0416                 ; skip forward if so.
1299
1300            LD      (HL),A                  ; else insert the character.
1301
1302            INC     HL                      ; point to next position.
1303            JR      L041C                   ; forward
1304
1305    ; ---
1306
1307    ; a carriage return
1308
1309    L0416:  INC     HL                      ; increment screen address.
1310            LD      A,L                     ; fetch low byte of address and mask.
1311            AND     $1F                     ; a zero result indicates a line skip.
1312            JR      NZ,L0416                ; loop until a new line of 32 columns
1313                                            ; is started.
1314
```

```
; both paths converge.

L041C:  LD      ($3C1C),HL              ; update SCRPOS

        EXX                             ; back to main set.

        RET                             ; return.

; -----------------------------------
; The 'UPPER DISPLAY SCROLLING' ROUTINE
; -----------------------------------

L0421:  PUSH    AF                      ; save character

        LD      HL,$3C1C                ; address the low order byte SCRPOS

        CALL    L0443                   ; routine cursor up
                                        ; i.e. SCRPOS = SCRPOS - 32

        POP     AF                      ; restore character

; now calculate the number of characters to scroll in the upper display.

        LD      HL,($3C24)              ; fetch L_HALF the start of input buffer
        LD      DE,$2420                ; second line in video display

;
; => scroll lower display enters here
L042F:  AND     A                       ; prepare for true subtraction.
        SBC     HL,DE                   ; find number of characters to scroll.

        LD      B,H                     ; result to BC
        LD      C,L

        LD      HL,$FFE0                ; set HL to -32d
        ADD     HL,DE                   ; now HL = DE -32d
        EX      DE,HL                   ; switch so DE = HL - 32

        LDIR                            ; scroll the lines up.

        LD      B,$20                   ; blank a line of 32 characters

L043D:  DEC     HL                      ; decrement screen address.
        LD      (HL),$20                ; insert a space character
        DJNZ    L043D                   ; and loop for all 32 characters

        RET                             ; return.

; --------------------------------
; THE 'SCREEN POINTERS' SUBROUTINE
; --------------------------------
;

L0443:  LD      A,(HL)                  ; fetch low byte of screen address
        SUB     $20                     ; subtract thirty two characters.
        LD      (HL),A                  ; and put back.

        INC     HL                      ; address high-order byte.
        JR      NC,L044B                ; forward if low byte did not wrap

        DEC     (HL)                    ; else decrement the high byte as the
                                        ; position has moved across a third of
                                        ; the display.

L044B:  INC     HL                      ; address following System Variable
        RET                             ; return.

; -----------------------------------
; THE 'INDEX SYSTEM VARIABLE' ROUTINE
; -----------------------------------
; This routine is used by words CONTEXT, CURRENT, BASE etc. to index and then
; stack a system variable associated with a FORTH word. See shortly.
;
```

```
; It is a bit overblown considering the eventual position of the System
; Variables and ld d,$3c; rst 10h; jp (iy) could have been used instead of
; the long-winded addition below.

L044D:  EX      DE,HL                   ; HL addresses the offset byte.
        LD      E,(HL)                  ; fetch to E register
;
        LD      D,$00                   ; prepare to add.
        LD      HL,$3C00                ; the address of start of SYSVARS
        ADD     HL,DE                   ; add the 8-bit offset
        EX      DE,HL                   ; location to DE.
        RST     10H                     ; push word DE

        JP      (IY)                    ; to 'next'.

; ---------------
; THE 'HERE' WORD
; ---------------
; ( -- address)
; Leaves the address of one past the end of the dictionary.

L0459:  DEFM    "HER"                   ; 'name field'
        DEFB    'E' + $80

        DEFW    L00AA                   ; 'link field'

L045F:  DEFB    $04                     ; 'name length field'

L0460:  DEFW    L0462                   ; 'code field'

; ---

L0462:  LD      DE,($3C37)              ; system variable STKBOT.
        RST     10H                     ; push word DE

        JP      (IY)                    ; to 'next'.

; ------------------
; THE 'CONTEXT' WORD
; ------------------
; (  -- 15411 )
; A system variable pointing to the context vocabulary.
; $3C33 CONTEXT

L0469:  DEFM    "CONTEX"                ; 'name field'
        DEFB    'T' + $80

        DEFW    L045F                   ; 'link field'

L0472:  DEFB    $07                     ; 'name length field'

L0473:  DEFW    L044D                   ; 'code field'

; ---

L0475:  DEFB    $33                     ; low byte of system variable.

; ------------------
; THE 'CURRENT' WORD
; ------------------
; (  -- 15409 )
; A system variable pointing to the current vocabulary.
; $3C31 CURRENT

L0476:  DEFM    "CURREN"                ; 'name field'
        DEFB    'T' + $80

        DEFW    L0472                   ; 'link field'

L047F:  DEFB    $07                     ; 'name length field'

L0480:  DEFW    L044D                   ; 'code field'
```

```
; ---

L0482:  DEFB    $31                     ; a single parameter low-byte of $3C31.

; ---------------
; THE 'BASE' WORD
; ---------------
; ( -- 15423)
; A one-byte variable containing the system number base.
; $3C3F BASE

L0483:  DEFM    "BAS"                   ; 'name field'
        DEFB    'E' + $80

        DEFW    L047F                   ; 'link field'

L0489:  DEFB    $04                     ; 'name length field'

L048A:  DEFW    L044D                   ; 'code field'

; ---

L048C:  DEFB    $3F                     ; low-byte of system variable BASE

; ---

; These two Internal Words are used to stack the value of FLAGS and DICT.

; -------------------------
; The 'flags' Internal Word
; -------------------------

L048D:  DEFW    L044D                   ; headerless 'code field'

; ---

L048F:  DEFB    $3E                     ; low-order byte of FLAGS $3C3E

; -------------------------
; The 'dict' Internal Word
; -------------------------

L0490:  DEFW    L044D                   ; headerless 'code field'

; ---

L0492:  DEFB    $39                     ; low-order byte of DICT $3C39


; --------------
; THE 'PAD' WORD
; --------------
; (  -- 9985 )
; Stacks the address of the 254-byte workpad.
; On most FORTH systems the PAD floats about in memory but on the Ace it is
; fixed in location and size. Its definition is simply a constant.

l0493   DEFM    "PA"                    ; 'name field'
        DEFB    'D' + $80

        DEFW    L0489                   ; 'link field'

L0498:  DEFB    $03                     ; 'name length field'

L0499:  DEFW    L0FF5                   ; 'code field' - stack word

; ---

L049B:  DEFW    $2701                   ; parameter is 9985 decimal -
                                        ; work pad address

; ------------
; THE ';' WORD
```

```
; ------------
; Terminates colon, DEFINER and COMPILER definitions.

L049D:  DEFB    ';' + $80               ; 'name field'

        DEFW    L0498                   ; 'link field'

L04A0:  DEFB    $41                     ; length 1 + $40 (immediate word)

L04A1:  DEFW    L1108                   ; 'code field' - compile

; ---

L04A3:  DEFW    L04B6                   ; exit

L04A5:  DEFW    L12D8                   ; check-for
        DEFB    $0A                     ; ten                   marker byte?
        DEFW    L1A0E                   ; end-forth.

; code gels

L04AA:  LD      HL,$3C3E                ; address FLAGS
        LD      A,(HL)                  ; fetch FLAGS value.

        AND     $BB                     ; AND %10111011
                                        ; reset bit 2 - show definition complete
                                        ; reset bit 6 - show in interpreter mode

        LD      (HL),A                  ; update FLAGS value.

        JP      (IY)                    ; to 'next'.

; ----
; Note. these backward links to the beginning of words will probably be less
; of a mystery when the syntax checking and listing modules are more fully
; explored. A value of $FFFF sometimes occurs.

x04b3   DEFB    $00                     ;;

x04b4   DEFB    $E8                     ;;
x04b5   DEFB    $FF                     ;; 04b5 + ffe8 = 049d  = ';'

; ----------------------------------
; THE 'ADDRESS' INTERPRETER ROUTINES
; ----------------------------------

; ------------------------
; The 'Exit' Internal Word
; ------------------------
; Drops the 'Next Word' pointer from the Return Stack thereby ending a
; subroutine and returning to next word in calling thread.

L04B6:  DEFW    L04B8                   ; headerless 'code field'

; ---

L04B8:  POP     HL                      ; discard the next word pointer.

; ----------------------------
; THE 'ADDRESS INTERPRETER' LOOP
; ----------------------------
; Sometimes known as the Sequencer.
;
; iy_fast

L04B9:  POP     HL                      ; word pointer.

; =====> from DOCOLON and BRANCH

L04BA:  LD      E,(HL)
        INC     HL
        LD      D,(HL)
        INC     HL
```

```
1607
1608            PUSH    HL                      ; word pointer.
1609
1610    ; ==>
1611    ;
1612    L04BF:  EX      DE,HL
1613            LD      E,(HL)
1614            INC     HL
1615            LD      D,(HL)
1616            INC     HL
1617            EX      DE,HL
1618
1619            JP      (HL)                    ; jump to machine code (4 clock cycles)
1620                                            ; which will terminate with a JP (IY)
1621                                            ; instruction (8 clock cycles).
1622
1623
1624
1625    ; --------------------------------
1626    ; The 'Memory Check' Internal Word
1627    ; --------------------------------
1628    ; This internal word which also checks the BREAK key is only used from the
1629    ; start of the LINE definition. However the machine code entry point is the
1630    ; normal value of the IY register and so this code is executed at the end of
1631    ; every word.
1632
1633    L04C6:  DEFW    L04C8                   ; headerless 'code field'
1634
1635    ; iy_slow
1636
1637    L04C8:  LD      BC,$000B                ; allow overhead of eleven bytes
1638            LD      DE,($3C3B)              ; SPARE
1639            LD      HL,($3C37)              ; STKBOT
1640            ADD     HL,BC                   ; add the overhead
1641            SBC     HL,DE                   ; subtract the SPARE value
1642            JR      C,L04D9                 ; forward if the original 12 byte gap
1643                                            ; remains.
1644
1645    ; else stack underflow has occurred.
1646
1647    L04D7:  RST     20H                     ; Error 2
1648            DEFB    $02                     ; Data stack underflow.
1649
1650    ; ---
1651
1652    L04D9:  LD      BC,$0000                ; allow no overhead.
1653
1654            CALL    L0F8C                   ; check free memory
1655            CALL    L04E4                   ; check BREAK key.
1656            JR      L04B9                   ; back to iy_fast
1657
1658    ; ------------------------------------
1659    ; THE 'CHECK FOR BREAK KEY' SUBROUTINE
1660    ; ------------------------------------
1661    ; Check for the key combination SHIFT/SPACE.
1662
1663    L04E4:  LD      A,$FE                   ; read port $FEFE -
1664            IN      A,($FE)                 ; keys SPACE, SYMSHIFT, M, N, B.
1665
1666            RRA                             ; test bit for outermost key
1667            RET     C                       ; return if not pressed.
1668
1669            LD      A,$7F                   ; read port $7FFE -
1670            IN      A,($FE)                 ; keys SHIFT, Z, X, C, V.
1671
1672            RRA                             ; test bit for outermost key
1673            RET     C                       ; return if not pressed.
1674
1675    L04F0:  RST     20H                     ; Error 3.
1676            DEFB    $03                     ; BREAK pressed.
1677
1678    ; -------------------------
1679    ; THE 'MAIN EXECUTION' LOOP
```

```
1680   ; -------------------------
1681   ; The final part of the QUIT definition, as in all FORTH implementations,
1682   ; just loops through two FORTH words.
1683
1684   ; The first call - to the Address Interpreter - does not return.
1685   ; The return address is the next word QUERY which the interpreter pops off
1686   ; the Return Stack and then before executing puts the address of the next word
1687   ; on Return Stack. The default action of the Address Interpreter is to execute
1688   ; words in turn until some word, such as branch, alters this default behaviour.
1689
1690   L04F2:  CALL    L04B9                   ; forth.
1691
1692   L04F5:  DEFW    L058C                   ; QUERY         - input buffer
1693           DEFW    L0506                   ; LINE          - interpret buffer
1694           DEFW    L0536                   ; prOK          - print OK
1695           DEFW    L1276                   ; branch        - relative jump
1696
1697   L04FD:  DEFW    $FFF7                   ; back to L04F5
1698
1699   ; ---
1700   ; the first high-level interpreted word.
1701   ; ---
1702
1703   ; ---------------
1704   ; THE 'LINE' WORD
1705   ; ---------------
1706   ; Interprets input buffer as a normal FORTH line.
1707
1708   L04FF:  DEFM    "LIN"                   ; 'name field'
1709           DEFB    'E' + $80
1710
1711           DEFW    L04A0                   ; 'link field'
1712
1713   L0505:  DEFB    $04                     ; 'name length field'
1714
1715   L0506:  DEFW    L0EC3                   ; 'code field' - docolon
1716
1717   ; ---
1718
1719   L0508:  DEFW    L04C6                   ; check mem each time through loop
1720                                           ; as dictionary could be expanding.
1721
1722           DEFW    L063D                   ; FIND          - search the dictionary
1723           DEFW    L08EE                   ; ?DUP          - duplicate if found
1724           DEFW    L1283                   ; ?branch       - forward if not a
1725   L0510:  DEFW    $0007                   ; to L0518      - word.
1726
1727           DEFW    L054F                   ; test and stack??
1728           DEFW    L1276                   ; branch
1729   L0516:  DEFW    $FFF1                   ; back to L0508
1730
1731   L0518:  DEFW    L06A9                   ; NUMBER
1732           DEFW    L08EE                   ; ?DUP
1733           DEFW    L1283                   ; ?branch       - forward if not a
1734   L051E:  DEFW    $0007                   ; to L0526      - number.
1735
1736           DEFW    L0564                   ; pop de with test
1737           DEFW    L1276                   ; branch
1738   L0524:  DEFW    $FFE3                   ; loop back to L0508
1739
1740   L0526:  DEFW    L061B                   ; stack-length
1741           DEFW    L0C1A                   ; 0=
1742           DEFW    L1283                   ; ?branch       - forward with anything
1743   L052C:  DEFW    $0003                   ; to L0530      - else
1744
1745   L052E:  DEFW    L04B6                   ; EXIT                          >>>
1746
1747   ; ---
1748
1749   L0530:  DEFW    L0578                   ; RETYPE        - [?] at relevant place
1750           DEFW    L1276                   ; branch        - once corrected back
1751   L0534:  DEFW    $FFD3                   ; to L0508      - to the loop.
1752
```

```
; ---------------------------
; The 'Print OK' Internal Word
; ---------------------------
; prints the OK message after successful execution.

L0536:  DEFW    L0538                   ; headerless 'code field'

L0538:  LD      A,($3C3E)               ; fetch system variable FLAGS

        BIT     6,A                     ; test for 'COMPILER' mode.
        JR      NZ,L054D                ; forward if so.

        BIT     4,A                     ; test for 'INVIS' mode.
        JR      NZ,L054D                ; forward if so.

        CALL    L1808                   ; else print the inline string.

; ---

        DEFM    " OK"                   ; the OK message between two spaces.
        DEFB    ' ' + $80               ; last one inverted.

; ---

L054A:  LD      A,$0D                   ; prepare a carriage return.
        RST     08H                     ; and PRINT also.

L054D:  JP      (IY)                    ; to 'next'.

; -----------------------------
; The 'XXXXXXXXXX' Internal Word
; -----------------------------
; to handle a Word from LINE

L054F:  DEFW    L0551                   ; headerless 'code field'

; ---

L0551:  RST     18H                     ; pop address from Data Stack to DE

        DEC     DE                      ; point to the 'name length field'

        LD      A,(DE)                  ; fetch contents of the address.

        CPL                             ; complement.

        AND     (IX+$3E)                ; FLAGS

        AND     $40                     ; isolate BIT 6 of FLAGS, set if in
                                        ; compiler mode.

        INC     DE                      ; increment address to 'code field'

        JR      Z,L0561                 ; forward if not in compiling mode

        RST     10H                     ; push word DE          - add to dict
        LD      DE,L0F4E                ; ','                   - enclose

L0561:  JP      L04BF                   ; next word.

; ----------------------
; The '???' Internal Word
; ----------------------
; after handling a number from LINE

L0564:  DEFW    L0566                   ; headerless 'code field'

; ---

L0566:  RST     18H                     ; pop word DE

        BIT     6,(IX+$3E)              ; test FLAGS - compiler mode ?
```

```
1826            JR      NZ,L0561                ; loop back while in compiler mode.
1827
1828            JP      (IY)                    ; to 'next'.
1829
1830    ; -----------------
1831    ; THE 'RETYPE' WORD
1832    ; -----------------
1833    ; Allows user to edit the input line. Turns cursor to [?].
1834
1835    L056F:  DEFM    "RETYP"                 ; 'name field'
1836            DEFB    'E' + $80
1837
1838            DEFW    L058B                   ; 'link field'
1839
1840    L0577:  DEFB    $06                     ; 'name length field'
1841
1842    L0578:  DEFW    L057A                   ; 'code field'
1843
1844    ; ---
1845
1846    L057A:  CALL    L02EA                   ; routine sets logical line.
1847
1848            CALL    L0276                   ; routine pr_cursor
1849
1850            LD      (HL),$BF                ; the inverse [?] character
1851
1852            JR      L0594                   ; forward to join the QUERY routine.
1853
1854    ; ----------------
1855    ; THE 'QUERY' WORD
1856    ; ----------------
1857    ; Clears input buffer, then accepts characters until ENTER pressed.
1858    ; Buffer can be edited as usual and is limited to 22 lines.
1859
1860    L0584:  DEFM    "QUER"                  ; 'name field'
1861            DEFB    'Y' + $80
1862
1863            DEFW    L0505                   ; 'link field'
1864
1865    L058B:  DEFB    $05                     ; 'name length field'
1866
1867    L058C:  DEFW    L058E                   ; 'code field'
1868
1869    ; ---
1870
1871    L058E:  CALL    L02D8                   ; routine SETBUF
1872
1873            CALL    L0276                   ; routine pr_cursor
1874
1875    ; ->
1876    L0594:  LD      HL,$3C28                ; fetch STATIN
1877            SET     0,(HL)                  ;
1878            RES     5,(HL)                  ; (bit 5 set by interrupt when the user
1879                                            ; presses the ENTER key)
1880
1881    L059B:  BIT     5,(HL)                  ; wait for interrupt to set the bit.
1882            JR      Z,L059B                 ; loop until.
1883
1884            CALL    L0225                   ; routine DEL-CURSOR
1885            JP      (IY)                    ; to 'next'.
1886
1887    ; ---------------
1888    ; THE 'WORD' WORD
1889    ; ---------------
1890    ; WORD text
1891    ; ( delimiter -- address )
1892    ; Takes text out of the input buffer up as far as a delimiter, and copies it
1893    ; to pad, starting at the second byte there. Puts the length (not including
1894    ; the delimiter) in the first byte of the pad, and stacks the address of the
1895    ; first byte of the pad.
1896    ; At most 253 characters are taken from the input buffer. If there are more
1897    ; left before the delimiter, then the first byte of the pad shows 254.
1898    ; Initial delimiters are ignored.
```

```
1899
1900   L05A4:  DEFM    "WOR"                   ; 'name field'
1901           DEFB    'D' + $80
1902
1903           DEFW    L0577                   ; 'link field'
1904
1905   L05AA:  DEFB    $04                     ; 'name length field'
1906
1907   L05AB:  DEFW    L05AD                   ; 'code field'
1908
1909   ; ---
1910
1911   L05AD:  RST     18H                     ; pop word DE
1912           LD      HL,$27FE                ; set HL to penultimate byte of 'pad'.
1913           LD      B,$FD                   ; the count is 253.
1914
1915   L05B3:  LD      (HL),$20                ; insert a space in pad.
1916           DEC     HL                      ; decrement the address.
1917           DJNZ    L05B3                   ; repeat for the 253 locations.
1918
1919           PUSH    DE                      ; save the delimiter.
1920           EX      DE,HL                   ; save in HL also, DE is start of pad.
1921
1922           RST     10H                     ; stack data word DE
1923           POP     DE                      ; retrieve the delimiter.
1924
1925           CALL    L05E1                   ;
1926
1927           INC     B
1928           DEC     B
1929           JR      Z,L05C6                 ;
1930
1931           LD      BC,$00FF
1932
1933   L05C6:  LD      HL,$2701
1934           LD      (HL),C
1935           INC     HL
1936           LD      A,$FC
1937           CP      C
1938           JR      NC,L05D1                ;
1939
1940           LD      C,A
1941
1942   L05D1:  INC     C
1943           PUSH    DE
1944           PUSH    BC
1945           EX      DE,HL
1946           LDIR
1947           POP     BC
1948           POP     DE
1949           DEC     C
1950           CALL    L07DA                   ;
1951           JP      (IY)                    ; to 'next'.
1952
1953   ; -------------------------------
1954   ; THE 'GET BUFFER TEXT' SUBROUTINE
1955   ; -------------------------------
1956   ; Called from FIND, NUMBER and XXXXX. Word may have leading spaces and is
1957   ; terminated by a space or newline (zero).
1958   ; It is also used to find the end of a comment delimited by ')'.
1959   ;
1960   ; =>
1961   L05DF:  LD      E,$20                   ; set a space as the skip character.
1962
1963   ; =>called with E holding delimiter.
1964   ;
1965   L05E1:  LD      HL,($3C24)              ; fetch L_HALF - start of screen buffer.
1966           LD      ($3C1E),HL              ; make INSCRN start of logical line the
1967                                           ; same.
1968
1969           LD      BC,$0000                ; initialize letter count to zero.
1970
1971   ; -> loop
```

```
L05EA:  INC     HL                      ; increment screen address.
        LD      A,(HL)                  ; fetch character to A.
        CP      E                       ; compare to character in E.
        JR      Z,L05EA                 ; loop while character matches.

        AND     A                       ; test for zero (at $2700?)
        JR      Z,L0600                 ; forward if so.

; a word has been found on the screen line.

        PUSH    HL                      ; save pointer to start of word.

L05F3:  INC     BC                      ; increment the letter count.
        INC     HL                      ; increment the screen pointer.

        LD      A,(HL)                  ; fetch new character
        AND     A                       ; test for zero.
        JR      Z,L05FC                 ; skip forward as at end of word.

        CP      E                       ; compare to the skip character.
        JR      NZ,L05F3                ; loop back if still within a word.

L05FC:  POP     DE                      ; retrieve pointer to start of word.

        XOR     A                       ;; clear A
        CP      B                       ;; compare to B zero

        RET                             ; return. with carry reset for success.

; ---

L0600:  PUSH    DE                      ; save delimiter

        CALL    L02B0                   ; routine find zerobyte
        JP      PO,L0614                ; jump if found to exit failure

        LD      DE,($3C24)              ; else set DE from L_HALF
        CALL    L07FA                   ; routine SPACE_FILL (DE-HL)
        LD      ($3C24),HL              ; set L_HALF to next line

        POP     DE                      ; restore delimiter

        JR      L05E1                   ; loop back using new line.

; ---

; branch here if a word not found.

L0614:  EX      DE,HL                   ; DE addresses cursor.
        POP     BC                      ; discard saved delimiter
        LD      BC,$0000                ; set BC, to zero
        SCF                             ; signal not found
        RET                             ; return.

; --------------------------------
; The 'stack length' Internal Word
; --------------------------------
; used once only from LINE to check for any extraneous text that is not a Word
; or a Number.

L061B:  DEFW    L061D                   ; headerless 'code field'

; ---

L061D:  CALL    L05DF                   ; get buffer

        LD      D,B                     ; transfer length of word
        LD      E,C                     ; from BC to DE
        RST     10H                     ; push word DE
        JP      (IY)                    ; to 'next'.


; ---------------
```

```
; THE 'VLIST' WORD
; ----------------
; List dictionary to screen, including words in ROM.
; (no pause after 18 lines)

L0625:  DEFM    "VLIS"                  ; 'name field'
        DEFB    'T' + $80

        DEFW    L05AA                   ; 'link field'

L062C:  DEFB    $05                     ; 'name length field'

L062D:  DEFW    L062F                   ; 'code field'

; ---

L062F:  LD      A,$0D                   ; prepare a newline

        RST     08H                     ; print it.

        LD      C,$00                   ; set a flag for 'do all names'.

        JR      L0644                   ; forward to FIND.


; ---------------
; THE 'FIND' WORD
; ---------------
; ( -- compilation address )
; Leaves compilation address of first word in input buffer, if defined in
; context vocabulary; else 0.

L0636:  DEFM    "FIN"                   ; 'name field'
        DEFB    'D' + $80

        DEFW    L062C                   ; 'link field'

L063C:  DEFB    $04                     ; 'name length field'

L063D:  DEFW    L063F                   ; 'code field'

; ---

L063F:  CALL    L05DF                   ; get buffer word, gets length in C.

        JR      C,L068A                 ; back if null to stack word zero

; ->

L0644:  LD      HL,(CONTEXT)            ; fetch value of system variable CONTEXT
        LD      A,(HL)                  ; extract low byte of address.
        INC     HL                      ; increment pointer.
        LD      H,(HL)                  ; extract high byte of address.
        LD      L,A                     ; address now in HL.

; The address points to the 'name length field' of the most recent word in the
; Dictionary.


L064B:  LD      A,(HL)                  ; fetch addressed byte.
        AND     $3F                     ; discount bit 6, the immediate word
                                        ; indicator, to give length 1-31

        JR      Z,L067F                 ; a 'zero' length indicates this is a
                                        ; link like the example at the end of
                                        ; this ROM.

        XOR     C                       ; match against C.
        JR      Z,L0657                 ; skip forward if lengths match.

        LD      A,C                     ; test flag C
        AND     A                       ; for value zero.
        JR      NZ,L067F                ; forward if C not zero.
```

```
2118
2119    ; else a name that matches the search length or all names are required - VLIST.
2120
2121
2122    L0657:  PUSH    DE                      ; preserve DE
2123            PUSH    HL                      ; preserve 'name length field' pointer.
2124
2125            CALL    L15E8                   ; routine WORDSTART finds start of name.
2126                                            ; A is returned as zero.
2127
2128            OR      C                       ; test C for zero
2129            JR      Z,L0676                 ; branch forward to print if in VLIST.
2130
2131    ; else the search is for a specific word and a word with same length, at least,
2132    ; has been found.
2133
2134            LD      B,C                     ; copy the length to counter B.
2135
2136    L0660:  LD      A,(DE)                  ; fetch first letter of match word.
2137
2138            CALL    L0807                   ; routine UPPERCASE
2139
2140            INC     DE                      ; update pointer (in lower screen)
2141            XOR     (HL)                    ; match against letter (in dictionary).
2142            AND     $7F                     ; disregard any inverted bit.
2143            INC     HL                      ; increment dictionary pointer.
2144
2145            JR      NZ,L067D                ; exit loop to try next link if no match
2146
2147            DJNZ    L0660                   ; else loop back for all letters.
2148
2149    ; Oh Frabjous day - a match.
2150
2151            POP     DE                      ; pop 'name length field' pointer.
2152            INC     DE                      ; increment to point to compilation
2153                                            ; address.
2154            RST     10H                     ; stack date word DE.
2155
2156    ; the remaining task is to clean up the input buffer in the lower screen.
2157
2158            POP     DE                      ; pop the DE - screen pointer.
2159
2160            CALL    L07DA                   ; clean up - backfill with spaces.
2161
2162            JP      (IY)                    ; to 'next'.
2163
2164    ; ----------------------
2165    ; THE 'PRINT NAME' BRANCH
2166    ; ----------------------
2167    ; This branch is taken from the above loop when all found words are to be
2168    ; printed by VLIST. It takes its time as if the user has expanded the
2169    ; dictionary then the list will scroll off the top of the screen. By waiting
2170    ; for an interrupt each time, it ensures that a standard listing takes about
2171    ; three seconds and there is ample opportunity to press BREAK to stop at a
2172    ; certain point.
2173
2174    L0676:  CALL    L17FB                   ; routine print string and space
2175
2176            HALT                            ; wait for an interrupt.
2177
2178            CALL    L04E4                   ; routine checks BREAK key.
2179
2180    L067D:  POP     HL                      ; restore 'name length field' pointer
2181            POP     DE                      ; restore DE
2182
2183    L067F:  DEC     HL                      ; point to high byte of 'link field'
2184            LD      A,(HL)                  ; hold it in A.
2185            DEC     HL                      ; point to low byte of 'link field'
2186            LD      L,(HL)                  ; transfer address of the new
2187            LD      H,A                     ; 'name length field' to HL pointer.
2188
2189            OR      L                       ; test if address is zero - for the
2190                                            ; last entry in the linked list.
```

```
2191
2192            JR      NZ,L064B                ; loop back while this is not the
2193                                            ; last entry in the vocabulary.
2194
2195    L0687:  DEFB    $C3                     ; A JP instruction i.e. JP L068A
2196
2197    ; Note. The intention is to jump past the headerless code word for the internal
2198    ; word stk_zero. Since the word that would follow the first byte of the jump
2199    ; instruction would be identical to the word it is jumping over then the word
2200    ; can be omitted. Only saves one byte but this is back in 1983.
2201
2202    ; ----------------------------
2203    ; The 'stk-zero' Internal Word
2204    ; ----------------------------
2205    ; (  -- 0 )
2206
2207    L0688:  DEFW    L068A                   ; headerless 'code field'
2208
2209    ; ---
2210
2211    L068A:  LD      DE,$0000                ; load DE with the value zero.
2212            RST     10H                     ; stack Data Word DE
2213
2214            JP      (IY)                    ; to 'next'.
2215
2216    ; ------------------
2217    ; THE 'EXECUTE' WORD
2218    ; ------------------
2219    ; ( compilation address --  )
2220    ; Executes the word with the given compilation address.
2221
2222    L0690:  DEFM    "EXECUT"                ; 'name field'
2223            DEFB    'E' + $80
2224
2225            DEFW    L063C                   ; 'link field'
2226
2227    L0699:  DEFB    $07                     ; 'name length field'
2228
2229    L069A:  DEFW    L069C                   ; 'code field'
2230
2231    ; ---
2232
2233    L069C:  RST     18H
2234
2235            JP      L04BF                   ;
2236
2237    ; -----------------
2238    ; THE 'NUMBER' WORD
2239    ; -----------------
2240    ; Takes a number from the start of the input buffer. Leaves the number and
2241    ; a non-zero address on the stack. (The address is the compilation address
2242    ; of a literal compiler, so that if you then say EXECUTE, the literal compiler
2243    ; compiles the number into the dictionary as a literal - for an integer it
2244    ; is 4102, for a floating point number it is 4181).
2245    ; If no valid number then leaves just 0 on the stack.
2246
2247    L06A0:  DEFM    "NUMBE"                 ; 'name field'
2248            DEFB    'R' + $80
2249
2250            DEFW    L0699                   ; 'link field'
2251
2252    L06A8:  DEFB    $06                     ; 'name length field'
2253
2254    L06A9:  DEFW    L06AB                   ; 'code field'
2255
2256    ; ---
2257
2258    L06AB:  CALL    L05DF                   ; get buffer
2259
2260            JR      C,L068A                 ; if empty stack word zero.
2261
2262            PUSH    BC
2263            PUSH    DE
```

```
2264
2265            CALL    L074C                   ;
2266
2267            JR      NZ,L06BC                ;
2268
2269            LD      DE,$1006                ; addr literal?
2270            JR      L0714                   ;
2271
2272    ; ---
2273
2274    L06BC:  RST     18H                     ; pop word DE
2275            LD      DE,$0000
2276            RST     10H                     ; push word DE
2277            LD      DE,$4500
2278            POP     BC
2279            PUSH    BC
2280            LD      A,(BC)
2281            CP      $2D                     ; is it '-' ?
2282            JR      NZ,L06CE                ;
2283
2284            LD      D,$C5
2285            INC     BC
2286    L06CE:  RST     10H                     ; push word DE
2287            LD      D,B
2288            LD      E,C
2289            DEC     HL
2290            DEC     HL
2291
2292    L06D3:  CALL    L0723                   ; routine GET_DECIMAL
2293
2294            INC     HL
2295            INC     (HL)
2296            DEC     HL
2297            JR      NC,L06D3                ;
2298
2299            CP      $FE
2300            JR      NZ,L071C                ;
2301
2302    L06DF:  CALL    L0723                   ; routine GET_DECIMAL
2303
2304            JR      NC,L06DF                ;
2305
2306            ADD     A,$30                   ; add '0' converting to letter.
2307            CALL    L077B                   ;
2308            JR      NZ,L06EF                ;
2309
2310            LD      E,$00
2311            JR      L06FD                   ;
2312
2313    L06EF:  AND     $DF                     ;
2314
2315            CP      $45                     ; is it 'E' - extended format?
2316            JR      NZ,L071C                ;
2317
2318            PUSH    HL
2319
2320            CALL    L074C                   ;
2321
2322            RST     18H                     ; pop word DE
2323            POP     HL
2324            JR      NZ,L071C                ;
2325
2326    L06FD:  CALL    L0740                   ;
2327            JR      Z,L0711                 ;
2328
2329            INC     HL
2330            LD      A,(HL)
2331            AND     $7F
2332            ADD     A,E
2333
2334            JP      M,L071C                 ; forward +->
2335
2336            JR      Z,L071C                 ; forward +->
```

```
2337
2338            XOR     (HL)
2339            AND     $7F
2340            XOR     (HL)
2341            LD      (HL),A
2342    L0711:  LD      DE,L1055                ; stk_fp
2343    L0714:  RST     10H                     ; push word DE
2344            POP     DE
2345            POP     BC
2346            CALL    L07DA                   ;
2347            JP      (IY)                    ; to 'next'.
2348
2349    ; ---
2350
2351    ; +->
2352    L071C:  POP     HL
2353            POP     HL
2354            RST     18H                     ; pop word DE
2355            RST     18H                     ; pop word DE
2356            JP      L068A                   ;
2357
2358    ; ----------------------------
2359    ; THE 'GET DECIMAL' SUBROUTINE
2360    ; ----------------------------
2361    ; Fetch character and return with carry set if after conversion is not in
2362    ; range 0 to 9.
2363
2364    L0723:  LD      A,(DE)
2365            INC     DE
2366            SUB     $30                     ; subtract '0'
2367            RET     C                       ; return if was less than '0'
2368
2369            CP      $0A                     ; compare to ten.
2370            CCF                             ; complement
2371            RET     C                       ; return - with carry set if over 9.
2372
2373    ; ---------
2374    ; normalize?
2375    ; ---------
2376    ; => from below only.
2377    L072C:  LD      C,A
2378            LD      A,(HL)
2379            AND     $F0
2380            RET     NZ
2381
2382            LD      A,C
2383
2384    ; => (int/print_fp)
2385    L0732:  DEC     HL
2386            DEC     HL
2387            LD      C,$03
2388
2389    L0736:  RLD                             ;  A = xxxx3210  <--   7654<-3210 (HL)
2390
2391            INC     HL                      ;
2392            DEC     C
2393            JR      NZ,L0736                ;
2394
2395            DEC     (HL)                    ; decrement exponent
2396            DEC     HL                      ; point to start of BCD nibbles
2397            CP      A
2398            RET
2399
2400    ; ---
2401
2402    ; from ufloat to normalize 6-nibble mantissa
2403
2404    L0740:  LD      B,$06                   ; six nibbles
2405
2406    L0742:  XOR     A
2407
2408            CALL    L072C                   ;
2409
```

```
2410            RET     NZ
2411
2412            DJNZ    L0742                   ;
2413
2414            INC     HL
2415            LD      (HL),B
2416
2417            RET
2418
2419    ; ---------------------------
2420    ; THE 'GET NUMBER' SUBROUTINE
2421    ; ---------------------------
2422    ; can be called twice by the above code for the word 'NUMBER'.
2423    ; Once to get the first number encountered and sometimes, if in extended
2424    ; format, the exponent as well.
2425
2426    L074C:  RST     10H                     ; push word DE
2427
2428            CALL    L04B9                   ; forth
2429
2430    L0750:  DEFW    L086B                   ; dup
2431            DEFW    L0896                   ; C@
2432            DEFW    L104B                   ; stk-data
2433            DEFB    $2D                     ;  chr '-'
2434            DEFW    L0C4A                   ; =
2435            DEFW    L086B                   ; dup
2436            DEFW    L0DA9                   ; negate
2437            DEFW    L08D2                   ; >R
2438            DEFW    L0DD2                   ; +
2439            DEFW    L0E1F                   ; 1-
2440            DEFW    L0688                   ; stk-zero
2441            DEFW    L0688                   ; stk-zero
2442            DEFW    L08FF                   ; rot
2443    L0769:  DEFW    L078A                   ; convert
2444            DEFW    L08FF                   ; rot
2445            DEFW    L08DF                   ; R>
2446            DEFW    L0D94                   ; pos
2447            DEFW    L08FF                   ; rot
2448            DEFW    L0879                   ; drop
2449            DEFW    L0885                   ; swap
2450            DEFW    L1A0E                   ; end-forth.
2451
2452    L0779:  RST     18H                     ; pop word DE
2453            LD      A,(DE)
2454
2455    L077B:  CP      $20
2456            RET     Z
2457
2458            AND     A
2459            RET
2460
2461    ; ------------------
2462    ; THE 'CONVERT' WORD
2463    ; ------------------
2464    ; (  ud1, addr1 -- ud2, addr2  )
2465    ; Accumulates digits from text into an unsigned double length
2466    ; number ud1: for each digit, the double length accumulator is
2467    ; multiplied by the system number base and the digit (converted
2468    ; from ASCII) is added on. The text starts at addr1 + 1. addr2 is
2469    ; the address of the first unconvertible character, ud2 is the
2470    ; final value of the accumulator.
2471
2472    L0780:  DEFM    "CONVER"                ; 'name field'
2473            DEFB    'T' + $80
2474
2475            DEFW    L06A8                   ; 'link field'
2476
2477    L0789:  DEFB    $07                     ; 'name length field'
2478
2479    L078A:  DEFW    L0EC3                   ; 'code field' - docolon
2480
2481    ; ---
2482
```

```
2483   L078C:  DEFW    L0E09                   ; 1+
2484   L078E:  DEFW    L086B                   ; dup
2485   L0790:  DEFW    L08D2                   ; >R
2486   L0792:  DEFW    L0896                   ; C@
2487   L0794:  DEFW    L07B8                   ; stk_digit
2488   L0796:  DEFW    L1283                   ; ?branch
2489   L0798:  DEFW    $001B                   ; to 0799 + 1B = $07B4
2490
2491   L079A:  DEFW    L0885                   ; swap
2492   L079C:  DEFW    L048A                   ; get base
2493   L079E:  DEFW    L0896                   ; C@
2494   L07A0:  DEFW    L0CA8                   ; u*
2495   L07A2:  DEFW    L0879                   ; drop
2496   L07A4:  DEFW    L08FF                   ; rot
2497   L07A6:  DEFW    L048A                   ; get base
2498   L07A8:  DEFW    L0896                   ; C@
2499   L07AA:  DEFW    L0CA8                   ; U*
2500   L07AC:  DEFW    L0DEE                   ; D+
2501   L07AE:  DEFW    L08DF                   ; R>
2502   L07B0:  DEFW    L1276                   ; branch
2503   L07B2:  DEFW    $FFD9                   ; loop back to L078C
2504
2505   L07B4:  DEFW    L08DF                   ; R>
2506   L07B6:  DEFW    L04B6                   ; exit
2507
2508   ; ----------------------------
2509   ; The 'stk_digit' Internal Word
2510   ; ----------------------------
2511
2512   L07B8:  DEFW    L07BA                   ; headerless 'code field'
2513
2514   ; ---
2515
2516   L07BA:  RST     18H                     ; pop word DE
2517
2518           LD      A,E                     ; character to A
2519
2520           CALL    L0807                   ; to_upper
2521
2522           ADD     A,$D0                   ; add to give carry with '0' and more.
2523
2524           JR      NC,L07D7                ; if less than '0' push byte 0 false.
2525
2526           CP      $0A                     ; compare to ten.
2527           JR      C,L07CD                 ; forward to stack bytes 0 - 9.
2528
2529           ADD     A,$EF                   ;
2530           JR      NC,L07D7                ; push word false 0.
2531
2532           ADD     A,$0A
2533
2534   L07CD:  CP      (IX+$3F)                ; compare to BASE
2535           JR      NC,L07D7                ; push word false 0.
2536
2537   ; else digit is within range of number base
2538
2539           LD      D,$00
2540           LD      E,A
2541           RST     10H                     ; push word DE
2542           SCF                             ; set carry to signal true
2543
2544   L07D7:  JP      L0C21                   ; push word 1 or 0
2545
2546   ; ---
2547   ;       ??
2548   ; ---
2549
2550   L07DA:  LD      H,D
2551           LD      L,E
2552           INC     BC
2553           ADD     HL,BC
2554           PUSH    HL
2555           BIT     4,(IX+$3E)              ; FLAGS
```

```
2556            CALL    Z,L097F                 ; pr_string
2557
2558            CALL    L02B0                   ; curs?
2559
2560            POP     DE
2561            AND     A
2562            SBC     HL,DE
2563            LD      B,H
2564            LD      C,L
2565            LD      HL,($3C1E)              ; INSCRN
2566            INC     HL
2567            EX      DE,HL
2568            JR      C,L07FB                 ;
2569
2570            JR      Z,L07FA                 ; forward to SPACE_FILL.
2571
2572            LDIR
2573
2574    ; ------------------------
2575    ; The 'SPACE FILL' routine
2576    ; ------------------------
2577    ; -> from cls
2578
2579    L07FA:  AND     A                       ; prepare to subtract two screen
2580                                            ; pointers.
2581
2582    L07FB:  SBC     HL,DE                   ; number of bytes in HL.
2583            EX      DE,HL                   ; now in DE, HL = start of area.
2584
2585    L07FE:  LD      A,D                     ; check if the
2586            OR      E                       ; counter is zero.
2587            RET     Z                       ; return if so.                 >>
2588
2589            LD      (HL),$20                ; insert a space character.
2590            INC     HL                      ; next address.
2591            DEC     DE                      ; decrement byte counter.
2592            JR      L07FE                   ; loop back to exit on zero.
2593
2594    ; --------------------------
2595    ; THE 'UPPERCASE' SUBROUTINE
2596    ; --------------------------
2597    ; converts characters to uppercase.
2598
2599    L0807:  AND     $7F                     ; ignore inverse bit 7
2600            CP      $61                     ; compare to 'a'
2601            RET     C                       ; return if lower
2602
2603            CP      $7B                     ; compare to 'z' + 1
2604            RET     NC                      ; return if higher than 'z'
2605
2606            AND     $5F                     ; make uppercase
2607            RET                             ; return.
2608
2609    ; --------------
2610    ; THE 'VIS' WORD
2611    ; --------------
2612    ; Allows copy-up mechanism and 'OK'.
2613
2614    L0812:  DEFM    "VI"                    ; 'name field'
2615            DEFB    'S' + $80
2616
2617            DEFW    L0789                   ; 'link field'
2618
2619    L0817:  DEFB    $03                     ; 'name length field'
2620
2621    L0818:  DEFW    L081A                   ; 'code field'
2622
2623    ; ---
2624
2625    L081A:  RES     4,(IX+$3E)              ; update FLAGS signal visible mode.
2626            JP      (IY)                    ; to 'next'.
2627
2628    ; ---------------
```

```
; THE 'INVIS' WORD
; ----------------
; Suppresses copy-up mechanism and 'OK'.

L0820:  DEFM    "INVI"                  ; 'name field'
        DEFB    'S' + $80

        DEFW    L0817                   ; 'link field'

L0827:  DEFB    $05                     ; 'name length field'

L0828:  DEFW    L082A                   ; 'code field'

; ---

L082A:  SET     4,(IX+$3E)              ; update FLAGS signal invisible mode.

        JP      (IY)                    ; to 'next'.


; ---------------
; THE 'FAST' WORD
; ---------------
; Fast mode - runs without error checks.
; Debugged programs run 25% faster.

L0830:  DEFM    "FAS"                   ; 'name field'
        DEFB    'T' + $80

        DEFW    L0827                   ; 'link field'

L0836:  DEFB    $04                     ; 'name length field'

L0837:  DEFW    L0839                   ; 'code field'

; ---

L0839:  LD      IY,L04B9                ; miss memory checks on return

        JP      (IY)                    ; to 'next'.

; ---------------
; THE 'SLOW' WORD
; ---------------
; ( -- )
; Slow mode with error checking.
; Make IY point to a return routine that performs housekeeping.


L083F:  DEFM    "SLO"                   ; 'name field'
        DEFB    'W' + $80

        DEFW    L0836                   ; 'link field'

L0845:  DEFB    $04                     ; 'name length field'


L0846:  DEFW    L0848                   ; 'code field'

; ---

L0848:  LD      IY,L04C8                ; set vector to memory checks each pass

        JP      (IY)                    ; to 'next'.

; ---------------------------------
; THE 'DATA STACK TO BC' SUBROUTINE
; ---------------------------------
; Called on twenty occasions to fetch a word from the Data Stack into the
; BC register pair. Very similar to RST 18H which does the same thing with the
; DE register pair as the destination on 73 occasions.
; In fact, as two Z80 restarts are unused, then 40 bytes of ROM code could have
; been saved by making this a restart also.
```

```
2702
2703   L084E:  LD      HL,($3C3B)              ; fetch SPARE - start of Spare Memory.
2704           DEC     HL                      ; decrement to point to last stack item
2705           LD      B,(HL)                  ; load high byte to B.
2706           DEC     HL                      ; address low byte of word.
2707           LD      C,(HL)                  ; and load to C.
2708           LD      ($3C3B),HL              ; update the system variable SPARE to
2709                                           ; a location two bytes less than it was.
2710           RET                             ; return.
2711
2712   ; ----------------------------------------
2713   ; THE 'CONTINUATION OF THE RST 18H' RESTART
2714   ; ----------------------------------------
2715   ; complete the operation of popping a word to DE from the data stack.
2716
2717   L0859:  DEC     HL                      ;
2718           LD      E,(HL)                  ;
2719           LD      ($3C3B),HL              ; update SPARE
2720           RET                             ; return.
2721
2722   ; ----------------------------------------
2723   ; THE 'CONTINUATION OF THE RST 10H' RESTART
2724   ; ----------------------------------------
2725   ; complete the operation of pushing a word in DE to the data stack.
2726
2727   L085F:  LD      (HL),D                  ;
2728           INC     HL                      ;
2729           LD      ($3C3B),HL              ; update SPARE
2730           RET                             ; return.
2731
2732   ; --------------
2733   ; THE 'DUP' WORD
2734   ; --------------
2735   ; ( n -- n, n )
2736   ; Duplicates the top of the stack.
2737
2738   L0865:  DEFM    "DU"                    ; 'name field'
2739           DEFB    'P' + $80
2740
2741           DEFW    L0845                   ; 'link field'
2742
2743   L086A:  DEFB    $03                     ; 'name length field'
2744
2745   L086B:  DEFW    L086D                   ; 'code field'
2746
2747   ; ---
2748
2749   L086D:  RST     18H                     ; unstack Data Word DE
2750           RST     10H                     ; stack Data Word DE
2751           RST     10H                     ; stack Data Word DE
2752
2753           JP      (IY)                    ; to 'next'.
2754
2755   ; ---------------
2756   ; THE 'DROP' WORD
2757   ; ---------------
2758   ; ( n -- )
2759   ; Throws away the top of the stack.
2760
2761   L0872:  DEFM    "DRO"                   ; 'name field'
2762           DEFB    'P' + $80
2763
2764           DEFW    L086A                   ; 'link field'
2765
2766   L0878:  DEFB    $04                     ; 'name length field'
2767
2768   L0879:  DEFW    L087B                   ; 'code field'
2769
2770   ; ---
2771
2772   L087B:  RST     18H                     ; unstack Data Word DE
2773           JP      (IY)                    ; to 'next'.
2774
```

```
2775   ; ---------------
2776   ; THE 'SWAP' WORD
2777   ; ---------------
2778   ; (n1, n2 -- n2, n1)
2779
2780   L087E:  DEFM    "SWA"                   ; 'name field'
2781           DEFB    'P' + $80
2782
2783           DEFW    L0878                   ; 'link field'
2784
2785   L0884:  DEFB    $04                     ; 'name length field'
2786
2787   L0885:  DEFW    L0887                   ; 'code field'
2788
2789   ; ---
2790
2791   L0887:  RST     18H                     ; pop word DE
2792           CALL    L084E                   ; stk_to_bc
2793           RST     10H                     ; push word DE
2794           LD      D,B                     ;
2795           LD      E,C                     ;
2796           RST     10H                     ; push word DE
2797
2798           JP      (IY)                    ; to 'next'.
2799
2800   ; -------------
2801   ; THE 'C@' WORD
2802   ; -------------
2803   ; (address -- byte)
2804   ; Fetches the contents of a given address.
2805
2806   L0891:  DEFB    'C'                     ; 'name field'
2807           DEFB    '@' + $80
2808
2809           DEFW    L0884                   ; 'link field'
2810
2811   L0895:  DEFB    $02                     ; 'name length field'
2812
2813   L0896:  DEFW    L0898                   ; 'code field'
2814
2815   ; ---
2816
2817   L0898:  RST     18H                     ; pop word DE
2818           LD      A,(DE)
2819           LD      E,A
2820           LD      D,$00
2821
2822           RST     10H                     ; push word DE
2823
2824           JP      (IY)                    ; to 'next'.
2825
2826   ; -------------
2827   ; THE 'C!' WORD
2828   ; -------------
2829   ; (n, address -- )
2830   ; Stores the less significant byte on n at a given address.
2831
2832   L08A0:  DEFB    'C'                     ; 'name field'
2833           DEFB    '!' + $80
2834
2835           DEFW    L0895                   ; 'link field'
2836
2837   L08A4:  DEFB    $02                     ; 'name length field'
2838
2839   L08A5:  DEFW    L08A7                   ; 'code field'
2840
2841   ; ---
2842
2843   L08A7:  RST     18H                     ; pop word DE
2844           CALL    L084E                   ; stk_to_bc
2845           LD      A,C
2846           LD      (DE),A
2847
```

```
2848            JP      (IY)                    ; to 'next'.
2849
2850    ; ------------
2851    ; THE '@' WORD
2852    ; ------------
2853    ; (address -- n)
2854    ; Leaves on stack the single length integer at the given address.
2855
2856    L08AF:  DEFB    '@' + $80               ; 'name field'
2857
2858            DEFW    L08A4                   ; 'link field'
2859
2860    L08B2:  DEFB    $01                     ; 'name length field'
2861
2862    L08B3:  DEFW    L08B5                   ; 'code field'
2863
2864    ; ---
2865
2866    L08B5:  RST     18H                     ; pop word DE
2867
2868            EX      DE,HL
2869            LD      E,(HL)
2870            INC     HL
2871            LD      D,(HL)
2872
2873            RST     10H                     ; push word DE
2874
2875            JP      (IY)                    ; to 'next'.
2876
2877    ; ------------
2878    ; THE '!' WORD
2879    ; ------------
2880    ; (n,address --)
2881    ; Stores the single-length integer n at the given address in memory.
2882
2883    L08BD:  DEFB    '!' + $80               ; 'name field'
2884
2885            DEFW    L08B2                   ; 'link field'
2886
2887    L08C0:  DEFB    $01                     ; 'name length field'
2888
2889    L08C1:  DEFW    L08C3                   ; 'code field'
2890
2891    ; ---
2892
2893    L08C3:  RST     18H                     ; pop word DE
2894            CALL    L084E                   ; stk_to_bc
2895            EX      DE,HL
2896            LD      (HL),C
2897            INC     HL
2898            LD      (HL),B
2899
2900            JP      (IY)                    ; to 'next'.
2901
2902    ; -------------
2903    ; THE '>R' WORD
2904    ; -------------
2905    ; (n -- )
2906    ; Transfers top entry on data stack to return stack.
2907    ; It can be copied back using 'I'.
2908
2909    L08CD:  DEFB    '>'                     ; 'name field'
2910            DEFB    'R' + $80
2911
2912            DEFW    L08C0                   ; 'link field'
2913
2914    L08D1:  DEFB    $02                     ; 'name length field'
2915
2916    L08D2:  DEFW    L08D4                   ; 'code field'
2917
2918    ; ---
2919
2920    L08D4:  RST     18H
```

```
2921            POP     BC
2922            PUSH    DE
2923            PUSH    BC
2924            JP      (IY)                    ; to 'next'.
2925
2926    ; -------------
2927    ; THE 'R>' WORD
2928    ; -------------
2929    ; ( -- entry from return stack)
2930    ; Transfers top entry on return stack to data stack.
2931
2932    L08DA:  DEFB    'R'                     ; 'name field'
2933            DEFB    '>' + $80
2934
2935            DEFW    L08D1                   ; 'link field'
2936
2937    L08DE:  DEFB    $02                     ; 'name length field'
2938
2939    L08DF:  DEFW    L08E1                   ; 'code field'
2940
2941    ; ---
2942
2943    L08E1:  POP     BC
2944            POP     DE
2945            PUSH    BC
2946            RST     10H                     ; push word DE
2947            JP      (IY)                    ; to 'next'.
2948
2949    ; ---------------
2950    ; THE '?DUP' WORD
2951    ; ---------------
2952    ; (n -- n, n)    if n!=0.
2953    ; (n -- n)       if n=0.
2954
2955    L08E7:  DEFM    "?DU"                   ; 'name field'
2956            DEFB    'P' + $80
2957
2958            DEFW    L08DE                   ; 'link field'
2959
2960    L08ED:  DEFB    $04                     ; 'name length field'
2961
2962    L08EE:  DEFW    L08F0                   ; 'code field'
2963
2964    ; ---
2965
2966
2967    L08F0:  RST     18H                     ; fetch word DE
2968            RST     10H                     ; push it back
2969            LD      A,D                     ; test if fetched
2970            OR      E                       ; word is zero
2971            CALL    NZ,L0010                ; push word DE if non-zero
2972            JP      (IY)                    ; to 'next'.
2973
2974    ; --------------
2975    ; THE 'ROT' WORD
2976    ; --------------
2977    ; (n1, n2, n3 -- n2, n3, n1)
2978
2979    L08F9:  DEFM    "RO"                    ; 'name field'
2980            DEFB    'T' + $80
2981
2982            DEFW    L08ED                   ; 'link field'
2983
2984    L08FE:  DEFB    $03                     ; 'name length field'
2985
2986    L08FF:  DEFW    L0EC3                   ; 'code field' - docolon
2987
2988    ; ---
2989
2990    L0901:  DEFW    L08D2                   ; >R
2991    L0903:  DEFW    L0885                   ; swap
2992    L0905:  DEFW    L08DF                   ; R>
2993    L0907:  DEFW    L0885                   ; swap
```

```
2994   L0909:  DEFW    L04B6                   ; exit
2995
2996   ; ---------------
2997   ; THE 'OVER' WORD
2998   ; ---------------
2999   ; (n1, n2 -- n1, n2, n1)
3000
3001   L090B:  DEFM    "OVE"                   ; 'name field'
3002           DEFB    'R' + $80
3003
3004           DEFW    L08FE                   ; 'link field'
3005
3006   L0911:  DEFB    $04                     ; 'name length field'
3007
3008   L0912:  DEFW    L0EC3                   ; 'code field' - docolon
3009
3010   ; ---
3011
3012   L0914:  DEFW    L08D2                   ; >R
3013   L0916:  DEFW    L086B                   ; dup
3014   L0918:  DEFW    L08DF                   ; R>
3015   L091A:  DEFW    L0885                   ; swap
3016   L091C:  DEFW    L04B6                   ; exit
3017
3018   ; ---------------
3019   ; THE 'PICK' WORD
3020   ; ---------------
3021   ; (n1 -- n2)
3022   ; Copies the n1-th stack entry (after dropping n1 itself) to the top.
3023   ; Error 7 if n1 <= 0.
3024
3025   L091E:  DEFM    "PIC"                   ; 'name field'
3026           DEFB    'K' + $80
3027
3028           DEFW    L0911                   ; 'link field'
3029
3030   L0924:  DEFB    $04                     ; 'name length field'
3031
3032           DEFW    L0927                   ; 'code field'
3033
3034   ; ---
3035
3036   L0927:  CALL    L094D                   ;
3037           JP      (IY)                    ; to 'next'.
3038
3039   ; ---------------
3040   ; THE 'ROLL' WORD
3041   ; ---------------
3042   ; (n -- )
3043   ; Extracts the nth stack value to the top of the stack, after dropping n
3044   ; itself, and moves the remaining values down to fill the vacated position.
3045   ; Error 7 if n <= 0.
3046
3047   L092C:  DEFM    "ROL"                   ; 'name field'
3048           DEFB    'L' + $80
3049
3050           DEFW    L0924                   ; 'link field'
3051
3052   L0932:  DEFB    $04                     ; 'name length field'
3053
3054   L0933:  DEFW    L0935                   ; 'code field'
3055
3056   ; ---
3057
3058   L0935:  CALL    L094D                   ;
3059           EX      DE,HL
3060           LD      HL,($3C37)              ; STKBOT
3061           SBC     HL,DE
3062           JP      NC,L04D7                ; jump back to Error 2
3063
3064           LD      H,D
3065           LD      L,E
3066           INC     HL
```

```
3067            INC     HL
3068            LDIR
3069            LD      ($3C3B),DE              ; SPARE
3070            JP      (IY)                    ; to 'next'.
3071
3072    ; ---
3073
3074    L094D:  CALL    L084E                   ; stk_to_bc
3075            DEC     BC
3076            SLA     C
3077            RL      B
3078            INC     BC
3079            INC     BC
3080            JR      NC,L095B                ; skip the error routine
3081
3082            RST     20H                     ; Error 7
3083            DEFB    $07                     ; PICK or ROLL used with operand 0
3084                                            ; or negative
3085
3086    ; ---
3087
3088    L095B:  LD      HL,($3C3B)              ; SPARE
3089            SBC     HL,BC
3090            PUSH    HL
3091            LD      E,(HL)
3092            INC     HL
3093            LD      D,(HL)
3094            RST     10H                     ; push word DE
3095            POP     HL
3096            RET
3097
3098    ; ---------------
3099    ; THE 'TYPE' WORD
3100    ; ---------------
3101    ; (address, n -- )
3102    ; EMITs n characters from memory starting at the address.
3103
3104
3105    L0967:  DEFM    "TYP"                   ; 'name field'
3106            DEFB    'E' + $80
3107
3108            DEFW    L0932                   ; 'link field'
3109
3110    L096D:  DEFB    $04                     ; 'name length field'
3111
3112    L096E:  DEFW    L0970                   ; 'code field'
3113
3114    ; ---
3115
3116    L0970:  CALL    L084E                   ; stk_to_bc
3117            RST     18H                     ; pop word DE
3118            CALL    L097F                   ; routine pr_string (below)
3119
3120            JP      (IY)                    ; to 'next'.
3121
3122    ; -------------------------
3123    ; THE 'PRINT STRING' ROUTINE
3124    ; -------------------------
3125    ; The first entry point prints strings embedded in the Dictionary with the
3126    ; DE pointing to the preceding length word.
3127    ;
3128    ; The second entry point prints a string with length in BC and start in DE.
3129    ; It is called by TYPE above and to print comment fields.
3130
3131    ; ->
3132
3133    L0979:  LD      A,(DE)
3134            LD      C,A
3135            INC     DE
3136            LD      A,(DE)
3137            LD      B,A
3138            INC     DE
3139
```

```
3140   ; -->
3141   L097F:  LD      A,B
3142           OR      C
3143           RET     Z
3144
3145           LD      A,(DE)
3146           INC     DE
3147           DEC     BC
3148           RST     08H                     ; print_ch
3149
3150           JR      L097F                   ;
3151
3152   ; -------------
3153   ; THE '<#' WORD
3154   ; -------------
3155   ; (  --  )
3156   ; Initiates formatted output.
3157
3158   L0988:  DEFB    '<'                     ; 'name field'
3159           DEFB    '#' + $80
3160
3161           DEFW    L096D                   ; 'link field'
3162
3163   L098C:  DEFB    $02                     ; 'name length field'
3164
3165   L098D:  DEFW    L098F                   ; 'code field'
3166
3167   ; ---
3168
3169   L098F:  LD      HL,$27FF                ; end of pad
3170           LD      ($3C1A),HL              ; update system variable HLD
3171           JP      (IY)                    ; to 'next'.
3172
3173   ; -------------
3174   ; THE '#>' WORD
3175   ; -------------
3176   ; (ud -- address, n)
3177   ; Finishes formatted output, leaving the address and length (n) of the
3178   ; resultant string.
3179
3180   L0997:  DEFB    '#'                     ; 'name field'
3181           DEFB    '>' + $80
3182
3183           DEFW    L098C                   ; 'link field'
3184
3185   L099B:  DEFB    $02                     ; 'name length field'
3186
3187   L099C:  DEFW    L099E                   ; 'code field'
3188
3189   ; ---
3190
3191   L099E:  RST     18H                     ; pop word DE
3192           RST     18H                     ; pop word DE
3193           LD      DE,($3C1A)              ; HLD
3194           RST     10H                     ; push word DE (address)
3195           LD      HL,$27FF                ; end of pad.
3196           AND     A                       ; prepare to subtract.
3197           SBC     HL,DE                   ; find length of string.
3198           EX      DE,HL                   ; transfer to DE
3199           RST     10H                     ; push word DE (n)
3200
3201           JP      (IY)                    ; to 'next'.
3202
3203   ; ------------
3204   ; THE '.' WORD
3205   ; ------------
3206   ;
3207
3208   L09AF:  DEFB    '.' + $80               ; 'name field'
3209
3210           DEFW    L0A49                   ; 'link field'
3211
3212   L09B2:  DEFB    $01                     ; 'name length field'
```

```
3213
3214   L09B3:  DEFW    L0EC3                   ; 'code field' - docolon
3215
3216   ; ---
3217
3218   L09B5:  DEFW    L098D                   ; <#
3219           DEFW    L086B                   ; dup
3220           DEFW    L0C0D                   ; abs
3221           DEFW    L0688                   ; stk-zero
3222           DEFW    L09E1                   ; #s
3223           DEFW    L08FF                   ; rot
3224           DEFW    L0A4A                   ; sign
3225
3226   L09C3:  DEFW    L099C                   ; #>
3227           DEFW    L096E                   ; type
3228           DEFW    L0A73                   ; space
3229           DEFW    L04B6                   ; exit
3230
3231   ; -------------
3232   ; THE 'U.' WORD
3233   ; -------------
3234   ; (un -- )
3235   ; Prints the unsigned single length integer 'un' to the television screen,
3236   ; followed by a space.
3237
3238   L09CB:  DEFB    'U'                     ; 'name field'
3239           DEFB    '.' + $80
3240
3241           DEFW    L09B2                   ; 'link field'
3242
3243   L09CF:  DEFB    $02                     ; 'name length field'
3244
3245   L09D0:  DEFW    L0EC3                   ; 'code field' - docolon
3246
3247   ; ---
3248
3249   L09D2:  DEFW    L0688                   ; stk-zero
3250   L09D4:  DEFW    L098D                   ; <#
3251   L09D6:  DEFW    L09E1                   ; #S
3252   L09D8:  DEFW    L1276                   ; branch
3253   L09DA:  DEFW    $FFE8                   ; -> 09C3
3254
3255
3256   ; -------------
3257   ; THE '#S' WORD
3258   ; -------------
3259   ; (ud -- 0,0)
3260   ; Applies # repeatedly (at least once) until the double length number left
3261   ; on the stack is 0.
3262
3263   L09DC:  DEFB    '#'                     ; 'name field'
3264           DEFB    'S' + $80
3265
3266           DEFW    L09CF                   ; 'link field'
3267
3268   L09E0:  DEFB    $02                     ; 'name length field'
3269
3270   L09E1:  DEFW    L0EC3                   ; 'code field' - docolon
3271
3272   ; ---
3273
3274   L09E3:  DEFW    L09F7                   ; #
3275           DEFW    L0912                   ; over
3276           DEFW    L0912                   ; over
3277           DEFW    L0E36                   ; or
3278           DEFW    L0C1A                   ; 0=
3279           DEFW    L128D                   ; ?branch
3280
3281   L09EF:  DEFW    $FFF3                   ; back to L09E3
3282
3283           DEFW    L04B6                   ; exit
3284
3285   ; ------------
```

```
; THE '#' WORD
; ------------
; (ud1 -- ud2)
; used in formatted output. Generates one digit from the unsigned double
; length integer ud1 and holds it in the pad. The unsigned double length
; integer ud2 is the quotient when ud1 is divided by the number base.

L09F3:  DEFB    '#' + $80               ; 'name field'

        DEFW    L09E0                   ; 'link field'

L09F6:  DEFB    $01                     ; 'name length field'

L09F7:  DEFW    L0EC3                   ; 'code field' - docolon

; ---

L09F9:  DEFW    L048A                   ; get base
L09FB:  DEFW    L0896                   ; C@
L09FD:  DEFW    L0CC4                   ; div?
L09FF:  DEFW    L08FF                   ; rot
L0A01:  DEFW    L0A07                   ; stk-char
L0A03:  DEFW    L0A5C                   ; hold
L0A05:  DEFW    L04B6                   ; exit

; ----------------------------
; The 'stk-char' Internal Word
; ----------------------------
; used from above thread.

L0A07:  DEFW    L0A09                   ; headerless 'code field'

; ---

L0A09:  RST     18H                     ; data stack to DE
        LD      A,E                     ; character to A
        ADD     A,$30                   ; convert digit to ASCII
        CP      $3A                     ; compare to '9'
        JR      C,L0A13                 ; forward if digit
        ADD     A,$07                   ; else add for hex

L0A13:  LD      E,A                     ; back to E
        RST     10H                     ; push ASCII on data stack.
        JP      (IY)                    ; to 'next'.

; --------------
; THE 'CLS' WORD
; --------------
; ( -- )
; Clears the screen and sets the print position to the top left of
; the screen.

L0A17:  DEFM    "CL"                    ; 'name field'
        DEFB    'S' + $80

        DEFW    L09F6                   ; 'link field'

L0A1C:  DEFB    $03                     ; 'name length field'

        DEFW    L0A1F                   ; 'code field'

; ---

L0A1F:  CALL    L0A24                   ; routine CLS below.

        JP      (IY)                    ; to 'next'.


; --------------------
; THE 'CLS' SUBROUTINE
; --------------------
; Called from the 'CLS' word definition above and also from the initialization
; routine.
```

```
3359
3360   L0A24:  LD      DE,$26FF                ; point destination to end of video
3361                                           ; memory.
3362           LD      HL,($3C24)              ; set HL to first byte of input buffer
3363                                           ; from system variable L_HALF.
3364                                           ; (at initialization $26E0).
3365
3366           LD      BC,$0020                ; set count to thirty two.
3367
3368           ADD     HL,BC                   ; add to the low address.
3369           DEC     HL                      ; step back and
3370           LDDR                            ; copy the 32 bytes.
3371
3372   ; while BC is zero, set the plotting coordinates.
3373
3374           LD      ($3C2F),BC              ; set XCOORD and YCOORD to zero.
3375
3376   ; set the screen position to the start of video memory.
3377
3378           LD      HL,$2400                ; start of the 768 bytes of video RAM.
3379           LD      ($3C1C),HL              ; set system variable SCRPOS.
3380
3381           INC     DE                      ; the byte before logical line.
3382           EX      DE,HL                   ; transfer to HL.
3383           LD      ($3C24),HL              ; set L_HALF.
3384           JP      L07FA                   ; jump back to fill the locations
3385                                           ; from DE to HL -1 with spaces.
3386
3387   ; ---------------
3388   ; THE 'SIGN' WORD
3389   ; ---------------
3390   ; (n -- )
3391   ; In formatted output, holds a minus sign in the pad if n is negative.
3392
3393
3394   L0A43:  DEFM    "SIG"                   ; 'name field'
3395           DEFB    'N' + $80
3396
3397           DEFW    L099B                   ; 'link field'
3398
3399   L0A49:  DEFB    $04                     ; 'name length field'
3400
3401   L0A4A:  DEFW    L0A4C                   ; 'code field'
3402
3403   ; ---
3404
3405   L0A4C:  RST     18H                     ; pop word DE
3406           RL      D                       ; test sign bit
3407           LD      E,$2D                   ; prepare a '-'
3408           JR      C,L0A5F                 ; forward if minus
3409           JP      (IY)                    ; to 'next'.
3410
3411   ; ---------------
3412   ; THE 'HOLD' WORD
3413   ; ---------------
3414   ; (character -- )
3415   ; Used in formatted output to hold the character in the pad.
3416
3417   L0A55:  DEFM    "HOL"                   ; 'name field'
3418           DEFB    'D' + $80
3419
3420   L0A59:  DEFW    L0A1C                   ; 'link field'
3421
3422   L0A5B:  DEFB    $04                     ; 'name length field'
3423
3424   L0A5C:  DEFW    L0A5E                   ; 'code field'
3425
3426   ; ---
3427
3428   L0A5E:  RST     18H                     ; data stack to DE
3429
3430   L0A5F:  LD      HL,($3C1A)              ; HLD
3431           DEC     L
```

```
3432            JR      Z,L0A69                 ; forward when full
3433
3434            LD      ($3C1A),HL              ; update HLD
3435            LD      (HL),E                  ; and place character in buffer
3436
3437    L0A69:  JP      (IY)                    ; to 'next'.
3438
3439    ; ----------------
3440    ; THE 'SPACE' WORD
3441    ; ----------------
3442    ; (  --  )
3443    ; EMITs a space.
3444
3445    L0A6B:  DEFM    "SPAC"                  ; 'name field'
3446            DEFB    'E' + $80
3447
3448            DEFW    L0A5B                   ; 'link field'
3449
3450    L0A72:  DEFB    $05                     ; 'name length field'
3451
3452    L0A73:  DEFW    L0A75                   ; 'code field'
3453
3454    ; ---
3455
3456    L0A75:  LD      A,$20                   ; load accumulator with the ASCII
3457                                            ; code for space.
3458            RST     08H                     ; print_ch
3459
3460    L0A78:  JP      (IY)                    ; to 'next'.
3461
3462    ; -----------------
3463    ; THE 'SPACES' WORD
3464    ; -----------------
3465    ; (n -- )
3466    ; EMITs n spaces if n >= 1.
3467
3468    L0A7A:  DEFM    "SPACE"                 ; 'name field'
3469            DEFB    'S' + $80
3470
3471            DEFW    L0A72                   ; 'link field'
3472
3473    L0A82:  DEFB    $06                     ; 'name length field'
3474
3475            DEFW    L0A85                   ; 'code field'
3476
3477    ; ---
3478
3479    L0A85:  RST     18H                     ; fetch stack data to DE
3480
3481    L0A86:  DEC     DE                      ; decrement the counter.
3482            BIT     7,D                     ; test for a negative value
3483            JR      NZ,L0A78                ; back to a jp iy  when done    >>
3484
3485            LD      A,$20                   ; prepare a space
3486            RST     08H                     ; print it
3487            JR      L0A86                   ; loop back for more.
3488
3489    ; -------------
3490    ; THE 'CR' WORD
3491    ; -------------
3492    ; Outputs a carriage return character to the television.
3493
3494    L0A90:  DEFB    'C'                     ; 'name field'
3495            DEFB    'R' + $80
3496
3497            DEFW    L0A82                   ; 'link field'
3498
3499    L0A94:  DEFB    $02                     ; 'name length field'
3500
3501    L0A95:  DEFW    L0A97                   ; 'code field'
3502
3503    ; ---
3504
```

```
3505  L0A97:  LD      A,$0D                   ; prepare a CR
3506          RST     08H                     ; print it.
3507
3508          JP      (IY)                    ; to 'next'.
3509
3510  ; ---------------
3511  ; THE 'EMIT' WORD
3512  ; ---------------
3513  ; (character -- )
3514  ; writes the character to the television screen.
3515
3516  L0A9C:  DEFM    "EMI"                   ; 'name field'
3517          DEFB    'T' + $80
3518
3519          DEFW    L0A94                   ; 'link field'
3520
3521  L0AA2:  DEFB    $04                     ; 'name length field'
3522
3523  L0AA3:  DEFW    L0AA5                   ; 'code field'
3524
3525  ; ---
3526
3527  L0AA5:  RST     18H                     ; pop de off data stack
3528          LD      A,E                     ; character to A
3529          RST     08H                     ; print it.
3530
3531          JP      (IY)                    ; to 'next'.
3532
3533  X009B:  LD      SP,($3C18)              ; set stack-pointer to RAMTOP.
3534      IM  1
3535      EI
3536
3537      CALL    L02D8           ; SET_BUF
3538      CALL    L0276           ; pr_cursor
3539
3540      LD  HL,BCMD
3541  BCMDLP: LD  A,(HL)
3542      AND A
3543      JR  Z,BCMDINT
3544      PUSH    HL
3545      CALL    L0196
3546      CALL    L0282
3547      POP HL
3548      INC HL
3549      JR  BCMDLP
3550
3551  BCMDINT:CALL    L0225               ; routine DEL-CURSOR
3552          CALL    DO_FORTH
3553      DEFW    L0506           ; interpret buffer
3554      DEFW    L0536           ; print OK
3555          DEFW    F_FORTH2ASM         ; end-forth.
3556
3557      JP  L04F2           ; jump forward to the main execution
3558                                      ; loop.
3559
3560  BCMD:   DEFM    'INIT ALOAD INIT'
3561      DEFB    0
3562
3563  ; ------------
3564  ; THE 'V' WORD
3565  ; ------------
3566
3567  N_V:
3568      .BYTE   'V' | INVERSE       ; Word Name (last letter inverse)
3569          .WORD   L_BSAVE                 ; Link Field
3570  L_V:
3571      .BYTE   $ - N_V - 2             ; Name Length Field
3572  ;;; --- code ---
3573          DEFW    F_DOCOLON
3574      DEFW    F_FLUSH
3575      DEFW    F_FORTH2ASM
3576
3577      CALL    WRD2PAD
```

```
3578        JR  Z,LVDR2
3579        CALL    WRBLK           ; send contents of PAD
3580    LVDRL:  CALL    WRD2PAD
3581        JR  Z,LVDR2
3582
3583        LD  A,$20
3584        CALL    WRCHR
3585        CALL    WRBLK           ; send contents of PAD
3586        JR  LVDRL           ; next word
3587
3588    LVDR2:  CALL    SENDCR
3589        LD  A,$0D
3590        JP  IPRLP
3591
3592        DEFS    $0B14 - $       ; spare
3593
3594    ; -------------
3595    ; THE 'AT' WORD
3596    ; -------------
3597    ; (line, column -- )
3598    ; Sets print position to line and column numbers on the stack.
3599    ; There are 23 lines (0 to 22) and 32 columns (0 to 31). The
3600    ; column number is taken modulo 32, and ERROR 9 if trying to print
3601    ; in the input buffer at the bottom.
3602
3603    L0B14:  DEFB    'A'                     ; 'name field'
3604            DEFB    'T' + $80
3605
3606        DEFW    L0AA2
3607
3608    L0B18:  DEFB    $02                     ; 'name length field'
3609
3610            DEFW    L0B1B                   ; 'code field'
3611
3612    ; ---
3613
3614    L0B1B:  RST     18H                     ; pop word DE
3615
3616            CALL    L084E                   ; stk_to_bc
3617
3618            LD      A,C
3619
3620            CALL    L0B28                   ;
3621
3622            LD      ($3C1C),HL              ; update system variable SCRPOS
3623
3624            JP      (IY)                    ; to 'next'.
3625
3626    ; ---
3627
3628    ; plotsub
3629
3630    L0B28:  ADD     A,$20
3631            LD      L,A
3632            LD      H,$01
3633            ADD     HL,HL
3634            ADD     HL,HL
3635            ADD     HL,HL
3636            ADD     HL,HL
3637            ADD     HL,HL
3638            LD      D,$00
3639            LD      A,E
3640            AND     $1F
3641            LD      E,A
3642            ADD     HL,DE
3643            LD      DE,($3C24)              ; fetch start of lower half from L_HALF
3644            SBC     HL,DE
3645            ADD     HL,DE
3646            RET     C
3647
3648    ;
3649
3650            RST     20H                     ; Error 9
```

```
3651            DEFB    $09                     ; Erroneous 'AT' Command.
3652
3653    ; ---------------
3654    ; THE 'PLOT' WORD
3655    ; ---------------
3656    ; (x, y, n -- )
3657    ; Plots pixel (x, y) with plot mode n.
3658    ; n =   0       unplot
3659    ;       1       plot
3660    ;       2       move
3661    ;       3       change
3662    ; If n>3, takes value modulo 4.
3663
3664    L0B43:  DEFM    "PLO"                   ; 'name field'
3665            DEFB    'T' + $80
3666
3667            DEFW    L0B18                   ; 'link field'
3668
3669    L0B49:  DEFB    $04                     ; 'name length field'
3670
3671            DEFW    L0B4C                   ; 'code field'
3672
3673    ; ---
3674
3675    L0B4C:  CALL    L084E                   ; stk_to_bc
3676
3677            RST     18H                     ; pop word DE
3678            LD      (IX+$30),E              ; YCOORD
3679            SRL     E
3680            RL      C
3681            LD      A,$16                   ; 24
3682            SUB     E
3683
3684            RST     18H                     ; pop word DE
3685            LD      (IX+$2F),E              ; XCOORD
3686            SRL     E
3687            RL      C
3688
3689            CALL    L0B28                   ;
3690
3691            LD      A,(HL)
3692            AND     $78                     ; 01111000
3693            CP      $10
3694            LD      A,(HL)
3695            JR      Z,L0B6F                 ;
3696
3697            LD      A,$10
3698
3699    L0B6F:  LD      E,A
3700            LD      D,$87
3701            LD      A,C
3702            AND     $03
3703            LD      B,A
3704            JR      Z,L0B7F                 ;
3705
3706            CPL
3707
3708            ADD     A,$02
3709            ADC     A,$03
3710            LD      D,A
3711            LD      B,E
3712    L0B7F:  LD      A,C
3713            RRCA
3714            RRCA
3715            RRCA
3716            SBC     A,A
3717            BIT     3,C
3718            JR      NZ,L0B8C                ;
3719            XOR     E
3720            RLCA
3721            SBC     A,A
3722            XOR     B
3723
```

```
3724  L0B8C:  AND     D
3725          XOR     E
3726          LD      (HL),A
3727          JP      (IY)                    ; to 'next'.
3728
3729  ; ---------------
3730  ; THE 'BEEP' WORD
3731  ; ---------------
3732  ; ( m, n --  )
3733  ; Plays a note on the loudspeaker. 8 * m = period in microseconds,
3734  ; n = time in milliseconds.
3735
3736  L0B91:  DEFM    "BEE"                   ; 'name field'
3737          DEFB    'P' + $80
3738
3739          DEFW    L0B49                   ; 'link field'
3740
3741  L0B97:  DEFB    $04                     ; 'name length field'
3742
3743          DEFW    L0EC3                   ; 'code field'  m, n.
3744
3745  ; ---
3746
3747  L0B9A:  DEFW    L0912                   ; OVER          m, n, m.
3748          DEFW    L104B                   ; stk-data      m, n, m, 125.
3749          DEFB    $7D                     ;  (125)
3750          DEFW    L0885                   ; SWAP          m, n, 125, m.
3751          DEFW    L0D7A                   ; */            m, (n*125)/m
3752          DEFW    L1A0E                   ; end
3753
3754  ; ---
3755
3756  L0BA5:  RST     18H                     ; pop word DE
3757
3758          CALL    L084E                   ; stk_to_bc
3759
3760          LD      HL,$00F9                ;
3761          ADD     HL,BC                   ;
3762          INC     L                       ;
3763
3764          DI                              ; Disable Interrupts.
3765
3766  L0BAF:  LD      A,$7F                   ; place $7FFE on address bus and read
3767          IN      A,($FE)                 ; from port, pushing the loudspeaker
3768                                          ; diaphragm in.
3769
3770          RRCA                            ; test the read 'SPACE' key bit.
3771
3772          JR      NC,L0BC7                ; forward if BREAK pressed.
3773
3774          CALL    L0BC9                   ; routine delay_HL
3775
3776          DEC     DE                      ; decrement counter.
3777
3778          LD      A,D                     ; all even addresses are reserved for
3779                                          ; Jupiter Ace so any value does for the
3780                                          ; high order byte. $FE is low value.
3781
3782          OUT     ($FE),A                 ; push the loudspeaker diaphragm out.
3783
3784          CALL    L0BC9                   ; routine delay_HL
3785
3786          OR      E                       ; test for counter DE reaching zero.
3787          JP      NZ,L0BAF                ; loop back if not.
3788
3789          EI                              ; Enable Interrupts.
3790
3791          JP      (IY)                    ; to 'next'.
3792
3793  ; ---
3794
3795  L0BC7:  RST     20H                     ; Error 3
3796          DEFB    $03                     ; BREAK pressed.
```

```
3797
3798    ; --------------------------
3799    ; THE 'BEEP DELAY' SUBROUTINE
3800    ; --------------------------
3801    ; called twice from the above BEEP routine.
3802
3803    L0BC9:  LD      B,L                     ; transfer the value of
3804            LD      C,H                     ; the HL register to BC.
3805
3806    L0BCB:  DJNZ    L0BCB                   ; self-loop for B times
3807
3808            DEC     B                       ; set B to $FF for future loops
3809            DEC     C                       ; decrement outer loop counter C
3810            JP      NZ,L0BCB                ; JUMP back if not zero          (10)
3811
3812            RET                             ; return
3813
3814    ; ---------------
3815    ; THE 'INKEY' WORD
3816    ; ---------------
3817    ; ( -- ASCII code)
3818    ; Reads the keyboard. Puts ASCII value on the stack if a key is pressed, 0
3819    ; otherwise.
3820
3821
3822    L0BD3:  DEFM    "INKE"                  ; 'name field'
3823            DEFB    'Y' + $80
3824
3825            DEFW    L0B97                   ; 'link field'
3826
3827    L0BDA:  DEFB    $05                     ; 'name length field'
3828
3829    L0BDB:  DEFW    L0BDD                   ; 'code field'
3830
3831    ; ---
3832
3833    L0BDD:  CALL    L0336                   ; routine KEY-SCAN
3834
3835            LD      E,A                     ; transfer the key code to E.
3836            LD      D,$00                   ; make high order byte zero.
3837
3838            RST     10H                     ; stack Data Word DE
3839
3840            JP      (IY)                    ; to 'next'.
3841
3842    ; -------------
3843    ; THE 'IN' WORD
3844    ; -------------
3845    ; (port address -- data byte)
3846    ; Inputs a data byte from an I/O port.
3847
3848    L0BE6:  DEFB    'I'                     ; 'name field'
3849            DEFB    'N' + $80
3850
3851            DEFW    L0BDA                   ; 'link field'
3852
3853    L0BEA:  DEFB    $02                     ; 'name length field'
3854
3855            DEFW    L0BED                   ; 'code field'
3856
3857    ; ---
3858
3859    L0BED:  CALL    L084E                   ; stk_to_bc
3860            LD      D,$00                   ; make high order byte zero.
3861
3862            IN      E,(C)                   ; read the port to E.
3863
3864            RST     10H                     ; stack Data Word DE.
3865
3866    L0BF5:  JP      (IY)                    ; to 'next'.
3867
3868    ; --------------
3869    ; THE 'OUT' WORD
```

```
; --------------
; (data byte, port address -- )
; Outputs a data byte to an I/O port.

L0BF7:  DEFM    "OU"                    ; 'name field'
        DEFB    'T' + $80

        DEFW    L0BEA                   ; 'link field'

L0BFC:  DEFB    $03                     ; 'name length field'

        DEFW    L0BFF                   ; 'code field'

; ---

L0BFF:  CALL    L084E                   ; stk_to_bc
                                        ; all 16 bits are placed on the
                                        ; Z80A address bus.
        RST     18H                     ; pop word DE

        OUT     (C),E                   ; output byte to port address.

        JP      (IY)                    ; to 'next'.

; --------------
; THE 'ABS' WORD
; --------------
; (n -- absolute value of n)

L0C07:  DEFM    "AB"                    ; 'name field'
        DEFB    'S' + $80

        DEFW    L0BFC                   ; 'link field'

L0C0C:  DEFB    $03                     ; 'name length field'

L0C0D:  DEFW    L0EC3                   ; 'code field' - docolon

; ---

        DEFW    L086B                   ; DUP
        DEFW    L0D94                   ; pos
        DEFW    L04B6                   ; EXIT

; -------------
; THE '0=' WORD
; -------------
; (n -- flag)
; flag is 1 in n = 0.

L0C15:  DEFB    '0'                     ; 'name field'
        DEFB    '=' + $80

        DEFW    L0C0C                   ; 'link field'

L0C19:  DEFB    $02                     ; 'name length field'

L0C1A:  DEFW    L0C1C                   ; 'code field'

; ---

L0C1C:  RST     18H                     ; pop word DE
        LD      A,D                     ; test for
        OR      E                       ; zero
        CP      $01                     ; sets carry if word is zero

; -> zero_or_one

L0C21:  LD      A,$00                   ; make accumulator zero.
        LD      D,A                     ; set D to zero
        RLA                             ; pick up carry (1/0)
        LD      E,A                     ; set DE to one or zero
        RST     10H                     ; push word DE
```

```
3943
3944            JP      (IY)                    ; to 'next'.
3945
3946    ; -------------
3947    ; THE '0<' WORD
3948    ; -------------
3949    ; (n -- flag)
3950    ; flag is 1 if n is negative
3951
3952    L0C29:  DEFB    '0'                     ; 'name field'
3953            DEFB    '<' + $80
3954
3955            DEFW    L0C19                   ; 'link field'
3956
3957    L0C2D:  DEFB    $02                     ; 'name length field'
3958
3959    L0C2E:  DEFW    L0C30                   ; 'code field'
3960
3961    ; ---
3962
3963    L0C30:  RST     18H                     ; pop word DE
3964            RL      D                       ; test the sign bit.
3965
3966            JR      L0C21                   ; back to above routine to stack the
3967                                            ; carry as one (true) or zero (false).
3968
3969    ; -------------
3970    ; THE '0>' WORD
3971    ; -------------
3972    ; (n -- flag)
3973    ; flag is 1 if n is positive.
3974
3975
3976    L0C35:  DEFB    '0'                     ; 'name field'
3977            DEFB    '>' + $80
3978
3979            DEFW    L0C2D                   ; 'link field'
3980
3981    L0C39:  DEFB    $02                     ; 'name length field'
3982
3983    L0C3A:  DEFW    L0C3C                   ; 'code field'
3984
3985    ; ---
3986
3987    L0C3C:  RST     18H                     ; pop word DE
3988            LD      A,D
3989            OR      E
3990            JR      Z,L0C21                 ; to stack word one or zero
3991
3992            RL      D
3993            CCF
3994            JR      L0C21                   ; to stack word one or zero
3995
3996    ; ------------
3997    ; THE '=' WORD
3998    ; ------------
3999    ; (n1, n2 -- flag)
4000    ; flag is 1 if n1=n2.
4001
4002    L0C46:  DEFB    '=' + $80               ; 'name field'
4003
4004            DEFW    L0C39                   ; 'link field'
4005
4006    L0C49:  DEFB    $01                     ; 'name length field'
4007
4008    L0C4A:  DEFW    L0EC3                   ; 'code field' - docolon
4009
4010    ; ---
4011
4012    L0C4C:  DEFW    L0DE1                   ; -
4013            DEFW    L0C1A                   ; 0=
4014            DEFW    L04B6                   ; exit
4015
```

```
4016   ; ------------
4017   ; THE '>' WORD
4018   ; ------------
4019   ; (n1, n2 -- flag)
4020   ; flag is 1 if n1>n2.
4021
4022   L0C52:  DEFB    '>' + $80               ; 'name field'
4023
4024           DEFW    L0C49                   ; 'link field'
4025
4026   L0C55:  DEFB    $01                     ; 'name length field'
4027
4028   L0C56:  DEFW    L0C58                   ; 'code field'
4029
4030   ; ---
4031
4032   L0C58:  RST     18H                     ; pop word DE
4033           PUSH    DE                      ;
4034           RST     18H                     ; pop word DE
4035           POP     HL                      ;
4036
4037           CALL    L0C99                   ;
4038
4039           JR      L0C21                   ; to stack word one or zero
4040
4041   ; ------------
4042   ; THE '<' WORD
4043   ; ------------
4044   ; (n1, n2 -- flag)
4045   ; flag is 1 if n1 < n2.
4046
4047   L0C61:  DEFB    '<' + $80               ; 'name field'
4048
4049           DEFW    L0C55                   ; 'link field'
4050
4051   L0C64:  DEFB    $01                     ; 'name length field'
4052
4053   L0C65:  DEFW    L0EC3                   ; 'code field' - docolon
4054
4055   ; ---
4056
4057           DEFW    L0885                   ; swap
4058           DEFW    L0C56                   ; >
4059           DEFW    L04B6                   ; exit
4060
4061
4062   ; -------------
4063   ; THE 'U<' WORD
4064   ; -------------
4065   ; (un1, un2 -- flag)
4066   ; The flag is 1 if, of the two unsigned single length integers, un1 is less
4067   ; than un2.
4068
4069   L0C6D:  DEFB    'U'                     ; 'name field'
4070           DEFB    '<' + $80
4071
4072           DEFW    L0C64                   ; 'link field'
4073
4074   L0C71:  DEFB    $02                     ; 'name length field'
4075
4076   L0C72:  DEFW    L0C74                   ; 'code field'
4077
4078   ; ---
4079
4080   L0C74:  CALL    L084E                   ; stk_to_bc
4081
4082   L0C77:  RST     18H                     ; pop word DE
4083           EX      DE,HL
4084           AND     A
4085           SBC     HL,BC
4086           JR      L0C21                   ; to stack word one or zero
4087
4088   ; -------------
```

```
4089  ; THE 'D<' WORD
4090  ; -------------
4091  ; (d1, d2 -- flag)
4092  ; flag is 1 if the signed double integer, d1 < d2.
4093
4094  L0C7E:  DEFB    'D'                     ; 'name field'
4095          DEFB    '<' + $80
4096
4097          DEFW    L0C71                   ; 'link field'
4098
4099  L0C82:  DEFB    $02                     ; 'name length field'
4100
4101  L0C83:  DEFW    L0C85                   ; 'code field'
4102
4103  ; ---
4104
4105  L0C85:  RST     18H                     ; pop word DE
4106          PUSH    DE
4107          CALL    L084E                   ; stk_to_bc
4108          RST     18H                     ; pop word DE
4109          POP     HL
4110          AND     A
4111          SBC     HL,DE
4112          JR      Z,L0C77                 ;
4113
4114          ADD     HL,DE
4115          EX      DE,HL
4116
4117          CALL    L0C99                   ;
4118
4119          RST     18H                     ; pop word DE
4120          JR      L0C21                   ; to stack word one or zero
4121
4122  ; ---
4123  ; THE 'sign?' SUBROUTINE
4124  ; ---
4125
4126  L0C99:  LD      A,H
4127          XOR     D
4128          JP      M,L0CA0                 ;
4129
4130          SBC     HL,DE
4131
4132  L0CA0:  RL      H
4133          RET
4134
4135  ; -------------
4136  ; THE 'U*' WORD
4137  ; -------------
4138  ; (un1, un2 -- double length(un1 * un2))
4139  ; Multiplies two unsigned single length integers to give an unsigned
4140  ; double length product.
4141
4142  L0CA3:  DEFB    'U'                     ; 'name field'
4143          DEFB    '*' + $80
4144
4145          DEFW    L0C82                   ; 'link field'
4146
4147  L0CA7:  DEFB    $02                     ; 'name length field'
4148
4149  L0CA8:  DEFW    L0CAA                   ; 'code field'
4150
4151  ; => mult
4152
4153  L0CAA:  RST     18H                     ; pop word DE
4154          CALL    L084E                   ; stk_to_bc
4155          LD      HL,$0000
4156          LD      A,$10
4157  L0CB3:  ADD     HL,HL
4158          EX      DE,HL
4159          ADC     HL,HL
4160          EX      DE,HL
4161          JR      NC,L0CBE                ;
```

```
4162
4163            ADD     HL,BC
4164            JR      NC,L0CBE                ;
4165
4166            INC     DE
4167
4168    L0CBE:  DEC     A
4169            JR      NZ,L0CB3                ;
4170
4171            EX      DE,HL
4172            JR      L0CF3                   ;
4173
4174    ; ---
4175    ; The 'div?' Internal Word
4176    ; ---
4177
4178    L0CC4:  DEFW    L0CC6
4179
4180    L0CC6:  RST     18H                     ; pop word DE
4181            EXX
4182            RST     18H                     ; pop word DE
4183            PUSH    DE
4184            RST     18H                     ; pop word DE
4185            POP     HL
4186            LD      A,H
4187            OR      L
4188            LD      A,$21                   ; 33
4189            JR      NZ,L0CD5                ;
4190
4191            EX      DE,HL
4192            LD      A,$11                   ; 17
4193
4194    L0CD5:  EXX
4195            LD      B,A
4196            XOR     A
4197            LD      H,A
4198            LD      L,A
4199            LD      C,A
4200
4201    L0CDB:  ADC     HL,HL
4202            SBC     A,A
4203            AND     A
4204            SBC     HL,DE
4205            SBC     A,C
4206            JR      NC,L0CE5                ;
4207            ADD     HL,DE
4208
4209    L0CE5:  CCF
4210            EXX
4211            EX      DE,HL
4212            ADC     HL,HL
4213            EX      DE,HL
4214            ADC     HL,HL
4215            EXX
4216            DJNZ    L0CDB                   ;
4217
4218            EX      DE,HL
4219            RST     10H                     ; push word DE
4220            EXX
4221
4222    L0CF3:  PUSH    HL
4223            RST     10H                     ; push word DE
4224            POP     DE
4225            RST     10H                     ; push word DE
4226
4227            JP      (IY)                    ; to 'next'.
4228
4229    ; ---------------
4230    ; THE '/MOD' WORD
4231    ; ---------------
4232    ; (n1, n2 -- remainder, quotient of n1/n2)
4233    ; The remainder has the same sign as the dividend n1.
4234
```

```
4235   L0CF9:  DEFM    "/MO"                   ; 'name field'
4236           DEFB    'D' + $80
4237
4238           DEFW    L0CA7                   ; 'link field'
4239
4240   L0CFF:  DEFB    $04                     ; 'name length field'
4241
4242   L0D00:  DEFW    L0EC3                   ; 'code field' - docolon
4243
4244   ; ---
4245
4246   L0D02:  DEFW    L0885                   ; swap
4247           DEFW    L08D2                   ; >R
4248           DEFW    L12E9                   ; I
4249           DEFW    L0C0D                   ; abs
4250           DEFW    L104B                   ; stk_data
4251           DEFB    $00                     ; zero
4252   ; ->
4253   L0D0D:  DEFW    L08FF                   ; rot
4254           DEFW    L086B                   ; dup
4255           DEFW    L12E9                   ; I
4256           DEFW    L0E60                   ; xor
4257           DEFW    L08D2                   ; >R
4258           DEFW    L0C0D                   ; abs
4259           DEFW    L0D8C                   ; U/MOD
4260           DEFW    L08DF                   ; >R
4261           DEFW    L0D94                   ; pos
4262           DEFW    L0885                   ; swap
4263           DEFW    L08DF                   ; >R
4264           DEFW    L0D94                   ; pos
4265           DEFW    L0885                   ; swap
4266           DEFW    L04B6                   ; exit
4267
4268   ; ----------------
4269   ; THE '*/MOD' WORD
4270   ; ----------------
4271   ; (n1, n2, n3 -- remainder, quotient of (n1 * n2)/n3)
4272   ; As in */, n1 * n2 is held to double length.
4273
4274   L0D29:  DEFM    "*/MO"                  ; 'name field'
4275           DEFB    'D' + $80
4276
4277           DEFW    L0CFF                   ; 'link field'
4278
4279   L0D30:  DEFB    $05                     ; 'name length field'
4280
4281   L0D31:  DEFW    L0EC3                   ; 'code field' - docolon
4282
4283   ; ---
4284
4285           DEFW    L08FF                   ; rot
4286           DEFW    L08D2                   ; >R
4287           DEFW    L12E9                   ; I
4288           DEFW    L0C0D                   ; abs
4289           DEFW    L08FF                   ; rot
4290           DEFW    L086B                   ; dup
4291           DEFW    L08DF                   ; >R
4292           DEFW    L0E60                   ; xor
4293           DEFW    L08D2                   ; >R
4294           DEFW    L0C0D                   ; abs
4295           DEFW    L0CA8                   ; u*
4296           DEFW    L1276                   ; branch
4297
4298   L0D4B:  DEFW    $FFC1                   ; back to L0D0D  (in /MOD)
4299
4300
4301
4302
4303   ; ------------
4304   ; THE '/' WORD
4305   ; ------------
4306   ; (n1, n2 -- n1/n2)
4307   ; Single length signed integer division.
```

```
4308
4309   L0D4D:  DEFB    '/' + $80               ; 'name field'
4310
4311           DEFW    L0D30                   ; 'link field'
4312
4313   L0D50:  DEFB    $01                     ; 'name length field'
4314
4315   L0D51:  DEFW    L0EC3                   ; 'code field' - docolon
4316
4317   ; ---
4318
4319   L0D53:  DEFW    L0D00                   ; /MOD
4320           DEFW    L0885                   ; swap
4321           DEFW    L0879                   ; drop
4322           DEFW    L04B6                   ; exit
4323
4324   ; --------------
4325   ; THE 'MOD' WORD
4326   ; --------------
4327   ; (n1, n2 -- remainder n1/n2)
4328   ; The remainder has the same sign as the dividend.
4329
4330   L0D5B:  DEFM    "MO"                    ; 'name field'
4331           DEFB    'D' + $80
4332
4333           DEFW    L0D50                   ; 'link field'
4334
4335   L0D60:  DEFB    $03                     ; 'name length field'
4336
4337   L0D61:  DEFW    L0EC3                   ; 'code field' - docolon
4338
4339   ; ---
4340
4341           DEFW    L0D00                   ; /MOD
4342           DEFW    L0879                   ; drop
4343           DEFW    L04B6                   ; exit
4344
4345
4346   ; ------------
4347   ; THE '*' WORD
4348   ; ------------
4349   ; (n1, n2 -- n1*n2)
4350
4351   L0D69:  DEFB    '*' + $80               ; 'name field'
4352
4353           DEFW    L0D60                   ; 'link field'
4354
4355   L0D6C:  DEFB    $01                     ; 'name length field'
4356
4357           DEFW    L0EC3                   ; 'code field' - docolon
4358
4359   ; ---
4360
4361           DEFW    L0CA8                   ; u*
4362           DEFW    L0879                   ; drop
4363           DEFW    L04B6                   ; exit
4364
4365
4366   ; -------------
4367   ; THE '*/' WORD
4368   ; -------------
4369   ; (n1, n2, n3 -- (n1*n2)/n3)
4370   ; The intermediate product n1*n2 is held to double length.
4371
4372   L0D75:  DEFB    '*'                     ; 'name field'
4373           DEFB    '/' + $80
4374
4375           DEFW    L0D6C                   ; 'link field'
4376
4377   L0D79:  DEFB    $02                     ; 'name length field'
4378
4379   L0D7A:  DEFW    L0EC3                   ; 'code field' - docolon
4380
```

```
4381   ; ---
4382
4383           DEFW    L0D31                   ; */MOD
4384           DEFW    L0885                   ; swap
4385           DEFW    L0879                   ; drop
4386           DEFW    L04B6                   ; exit
4387
4388   ; --------------
4389   ; THE 'U/MOD' WORD
4390   ; --------------
4391   ; (ud1, un2 -- un3, un4)
4392   ; In unsigned arithmetic throughout, divides the double length integer ud1
4393   ; by the single length integer un2 to give a single length remainder un3
4394   ; and a single length quotient un4.
4395
4396   L0D84:  DEFM    "U/MO"                  ; 'name field'
4397           DEFB    'D' + $80
4398
4399           DEFW    L0D79                   ; 'link field'
4400
4401   L0D8B:  DEFB    $05                     ; 'name length field'
4402
4403   L0D8C:  DEFW    L0EC3                   ; 'code field' - docolon
4404
4405   ; ---
4406
4407   L0D8E:  DEFW    L0CC4                   ; div?
4408           DEFW    L0879                   ; drop
4409           DEFW    L04B6                   ; exit
4410
4411   ; ---
4412
4413   ; make positive
4414
4415   L0D94:  DEFW    L0EC3                   ; 'code field' - docolon
4416
4417   ; ---
4418
4419   L0D96:  DEFW    L0C2E                   ; 0<
4420           DEFW    L1283                   ; ?branch               (if false)
4421   L0D9A:  DEFW    $0003                   ; to L0D9E
4422
4423           DEFW    L0DA9                   ; negate
4424
4425   L0D9E:  DEFW    L04B6                   ; exit
4426
4427   ; -----------------
4428   ; THE 'NEGATE' WORD
4429   ; -----------------
4430   ; (n -- -n)
4431
4432
4433   L0DA0:  DEFM    "NEGAT"                 ; 'name field'
4434           DEFB    'E' +$80
4435
4436           DEFW    L0D8B                   ; 'link field'
4437
4438   L0DA8:  DEFB    $06                     ; 'name length field'
4439
4440   L0DA9:  DEFW    L0DAB                   ; 'code field'
4441
4442   ; ---
4443
4444   L0DAB:  LD      BC,$0002                ;
4445           JR      L0DBF                   ;
4446
4447   ; ------------------
4448   ; THE 'DNEGATE' WORD
4449   ; ------------------
4450   ; (d -- -d)
4451   ; Double length integer negation.
4452
4453   L0DB0:  DEFM    "DNEGAT"                ; 'name field'
```

```
4454            DEFB    'E' +$80
4455
4456            DEFW    L0DA8                   ; 'link field'
4457
4458    L0DB9:  DEFB    $07                     ; 'name length field'
4459
4460    L0DBA:  DEFW    L0DBC                   ; 'code field'
4461
4462    ; ---
4463
4464    L0DBC:  LD      BC,$0004
4465
4466    ; NEGATE joins here with bc=2
4467
4468    L0DBF:  LD      HL,($3C3B)              ; SPARE
4469            AND     A
4470            SBC     HL,BC
4471
4472    L0DC5:  LD      A,B
4473            SBC     A,(HL)
4474            LD      (HL),A
4475            INC     HL
4476            DEC     C
4477            JR      NZ,L0DC5                ;
4478
4479            JP      (IY)                    ; to 'next'.
4480
4481    ; ------------
4482    ; THE '+' WORD
4483    ; ------------
4484    ; (n1, n2 -- n1 + n2)
4485
4486    L0DCE:  DEFB    '+' + $80               ; 'name field'
4487
4488            DEFW    L0DB9                   ; 'link field'
4489
4490    L0DD1:  DEFB    $01                     ; 'name length field'
4491
4492    L0DD2:  DEFW    L0DD4                   ; 'code field'
4493
4494    ; ---
4495
4496    L0DD4:  RST     18H                     ; pop word DE
4497            PUSH    DE                      ; save on machine stack
4498            RST     18H                     ; pop word DE
4499            POP     HL                      ; first number to HL
4500
4501            ADD     HL,DE                   ; the actual addition
4502
4503            EX      DE,HL                   ; result to DE
4504            RST     10H                     ; push word DE
4505
4506            JP      (IY)                    ; to 'next'.
4507
4508    ; ------------
4509    ; THE '-' WORD
4510    ; ------------
4511    ; (n1, n2 -- n1-n2)
4512    ; flip the sign and do a plus.
4513
4514    L0DDD:  DEFB    '-' + $80               ; 'name field'
4515
4516            DEFW    L0DD1                   ; 'link field'
4517
4518    L0DE0:  DEFB    $01                     ; 'name length field'
4519
4520    L0DE1:  DEFW    L0EC3                   ; 'code field' - docolon
4521
4522    ; ---
4523
4524    L0DE3:  DEFW    L0DA9                   ; negate
4525            DEFW    L0DD2                   ; +
4526            DEFW    L04B6                   ; exit
```

```
4527
4528    ; -------------
4529    ; THE 'D+' WORD
4530    ; -------------
4531    ; (d1, d2 -- d1 + d2)
4532    ; double length integer addition.
4533
4534    L0DE9:  DEFB    'D'                     ; 'name field'
4535            DEFB    '+' + $80
4536
4537            DEFW    L0DE0                   ; 'link field'
4538
4539    L0DED:  DEFB    $02                     ; 'name length field'
4540
4541    L0DEE:  DEFW    L0DF0                   ; 'code field'
4542
4543    ; ---
4544
4545    L0DF0:  RST     18H                     ; pop word DE
4546
4547            PUSH    DE
4548            CALL    L084E                   ; stk_to_bc
4549            RST     18H                     ; pop word DE
4550            PUSH    DE
4551            RST     18H                     ; pop word DE
4552            EX      DE,HL
4553            ADD     HL,BC
4554            EX      DE,HL
4555            RST     10H                     ; push word DE
4556            POP     BC
4557            POP     HL
4558            ADC     HL,BC
4559            EX      DE,HL
4560            RST     10H                     ; push word DE
4561
4562            JP      (IY)                    ; to 'next'.
4563
4564    ; -------------
4565    ; THE '1+' WORD
4566    ; -------------
4567    ; (n -- n+1)
4568
4569    L0E04:  DEFB    '1'                     ; 'name field'
4570            DEFB    '+' + $80
4571
4572            DEFW    L0DED                   ; 'link field'
4573
4574    L0E08:  DEFB    $02                     ; 'name length field'
4575
4576    L0E09:  DEFW    L0E0B                   ; 'code field'
4577
4578    ; ---
4579
4580    L0E0B:  RST     18H                     ; get word 'n' in DE
4581            JR      L0E17                   ; forward to increment and stack
4582
4583    ; -------------
4584    ; THE '2+' WORD
4585    ; -------------
4586    ; (n -- n+2)
4587
4588    L0E0E:  DEFB    '2'                     ; 'name field'
4589            DEFB    '+' + $80
4590
4591            DEFW    L0E08                   ; 'link field'
4592
4593    L0E12:  DEFB    $02                     ; 'name length field'
4594
4595    L0E13:  DEFW    L0E15                   ; 'code field'
4596
4597    ; ---
4598
4599    L0E15:  RST     18H                     ; get word 'n' in DE.
```

```
4600            INC     DE                      ; increment n                   (4)
4601    ; ->
4602    L0E17:  INC     DE                      ; increment n                   (4)
4603            JR      L0E2E                   ; forward to push word DE and exit
4604
4605    ; -------------
4606    ; THE '1-' WORD
4607    ; -------------
4608    ; (n -- n-1)
4609
4610
4611    L0E1A:  DEFB    '1'                     ; 'name field'
4612            DEFB    '-' + $80
4613
4614            DEFW    L0E12                   ; 'link field'
4615
4616    L0E1E:  DEFB    $02                     ; 'name length field'
4617
4618    L0E1F:  DEFW    L0E21                   ; 'code field'
4619
4620    ; ---
4621
4622    L0E21:  RST     18H                     ;
4623            JR      L0E2D                   ;
4624
4625    ; -------------
4626    ; THE '2-' WORD
4627    ; -------------
4628    ; (n -- n-2)
4629
4630
4631    L0E24:  DEFB    '2'                     ; 'name field'
4632    L0E25:  DEFB    '-' + $80
4633
4634    L0E26:  DEFW    L0E1E                   ; 'link field'
4635
4636    L0E28:  DEFB    $02                     ; 'name length field'
4637
4638    L0E29:  DEFW    L0E2B                   ; 'code field'
4639
4640    ; ---
4641
4642    ;
4643    L0E2B:  RST     18H
4644            DEC     DE
4645
4646    ; ->
4647    L0E2D:  DEC     DE
4648
4649    ; ->
4650    L0E2E:  RST     10H                     ; push word DE
4651
4652            JP      (IY)                    ; to 'next'.
4653
4654    ; -------------
4655    ; THE 'OR' WORD
4656    ; -------------
4657    ; (n1, n2 -- n1 OR n2)
4658    ; Bitwise Boolean operation.
4659
4660
4661    L0E31:  DEFB    'O'                     ; 'name field'
4662            DEFB    'R' + $80
4663
4664            DEFW    L0E28                   ; 'link field'
4665
4666    L0E35:  DEFB    $02                     ; 'name length field'
4667
4668    L0E36:  DEFW    L0E38                   ; 'code field'
4669
4670    ; ---
4671
4672    L0E38:  RST     18H                     ; pop word DE
```

```
4673            CALL    L084E                   ; stk_to_bc
4674
4675            LD      A,E                     ;
4676            OR      C                       ; OR low order bytes
4677            LD      E,A                     ;
4678
4679            LD      A,D                     ;
4680            OR      B                       ; OR high order bytes
4681            LD      D,A                     ;
4682
4683            RST     10H                     ; push word DE
4684
4685            JP      (IY)                    ; to 'next'.
4686
4687    ; --------------
4688    ; THE 'AND' WORD
4689    ; --------------
4690    ; (n1, n2 -- n1 AND n2)
4691    ; Bitwise Boolean operation.
4692
4693
4694    L0E45:  DEFM    "AN"                    ; 'name field'
4695            DEFB    'D' + $80
4696
4697            DEFW    L0E35                   ; 'link field'
4698
4699    L0E4A:  DEFB    $03                     ; 'name length field'
4700
4701            DEFW    L0E4D                   ; 'code field'
4702
4703    ; ---
4704
4705    L0E4D:  RST     18H
4706            CALL    L084E                   ; stk_to_bc
4707
4708            LD      A,E                     ;
4709            AND     C                       ;
4710            LD      E,A                     ;
4711
4712            LD      A,D                     ;
4713            AND     B                       ;
4714            LD      D,A                     ;
4715
4716            RST     10H                     ; push word DE
4717            JP      (IY)                    ; to 'next'.
4718
4719    ; --------------
4720    ; THE 'XOR' WORD
4721    ; --------------
4722    ; (n1, n2 -- n1 XOR n2)
4723    ; Bitwise Boolean XOR (exclusive or)
4724
4725    L0E5A:  DEFM    "XO"                    ; 'name field'
4726            DEFB    'R' + $80
4727
4728            DEFW    L0E4A                   ; 'link field'
4729
4730    L0E5F:  DEFB    $03                     ; 'name length field'
4731
4732    L0E60:  DEFW    L0E62                   ; 'code field'
4733
4734    ; ---
4735
4736    L0E62:  RST     18H
4737            CALL    L084E                   ; stk_to_bc
4738
4739            LD      A,E                     ;
4740            XOR     C                       ;
4741            LD      E,A                     ;
4742
4743            LD      A,D                     ;
4744            XOR     B                       ;
4745            LD      D,A                     ;
```

```
4746
4747            RST     10H                     ; push word DE
4748            JP      (IY)                    ; to 'next'.
4749
4750    ; --------------
4751    ; THE 'MAX' WORD
4752    ; --------------
4753    ; (n1, n2 -- max (n1, n2))
4754    ; Calculates the larger of two numbers.
4755
4756    L0E72:  DEFM    "MA"                    ; 'name field'
4757            DEFB    'X' + $80
4758
4759            DEFW    L0E5F                   ; 'link field'
4760
4761    L0E74:  DEFB    $03                     ; 'name length field'
4762
4763    L0E75:  DEFW    L0EC3                   ; 'code field' - docolon
4764
4765    ; ---
4766
4767    L0E77:  DEFW    L0912                   ; over
4768            DEFW    L0912                   ; over
4769            DEFW    L0C65                   ; <
4770            DEFW    L1271                   ; branch
4771    L0E7F:  DEFW    $000F                   ; forward to L0E8F
4772
4773    ; --------------
4774    ; THE 'MIN' WORD
4775    ; --------------
4776    ; (n1, n2 -- min (n1, n2))
4777    ; Calculates the smaller of two numbers.
4778
4779    L0E81:  DEFM    "MI"                    ; 'name field'
4780            DEFB    'N' + $80
4781
4782            DEFW    L0E74                   ; 'link field'
4783
4784    L0E86:  DEFB    $03                     ; 'name length field'
4785
4786            DEFW    L0EC3                   ; 'code field' - docolon
4787
4788    ; ---
4789
4790    L0E89:  DEFW    L0912                   ; over
4791            DEFW    L0912                   ; over
4792            DEFW    L0C56                   ; >
4793    ; ->
4794    L0E8F:  DEFW    L1283                   ; ?branch
4795    L0E91:  DEFW    $0003                   ; forward to L0995
4796
4797            DEFW    L0885                   ; swap
4798
4799    L0995:  DEFW    L0879                   ; drop
4800            DEFW    L04B6                   ; exit
4801
4802    ; ------------------
4803    ; THE 'DECIMAL' WORD
4804    ; ------------------
4805    ; (  --  )
4806    ; Sets the system number base to ten.
4807
4808    L0E99:  DEFM    "DECIMA"                ; 'name field'
4809            DEFB    'L' + $80
4810
4811            DEFW    L0E86                   ; 'link field'
4812
4813    L0EA2:  DEFB    $07                     ; 'name length field'
4814
4815            DEFW    L0EA5                   ; 'code field'
4816
4817    ; ---
4818
```

```
4819   L0EA5:  LD      (IX+$3F),$0A            ; update system variable BASE to 10
4820
4821           JP      (IY)                    ; to 'next'.
4822
4823   ; ------------
4824   ; THE ':' WORD
4825   ; ------------
4826   ; Introduces colon definitions.
4827
4828   L0EAB:  DEFB    ':' + $80               ; 'name field'
4829
4830           DEFW    L0EA2                   ; 'link field'
4831
4832   L0EAE:  DEFB    $01                     ; 'name length field'
4833
4834   L0EAF:  DEFW    L1085                   ; 'code field' - create and enclose
4835
4836   ; ---
4837
4838   L0EB1:  DEFW    L0EC3                   ; do_colon
4839
4840           DEFW    L104B                   ; stk_data
4841           DEFB    $0A                     ; ten                   marker byte?
4842   ; ->
4843   L0EB6:  DEFW    L1A0E                   ; end_forth
4844
4845   L0EB8:  LD      HL,$3C3E                ; FLAGS
4846
4847           LD      A,(HL)                  ; update bits 6 and 2.
4848           OR      $44                     ; signal in compile mode, definition
4849                                           ; incomplete.
4850           LD      (HL),A                  ; update FLAGS.
4851
4852           JP      (IY)                    ; to 'next'.
4853
4854   ; ---
4855
4856   x0EC1   DEFB    $E9                     ;;
4857   x0Ec2   DEFB    $FF                     ;; 0ec2 + ffe9 =  0eab = ':'
4858
4859   ; -------------------------------
4860   ; THE 'ENTER' or 'DOCOLON' action
4861   ; -------------------------------
4862   ;
4863
4864   L0EC3:  EX      DE,HL                   ;
4865           JP      L04BA                   ;
4866
4867
4868   ; -----------------
4869   ; THE 'CREATE' WORD
4870   ; -----------------
4871   ; CREATE name
4872   ; (  --  )
4873   ; Defines a new word with a header and an empty parameter field.
4874   ; When executed, the new word stacks its parameter field address.
4875
4876   L0EC7:  DEFM    "CREAT"                 ; 'name field'
4877           DEFB    'E' + $80
4878
4879           DEFW    L0EAE                   ; 'link field'
4880
4881   L0ECF:  DEFB    $06                     ; 'name length field'
4882
4883   L0ED0:  DEFW    L0EC3                   ; 'code field' - docolon
4884
4885   ; ---
4886
4887   L0ED2:  DEFW    L104B                   ; stk_data
4888           DEFB    $20                     ; a space               delimiter
4889           DEFW    L05AB                   ; word to pad
4890           DEFW    L0EFB                   ; get-name              in dict
4891           DEFW    L0688                   ; stk-zero              link
```

```
4892            DEFW    L0F4E                   ; ,
4893            DEFW    L0480                   ; current
4894            DEFW    L08B3                   ; @
4895            DEFW    L086B                   ; dup
4896            DEFW    L08B3                   ; @
4897            DEFW    L0F4E                   ; ,
4898            DEFW    L0460                   ; here
4899            DEFW    L0885                   ; swap
4900            DEFW    L08C1                   ; !
4901            DEFW    L0499                   ; pad
4902            DEFW    L0896                   ; C@            fetch 1 byte
4903            DEFW    L0F5F                   ; C,
4904            DEFW    L1011                   ; stack next word
4905            DEFW    $0FEC                   ; ???
4906            DEFW    L0F4E                   ; ,
4907    L0EF9:  DEFW    L04B6                   ; exit
4908
4909    ; ----------------------------
4910    ; The 'get_name' Internal Word
4911    ; ----------------------------
4912    ; Used only by the above CREATE thread.
4913
4914    L0EFB:  DEFW    L0EFD                   ; headerless 'code field'
4915
4916    ; ---
4917
4918    L0EFD:  CALL    L0F2E                   ; blank stack
4919
4920            RST     18H                     ; pop word DE
4921
4922            LD      A,(DE)
4923            DEC     A                       ; zero becomes $FF
4924            CP      $3F                     ; max length is 64
4925            JR      C,L0F09                 ; forward if n range 1 - 64.
4926
4927            RST     20H                     ; Error 6
4928            DEFB    $06                     ; Name of new word too short or long.
4929
4930    ; ---
4931
4932    L0F09:  ADD     A,$08                   ; allow for prev/len/addr 3 missing
4933
4934            LD      C,A                     ;
4935            LD      B,$00                   ; length to BC
4936
4937    L0F0E:  CALL    L0F8C                   ; check free memory.
4938
4939    x0f11   LD      A,(DE)                  ; true length to A
4940            LD      C,A                     ; and BC again
4941
4942            LD      HL,($3C37)              ; STKBOT
4943
4944            PUSH    DE                      ;
4945            CALL    L0F9E                   ; routine MAKE ROOM
4946            POP     DE                      ;
4947
4948            LD      A,(DE)                  ; length of word in pad
4949            LD      B,A                     ; transfer to counter.
4950
4951    L0F1D:  INC     DE                      ; increase source
4952            LD      A,(DE)                  ; fetch character
4953
4954            CALL    L0807                   ; to_upper makes uppercase.
4955
4956            LD      (HL),A                  ; store in dictionary
4957            INC     HL                      ; increase destination
4958            DJNZ    L0F1D                   ; loop back for all letters.
4959
4960            LD      ($3C39),HL              ; store this location in SPARE
4961            DEC     HL                      ; step back to last letter of word.
4962            SET     7,(HL)                  ; and 'invert' it.
4963            JP      (IY)                    ; to 'next'.
4964
```

```
4965   ; ---
4966
4967
4968   L0F2E:  BIT     2,(IX+$3E)              ; test FLAGS incomplete definition ?
4969           JR      Z,L0F36                 ; forward if not.
4970
4971           RST     20H                     ; Error 12
4972           DEFB    $0C                     ; Incomplete definition in dictionary.
4973
4974   ; ---
4975
4976   L0F36:  LD      HL,($3C37)              ; fetch STKBOT
4977           LD      DE,($3C39)              ; fetch SPARE
4978
4979           XOR     A                       ; clear accumulator and carry flag
4980
4981           SBC     HL,DE                   ; subtract
4982
4983           EX      DE,HL                   ;
4984           LD      (HL),E                  ; place low byte at next STACK slot.
4985           INC     HL                      ;
4986           LD      (HL),D                  ; place high byte
4987           LD      H,A                     ; make HL zero
4988           LD      L,A                     ;
4989           LD      ($3C39),HL              ; update system variable SPARE to zero
4990
4991           RET                             ; return
4992
4993   ; ---------------------
4994
4995   ; ------------
4996   ; THE ',' WORD
4997   ; ------------
4998   ; ( n --   )
4999   ; Encloses the single length integer in the dictionary.
5000
5001   L0F4A:  DEFB    ',' + $80               ; 'name field'
5002
5003           DEFW    L0ECF                   ; 'link field'
5004
5005   L0F4D:  DEFB    $01                     ; 'name length field'
5006
5007   L0F4E:  DEFW    L0EC3                   ; 'code field' - docolon
5008
5009   ; ---
5010
5011   L0F50:  DEFW    L0F83                   ; allot2
5012
5013           DEFW    L0460                   ; here
5014           DEFW    L0E29                   ; 2-
5015           DEFW    L08C1                   ; !
5016           DEFW    L04B6                   ; exit
5017
5018
5019   ; -------------
5020   ; THE 'C,' WORD
5021   ; -------------
5022   ; ( n --   )
5023   ; Encloses the less significant byte of n in the dictionary.
5024
5025   L0F5A:  DEFB    'C'                     ; 'name field'
5026           DEFB    ',' + $80
5027
5028           DEFW    L0F4D                   ; 'link field'
5029
5030   L0F5E:  DEFB    $02                     ; 'name length field'
5031
5032   L0F5F:  DEFW    L0EC3                   ; 'code field' - docolon
5033
5034   ; ---
5035
5036   L0F61:  DEFW    L104B                   ; stk-data
5037           DEFB    $01                     ; one
```

```
5038            DEFW    L0F76                   ; allot
5039
5040    x0f66   DEFW    L0460                   ; here
5041            DEFW    L0E1F                   ; 1-
5042            DEFW    L08A5                   ; C!
5043            DEFW    L04B6                   ; exit
5044
5045    ; ---------------
5046    ; THE 'ALLOT' WORD
5047    ; ---------------
5048    ; (n -- )
5049    ; Encloses n bytes in the dictionary, without initializing them.
5050
5051    L0F6E:  DEFM    "ALLO"                  ; 'name field'
5052            DEFB    'T' + $80
5053
5054            DEFW    L0F5E                   ; 'link field'
5055
5056    L0F75:  DEFB    $05                     ; 'name length field'
5057
5058    L0F76:  DEFW    L0F78                   ; 'code field'
5059
5060    ; ---
5061
5062    L0F78:  CALL    L084E                   ; stk_to_bc
5063            LD      HL,($3C37)              ; STKBOT
5064            CALL    L0F9E                   ; routine MAKE ROOM
5065            JP      (IY)                    ; to 'next'.
5066
5067    ; --------------------------
5068    ; The 'allot2' Internal Word
5069    ; --------------------------
5070    ; Encloses 2 bytes in the dictionary, without initializing them.
5071
5072    L0F83:  DEFW    L0EC3                   ; headerless 'code field' - docolon
5073
5074    ; ---
5075
5076    L0F85:  DEFW    L104B                   ; stk_data
5077            DEFB    $02                     ; two bytes required
5078            DEFW    L0F76                   ; allot
5079            DEFW    L04B6                   ; exit
5080
5081    ; ---------------------------------
5082    ; THE 'DEFAULT MEMORY CHECK' ROUTINE
5083    ; ---------------------------------
5084    ; called each cycle in slow mode to check free memory.
5085
5086    L0F8C:  LD      HL,$001E                ; Allow a thirty byte overhead.
5087
5088    ; ---------------------------------
5089    ; THE 'CHECK FREE MEMORY' SUBROUTINE
5090    ; ---------------------------------
5091
5092    L0F8F:  PUSH    BC                      ; save bytes to check.
5093
5094            ADD     HL,BC                   ;
5095            LD      BC,($3C3B)              ; SPARE
5096            ADD     HL,BC                   ; carry indicates error - past 65535
5097
5098            POP     BC                      ; restore number of bytes
5099            JR      C,L0F9C                 ; forward with error
5100
5101            SBC     HL,SP                   ; now check against the return stack
5102                                            ; (machine stack)
5103            RET     C                       ; return if value is less
5104
5105    L0F9C:  RST     20H                     ; Error 1
5106            DEFB    $01                     ; Not enough memory
5107
5108    ; -------------------------
5109    ; THE 'MAKE ROOM' SUBROUTINE
5110    ; -------------------------
```

```
5111
5112  L0F9E:  EX      DE,HL                   ; first new location to DE
5113          LD      HL,$0028                ; overhead 40 bytes.
5114
5115  L0FA2:  CALL    L0F8F                   ; check free memory.
5116
5117  ; now increase the two data stack pointers.
5118
5119          LD      HL,($3C37)              ; fetch value of STKBOT
5120          ADD     HL,BC                   ; add required room.
5121          LD      ($3C37),HL              ; update STKBOT.
5122
5123          LD      HL,($3C3B)              ; fetch value of SPARE
5124          PUSH    HL                      ; take a copy of 'old' value
5125          ADD     HL,BC                   ; add required room.
5126          LD      ($3C3B),HL              ; update SPARE.
5127
5128          EX      (SP),HL                 ; new SPARE value to stack,
5129                                          ; old SPARE value to HL.
5130          PUSH    HL                      ; push old SPARE value.
5131          AND     A                       ; clear carry.
5132
5133          SBC     HL,DE                   ; get length of stack and 12
5134          LD      B,H                     ;
5135          LD      C,L                     ;
5136          POP     HL                      ; old spare
5137          POP     DE                      ; new spare
5138          RET     Z                       ; return if same.
5139
5140  ; else new SPARE must be higher than old spare.
5141
5142          DEC     HL                      ; point to end of data stack
5143          DEC     DE                      ; adjust destination.
5144          LDDR                            ; copy the Data Stack + gap upwards.
5145
5146  L0FC2:  INC     HL                      ; point to first new location.
5147
5148          RET                             ; return.
5149
5150  ; -------------------
5151  ; THE 'VARIABLE' WORD
5152  ; -------------------
5153  ; VARIABLE name
5154  ; (n -- )
5155  ; Sets up a variable with the given name, and initializes its value to n.
5156
5157  L0FC4:  DEFM    "VARIABL"               ; 'name field'
5158          DEFB    'E' + $80
5159
5160          DEFW    L0F75                   ; 'link field'
5161
5162  L0FCE:  DEFB    $08                     ; 'name length field'
5163
5164          DEFW    L1085                   ; 'code field' - create and enclose
5165
5166  ; ---
5167
5168  L0FD1:  DEFW    L0FF0                   ; push word DE
5169          DEFW    L0F4E                   ; ,
5170
5171          DEFW    L04B6                   ; exit
5172
5173  ; -------------------
5174  ; THE 'CONSTANT' WORD
5175  ; -------------------
5176  ; CONSTANT name
5177  ; (n -- )
5178  ; Defines a constant with the given name and value n.
5179
5180  L0FD7:  DEFM    "CONSTAN"               ; 'name field'
5181          DEFB    'T' + $80
5182
5183          DEFW    L0FCE                   ; 'link field'
```

```
5184
5185   L0FE1:  DEFB    $08                     ; 'name length field'
5186
5187   L0FE2:  DEFW    L1085                   ; 'code field' - create and enclose
5188
5189   ; ---
5190
5191   L0FE4:  DEFW    L0FF5                   ; pad??
5192           DEFW    L0F4E                   ; ,
5193           DEFW    L04B6                   ; exit
5194
5195   ; ---
5196   ; ???
5197
5198   x0fea   DEFB    $DC                     ;;
5199   x0feb   DEFB    $FE                     ;;  0feb + fedc = 0Ec7 = CREATE
5200
5201   ; ->
5202   L0FEC:  JR      L0FF0                   ; skip forward
5203
5204   x0fee   DEFB    $D5                     ;;
5205   x0fef   DEFB    $FF                     ;;  0fef + ffd5 = 0fc4 = VARIABLE
5206
5207   ; ---
5208
5209   L0FF0:  RST     10H                     ; push word DE
5210           JP      (IY)                    ; to 'next'.
5211
5212   ; ---
5213
5214   x0FF3   DEFB    $E3                     ;;
5215   x0ff4   DEFB    $FF                     ;;  0ff4 + ffe3 = 0fd7 = CONSTANT
5216
5217   ; --> pad
5218
5219   L0FF5:  EX      DE,HL
5220           LD      E,(HL)
5221           INC     HL
5222           LD      D,(HL)
5223           RST     10H                     ; push word DE
5224           JP      (IY)                    ; to 'next'.
5225
5226   ; ------------------
5227   ; THE 'LITERAL' WORD
5228   ; ------------------
5229   ; (n -- )
5230   ; Compiles the top of the stack into a word definition as a literal.
5231   ; Compiles integers. decimal 4102 = $1006. c.f. $1055
5232
5233   L0FFC:  DEFM    "LITERA"                ; 'name field'
5234           DEFB    'L' + $80
5235
5236           DEFW    L0FE1                   ; 'link field'
5237
5238   L1005:  DEFB    $47                     ; 'name length field'
5239
5240   L1006:  DEFW    L1108                   ; 'code field' - compile
5241
5242   ; ---
5243
5244   L1008:  DEFW    L1011                   ; stack next word
5245           DEFW    L0F4E                   ; ,
5246           DEFW    L04B6                   ; exit
5247
5248   ; ---
5249
5250   x100E:  DEFB    $02                     ;;
5251   x100f   DEFB    $FF                     ;; 100f + ff02 = 0f11 nah!
5252   x1010   DEFB    $FF                     ;;
5253
5254   ; ----------------------------------
5255   ; The 'Stack Next Word' Internal Word
5256   ; ----------------------------------
```

```
5257
5258  L1011:  DEFW    L1013                   ; headerless 'code field'
5259
5260  ; ---
5261
5262  L1013:  LD      B,$01                   ; counter - one word to push
5263
5264  L1015:  POP     HL                      ; drop the 'Next Word' pointer.
5265          LD      E,(HL)                  ; low byte to E.
5266          INC     HL                      ; increment pointer.
5267          LD      D,(HL)                  ; high byte to D.
5268
5269  ; -> E B=1 (one byte op)
5270
5271  L1019:  INC     HL                      ; increment the 'Next Word' pointer
5272
5273  L101A:  PUSH    HL                      ; the 'Next Word' pointer goes to
5274                                          ; the Return Stack.
5275          RST     10H                     ; stack Data Word DE
5276          DJNZ    L1015                   ; loop back if more than one.
5277
5278  L101E:  JP      (IY)                    ; to 'next'.
5279
5280
5281  ; ---------------
5282  ; THE 'ASCII' WORD
5283  ; ---------------
5284  ; Takes the next word from the input buffer, and yields the ASCII code
5285  ; of its first character. If compiling, then compiles this as a literal.
5286  ;
5287  ; e.g.      :STARS 0 DO ASCII * EMIT LOOP ;
5288  ; (--ASCII code)         (if interpreting)
5289  ; (--)                   (if compiling)
5290
5291  L1020:  DEFM    "ASCI"                  ; 'name field'
5292          DEFB    'I' + $80
5293
5294          DEFW    L1005                   ; 'link field'
5295
5296  L1027:  DEFB    $45                     ; 'name length field' (immediate mode)
5297
5298  L1029:  DEFW    L0EC3                   ; 'code field' - docolon
5299
5300  ; ---------------
5301
5302  L102A:  DEFW    L104B                   ; stk_data
5303          DEFB    $20                     ; space delimiter
5304          DEFW    L05AB                   ; word  to pad
5305          DEFW    L0E09                   ; 1+
5306          DEFW    L0896                   ; C@
5307          DEFW    L1A0E                   ; end-forth.
5308
5309          BIT     6,(IX+$3E)              ; FLAGS
5310          JR      Z,L101E                 ; back to a jp (iy)
5311
5312          CALL    L04B9                   ; forth
5313
5314  L103E:  DEFW    L1011                   ; stack next word
5315          DEFW    L104B                   ; (stk_data)
5316          DEFW    L0F4E                   ; ,
5317          DEFW    L0F5F                   ; c,
5318          DEFW    L04B6                   ; exit
5319
5320  ; ---
5321
5322  x1048   DEFB    $01                     ;; ?
5323
5324  x1049   DEFB    $D6                     ;; ?
5325  x104a   DEFB    $FF                     ;; ?  104a + ffd6 = 1020 = ASCII
5326
5327  ; ---------------------------
5328  ; The 'stk-data' Internal Word
5329  ; ---------------------------
```

```
5330   ; used succinctly to stack the following byte as a word.
5331
5332   L104B:  DEFW    L104D                   ; headerless 'code field'
5333
5334   ; ---
5335
5336   L104D:  POP     HL                      ; retrieve the 'Next Word' pointer.
5337
5338           LD      E,(HL)                  ; fetch the single byte from there.
5339           LD      D,$00                   ; set high order byte to zero.
5340
5341           LD      B,$01                   ; set counter to 1.
5342
5343           JR      L1019                   ; back to stack one word and
5344                                           ; put the incremented pointer back on
5345                                           ; the Return Stack.
5346
5347   ; --------------------------
5348   ; The 'stk_fp' Internal Word
5349   ; --------------------------
5350   ; stack and enclose a floating point number - two words.
5351
5352   L1055:  DEFW    L1108                   ; headerless 'code field' - compile
5353
5354   ; ---
5355
5356           DEFW    L1064                   ; stack two words.
5357           DEFW    L0885                   ; swap
5358           DEFW    L0F4E                   ; ,
5359           DEFW    L0F4E                   ; ,
5360           DEFW    L04B6                   ; exit
5361   ; ---
5362
5363   x1061   DEFB    $04                     ;;
5364   x1062   DEFB    $FF                     ;; 1062 + ff04 = 0f66 XX
5365   x1063   DEFB    $FF                     ;;
5366
5367   ; ----------------------------------
5368   ; The 'STACK TWO WORDS' Internal Word
5369   ; ----------------------------------
5370
5371   L1064:  DEFW    L1066                   ; headerless 'code field'
5372
5373   ; ---
5374
5375   L1066:  LD      B,$02                   ; set counter to two
5376
5377           JR      L1015                   ; back to stack 2 words
5378
5379
5380   ; -----------------
5381   ; THE 'DEFINER' WORD
5382   ; -----------------
5383   ; Used with 'DOES>' to define new defining words. i.e. words that themselves
5384   ; define new words.
5385   ; The format is
5386   ; DEFINER name
5387   ;       defining routine
5388   ; DOES>
5389   ;       action routine
5390   ; ;
5391   ; name is the name of the new defining word; when executed it will set up
5392   ; the header of a new word and use its defining routine to set up the
5393   ; parameter field. When this new word in its turn is executed, its parameter
5394   ; field will be put on the stack and the action routine will be executed.
5395
5396   L106A:  DEFM    "DEFINE"                ; 'name field'
5397           DEFB    'R' + $80
5398
5399           DEFW    L1027                   ; 'link field'
5400
5401   L1073:  DEFB    $07                     ; 'name length field'
5402
```

```
L1074:  DEFW    L1085                   ; 'code field' - create and enclose

; ---

L1076:  DEFW    L1085                   ; create and enclose
        DEFW    L0460                   ; here

        DEFW    L104B                   ; stk-data

        DEFB    $0C                     ; 12                    marker byte

        DEFW    L0F83                   ; allot2
        DEFW    L1276                   ; branch
L1081:  DEFW    $FE34                   ; back to L0EB6

; ---

x1083   DEFB    $E6                     ;;
x1084   DEFB    $FF                     ;; 1084 + ffe6 = 106a = DEFINER

; ---
;; createe and fill
; ----
; used seven times as a code word.

L1085:  CALL    L0FF0                   ; push word DE (save addr nxt wrd on DS)

        DEFW    L0ED0                   ; create
        DEFW    L086B                   ; dup
        DEFW    L08B3                   ; @
        DEFW    L0460                   ; here
        DEFW    L0E29                   ; 2-
        DEFW    L08C1                   ; !

L1094:  DEFW    L0E13                   ; 2+
        DEFW    L109A                   ; pop DE
        DEFW    L04B6                   ; exit

; -----------
; pop word DE
; -----------
; branch to addr on stack???

L109A:  DEFW    L109C                   ; headerless 'code field'

; ---

L109C:  RST     18H                     ; unstack Data Word DE

        JP      L0EC3                   ; start new thread.

; ---------------
; THE 'CALL' WORD
; ---------------
; (address -- )
; Executes Z80 machine code at address on the stack. The code is terminated
; by a jp (iy)
; e.g. in hex
; DEFINER CODE DOES> CALL ;
; CODE EI FB C, FD C, E9 C,
; The word EI will enable interrupts.

L10A0:  DEFM    "CAL"                   ; 'name field'
        DEFB    'L' + $80

        DEFW    L1073                   ; 'link field'

L10A6:  DEFB    $04                     ; 'name length field'

L10A7:  DEFW    L10A9                   ; 'code field'

; ---
```

```
5476   L10A9:  RST     18H
5477           EX      DE,HL
5478
5479           JP      (HL)
5480
5481   ; ----------------
5482   ; THE 'DOES>' WORD
5483   ; ----------------
5484   ; See DEFINER.
5485
5486   L10AC:  DEFM    "DOES"                  ; 'name field'
5487           DEFB    '>' + $80
5488
5489           DEFW    L10F4                   ; 'link field'
5490
5491   L10B3:  DEFB    $45                     ; 'name length field' (immediate mode)
5492
5493   L10B4:  DEFW    L1108                   ; 'code field' - compile
5494
5495   L10B6:  DEFW    L10E8                   ; exit
5496
5497           DEFW    L12D8                   ; check??
5498
5499           DEFB    $0C                     ; 12
5500
5501           DEFW    L10CD                   ;
5502           DEFW    L104B                   ; stk_data
5503
5504           DEFB    $CD                     ; data                  call ?
5505
5506           DEFW    L0F5F                   ; C,
5507           DEFW    L1011                   ; stack next word
5508           DEFW    L0FF0                   ; (push word DE)
5509           DEFW    L0F4E                   ; ,
5510           DEFW    L104B                   ; stk-data
5511
5512           DEFB    $0A                     ; ten                   marker byte.
5513
5514           DEFW    L04B6                   ; exit
5515
5516   ; ----------------------
5517   ; The '???' Internal Word
5518   ; ----------------------
5519
5520   L10CD:  DEFW    L0EC3                   ; headerless 'code field' - docolon
5521
5522   ; ---
5523
5524           DEFW    L086B                   ; dup
5525           DEFW    L0E29                   ; 2-
5526           DEFW    L15B5                   ; namefield
5527           DEFW    L0460                   ; here
5528           DEFW    L0DE1                   ; -
5529           DEFW    L0E1F                   ; 1-
5530           DEFW    L0F4E                   ; ,
5531           DEFW    L0460                   ; here
5532           DEFW    L0885                   ; swap
5533           DEFW    L08C1                   ; !
5534           DEFW    L04B6                   ; exit
5535
5536   ; ---
5537
5538   x10e5   DEFB    $05                     ;;
5539
5540   x10e6   DEFB    $C5                     ;;
5541   x10e7   DEFB    $FF                     ;; 10e7 + ffc5 = 10ac = DOES>
5542
5543   ; ---
5544
5545   L10E8:  DEFW    L04B8                   ; exit?
5546
5547   ; -------------------
5548   ; THE 'COMPILER' WORD
```

```
5549   ; ------------------
5550   ; Used with 'RUNS>' for defining new compiling words, i.e. words that are
5551   ; used within word definitions to give an immediate effect of compiling
5552   ; some information into the dictionary.
5553   ; (This is traditionally done with IMMEDIATE, but COMPILER...RUNS> works
5554   ; better with EDIT etc.)
5555
5556   L10EA:  DEFM    "COMPILE"               ; 'name field'
5557           DEFB    'R' + $80
5558
5559           DEFW    L10A6                   ; 'link field'
5560
5561   L10F4:  DEFB    $08                     ; 'name length field'
5562
5563   L10F5:  DEFW    L1085                   ; 'code field' - create and enclose
5564
5565   ; ---
5566
5567           DEFW    L1108                   ; compile
5568           DEFW    L1160                   ; immediate
5569           DEFW    L0460                   ; here
5570           DEFW    L104B                   ; stk_data
5571
5572   L10FF:  DEFB    $0B                     ; 11                    marker byte
5573
5574           DEFW    L0F83                   ; allot2
5575           DEFW    L1276                   ; branch
5576   L1104:  DEFW    $FDB1                   ; back to L0EB6
5577
5578   ; ---
5579
5580   x1106   DEFB    $E3                     ;;
5581   x1107   DEFB    $FF                     ;; 1107 + ffe3 = 10ea = COMPILER
5582
5583   ; ---------------------
5584   ; THE 'COMPILE' ROUTINE
5585   ; ---------------------
5586   ; Instead of executing code words as they are encountered, lay them down in
5587   ; the dictionary along with any parameters.
5588
5589   L1108:  BIT     6,(IX+$3E)              ; test FLAGS - compiler mode ?
5590           JR      NZ,L1110                ; skip error if so.
5591
5592           RST     20H                     ; Error 4.
5593           DEFB    $04                     ; Compiling word used in interpret mode.
5594
5595   L1110:  CALL    L0FF0                   ; push word DE (then jp (iy))
5596
5597           DEFW    L086B                   ; dup
5598           DEFW    L08B3                   ; @
5599           DEFW    L0F4E                   ; ,
5600           DEFW    L1276                   ; branch
5601   L111B:  DEFW    $FF78                   ; to L1094 - definer code
5602
5603   ; ----------------
5604   ; THE 'RUNS>' WORD
5605   ; ----------------
5606   ; See COMPILER
5607
5608   L111D:  DEFM    "RUNS"                  ; 'name field'
5609           DEFB    '>' + $80
5610
5611           DEFW    L10B3                   ; 'link field'
5612
5613   L1124:  DEFB    $45                     ; 'name length field' (immediate mode)
5614
5615   L1125:  DEFW    L1108                   ; 'code field' - compile
5616
5617   ; ---
5618
5619   L1127:  DEFW    L1140                   ; vv
5620           DEFW    L12D8                   ; check-for
5621           DEFB    $0B                     ; 11                    marker byte.
```

```
5622            DEFW    L0885                   ; swap
5623            DEFW    L0F5F                   ; c,
5624            DEFW    L10CD                   ; ?
5625            DEFW    L1011                   ; stack next word
5626            DEFW    L1142
5627            DEFW    L0F4E                   ; ,
5628
5629            DEFW    L104B                   ; stk-data
5630            DEFB    $0A                     ; ten.                  marker byte.
5631            DEFW    L04B6                   ; exit
5632
5633    ; ---
5634
5635    x113d   DEFB    $05                     ;;
5636
5637    x113e   DEFB    $DE                     ;;
5638    x113f   DEFB    $FF                     ;; 113f + ffde = 111d = RUNS>
5639
5640    ; ---
5641
5642    L1140:  DEFW    L04B8
5643
5644    L1142:  POP     HL
5645            PUSH    DE
5646            EX      DE,HL
5647
5648            RST     10H                     ; push word DE
5649            LD      B,D
5650            LD      C,E
5651            POP     DE
5652            PUSH    DE
5653            DEC     DE
5654            DEC     DE
5655
5656            CALL    L159E                   ;
5657
5658            POP     DE
5659            PUSH    BC
5660            JP      L0EC3                   ;
5661
5662    ; --------------------
5663    ; THE 'IMMEDIATE' WORD
5664    ; --------------------
5665    ; (  --  )
5666    ; The most recent word in the current vocabulary is made immediate, so that
5667    ; it will execute even in compile mode.
5668
5669    L1154:  DEFM    "IMMEDIAT"              ; 'name field'
5670            DEFB    'E' + $80
5671
5672            DEFW    L1124                   ; 'link field'
5673
5674    L115F:  DEFB    $09                     ; 'name length field'
5675
5676    L1160:  DEFW    L0EC3                   ; 'code field' - docolon
5677
5678    ; ---
5679
5680    L1162:  DEFW    L0480                   ; current
5681            DEFW    L08B3                   ; @
5682            DEFW    L08B3                   ; @
5683            DEFW    L1A0E                   ; end-forth.
5684
5685    L116A:  RST     18H                     ; pop word DE
5686            EX      DE,HL
5687            SET     6,(HL)
5688            JP      (IY)                    ; to 'next'.
5689
5690    ; ---------------------
5691    ; THE 'VOCABULARY' WORD
5692    ; ---------------------
5693    ; (  --  )
5694    ; Defines a new vocabulary with the given name.
```

```
5695
5696   L1170:  DEFM    "VOCABULAR"             ; 'name field'
5697           DEFB    'Y' + $80
5698
5699           DEFW    L115F                   ; 'link field'
5700
5701   L117C:  DEFB    $0A                     ; 'name length field'
5702
5703   L117D:  DEFW    L1085                   ; 'code field' - create and enclose
5704
5705   ; ---
5706
5707   L117F:  DEFW    L11B5                   ; set context
5708           DEFW    L0480                   ; current
5709           DEFW    L08B3                   ; @
5710           DEFW    L0E13                   ; 2+
5711           DEFW    L0F4E                   ; ,
5712           DEFW    L0688                   ; stk-zero
5713           DEFW    L0F5F                   ; C,
5714           DEFW    L0460                   ; here
5715           DEFW    L1011                   ; stack next word
5716           DEFW    $3C35                   ; (VOCLNK)
5717           DEFW    L086B                   ; dup
5718           DEFW    L08B3                   ; @
5719           DEFW    L0F4E                   ; ,
5720           DEFW    L08C1                   ; !
5721           DEFW    L04B6                   ; exit
5722
5723   ; ----------------------
5724   ; THE 'DEFINITIONS' WORD
5725   ; ----------------------
5726   ; (  --  )
5727   ; The CONTEXT vocabulary is made the CURRENT vocabulary as well.
5728
5729   L119D:  DEFM    "DEFINITION"            ; 'name field'
5730           DEFB    'S' + $80
5731
5732           DEFW    L117C                   ; 'link field'
5733
5734   L11AA:  DEFB    $0B                     ; 'name length field'
5735
5736   L11AB:  DEFW    L11AD                   ; 'code field'
5737
5738   ; ---
5739
5740   L11AD:  LD      HL,(CONTEXT)            ; CONTEXT
5741           LD      ($3C31),HL              ; CURRENT
5742           JP      (IY)                    ; to 'next'.
5743
5744   ; ---
5745
5746   L11B5:  LD      (CONTEXT),DE            ; CONTEXT
5747           JP      (IY)                    ; to 'next'.
5748
5749   ; ---
5750
5751   ; -------------
5752   ; THE 'IF' WORD
5753   ; -------------
5754   ; (n -- )
5755   ; Used in the form
5756   ; IF ... THEN
5757   ; or
5758   ; IF ... ELSE ... THEN
5759   ; In the first form, if n is non-zero then the words between IF and THEN
5760   ; are executed; otherwise they are skipped over.
5761   ; In the second form, if n is non-zero then the words between IF and ELSE
5762   ; are executed and those between ELSE and THEN are skipped over, while if
5763   ; n is zero then the words between IF and ELSE are skipped over and those
5764   ; between ELSE and THEN are executed.
5765
5766   L11BB:  DEFB    'I'                     ; 'name field'
5767           DEFB    'F' + $80
```

```
5768
5769            DEFW    L13E0                   ; 'link field'
5770
5771    L11BF:  DEFB    $42                     ; 'name length field' (immediate word)
5772
5773            DEFW    L1108                   ; 'code field' - compile
5774
5775    ; ---
5776
5777            DEFW    L1283                   ; ?branch
5778            DEFW    L0460                   ; here
5779
5780            DEFW    L104B                   ; stk_data
5781            DEFB    $02                     ; 2 locations required for jump length
5782            DEFW    L0F83                   ; allot2
5783            DEFW    L04B6                   ; exit
5784
5785    ; ----------------
5786    ; THE 'WHILE' WORD
5787    ; ----------------
5788    ; (n -- )
5789    ; Used in BEGIN ... WHILE ... REPEAT. If n = 0 then skips over to just past
5790    ; REPEAT.
5791
5792    L11CD:  DEFM    "WHIL"                  ; 'name field'
5793            DEFB    'E' + $80
5794
5795            DEFW    L11BF                   ; 'link field'
5796
5797    L11D4:  DEFB    $45                     ; 'name length field' (immediate mode)
5798
5799    L11D5:  DEFW    L1108                   ; 'code field' - compile
5800
5801    ; ---
5802
5803            DEFW    L1288                   ; ?branch
5804
5805            DEFW    L12D8                   ; check-for
5806            DEFB    $01                     ;  1
5807            DEFW    L0460                   ; here
5808            DEFW    L104B                   ; stk-data
5809            DEFB    $04                     ;  four
5810            DEFW    L0F83                   ; allot
5811            DEFW    L04B6                   ; exit
5812
5813    ; ---------------
5814    ; THE 'ELSE' WORD
5815    ; ---------------
5816    ; (  --  )
5817    ; Used with IF and THEN.
5818
5819    L11E5:  DEFM    "ELS"                   ; 'name field'
5820            DEFB    'E' + $80
5821
5822            DEFW    L11D4                   ; 'link field'
5823
5824    L11EB:  DEFB    $44                     ; 'name length field' (immediate mode)
5825
5826    L11EC:  DEFW    L1108                   ; 'code field' - compile
5827
5828    ; ---
5829
5830            DEFW    L1271                   ; branch
5831
5832            DEFW    L12D8                   ; check-for
5833            DEFB    $02                     ; two
5834            DEFW    L0F83                   ; allot2
5835            DEFW    L1225                   ; ?
5836            DEFW    L0460                   ; here
5837            DEFW    L0E29                   ; 2-
5838            DEFW    L104B                   ; stk-data
5839            DEFB    $02                     ; two
5840            DEFW    L04B6                   ; exit
```

```
5841
5842    ; ---------------
5843    ; THE 'THEN' WORD
5844    ; ---------------
5845    ; Used with IF.
5846
5847    L1200:  DEFM    "THE"                   ; 'name field'
5848            DEFB    'N' + $80
5849
5850            DEFW    L11EB                   ; 'link field'
5851
5852    L1206:  DEFB    $44                     ; 'name length field' (immediate mode)
5853
5854    L1207:  DEFW    L1108                   ; 'code field' - compile
5855
5856    ; ---
5857
5858            DEFW    L12A4                   ; end?
5859
5860            DEFW    L12D8                   ; check-for
5861            DEFB    $02
5862            DEFW    L1225                   ; ?
5863            DEFW    L04B6                   ; exit
5864
5865    ; ---------------
5866    ; THE 'BEGIN' WORD
5867    ; ---------------
5868    ; (  --  )
5869    ; Used with either UNTIL or WHILE...REPEAT.
5870
5871    L1212:  DEFM    "BEGI"                  ; 'name field'
5872            DEFB    'N' + $80
5873
5874            DEFW    L1206                   ; 'link field'
5875
5876    L1219:  DEFB    $45                     ; 'name length field' (immediate mode)
5877
5878    L121A:  DEFW    L1108                   ; 'code field' - compile
5879
5880    ; ---
5881
5882            DEFW    L129F
5883            DEFW    L0460                   ; here
5884            DEFW    L104B                   ; stk_data
5885            DEFB    $01                     ; 1
5886            DEFW    L04B6                   ; exit
5887
5888    ; ----------------------
5889    ; The '???' Internal Word
5890    ; ----------------------
5891
5892    L1225:  DEFW    L0EC3                   ; headerless 'code field' - docolon
5893
5894    ; ---
5895
5896            DEFW    L086B                   ; dup
5897            DEFW    L0460                   ; here
5898            DEFW    L0885                   ; swap
5899            DEFW    L0DE1                   ; -
5900            DEFW    L0E1F                   ; 1-
5901            DEFW    L0885                   ; swap
5902            DEFW    L08C1                   ; !
5903            DEFW    L04B6                   ; exit
5904
5905    ; ----------------------
5906    ; The '???' Internal Word
5907    ; ----------------------
5908
5909    L1237:  DEFW    L0EC3                   ; headerless 'code field' - docolon
5910
5911    ; ---
5912
5913            DEFW    L0460                   ; here
```

```
5914            DEFW    L0DE1                   ; -
5915            DEFW    L0E1F                   ; 1-
5916            DEFW    L0F4E                   ; ,
5917            DEFW    L04B6                   ; exit
5918
5919
5920    ; -----------------
5921    ; THE 'REPEAT' WORD
5922    ; -----------------
5923    ; (  --  )
5924    ; Used in construction BEGIN ... WHILE .. REPEAT.
5925    ; Causes a jump back to just after BEGIN.
5926
5927
5928    L1243:  DEFM    "REPEA"                 ; 'name field'
5929            DEFB    'T' + $80
5930
5931            DEFW    L1219                   ; 'link field'
5932
5933    L124B:  DEFB    $46                     ; 'name length field' (immediate mode)
5934
5935    L124C:  DEFW    L1108                   ; 'code field' - compile
5936
5937    ; ---
5938
5939    L124E   DEFW    L1276                   ; branch
5940    L1250:  DEFW    L12D8                   ; check_for
5941            DEFB    $04                     ; four
5942            DEFW    L0885                   ; swap
5943            DEFW    L1237                   ; ?
5944            DEFW    L1225                   ; ?
5945            DEFW    L04B6                   ; exit
5946
5947    ; ----------------
5948    ; THE 'UNTIL' WORD
5949    ; ----------------
5950    ; (n -- )
5951    ; Used in BEGIN ... UNTIL.
5952    ; Loops back to BEGIN if n = 0
5953
5954    L125B:  DEFM    "UNTI"                  ; 'name field'
5955            DEFB    'L' + $80
5956
5957            DEFW    L124B                   ; 'link field'
5958
5959    L1262:  DEFB    $45                     ; 'name length field' (immediate mode)
5960
5961    L1263:  DEFW    L1108                   ; 'code field' - compile
5962
5963    ; ---
5964
5965            DEFW    L128D                   ; ?branch
5966            DEFW    L12D8                   ; check_for
5967            DEFB    $01                     ;
5968            DEFW    L1237                   ; ?
5969            DEFW    L04B6                   ; exit
5970
5971    ; ---
5972
5973    x126E   DEFB    $02                     ;;
5974
5975    x126F   DEFB    $75                     ;;
5976    x1270   DEFB    $FF                     ;; 1270 + ff75 = 11e5 = ELSE
5977
5978    ; ---
5979
5980
5981    L1271:  DEFW    L1278                   ; ?
5982
5983    ; ---
5984    x1273   DEFB    $02                     ;;
5985
5986    x1274   DEFB    $CE                     ;;
```

```
x1275   DEFB    $FF                     ;; 1275 + ffce = 1243 = REPEAT

; -------------------------
; The 'branch' Internal Word
; -------------------------

L1276:  DEFW    L1278                   ; headerless 'code field'

; ---

L1278:  POP     HL                      ; drop next word pointer
        LD      E,(HL)                  ; read the 16-bit offset
        INC     HL                      ; that is
        LD      D,(HL)                  ; stored there.

L127C:  ADD     HL,DE                   ; add to current address.

        JP      L04BA                   ; jump back into address loop so that
                                        ; a new address gets stacked as IP.

; ---

x1280   DEFB    $02                     ;;

x1281   DEFB    $39                     ;;
x1282   DEFB    $FF                     ;; 1282 + ff39 = 11bb = IF

; ---

L1283:  DEFW    L128F                   ; from IF, convert, line, min, etc.

; ---

x1285   DEFB    $02                     ;;

x1286   DEFB    $46                     ;;
x1287   DEFB    $FF                     ;; 1287 + ff46 = 11cd = WHILE

; ---

L1288:  DEFW    L128F                   ; from WHILE

; ---

x128A   DEFB    $02                     ;;

x128B   DEFB    $CF                     ;;
x128C   DEFB    $FF                     ;; 128c + ffcf = 125b = UNTIL

; ---------------------------
; The '?branch' Internal Word
; ---------------------------

L128D:  DEFW    L128F                   ; headerless 'code field'

; ---

L128F:  CALL    L084E                   ; stk_to_bc

        LD      A,B                     ; test for
        OR      C                       ; zero

; -> from +loop
L1294:  JR      Z,L1278                 ; make the jump to "branch" if zero.

        POP     HL                      ; else drop the pointer.
        INC     HL                      ; step over.
        INC     HL                      ; the jump bytes
        JP      L04BA                   ; jump back into address loop so that
                                        ; a new address gets stacked as IP.

; ---
```

```
x129C   DEFB    $00                     ;;
x129D   DEFB    $74                     ;;
x129E   DEFB    $FF                     ;; 129e + ff74 = 1212 = BEGIN

; ---

L129F:  DEFW    L04B9                   ; forth

; ---

x12A1   DEFB    $00                     ;;
x12A2   DEFB    $5D                     ;;
x12A3   DEFB    $FF                     ;; 12a3 + ff5d = 1200 = THEN

; ---

L12A4:  DEFW    L04B9

; -------------
; THE 'DO' WORD
; -------------
; (limit, initial value -- )
; Sets up a DO loop, initializing the loop counter to the initial value.
; The limit and loop counter are stored on the return stack.
; See LOOP and +LOOP.

L12A6:  DEFB    'D'                     ; 'name field'
        DEFB    'O' + $80

        DEFW    L1262                   ; 'link field'

L12AA:  DEFB    $42                     ; 'name length field' (immediate mode)

L12AB:  DEFW    L1108                   ; 'code field' - compile

; ---

        DEFW    L1323                   ; shuffle
        DEFW    L0460                   ; here
        DEFW    L104B                   ; stk_data
        DEFB    $03                     ; 3                     marker byte.
        DEFW    L04B6                   ; exit

; ---------------
; THE 'LOOP' WORD
; ---------------
; (  --  )
; Like +LOOP (below) but the number added onto the loop counter is 1.

L12B6:  DEFM    "LOO"                   ; 'name field'
        DEFB    'P' + $80

        DEFW    L12AA                   ; 'link field'

L12BC:  DEFB    $44                     ; 'name length field' (immediate mode)

L12BD:  DEFW    L1108                   ; 'code field' - compile

; ---

        DEFW    L1332                   ; shuffle more

L12C1:  DEFW    L12D8                   ; check-for
        DEFB    $03                     ; 3                     marker byte
        DEFW    L1237                   ; ?
        DEFW    L04B6                   ; exit

; ----------------
; THE '+LOOP' WORD
; ----------------
; (n -- )
; Used with DO. Adds n to the loop counter, and loops back if the loop counter
; is now less than the limit (if n >= 0) or greater than the limit (if n < 0).
```

```
6133
6134  L12C8:  DEFM    "+LOO"                  ; 'name field'
6135          DEFB    'P' + $80
6136
6137          DEFW    L12BC                   ; 'link field'
6138
6139  L12CF:  DEFB    $45                     ; 'name length field' (immediate mode)
6140
6141  L12D0:  DEFW    L1108                   ; 'code field' - compile
6142
6143  ; ---
6144
6145  L12D2:  DEFW    L133C                   ; ?
6146          DEFW    L1276                   ; branch
6147
6148  L12D6:  DEFW    $FFEA                   ; back to L12C1
6149
6150  ; ----------------------------
6151  ; The 'check-for' Internal Word
6152  ; ----------------------------
6153  ; Checks for expected marker byte which indicates stack is balanced and that
6154  ; a previous mandatory word was present.
6155
6156  L12D8:  DEFW    L12DA                   ; headerless 'code field'
6157
6158  ; ---
6159
6160  L12DA:  RST     18H                     ; pop word DE
6161          POP     HL                      ;
6162          LD      A,(HL)                  ;
6163          INC     HL                      ;
6164          PUSH    HL                      ;
6165          SUB     E                       ;
6166          OR      D                       ;
6167
6168          JR      Z,L132D                 ; to next via jp (iy).
6169
6170  ; else...
6171
6172          RST     20H                     ; Error 5
6173          DEFB    $05                     ; Word is not properly structured.
6174
6175  ; ------------
6176  ; THE 'I' WORD
6177  ; ------------
6178  ; ( -- loop counter)
6179  ; Copies the top of the return stack to the data stack. This will be either
6180  ; the loop counter for the innermost DO...LOOP, or the number most recently
6181  ; transferred by >R.
6182
6183
6184  L12E5:  DEFB    'I' + $80               ; 'name field'
6185
6186          DEFW    L11AA                   ; 'link field'
6187
6188  L12E8:  DEFB    $01                     ; 'name length field'
6189
6190  L12E9:  DEFW    L12EB                   ; 'code field'
6191
6192  ; ---
6193
6194  L12EB:  POP     BC                      ; pop return address
6195          POP     DE                      ; pop the loop counter to DE.
6196          PUSH    DE                      ; now restore the stack
6197          PUSH    BC                      ; exactly as it was.
6198
6199          RST     10H                     ; push Data Word DE - inner loop counter
6200
6201          JP      (IY)                    ; to 'next'.
6202
6203  ; -------------
6204  ; THE 'I'' WORD
6205  ; -------------
```

```
6206   ; ( -- limit)
6207   ; Copies the second number down on the return stack to the data stack
6208   ; (so in a DO loop it copies  the limit of the loop).
6209
6210   L12F2:  DEFB    'I'                     ; 'name field'
6211           DEFB    $A7                     ; "'" + $80
6212
6213           DEFW    L12E8                   ; 'link field'
6214
6215   L12F6:  DEFB    $02                     ; 'name length field'
6216
6217   L12F7:  DEFW    L12F9                   ; 'code field'
6218
6219   ; ---
6220
6221   L12F9:  LD      HL,$0004                ; two bytes per entry.
6222           JR      L1307                   ; forward to use the 'J' indexing
6223                                           ; routine
6224
6225   ; ------------
6226   ; THE 'J' WORD
6227   ; ------------
6228   ; ( -- loop counter)
6229   ; Copies the third entry on the return stack to the data stack.
6230   ; This will be either the loop counter for the second innermost DO loop
6231   ; or the number put on the return stack by the most recent >R.
6232
6233   L12FE:  DEFB    'J' + $80               ; 'name field'
6234
6235           DEFW    L12F6                   ; 'link field'
6236
6237   L1301:  DEFB    $01                     ; 'name length field'
6238
6239   L1302:  DEFW    L1304                   ; 'code field'
6240
6241   ; ---
6242
6243   L1304:  LD      HL,$0006                ; two bytes per entry
6244
6245   ; -> I' joins here with HL=4
6246
6247   L1307:  ADD     HL,SP                   ; index the stack pointer.
6248           LD      E,(HL)                  ; low order byte to E
6249           INC     HL                      ; address high byte.
6250           LD      D,(HL)                  ; DE now holds a copy of the required
6251                                           ; entry from the Return Stack
6252
6253           RST     10H                     ; stack Data Word DE
6254
6255           JP      (IY)                    ; to 'next'.
6256
6257   ; ----------------
6258   ; THE 'LEAVE' WORD
6259   ; ----------------
6260   ; (  --  )
6261   ; Forces termination of a DO loop at the next LOOP or +LOOP by setting the
6262   ; loop counter equal to the limit.
6263
6264   L130E:  DEFM    "LEAV"                  ; 'name field'
6265           DEFB    'E' + $80
6266
6267           DEFW    L1301                   ; 'link field'
6268
6269   L1315:  DEFB    $05                     ; 'name length field'
6270
6271   L1316:  DEFW    L1318                   ; 'code field'
6272
6273   ; ---
6274
6275   L1318:  POP     BC                      ; pop return address to BC.
6276           POP     HL                      ; pop the loop counter.
6277           POP     HL                      ; now the limit.
6278           PUSH    HL                      ; push unaltered limit.
```

```
6279            PUSH    HL                      ; push counter - now limit.
6280            PUSH    BC                      ; restore return address.
6281
6282            JP      (IY)                    ; to 'next'.
6283
6284    ; ---
6285
6286
6287    x1320   DEFB    $00                     ;;
6288    x1321   DEFB    $84                     ;;
6289    x1322   DEFB    $FF                     ;; 1322 + ff84 = 12a6 = DO
6290
6291    ; ----------------------
6292    ; The '???' Internal Word
6293    ; ----------------------
6294
6295    L1323:  DEFW    L1325                   ; headerless 'code field'
6296
6297    ; ---
6298
6299    L1325:  CALL    L084E                   ; stk_to_bc
6300            RST     18H                     ; pop word DE
6301            POP     HL
6302            PUSH    DE
6303
6304    L132B:  PUSH    BC
6305            PUSH    HL
6306
6307    L132D:  JP      (IY)                    ; to 'next'.
6308
6309    ; ---
6310
6311    x132F   DEFB    $02                     ;;
6312    x1330   DEFB    $85                     ;;
6313    x1331   DEFB    $FF                     ;; 1331 + ff85 = 12b6 = LOOP
6314
6315    ; ----------------------
6316    ; The '???' Internal Word
6317    ; ----------------------
6318
6319    L1332:  DEFW    L1334                   ; headerless 'code field'
6320
6321    ; ---
6322
6323    L1334:  LD      DE,$0001
6324            JR      L133F                   ; forward =>
6325
6326    ; ---
6327
6328    x1339   DEFB    $02
6329    x133A   DEFB    $8D
6330    x133B   DEFB    $FF
6331
6332    ; ----------------------
6333    ; The '???' Internal Word
6334    ; ----------------------
6335    ; loop counter + n
6336    ; Note. ADC HL,DE is used in preference to ADD HL,DE as affects P/O flag
6337
6338    L133C:  DEFW    L133E                   ; headerless 'code field'
6339
6340    ; ---
6341
6342    L133E:  RST     18H                     ; pop word DE - number to be added (n)
6343    ; =>
6344    L133F:  POP     BC                      ; pop return address to BC.
6345            POP     HL                      ; loop counter to HL.
6346            AND     A                       ; clear carry.
6347            ADC     HL,DE                   ; add the number specified.
6348            LD      A,D                     ; save MSB of (n) in A.
6349            POP     DE                      ; now pop the limit to DE.
6350            SCF                             ; set carry.
6351            JP      PE,L1358                ; jump forward with overflow.
```

```
6352
6353            PUSH    DE                      ; push limit
6354            PUSH    HL                      ; push adjusted counter.
6355            RLCA                            ; now test sign of number (n)
6356            JR      NC,L1350                ;
6357
6358            EX      DE,HL
6359
6360    L1350:  CALL    L0C99                   ;
6361
6362            CCF
6363
6364            JR      NC,L1358                ;
6365
6366            POP     HL
6367            POP     HL
6368
6369    L1358:  PUSH    BC
6370            SBC     A,A
6371            JP      L1294                   ; jump to branch on zero.
6372
6373    ; ------------
6374    ; THE '(' WORD
6375    ; ------------
6376    ; Starts a comment terminated by ')'
6377
6378    L135D:  DEFB    '(' + $80               ; 'name field'
6379
6380            DEFW    L13D4                   ; 'link field'
6381
6382    L1360:  DEFB    $41                     ; 'name length field' (immediate mode)
6383
6384    L1361:  DEFW    L1108                   ; 'code field' - compile
6385
6386    ; ---
6387
6388    L1363:  DEFW    L1379                   ;
6389            DEFW    L104B                   ; stk_data
6390
6391            DEFB    $29                     ; character ')'         - delimiter
6392
6393    L1368:  DEFW    L0460                   ; here
6394            DEFW    L0885                   ; swap
6395            DEFW    L0F83                   ; allot2
6396            DEFW    L139F                   ; find)
6397            DEFW    L0885                   ; swap
6398            DEFW    L08C1                   ; !
6399
6400            DEFW    L04B6                   ; exit
6401
6402    ; ---
6403
6404    x1376   DEFB    $FF                     ;;
6405    x1377   DEFB    $E5                     ;;
6406    x1378   DEFB    $FF                     ;; 1378 + ffe5 = 135d = '('
6407
6408    ; ----------------------
6409    ; The '???' Internal Word
6410    ; ----------------------
6411
6412    L1379:  DEFW    L137B                   ; headerless 'code field'
6413
6414    ; ---
6415
6416    L137B:  POP     HL
6417            LD      E,(HL)
6418            INC     HL
6419            LD      D,(HL)
6420
6421            INC     DE
6422
6423            JP      L127C                   ;
6424
```

```
6425   ; -------------
6426   ; THE '."' WORD
6427   ; -------------
6428   ; (  --  )
6429   ; Prints the following string terminated by ".
6430
6431   L1383:  DEFB    '.'                     ; 'name field'
6432           DEFB    '"' + $80
6433
6434           DEFW    L1360                   ; 'link field'
6435
6436   L1387:  DEFB    $42                     ; 'name length field' (immediate mode)
6437
6438   L1388:  DEFW    L1108                   ; 'code field' - compile
6439
6440   ; ---
6441
6442   L138A:  DEFW    L1396                   ; pr_embedded string.
6443           DEFW    L104B                   ; stk_data
6444           DEFB    $22                     ; '"'                   - delimiter
6445
6446           DEFW    L1276                   ; branch
6447   L1391:  DEFW    $FFD6                   ; back to 1368 (1392+$FFD6)
6448                                           ; same routine as for matching comments
6449
6450   ; ---
6451
6452   x1393   DEFB    $FF                     ;;
6453   x1394   DEFB    $EE                     ;;
6454   x1395   DEFB    $FF                     ;; 1395 + ffee = 1383 = ."
6455
6456   ; ----------------------
6457   ; The '???' Internal Word
6458   ; ----------------------
6459   ; print string embedded in Dictionary
6460
6461   L1396:  DEFW    L1398                   ; headerless 'code field'
6462
6463   ; ---
6464
6465   L1398:  POP     DE
6466           CALL    L0979                   ; pr_string1
6467           PUSH    DE
6468           JP      (IY)                    ; to 'next'.
6469
6470   ; ----------------------
6471   ; The '???' Internal Word
6472   ; ----------------------
6473   ; enclose comment
6474   ; comments may be multiple
6475   ; e.g. : SV ( system) ( variables) CLS BEGIN 0 0 AT 15360 80 TYPE 0 UNTIL ;
6476
6477
6478   L139F:  DEFW    L13A1                   ; headerless 'code field'
6479
6480   ; ---
6481
6482   L13A1:  RST     18H                     ; pop word DE
6483           PUSH    DE                      ; save delimiter.
6484
6485           CALL    L05E1                   ; find the ')' delimiter
6486
6487           LD      H,D
6488           LD      L,E
6489           ADD     HL,BC
6490           LD      A,(HL)
6491           POP     HL                      ; pop the delimiter.
6492           CP      L
6493           JR      Z,L13B8                 ; forward with a match.         =->
6494
6495           EX      DE,HL                   ;
6496           RST     10H                     ; push word DE
6497           LD      DE,$0578                ; addr retype?
```

```
6498
6499            CALL    L1815                   ; pr2
6500
6501            JR      L13A1                   ; loop back
6502
6503    ; ---
6504    ; =->
6505
6506    L13B8:  PUSH    DE
6507            PUSH    BC
6508            LD      HL,($3C37)              ; STKBOT
6509
6510            CALL    L0F9E                   ; routine MAKE ROOM
6511
6512            POP     BC
6513            POP     DE
6514            PUSH    DE
6515            PUSH    BC
6516            EX      DE,HL
6517            LDIR                            ; copy comment to dictionary.
6518            POP     BC
6519            LD      D,B
6520            LD      E,C
6521            RST     10H                     ; push word DE
6522            POP     DE
6523
6524            CALL    L07DA                   ;
6525
6526            JP      (IY)                    ; to 'next'.
6527
6528    ; ------------
6529    ; THE '[' WORD
6530    ; ------------
6531    ; (  --  )
6532    ; Enters interpret mode.
6533
6534    L13D1:  DEFB    '[' + $80               ; 'name field'
6535
6536            DEFW    L12CF                   ; 'link field'
6537
6538    L13D4:  DEFB    $41                     ; 'name length field' (immediate mode)
6539
6540    L13D5:  DEFW    L13D7                   ; 'code field'
6541
6542    ; ---
6543
6544    L13D7:  RES     6,(IX+$3E)              ; FLAGS
6545            JP      (IY)                    ; to 'next'.
6546
6547    ; ------------
6548    ; THE ']' WORD
6549    ; ------------
6550    ; (  --  )
6551    ; Enters compile mode.
6552
6553    L13DD:  DEFB    ']' + $80               ; 'name field'
6554
6555            DEFW    L1315                   ; 'link field'
6556
6557    L13E0:  DEFB    $01                     ; 'name length field'
6558
6559    L13E1:  DEFW    L13E3                   ; 'code field'
6560
6561    ; ---
6562
6563    L13E3:  SET     6,(IX+$3E)              ; FLAGS
6564            JP      (IY)                    ; to 'next'.
6565
6566
6567    ; ---------------
6568    ; THE 'EXIT' WORD
6569    ; ---------------
6570    ; (  --  )
```

```
; Exits immediately from the word in whose definition it is contained.
; Cannot be used between DO and LOOP or +LOOP, nor between >R and R>.

L13E9:  DEFM    "EXI"                   ; 'name field'
        DEFB    'T' + $80

        DEFW    L1387                   ; 'link field'

L13EF:  DEFB    $04                     ; 'name length field'

L13F0:  DEFW    L04B8                   ; 'code field'

; -------------------
; THE 'REDEFINE' WORD
; -------------------
; REDEFINE name
; (  --  )
; Takes word 'name' and replaces it with the most recent word in the
; dictionary. Updates entire dictionary to take changes into account.
; Most commonly used as
;  EDIT name
;  REDEFINE name

L13F2:  DEFM    "REDEFIN"               ; 'name field'
        DEFB    'E' + $80

        DEFW    L13EF                   ; 'link field'

L13FC:  DEFB    $08                     ; 'name length field'

L13FD:  DEFW    L13FF                   ; 'code field'

; ---

L13FF:  CALL    L0F2E                   ; blank stack

        LD      HL,($3C31)              ; CURRENT

        LD      E,(HL)
        INC     HL
        LD      D,(HL)

        EX      DE,HL                   ; transfer value to HL
        INC     HL
        LD      ($2705),HL              ; store in pad

        PUSH    HL                      ; (*)

        CALL    L15C0                   ; get 'name field' address

        LD      ($270D),HL              ; name field addr
        LD      ($2707),BC              ; parameter field addr
        LD      ($270B),DE              ; length field value

        LD      HL,($3C37)              ; STKBOT
        SBC     HL,DE
        JP      NZ,L14DA                ; forward if not matched to Error 11.

        POP     DE                      ; (*)

        RST     10H                     ; push word DE

        CALL    L04B9                   ; forth

L1429:  DEFW    L1610                   ; prvcur
        DEFW    L063D                   ; find
        DEFW    L1A0E                   ; end-forth.

; ---

L1425:  RST     18H                     ; pop word DE
        LD      HL,$C3AF
        ADD     HL,DE
```

```
6644            JP      NC,L14CF                ;
6645
6646            EX      DE,HL
6647            LD      ($2703),HL
6648
6649            CALL    L15C0                   ; get 'name field' address
6650
6651            LD      ($2701),HL
6652
6653    L1441:  PUSH    HL
6654            LD      ($2709),DE
6655            LD      A,B
6656            OR      C
6657            LD      DE,($2707)
6658            JR      Z,L1452                 ;
6659
6660            LD      A,D
6661            OR      E
6662            JR      Z,L14CF                 ;
6663
6664    L1452:  POP     HL
6665            LD      BC,($270D)
6666            SBC     HL,BC
6667            EX      DE,HL
6668            ADD     HL,DE
6669            LD      ($2707),HL
6670            LD      HL,($270B)
6671            ADD     HL,DE
6672            LD      BC,($2709)
6673            AND     A
6674            SBC     HL,BC
6675            LD      ($270B),HL
6676            LD      BC,$002E                ; 46d
6677            ADD     HL,BC
6678            BIT     7,H
6679            JR      NZ,L147F                ;
6680
6681            LD      BC,($3C3B)              ; SPARE
6682            ADD     HL,BC
6683            JR      C,L14CF                 ;
6684
6685            SBC     HL,SP
6686            JR      NC,L14CF                ;
6687
6688    L147F:  LD      HL,($2703)
6689            PUSH    HL
6690            DEC     HL
6691            DEC     HL
6692            LD      B,(HL)
6693            DEC     HL
6694            LD      C,(HL)
6695            LD      HL,($2705)
6696            PUSH    HL
6697            DEC     HL
6698            DEC     HL
6699            LD      (HL),B
6700            DEC     HL
6701            LD      (HL),C
6702            POP     HL
6703            ADD     HL,DE
6704            POP     BC
6705            AND     A
6706            SBC     HL,BC
6707            LD      ($2705),HL
6708            LD      DE,($2701)
6709            LD      HL,($2709)
6710            AND     A
6711            SBC     HL,DE
6712            LD      B,H
6713            LD      C,L
6714            PUSH    DE
6715            PUSH    BC
6716
```

```
6717            CALL    L14DC                   ; RECLAIM
6718
6719            LD      HL,($270B)
6720            POP     BC
6721            ADD     HL,BC
6722            LD      B,H
6723            LD      C,L
6724            POP     HL
6725            PUSH    BC
6726
6727            CALL    L0F9E                   ; routine MAKE ROOM
6728
6729            EX      DE,HL                   ;
6730            LD      HL,($270D)              ;
6731            LD      BC,($270B)              ;
6732            ADD     HL,BC                   ;
6733            POP     BC                      ;
6734            PUSH    BC                      ;
6735            PUSH    HL                      ;
6736
6737            LDIR                            ;
6738
6739            POP     DE
6740            POP     BC
6741
6742            CALL    L14DC                   ; RECLAIM
6743            CALL    L14F8                   ;
6744
6745            JP      (IY)                    ; to 'next'.
6746
6747    ; ---
6748
6749    L14CF:  LD      HL,($3C31)              ; CURRENT
6750            LD      DE,($2705)
6751            DEC     DE
6752            LD      (HL),E
6753            INC     HL
6754            LD      (HL),D
6755
6756    L14DA:  RST     20H                     ; Error 11
6757            DEFB    $0B                     ; Error in REDEFINE or FORGET
6758
6759    ; ----------------------------
6760    ; THE 'RECLAIMING' SUBROUTINE
6761    ; ----------------------------
6762
6763    L14DC:  LD      HL,($3C37)              ; fetch STKBOT
6764            AND     A                       ; clear carry flag
6765            SBC     HL,BC                   ; subtract number of bytes to reclaim.
6766            LD      ($3C37),HL              ; update STKBOT
6767
6768            LD      HL,($3C3B)              ; fetch SPARE
6769            SBC     HL,BC                   ; subtract number of bytes to reclaim.
6770            LD      ($3C3B),HL              ; update SPARE
6771
6772            SBC     HL,DE                   ; subtract
6773            RET     Z                       ; return if same address.
6774
6775            PUSH    BC                      ;
6776            LD      B,H                     ;
6777            LD      C,L                     ;
6778            POP     HL                      ;
6779            ADD     HL,DE                   ;
6780
6781            LDIR                            ;
6782
6783            RET                             ;
6784
6785    ; ---
6786    ;
6787    ; ---
6788
6789    L14F8:  LD      BC,$3C31                ; CURRENT
```

```
6790
6791            CALL    L1557                   ;
6792            CALL    L1557                   ;
6793
6794            LD      BC,$3C40                ; addr. of "FORTH" in RAM.
6795
6796    L1504:  LD      HL,($3C37)              ; STKBOT
6797            SCF                             ;
6798            SBC     HL,BC                   ;
6799            RET     C                       ;
6800
6801    L150B:  LD      A,(BC)                  ;
6802            RLA                             ;
6803            INC     BC                      ;
6804            JR      NC,L150B                ;
6805
6806            INC     BC                      ;
6807            INC     BC                      ;
6808            CALL    L1557                   ;
6809            INC     BC                      ;
6810            CALL    L1557                   ;
6811
6812    L1519:  CALL    L15FB                   ; routine INDEXER
6813
6814    ; ------------------------------------------------------
6815
6816            DEFW    L0EC3                   ; DE value
6817    L151E:  DEFB    $1C                     ; to L153A
6818
6819            DEFW    L1085                   ; DE value
6820    L1521:  DEFB    $16                     ; to L1537
6821
6822            DEFW    L1108                   ; DE value
6823    L1524:  DEFB    $13                     ; to L1537
6824
6825            DEFW    L11B5                   ; DE value
6826    L1527:  DEFB    $18                     ; to L153F
6827
6828            DEFW    $0000                   ; zero end marker
6829
6830    ; ------------------------------------------------------
6831
6832    L152A:  LD      HL,$FFF9
6833            ADD     HL,BC
6834
6835            LD      C,(HL)
6836            INC     HL
6837            LD      B,(HL)
6838            DEC     HL
6839
6840            ADD     HL,BC
6841
6842            LD      B,H
6843            LD      C,L
6844            JR      L1504                   ;
6845
6846    ; ------------------------------------------------------
6847
6848    L1537:  CALL    L1557                   ;
6849
6850    ; ->
6851
6852    L153A:  CALL    L1548                   ;
6853            JR      L1504                   ;
6854
6855    ; ---
6856
6857    L153F:  CALL    L1557                   ;
6858            INC     BC                      ;
6859            CALL    L1557                   ;
6860            JR      L1504                   ;
6861
6862    ; ------------------------------------------------------
```

```
6863
6864   ; XXX?
6865
6866   L1548:  CALL    L1557                   ;
6867           LD      HL,L04B6                ;
6868           AND     A                       ;
6869           SBC     HL,DE                   ;
6870           RET     Z                       ;
6871
6872           CALL    L159E                   ;
6873
6874           JR      L1548                   ;
6875
6876   ; ---
6877   ; often called twice
6878   ; ---
6879
6880
6881   L1557:  LD      A,(BC)                  ; lo byte
6882           LD      E,A                     ;
6883           INC     BC                      ;
6884           LD      A,(BC)                  ; hi byte
6885           LD      D,A                     ;
6886           DEC     BC                      ; BC now unchanged, DE contents
6887
6888           CALL    L1568                   ; routine below. header?
6889
6890           EX      DE,HL                   ; value to DE
6891           LD      A,E                     ;
6892           LD      (BC),A                  ; lo byte
6893           INC     BC                      ;
6894           LD      A,D                     ;
6895           LD      (BC),A                  ; hi byte
6896           INC     BC                      ;
6897           RET                             ; to next - BC+=2
6898
6899   ; ---
6900
6901   L1568:  LD      HL,($2701)              ; first bytes of pad.
6902           AND     A                       ;
6903           SBC     HL,DE                   ; subtract the DE value read from
6904                                           ; memory
6905           LD      H,D                     ;
6906           LD      L,E                     ; transfer that DE to HL as well
6907
6908           RET     NC                      ; return if HL was higher than DE
6909
6910           LD      HL,($2709)              ; tape header
6911           SBC     HL,DE
6912           JR      NC,L1584                ; forward if higher to
6913
6914           LD      HL,($270D)
6915           SBC     HL,DE
6916           JR      C,L1592                 ; forward if lower to
6917
6918           LD      HL,($270B)              ;
6919           ADD     HL,DE
6920           RET                             ; return
6921
6922   ; ---
6923
6924   L1584:  LD      HL,($2703)
6925           SBC     HL,DE
6926           LD      HL,($2707)
6927           RET     C
6928
6929           LD      HL,($2705)
6930           ADD     HL,DE
6931           RET
6932
6933   ; ---
6934
6935   L1592:  LD      HL,($2701)
```

```
6936            ADD     HL,DE
6937            LD      DE,($270D)
6938            AND     A
6939            SBC     HL,DE
6940            RET
6941
6942    ; ---
6943
6944    L159E:  DEC     DE
6945            LD      A,(DE)
6946            RLA
6947            RET     NC
6948
6949    L15A2:  DEC     DE
6950            DEC     DE
6951            LD      A,(DE)
6952            LD      L,A                     ; low byte
6953            LD      H,$00                   ; make high byte zero
6954            INC     A                       ; test offset for $FF.
6955            JR      NZ,L15B1                ; forward if not.
6956
6957            LD      A,(BC)
6958            LD      L,A
6959            INC     BC
6960            LD      A,(BC)
6961            LD      H,A
6962            INC     BC
6963
6964    L15B1:  ADD     HL,BC
6965
6966            LD      B,H
6967            LD      C,L
6968            RET
6969
6970    ; ---
6971    ;
6972    ; ---
6973
6974
6975    L15B5:  DEFW    L15B7
6976
6977    ; ---
6978
6979    L15B7:  RST     18H                     ; pop word DE
6980
6981            EX      DE,HL
6982
6983            CALL    L15E7                   ; WORDSTART1
6984
6985            EX      DE,HL
6986
6987            RST     10H                     ; push word DE
6988            JP      (IY)                    ; to 'next'.
6989
6990    ; ---
6991    ;
6992    ; ---
6993
6994    L15C0:  PUSH    HL
6995            LD      E,(HL)
6996            INC     HL
6997            LD      D,(HL)
6998
6999    L15C4:  CALL    L15FB                   ; routine INDEXER
7000
7001    ; ------------------------------------------------------
7002
7003            DEFW    L1108
7004    L15C9:  DEFB    $0B                     ; to L15D4 - find parameter field
7005
7006            DEFW    L1085
7007    L15CC:  DEFB    $08                     ; to L15D4 - find parameter field
7008
```

```
7009            DEFW    $0000                   ; zero end_marker.
7010
7011    ; ------------------------------------------------------
7012
7013    L15CF:  LD      BC,$0000                ; zero indicates no parameter field.
7014            JR      L15DB                   ; forward to consider total length.
7015
7016    ; ------------------------------------------------------
7017
7018    L15D4:  POP     HL                      ; retrieve the code field address
7019            PUSH    HL                      ; save it again
7020
7021            INC     HL                      ; step past the
7022            INC     HL                      ; address word
7023            LD      C,(HL)                  ; and get following address
7024            INC     HL                      ; which if in RAM could be the
7025            LD      B,(HL)                  ; parameter field to              BC.
7026
7027    ; ->
7028
7029    L15DB:  POP     HL                      ; retrieve the code field address
7030            PUSH    HL                      ; and save it again
7031
7032            DEC     HL                      ; the name length field
7033            DEC     HL                      ; link field high order byte
7034            DEC     HL                      ; link field low order byte
7035            DEC     HL                      ; possible length field high
7036            LD      D,(HL)                  ; save in D
7037            DEC     HL                      ; possible length field low
7038            LD      E,(HL)                  ; save in E
7039            ADD     HL,DE                   ; add this length
7040            EX      DE,HL                   ; and save result in             DE.
7041
7042            POP     HL                      ; retrieve code field address
7043
7044    ; ->
7045    ; indexes the header information of a FORTH word
7046
7047    L15E7:  DEC     HL                      ; point to name length field
7048
7049    ; =>
7050    L15E8:  LD      A,H                     ; fetch high order byte of the
7051                                            ; header address.
7052            CP      $3C                     ; compare to RAM location
7053            LD      A,(HL)                  ; fetch length byte.
7054            RES     6,A                     ; reset the immediate mode bit
7055            JR      C,L15F2                 ; forward if definition is in ROM.
7056
7057            ADD     A,$02                   ; else add extra for 'length field'
7058
7059    L15F2:  DEC     HL                      ; step past the
7060            DEC     HL                      ; link to previous word.
7061
7062    L15F4:  DEC     HL                      ; now address last letter on name.
7063            DEC     A                       ; decrement the length
7064            JR      NZ,L15F4                ; loop back until at first letter  HL.
7065
7066            RET                             ; return.
7067
7068    ; -------
7069    ; INDEXER
7070    ; -------
7071
7072    ; indexerloop
7073
7074    L15F9:  INC     HL                      ; step past the
7075            PUSH    HL                      ; offset byte.
7076
7077    ; -> Call Entry point
7078
7079    L15FB:  POP     HL                      ; drop return address - points to byte
7080                                            ; after the call.
7081            LD      A,(HL)                  ; read low-order byte
```

```
7082            INC     HL                      ; increment address once
7083            PUSH    HL                      ; push return address
7084
7085            LD      H,(HL)                  ; read high-order byte.
7086            LD      L,A                     ; now HL holds the read word
7087            OR      H                       ; test for two zeros.
7088            RET     Z                       ; two zeros - return
7089                                            ; (ret addr is second NOP)
7090
7091            SBC     HL,DE                   ; compare to value passed in DE
7092
7093            POP     HL                      ; now increment the
7094            INC     HL                      ; return address on machine stack.
7095
7096            JR      NZ,L15F9                ; loop back if read word is not
7097                                            ; equal to DE
7098
7099            PUSH    DE                      ; else preserve DE
7100
7101            LD      D,$00                   ; a 1 byte relative jump.
7102            LD      E,(HL)                  ; read one-byte offset.
7103            ADD     HL,DE                   ; add to read address.
7104
7105            POP     DE                      ; restore DE
7106
7107            JP      (HL)                    ; >>>
7108
7109    ; ---
7110
7111    L1610:  DEFW    L0EC3                   ; 'code field' - docolon
7112
7113            DEFW    L0E1F                   ; 1-
7114            DEFW    L0E29                   ; 2-
7115            DEFW    L08B3                   ; @
7116            DEFW    L0480                   ; current
7117            DEFW    L08B3                   ; @
7118            DEFW    L08C1                   ; !
7119            DEFW    L04B6                   ; exit
7120
7121    ; ---------------------------------
7122    ; THE 'FIND WORD IN RAM' SUBROUTINE
7123    ; ---------------------------------
7124    ; This subroutine is used by FORGET, EDIT and LIST.
7125    ; First use the standard FORTH word find to get address of word (in pad).
7126    ; If word does not exist then returned value will be zero.
7127    ; The lowest word in RAM is the FORTH word at L3C51 so a check is made
7128    ; against this address.
7129
7130    L1620:  CALL    L04B9                   ; forth
7131            DEFW    L063D                   ; find
7132
7133    L1625:  DEFW    L1A0E                   ; end-forth.
7134
7135            RST     18H                     ; pop word DE
7136
7137            LD      HL,$C3AF                ; i.e $0000 - $3C51
7138
7139            ADD     HL,DE                   ; add to test value.
7140            RET     C                       ; carry signals that word exists in RAM.
7141                                            ; return the address in DE.
7142
7143    ; else generate an error code.
7144
7145            RST     20H                     ; Error 13
7146            DEFB    $0D                     ; Error word not found or is in ROM.
7147
7148    ; -----------------
7149    ; THE 'FORGET' WORD
7150    ; -----------------
7151    ; FORGET name.
7152    ; Erases the word 'name' and all subsequently defined names from the dictionary.
7153
7154    L162F:  DEFM    "FORGE"                 ; 'name field'
```

```
7155            DEFB    'T' + $80
7156
7157            DEFW    L13FC                   ; 'link field'
7158
7159    L1637:  DEFB    $06                     ; 'name length field'
7160
7161    L1638:  DEFW    L163A                   ; 'code field'
7162
7163    ; ---
7164
7165    L163A:  LD      HL,($3C31)              ; CURRENT
7166            LD      DE,(CONTEXT)            ; CONTEXT
7167            AND     A
7168            SBC     HL,DE
7169
7170            JP      NZ,L14DA                ;
7171
7172            CALL    L1620                   ; findramword
7173
7174            LD      HL,$FFFB
7175            ADD     HL,DE
7176            LD      ($3C39),HL              ; SPARE
7177            SET     2,(IX+$3E)              ; FLAGS
7178
7179            RST     20H                     ; Invoke error routine.
7180            DEFB    $FF                     ; No error
7181
7182    ; ---------------
7183    ; THE 'EDIT' WORD
7184    ; ---------------
7185    ; EDIT name
7186    ; Lists word 'name' at bottom of the screen to be edited. Lists 18 lines at
7187    ; a time, then waits for editing until ENTER is pressed.
7188    ; A new version of the word is entered at the end of the dictionary.
7189    ; While editing, cursor up and cursor down are needed to move the cursor
7190    ; from one line to another. DELETE LINE deletes one line.
7191
7192    L1657:  DEFM    "EDI"                   ; 'name field'
7193            DEFB    'T' + $80
7194
7195            DEFW    L1637                   ; 'link field'
7196
7197    L165D:  DEFB    $04                     ; 'name length field'
7198
7199    L165E:  DEFW    L1660                   ; 'code field'
7200
7201    ; ---
7202
7203    L1660:  CALL    L1620                   ; findramword
7204
7205            SET     3,(IX+$3E)              ; update FLAGS output -> input buffer
7206            JR      L1675                   ; forward to list routine the difference
7207                                            ; being that the listing will go to the
7208                                            ; lower screen.
7209
7210    ; ---------------
7211    ; THE 'LIST' WORD
7212    ; ---------------
7213    ; LIST name
7214    ; (  --  )
7215    ; Lists word 'name' on the screen. It must have been defined by :, DEFINER,
7216    ; or COMPILER. Lists about 18 lines at a time and waits for key depression
7217    ; (shifted space breaks).
7218
7219    L1669:  DEFM    "LIS"                   ; 'name field'
7220            DEFB    'T' + $80
7221
7222            DEFW    L165D                   ; 'link field'
7223
7224    L166F:  DEFB    $04                     ; 'name length field'
7225
7226    L1670:  DEFW    L1672                   ; 'code field'
7227
```

```
7228   ; ---
7229
7230   L1672:  CALL    L1620                   ; findramword
7231
7232   ; edit path joins here but carriage returns are printed as zeros.
7233
7234   L1675:  LD      A,$0D                   ; prepare a carriage return.
7235           RST     08H                     ; print_ch
7236
7237           BIT     3,(IX+$3E)              ; test FLAGS output->input buffer?
7238
7239           PUSH    DE
7240
7241           CALL    NZ,L02D8                ; call if so to initialize buffer
7242
7243           POP     BC                      ; LD DE,(BC)
7244
7245           LD      A,(BC)
7246           LD      E,A
7247           INC     BC
7248           LD      A,(BC)
7249           LD      D,A
7250           DEC     BC
7251
7252           CALL    L15FB                   ; routine INDEXER
7253
7254   ; -----------------------------------------------------
7255
7256   L168A:  DEFW    L0EC3                   ; DE value
7257   L168C:  DEFB    $0B                     ; offset to L1697
7258
7259   L168D:  DEFW    L1108                   ; DE value
7260   L168F:  DEFB    $0D                     ; offset to L169C
7261
7262   L1690:  DEFW    L1085                   ; DE value
7263   L1692:  DEFB    $1F                     ; offset to L16B1
7264
7265           DEFW    $0000                   ; zero end-marker
7266
7267   ; -----------------------------------------------------
7268
7269   L1695:  RST     20H                     ; Error 14
7270           DEFB    $0E                     ; Word unlistable.
7271
7272   ; Only words defined by ':', 'DEFINER' or 'COMPILER' are listable.
7273
7274   ; -----------------------------------------------------
7275
7276   ; ':'
7277   L1697:  LD      HL,$0002
7278           JR      L16B4                   ;
7279   ; ---
7280
7281   L169C:  PUSH    DE
7282           LD      HL,$0002
7283           ADD     HL,BC
7284           LD      A,(HL)
7285           INC     HL
7286           LD      H,(HL)
7287           LD      L,A
7288           DEC     HL
7289           DEC     HL
7290           DEC     HL
7291
7292           LD      L,(HL)
7293           LD      A,L
7294           RLCA
7295           SBC     A,A
7296           LD      H,A
7297
7298           CALL    L180E                   ; pr_int_hl?
7299
7300           POP     DE
```

```
7301
7302   L16B1:  LD      HL,$0004
7303
7304
7305   L16B4:  ADD     HL,BC
7306           PUSH    HL
7307           PUSH    BC
7308
7309           CALL    L17E4                   ;
7310
7311           POP     DE
7312           POP     BC
7313
7314           CALL    L17E4                   ;
7315
7316           LD      (IX+$14),$01            ; LISTWSx
7317
7318   L16C3:  LD      (IX+$16),$10            ; LISTWSx
7319
7320   L16C7:  CALL    L1708                   ; index_table
7321
7322           JR      C,L16D2                 ;
7323
7324           DEC     (IX+$16)                ; LISTWSx
7325           JP      P,L16C7                 ;
7326
7327   L16D2:  BIT     3,(IX+$3E)              ; FLAGS
7328           JR      NZ,L16E8                ; branch forward  =->
7329
7330           JR      C,L1702                 ;
7331
7332           LD      HL,$3C26                ; KEYCOD
7333           LD      (HL),$00                ;
7334
7335   L16DF:  LD      A,(HL)                  ;
7336           AND     A                       ;
7337           JR      Z,L16DF                 ; loop back while zero
7338
7339           CALL    L04E4                   ; check break
7340
7341           JR      L16C3                   ; loop back
7342
7343   ; =->
7344
7345   L16E8:  PUSH    AF
7346           RES     3,(IX+$3E)              ; FLAGS
7347           PUSH    BC
7348
7349           CALL    L04B9                   ; forth
7350
7351           DEFW    L0578                   ; retype        - allow user to retype
7352           DEFW    L0506                   ; line          - interpret buffer
7353           DEFW    L1A0E                   ; end-forth.
7354
7355
7356           SET     3,(IX+$3E)              ; FLAGS
7357
7358           CALL    L02D8                   ;
7359
7360           POP     BC
7361           POP     AF
7362           JR      NC,L16C3                ;
7363
7364   L1702:  RES     3,(IX+$3E)              ; FLAGS
7365           JP      (IY)                    ; to 'next'.
7366
7367   ; ------------------------------------------------------
7368
7369   ; called once
7370
7371   L1708:  LD      A,($3C14)               ; LISTWS2
7372           LD      ($3C15),A               ; LISTWS3
7373
```

```
7374            LD      (IX+$13),$05            ; LISTWS
7375
7376    L1712:  LD      A,(BC)
7377            LD      E,A
7378            INC     BC
7379            LD      A,(BC)
7380            LD      D,A
7381            INC     BC
7382
7383    L1718:  CALL    L15FB                   ; routine INDEXER
7384
7385    ; -----------------------------------------------------
7386
7387    L171B:  DEFW    L1283                   ;
7388    L171D:  DEFB    $40                     ; offset to L175D
7389
7390    L171E:  DEFW    L1271                   ;
7391    L1720:  DEFB    $44                     ; offset to L1764
7392
7393    L1721:  DEFW    L12A4                   ;
7394    L1723:  DEFB    $48                     ; offset to L176B
7395
7396    L1724:  DEFW    L129F                   ;
7397    L1726:  DEFB    $37                     ; offset to L175D
7398
7399    L1727:  DEFW    L128D                   ;
7400    L1729:  DEFB    $42                     ; offset to L176B
7401
7402    L172A:  DEFW    L1288                   ;
7403    L172C:  DEFB    $38                     ; offset to L1764
7404
7405    L172D:  DEFW    L1276                   ;
7406    L172F:  DEFB    $3C                     ; offset to L176B
7407
7408    L1730:  DEFW    L1323                   ;
7409    L1732:  DEFB    $2B                     ; offset to L175D
7410
7411    L1733:  DEFW    L1332                   ;
7412    L1735:  DEFB    $36                     ; offset to L176B
7413
7414    L1736:  DEFW    L133C                   ;
7415    L1738:  DEFB    $33                     ; offset to L176B
7416
7417    L1739:  DEFW    L10E8                   ;
7418    L173B:  DEFB    $29                     ; offset to L1764
7419
7420    L173C:  DEFW    L1140                   ;
7421    L173E:  DEFB    $26                     ; offset to L1764
7422
7423    L173F:  DEFW    L1011                   ;
7424    L1741:  DEFB    $3B                     ; offset to L177C
7425
7426    L1742:  DEFW    L1064                   ;
7427    L1744:  DEFB    $47                     ; offset to L178B
7428
7429    L1745:  DEFW    L104B                   ;
7430    L1747:  DEFB    $51                     ; offset to L1798
7431
7432    L1748:  DEFW    L1379                   ;
7433    L174A:  DEFB    $62                     ; offset to L17AC
7434
7435    L174B:  DEFW    L1396                   ;
7436    L174D:  DEFB    $63                     ; offset to L17B0
7437
7438    L174E:  DEFW    L04B6                   ;
7439    L1750:  DEFB    $54                     ; offset to L17A4
7440
7441    L1751:  DEFW    $0000                   ; zero end-marker
7442
7443    ; -----------------------------------------------------
7444
7445    ; default action
7446
```

```
7447   L1753:  CALL    L17E1                   ;
7448
7449   L1756:  DEC     (IX+$13)                ; LISTWS
7450           JR      NZ,L1712                ;
7451           AND     A
7452           RET
7453
7454   ; ---
7455
7456   L175D:  LD      HL,($3C14)              ; LISTWS2
7457           LD      H,L
7458           INC     L
7459           JR      L1770                   ;
7460
7461   ; ---
7462
7463   L1764:  LD      HL,($3C14)              ; LISTWS2
7464           LD      H,L
7465           DEC     H
7466           JR      L1770                   ;
7467
7468   ; ---
7469
7470   L176B:  LD      HL,($3C14)              ; LISTWS2
7471           DEC     L
7472           LD      H,L
7473
7474   L1770:  LD      ($3C14),HL              ; LISTWS2
7475           LD      (IX+$13),$01            ; LISTWS
7476           DEC     (IX+$16)                ; LISTWSx
7477           JR      L1753                   ;
7478
7479   ; ---
7480
7481   L177C:  CALL    L17DA                   ;
7482
7483           RST     10H                     ; push word DE
7484           LD      DE,$09B3                ; '.' addr
7485
7486   L1783:  CALL    L17C1                   ; routine INDENT
7487           CALL    L1815                   ; pr2
7488
7489           JR      L1756                   ;
7490
7491   ; ---
7492
7493   L178B:  CALL    L17DA                   ;
7494           RST     10H                     ; push word DE
7495           CALL    L17DA                   ;
7496           RST     10H                     ; push word DE
7497           LD      DE,$0AAF                ; 'F.' addr
7498           JR      L1783                   ;
7499
7500   ; ---
7501
7502   L1798:  LD      A,(BC)
7503           PUSH    AF
7504
7505           CALL    L17E1                   ;
7506
7507           POP     AF
7508           RST     08H                     ; print_ch
7509
7510           LD      A,$20                   ; a space character
7511           RST     08H                     ; print_ch
7512
7513           JR      L1756                   ;
7514
7515   ; ---
7516
7517   L17A4:  CALL    L1808                   ; pr_inline
7518
7519           DEFB    $0D                     ; newline
```

```
7520            DEFB    ';'                     ; ;
7521            DEFB    $8D                     ; inverted newline
7522
7523            SCF                             ;
7524            RET                             ;
7525
7526    ; ---
7527
7528    L17AC:  LD      A,$29                   ; character ')' - end of comment.
7529            JR      L17B2                   ;
7530
7531    L17B0:  LD      A,$22                   ; character '"' - quote
7532
7533    L17B2:  PUSH    AF
7534            PUSH    BC
7535            CALL    L17E1                   ;
7536            POP     DE
7537            CALL    L0979                   ; pr_string1
7538            LD      B,D
7539            LD      C,E
7540            POP     AF
7541
7542            RST     08H                     ; print_ch
7543
7544            AND     A
7545            RET
7546
7547    ; -----------------------------------------------------
7548
7549    L17C1:  LD      A,($3C15)               ; LISTWS3
7550            AND     A
7551            RET     M
7552
7553            PUSH    BC                      ; preserve BC
7554            LD      B,A                     ; transfer count to B
7555
7556            LD      A,$0D                   ; carriage return.
7557            RST     08H                     ; print_ch
7558
7559            INC     B                       ; test indentation.
7560            DEC     B                       ;
7561            JR      Z,L17D4                 ;
7562
7563    L17CF:  LD      A,$20                   ; a space character
7564            RST     08H                     ; print_ch
7565
7566            DJNZ    L17CF                   ;
7567
7568    L17D4:  LD      (IX+$15),$FF            ; LISTWS3
7569
7570            POP     BC                      ; restore BC
7571            RET                             ; return.
7572
7573    ; ---
7574
7575    L17DA:  LD      A,(BC)
7576            LD      E,A
7577            INC     BC
7578            LD      A,(BC)
7579            LD      D,A
7580    L17DF:  INC     BC
7581            RET
7582
7583    ; ---
7584
7585    L17E1:  CALL    L17C1                   ; routine INDENT
7586
7587    L17E4:  EX      DE,HL
7588            DEC     HL
7589            LD      A,(HL)
7590            BIT     7,A
7591            JR      NZ,L17F0                ;
7592
```

```
7593            CALL    L15E8                   ; routine WORDSTART
7594
7595            JR      L17FB                   ;
7596
7597    ; ---
7598
7599    L17F0:  EX      DE,HL
7600
7601            CALL    L15A2                   ;
7602
7603            INC     DE
7604            LD      A,(DE)
7605            LD      L,A
7606            INC     DE
7607            LD      A,(DE)
7608            LD      H,A
7609            ADD     HL,DE
7610
7611    ; pr_string_sp
7612
7613    L17FB:  LD      A,(HL)
7614            AND     $7F
7615            RST     08H                     ; print_ch
7616            BIT     7,(HL)
7617            INC     HL
7618            JR      Z,L17FB                 ;
7619
7620            LD      A,$20
7621            RST     08H                     ; print_ch
7622            RET
7623
7624    ; --------------------------------------
7625    ; THE 'INLINE PRINT STRING SPACE' ROUTINE
7626    ; --------------------------------------
7627    ;
7628
7629    L1808:  EX      (SP),HL
7630            CALL    L17FB                   ; pr_string_sp
7631            EX      (SP),HL
7632            RET
7633
7634    ; ---------------------------
7635    ; THE 'PRINT INTEGER' ROUTINE
7636    ; ---------------------------
7637    ; in HL
7638
7639    ; -> called twice
7640    L180E:  LD      DE,$09B3                ; '.' addr
7641            PUSH    DE                      ; but save it as we need DE?
7642
7643            EX      DE,HL                   ; transfer HL to DE.
7644            RST     10H                     ; push word DE, was HL, on Data Stack.
7645            POP     DE                      ; restore L09B3 again
7646
7647    ; -> called twice.
7648    L1815:  PUSH    BC                      ; preserve BC.
7649
7650            CALL    L04BF                   ; executes '.' word
7651
7652    ; the '.' exits so expects another word here
7653
7654
7655    L1819:  DEFW    L181B
7656
7657    L181B:  DEFW    L181D
7658
7659    L181D   POP     BC                      ;
7660            POP     BC                      ; restore BC.
7661
7662    L181F:  RET                             ; return.
7663
7664    ;;; Vdrive stuff
7665
```

```
7666  F_FORTHEND: .EQU    L04B6
7667  DO_FORTH:   .EQU    L04B9
7668  F_WORD:     .EQU    L05AB
7669  F_DROP:     .EQU    L0879
7670  F_DOCOLON:  .EQU    L0EC3
7671  PR_INLINE:  .EQU    L1808
7672  F_FORTH2ASM:    .EQU    L1A0E
7673  F_WORD2PAD: .EQU    L1A10
7674  CURRENT:    .EQU    $3C31
7675  STKBOT:     .EQU    $3C37
7676  SPARE:      .EQU    $3C3B
7677
7678  ;-------------
7679
7680  IO:     .EQU    $7B         ; IN-PORT VDRIVE
7681  OFF:        .EQU    %10001100   ; VDR:D1..D0;NU:D7..D2
7682  CSDO:       .EQU    %10001111   ; for reading
7683  CS00:       .EQU    %10001110   ; for writing
7684  CS00RS:     .EQU    %01000111   ; RIGHT-SHIFT
7685
7686  ;-------------
7687
7688  TXTDIR: DEFM    "DIR "
7689  TXTRD:  DEFM    "RD "
7690  TXTOPR: DEFM    "OPR "
7691  TXTOPW: DEFM    "OPW "
7692  TXTRDF: DEFM    "RDF ",0,0,0,1      ; by default, request one byte
7693  TXTWRF: DEFM    "WRF "
7694  TXTCLF: DEFM    "CLF "
7695  TXTSEK: DEFM    "SEK ",0,0,0,0      ; by default, seek to the start of the file
7696
7697  ;-------------
7698
7699  RDCHRH: HALT                ; (called from PRTC, RDFLUSH, INIT)
7700  RDCHRC: CALL    L04E4           ; check break
7701      SCF
7702  URWC:   EXX
7703          LD      E,CS00RS
7704          RL      E                   ; ROTATE CARRY TO READ, AND NC TO WRITE
7705          LD      D,A             ; D=CHAR, E=R/W-CODE ; RD-Cycle:110 WR-Cycle:100
7706      LD      C,IO
7707  URWC1:  LD      B,OFF
7708      IN      B,(C)
7709      LD      B,CSDO
7710      IN      B,(C)
7711      LD      B,E             ; R/W-CODE
7712      NOP
7713      IN      B,(C)
7714      LD      B,CS00
7715      IN      B,(C)
7716      PUSH    DE          ; SAVE DE
7717      LD      A,D
7718      LD      E,8
7719  WRBIT:  LD      B,CS00RS
7720      RLCA
7721      RL      B
7722      RRCA                ; RESTORE A
7723      IN      B,(C)
7724      RL      B           ; D7->CY
7725      RL      A           ; CY->D0 NEU
7726      DEC     E
7727      JR      NZ,WRBIT
7728      LD      B,CS00
7729      IN      B,(C)
7730      RL      B           ; STATUS->CY: OK=0
7731      POP     DE
7732      LD      B,OFF
7733      IN      B,(C)
7734          EXX
7735      RET
7736
7737  SEND:   CALL    WRBLK
7738  SENDCR: LD  A,$0D
```

```
7739
7740  WRCHR:  AND A
7741      CALL    URWC
7742      JR  C,WRCHR         ; buffer was full?
7743      RET
7744
7745  WRBLK:  LD  A,(HL)
7746      CALL    WRCHR
7747      INC     HL
7748      DJNZ    WRBLK
7749      RET
7750
7751  RDCHR:  CALL    RDCHRC
7752      JR  C,RDCHR
7753      RET
7754
7755  ;-----------------------------------
7756  ; PRT=PROMPT: READ&PRINT UNTIL ">"#0D
7757  ;-----------------------------------
7758  PRT:    CALL    RDCR
7759      LD  A,B
7760      CP  $3E         ; ">"
7761      JR  NZ,PRT
7762      RET
7763
7764  PRTDN:  CALL    PRT
7765      JP  (IY)
7766
7767  ;--------------------------
7768  ; RDCR=READ&PRINT UNTIL #0D
7769  ;--------------------------
7770  RDCR:   LD  C,$0D           ; CR
7771  RDCR1:  LD  B,A         ; save previous character
7772      CALL    RDCHR
7773      PUSH    AF
7774      RST 08H         ; EMIT A->SCREEN - main registers are preserved
7775      POP AF
7776      CP  C
7777      JR  NZ,RDCR1
7778      RET
7779
7780  ;--------------------------
7781  ; PRTC=READ&PRINT UNTIL END
7782  ;--------------------------
7783  PRTC:   CALL    RDCHRH          ; wait for response
7784      JR  C,PRTC
7785  PRTC1:  RST     08H                     ; print it
7786          CALL    RDCHRH                  ; next
7787          JR      NC,PRTC1
7788          RET
7789
7790  ;-------------------
7791  ; RDFLUSH=FLUSH&SYNC
7792  ;-------------------
7793  F_FLUSH:DEFW    F_FLUSH+2
7794      CALL    RDFLUSH
7795      JP  (IY)            ; !
7796
7797  RDFLUSH:HALT                ; delay of at least 20 ms
7798      CALL    RDFLSH2
7799      LD  A,'E'
7800      CALL    SYNC
7801      JR  NZ,RDFLUSH
7802      LD  A,'e'
7803      CALL    SYNC
7804      JR  NZ,RDFLUSH
7805      RET
7806
7807  RDFLSH2:CALL    RDCHRH
7808      JR  NC,RDFLSH2
7809      RET
7810
7811  SYNC:   PUSH    AF
```

```
7812        CALL    WRCHR
7813        LD  A,$0D
7814        CALL    WRCHR
7815        LD  B,0
7816    SYNC1:  CALL    RDCHRH
7817        JR  NC,SYNC2
7818        DJNZ    SYNC1
7819        RST     20H                     ; Error 10
7820            DEFB    $0A                     ; Tape error
7821    SYNC2:  POP BC
7822        CP  B           ; test if received char is same as sent
7823        RET NZ
7824        CALL    RDCHRH
7825        CP  $0D
7826        RET
7827
7828
7829    ; ----------------
7830    ; THE 'WRCHR' WORD
7831    ; ----------------
7832
7833    N_WRCHR:
7834        .BYTE   "WRCH",'R' | INVERSE    ; Word Name (last letter inverse)
7835            .WORD   L_V                 ; Link Field
7836    L_WRCHR:
7837        .BYTE   $ - N_WRCHR - 2     ; Name Length Field
7838    ;;; --- code ---
7839            DEFW    C_WRCHR
7840    C_WRCHR:RST 18H
7841        LD  A,E
7842        CALL    WRCHR
7843        JP  (IY)
7844
7845    ; ----------------
7846    ; THE 'RDCHR' WORD
7847    ; ----------------
7848
7849    N_RDCHR:
7850        .BYTE   "RDCH",'R' | INVERSE    ; Word Name (last letter inverse)
7851            .WORD   L_WRCHR                 ; Link Field
7852    L_RDCHR:
7853        .BYTE   $ - N_RDCHR - 2     ; Name Length Field
7854    ;;; --- code ---
7855            DEFW    C_RDCHR
7856    C_RDCHR:CALL    RDCHR
7857        LD  D,0
7858        LD  E,A
7859        RST 10H
7860        JP  (IY)
7861
7862    ;;; for compatibility
7863        DEFS    $192D - $, $F3      ; free space; DI to force stop if executed
7864
7865    ;;; FORTH dictionary word header
7866    N_SAVE:
7867        .BYTE   "SAV",'E' | INVERSE     ; Word Name (last letter inverse)
7868            .WORD   L1989                   ; Link Field
7869    L_SAVE:
7870        .BYTE   $ - N_SAVE - 2      ; Name Length Field
7871    ;;; --- code ---
7872            DEFW    F_DOCOLON               ; 'code field' - docolon
7873            DEFW    F_WORD2PAD              ; word to pad
7874            DEFW    L1A4F                   ; the works
7875            DEFW    F_FORTHEND              ; exit
7876    E_SAVE:
7877    ;;; ---
7878
7879    ;;; for compatibility
7880        DEFS    $193C - $, $F3      ; free space; DI to force stop if executed
7881
7882    ;;; FORTH dictionary word header
7883    N_BSAVE:
7884        .BYTE   "BSAV",'E' | INVERSE    ; Word Name (last letter inverse)
```

```
7885            .WORD   L_SAVE                  ; Link Field
7886    L_BSAVE:
7887        .BYTE   $ - N_BSAVE - 2     ; Name Length Field
7888    ;;; --- code ---
7889            DEFW    F_DOCOLON               ; 'code field' - docolon
7890            DEFW    L1A3D                   ; prep_header
7891            DEFW    L1A4F                   ; the works
7892            DEFW    F_FORTHEND              ; exit
7893    E_BSAVE:
7894    ;;; ---
7895
7896
7897    ;;; for compatibility
7898        DEFS    $194C - $, $F3      ; free space; DI to force stop if executed
7899
7900    ; ----------------
7901    ; THE 'BLOAD' WORD
7902    ; ----------------
7903    ; BLOAD name
7904    ; (m, n -- )
7905    ; Load at most n bytes of bytes type cassette file 'name' starting at
7906    ; address m. ERROR 10 if the file has more than m bytes.
7907    ;
7908    N_BLOAD:
7909            DEFM    "BLOA"                  ; 'name field'
7910            DEFB    'D' + $80
7911            DEFW    F_ALOAD-1               ; 'link field'
7912            DEFB    $-N_BLOAD-2             ; 'name length field'
7913    F_BLOAD:
7914            DEFW    F_DOCOLON               ; 'code field' - docolon
7915            DEFW    L1A3D                   ; prep_header
7916        DEFW    F_FLENGTH       ; to check if it exists
7917        DEFW    F_DROP
7918        DEFW    F_FLUSH
7919        DEFW    F_VRD
7920            DEFW    L1A74                   ; ld-bytes??
7921            DEFW    L1AB8                   ; tapeFF
7922            DEFW    L04B6               ; exit
7923
7924    ;;; dummy VERIFY for compatibility
7925        DEFS    $1966 - $, $F3      ; free space; DI to force stop if executed
7926
7927        DEFB    0
7928        DEFW    L0EC3
7929        DEFW    L04B6
7930
7931    ;;; dummy BVERIFY for compatibility
7932        DEFS    $1978 - $, $F3      ; free space; DI to force stop if executed
7933
7934        DEFB    0
7935        DEFW    L0EC3
7936        DEFW    L04B6
7937
7938    ;;; for compatibility
7939        DEFS    $1983 - $, $F3      ; free space; DI to force stop if executed
7940
7941    ; ---------------
7942    ; THE 'LOAD' WORD
7943    ; ---------------
7944    ; LOAD name
7945    ; (  --  )
7946    ; Searches for a dictionary cassette file 'name' and loads it in, adding it
7947    ; to end of old dictionary. Writes to the screen all files found on tape.
7948    ; For best results turn the tone control on the tape recorder right down
7949    ; (as bass as possible) and the volume control to about three-quarters
7950    ; maximum.
7951
7952    N_LOAD:
7953            DEFM    "LOA"                   ; 'name field'
7954            DEFB    'D' + $80
7955            DEFW    F_BLOAD-1               ; 'link field'
7956    L1989:  DEFB    $-N_LOAD-2              ; 'name length field'
7957    F_LOAD:
```

```
7958            DEFW    F_DOCOLON               ; 'code field' - docolon
7959        DEFW    F_WORD2PAD              ; word to pad
7960        DEFW    F_FLENGTH       ; to check if it exists
7961        DEFW    F_DROP
7962        DEFW    F_FLUSH
7963            DEFW    F_FORTH2ASM             ; end-forth.
7964
7965            LD      HL,(STKBOT)             ; STKBOT
7966            LD      ($230E),HL
7967            EX      DE,HL
7968            LD      HL,$FFCC
7969            ADD     HL,SP
7970            AND     A
7971            SBC     HL,DE
7972            LD      ($230C),HL
7973
7974            CALL    DO_FORTH                ; forth
7975        DEFW    F_VRD
7976        DEFW    L1A74                   ; ld_bytes
7977            DEFW    L1AB8                   ; tapeFF
7978            DEFW    F_FORTH2ASM             ; end-forth.
7979
7980            LD      BC,(STKBOT)             ; STKBOT
7981            LD      HL,$3C50
7982            LD      ($2701),HL
7983            INC     HL
7984            LD      ($2709),HL
7985            LD      HL,($2325)
7986            ADD     HL,BC
7987            LD      (STKBOT),HL             ; STKBOT
7988            LD      HL,$C3AF
7989            ADD     HL,BC
7990            LD      ($270B),HL
7991            LD      DE,($2329)
7992            ADD     HL,DE
7993            LD      DE,($3C4C)
7994            LD      ($3C4C),HL
7995            PUSH    BC
7996            PUSH    DE
7997
7998    L19D4:  LD      ($270D),SP
7999            CALL    L1504                   ;
8000            POP     BC
8001            POP     HL
8002    L19DD:  BIT     7,(HL)
8003            INC     HL
8004            JR      Z,L19DD                 ;
8005            INC     HL
8006            INC     HL
8007            LD      (HL),C
8008            INC     HL
8009            LD      (HL),B
8010            LD      HL,(STKBOT)             ; STKBOT
8011            LD      BC,$000C                ; allow twelve bytes for underflow.
8012            ADD     HL,BC
8013            LD      (SPARE),HL              ; SPARE
8014
8015        JP  (IY)            ; done
8016
8017    ; ---
8018
8019
8020    L1A3D:  DEFW    L0EC3                   ; 'code field' - docolon
8021            DEFW    L19F3                   ; word to pad
8022            DEFW    L1011                   ; stack next word
8023            DEFW    $230C                   ; header location
8024            DEFW    L08C1                   ; !     store int at address
8025            DEFW    L1011                   ; stack next word
8026            DEFW    $230E                   ; header location
8027            DEFW    L08C1                   ; !     store int at address
8028            DEFW    L04B6                   ; exit
8029
8030    ;;; for compatibility
```

```
8031        DEFS    $1A0E - $, $F3      ; free space; DI to force stop if executed
8032    L1A0E:  DEFW    L181F           ; address of RET
8033
8034    F_VRD:
8035        DEFW    VRD
8036    VRD:
8037        LD  B,$03
8038        LD  HL,TXTRD
8039            CALL    WRBLK           ; "RD " -> VDR
8040        EX  AF,AF'          ; get length of name
8041        LD  B,A
8042        LD  HL,$2302
8043        CALL    SEND
8044        JP  (IY)
8045
8046    ; ------------------
8047    ; INIT=VERBOSE FLUSH
8048    ; ------------------
8049    N_INIT:
8050        DEFM    "INI"
8051        DEFB    'T' + $80
8052        DEFW    L166F
8053        DEFB    $-N_INIT-2
8054    F_INIT: DEFW    C_INIT
8055
8056    ;;; first wait for two different bytes to appear;
8057    ;;; the Vdrive may not be connected, or its buffer may be empty
8058
8059    C_INIT: LD  B,0
8060        CALL    RDCHRH
8061    INIT1:  LD  C,A
8062        CALL    RDCHRH
8063        CP  C
8064        JR  NZ,INIT2
8065        DJNZ    INIT1
8066
8067        JR  ISYNC
8068
8069    INIT2:  LD  B,A
8070        LD  A,C
8071        RST 08H         ; also print the first byte received
8072        LD  A,B
8073
8074    IPRLP:  LD  B,100           ; allow 2 sec between init messages
8075    IPRLP1: RST 08H
8076    IPRLP2: CALL    RDCHRH
8077        JR  NC,IPRLP1
8078        DJNZ    IPRLP2
8079
8080    ISYNC:  LD  A,'E'
8081        CALL    SYNC
8082        JR  Z,IDONE
8083        RST 20H
8084        DEFB    $0A
8085
8086    IDONE:  JP  (IY)
8087
8088
8089    ; ---
8090    ; READ BYTES FROM VIRTUAL TAPE
8091    ; ---
8092
8093    L18A7:  HALT                ; wait a little before reading
8094        DI
8095        EX  AF,AF'          ; save carry flag
8096        CALL    L18FC           ; get 1st header byte
8097        CALL    L18FC           ; get 2nd header byte
8098        LD  B,0         ; initialize checksum
8099        CALL    L18F0           ; start reading
8100        EI
8101        RET             ; carry will be checked by the calling routine
8102
8103    L18DF:  EX  AF,AF'          ; get carry flag
```

```
8104        JR  NC,L18E7        ; verify if not set
8105        LD  (HL),C          ; load to memory if set
8106        JR  L18EC           ; next
8107
8108    L18E7:  LD  A,(HL)          ; compare read byte with memory
8109        XOR C
8110        RET NZ          ; return if verify fails
8111
8112    L18EC:  INC HL          ; next memory location
8113        DEC DE          ; decrease number of bytes to read
8114        EX  AF,AF'          ; save carry flag
8115
8116    L18F0:  CALL    L18FC           ; get byte
8117        LD  A,D         ; check number of bytes to read
8118        OR  E
8119        JR  NZ,L18DF
8120        SUB B           ; check checksum
8121    L18FB:  RET
8122
8123    L18FC:  CALL    RDCHR           ; read byte from vdrive
8124        LD  C,A
8125        XOR B           ; checksum; resets carry flag
8126        LD  B,A
8127        RET
8128
8129    ; --------------
8130    ; THE 'DIR' WORD
8131    ; --------------
8132    N_DIR:
8133            DEFM    "DI"                    ; 'name field'
8134            DEFB    'R' + $80
8135            DEFW    F_INIT-1                ; 'link field'
8136            DEFB    $-N_DIR-2               ; 'name length field'
8137    F_DIR:
8138            DEFW    F_DOCOLON               ; 'code field' - docolon
8139        DEFW    F_WORD2PAD              ; word to pad
8140            DEFW    F_FORTH2ASM             ; end-forth.
8141
8142        EX  AF,AF'          ; get length of name
8143        CP  0
8144        JR  NZ,F_DIR2
8145
8146        LD  B,$03
8147        LD  HL,TXTDIR
8148            CALL    WRBLK               ; "DIR" -> VDR
8149        CALL    SENDCR          ; get directory listing
8150
8151    ;;;  Slightly modified from Rune's proposal
8152
8153        CALL    DO_FORTH
8154        DEFW    L0A1C+1         ; CLS
8155        DEFW    L129F           ; BEGIN
8156        DEFW    L1011           ; stack next word
8157        DEFW    $3C1C           ; SCRAPS
8158        DEFW    L08B3           ; @
8159        DEFW    L1011           ; stack next word
8160        DEFW    $267F           ; address in display
8161        DEFW    L0C56           ; >
8162        DEFW    L1283           ; ?branch
8163        DEFW    $001C           ; offset
8164        DEFW    L129F           ; BEGIN
8165        DEFW    L1011           ; stack next word
8166        DEFW    $267F
8167        DEFW    L1011           ; stack next word
8168        DEFW    $3C1C
8169        DEFW    L08C1           ; !
8170        DEFW    L1396           ; pr_embedded string
8171        DEFB    1,0,$2A
8172        DEFW    L0BDB           ; inkey
8173        DEFW    L128D           ; ?branch
8174        DEFW    $FFEC           ; offset
8175        DEFW    L0A1C+1         ; CLS
8176        DEFW    L12A4           ; THEN
```

```
8177        DEFW    L_RDCHR+1       ; RDCHR
8178        DEFW    L086B           ; DUP
8179        DEFW    L0AA3           ; EMIT
8180        DEFW    L1011           ; stack next word
8181        DEFW    $003E           ; >
8182        DEFW    L0C4A           ; =
8183        DEFW    L128D           ; ?branch
8184        DEFW    $FFC6           ; offset
8185        DEFW    F_FORTHEND
8186
8187    F_DIR2: EX  AF,AF'          ; save length again
8188        CALL    DO_FORTH
8189        DEFW    F_FLENGTH
8190        DEFW    L09D0           ; U.
8191        DEFW    F_FORTH2ASM
8192        JP  PRTDN
8193
8194    F_FLENGTH:
8195        DEFW    FLENGTH
8196    FLENGTH:LD  B,$04
8197        LD  HL,TXTDIR
8198            CALL    WRBLK               ; "DIR " -> VDR
8199        EX  AF,AF'          ; get length of name
8200        LD  B,A
8201        EX  AF,AF'          ; save it again
8202        LD  HL,$2302        ; B must be set to file length
8203        CALL    SEND
8204
8205        LD  C,' '           ; name and length are separated by a space
8206        CALL    RDCR1           ; print intermediate characters (filename)
8207        CALL    RDCHR           ; get length (binary mode)
8208        LD  E,A
8209        CALL    RDCHR
8210        LD  D,A
8211        CALL    RDCHR
8212        LD  C,A
8213        CALL    RDCHR
8214        LD  B,A
8215        CP  0           ; the MSB should be zero, otherwise error
8216        JR  NZ,FLERR
8217        RST 10H         ; push DE to stack
8218        JP  (IY)            ; !
8219
8220    FLERR:  LD  A,E         ; output the chars that were apparently not the length
8221        RST 08H
8222        LD  A,D
8223        RST 08H
8224        LD  A,C
8225        RST 08H
8226        LD  A,B
8227        RST 08H
8228        CALL    RDCR
8229        RST 20H
8230        DEFB    $0D         ; not found
8231
8232    ; ----------------
8233    ; THE 'ALOAD' WORD
8234    ; ----------------
8235    N_ALOAD:
8236            DEFM    "ALOA"                  ; 'name field'
8237            DEFB    'D' + $80
8238            DEFW    F_DIR-1                 ; 'link field'
8239            DEFB    $-N_ALOAD-2             ; 'name length field'
8240    F_ALOAD:
8241            DEFW    F_DOCOLON               ; 'code field' - docolon
8242        DEFW    F_WORD2PAD              ; word to pad
8243        DEFW    F_FLENGTH       ; now length of file is available on the Forth stack
8244        DEFW    F_FLUSH
8245            DEFW    F_FORTH2ASM             ; end-forth.
8246
8247    ;;; the pad may be used for other purposes, so we put the current file name on the
        stack
8248
```

```
8249  NAMSAV: LD  B,5         ; save current name, max is 10 bytes, packed
8250      LD  HL,$230C        ; stack it with first letter having the lowest address
8251  NMSV1:  DEC HL
8252      LD  D,(HL)
8253      DEC HL
8254      LD  E,(HL)
8255      PUSH    DE
8256      DJNZ    NMSV1
8257      EX  AF,AF'          ;' store the actual length of the filename
8258      PUSH    AF
8259
8260      LD  A,$0D
8261      RST 08H
8262
8263      CALL    L02D8           ; routine SETBUF
8264          CALL    L0276               ; routine pr_cursor
8265
8266      RST 18H         ; get file length from stack in DE
8267      PUSH    DE          ; save it
8268
8269      LD  HL,TXTOPR
8270          CALL    SNDCMDP
8271
8272  ALDLP:  SCF             ; indicator of presence of control chars
8273  ALDLP1: PUSH    AF
8274  ALDLP2: LD  A,D
8275      OR  E
8276      DEC DE
8277      JR  Z,ALDEND
8278      CALL    RD1CHR
8279      CP  ' '         ; test for control char
8280      JR  NC,ALDLP3
8281      POP AF
8282      JR  C,ALDLP1
8283
8284      LD  HL,TXTCLF
8285      CALL    SNDCMDP
8286
8287      PUSH    DE
8288          CALL    L0225               ; routine DEL-CURSOR
8289          CALL    DO_FORTH
8290      DEFW    L0506           ; interpret buffer
8291      DEFW    L0536           ; print OK
8292          DEFW    F_FORTH2ASM         ; end-forth.
8293      CALL    L02D8           ; set buffer
8294          CALL    L0276               ; routine pr_cursor
8295      POP DE
8296
8297      LD  A,D
8298      OR  E
8299      JR  Z,ALDEND2
8300
8301      LD  HL,TXTOPR
8302      CALL    SNDCMDP
8303
8304      POP HL          ; get length of file
8305      PUSH    HL
8306      AND A
8307      SBC HL,DE           ; number of bytes to seek
8308      PUSH    DE
8309      EX  DE,HL
8310
8311      LD  B,6         ; seek to previous file position
8312      LD  HL,TXTSEK
8313      CALL    WRBLK
8314      LD  A,D
8315      CALL    WRCHR
8316      LD  A,E
8317      CALL    WRCHR
8318      LD  A,$0D
8319      CALL    WRCHR
8320      CALL    PRFL
8321
```

```
8322        POP DE
8323        JR  ALDLP
8324
8325    ALDLP3: POP HL          ; discard pushed AF
8326        PUSH    AF          ; with NC
8327        PUSH    DE
8328        CALL    L0196           ; to input buffer
8329        CALL    L0282           ; routine pr_cursor
8330        POP DE
8331        JR  ALDLP2
8332
8333    ALDEND: POP AF
8334
8335        LD  HL,TXTCLF
8336            CALL    SNDCMDP
8337
8338    ALDEND2:CALL    L0225
8339
8340        POP DE          ; length of file
8341
8342        POP AF          ; length of filename
8343    NAMLD:  LD  B,5         ; get current name, max is 10 bytes, packed
8344        LD  HL,$2302
8345    NMLD1:  POP DE
8346        LD  (HL),E
8347        INC HL
8348        LD  (HL),D
8349        INC HL
8350        DJNZ    NMLD1
8351
8352        JP  (IY)
8353
8354    SNDCMDS:LD  B,$04
8355        CALL    WRBLK
8356        LD  HL,7            ;!
8357        ADD HL,SP
8358        LD  B,(HL)
8359        INC HL
8360        JP  SEND
8361
8362    SNDCMDP:CALL    SNDCMDS
8363    PRFL:   CALL    RDCHR           ; efficient flush of prompt
8364        CP  $0D
8365        JR  NZ,PRFL
8366        RET
8367
8368    RD1CHR: LD  HL,TXTRDF
8369        LD  B,8
8370        CALL    SEND
8371        CALL    RDCHR           ; we now have the required byte available
8372        PUSH    AF
8373        CALL    PRFL
8374        POP AF
8375        RET
8376
8377    ; ---
8378
8379    L19F3:  DEFW    F_DOCOLON               ; 'code field' - docolon
8380        DEFW    F_FLUSH
8381            DEFW    L104B                   ; stk_data
8382            DEFB    $20                     ; a space delimiter
8383            DEFW    L05AB                   ; word          (to pad)
8384            DEFW    L1A0E                   ; end-forth.
8385
8386    ; ---
8387
8388    L19FC:  CALL    L0F2E                   ; blank stack
8389
8390    L19FF:  RST     18H                     ; pop word DE
8391
8392        LD  A,(DE)
8393        EX  AF,AF'          ; save length
8394
```

```
8395            LD      A,$20                   ;
8396            LD      (DE),A                  ;
8397            LD      DE,$270C                ;
8398            LD      HL,$27FF                ;
8399
8400            CALL    L07FA                   ; routine SPACE_FILL
8401
8402            JP      (IY)                    ; to 'next'.
8403
8404    ; ---
8405
8406    L1A10:  DEFW    L0EC3                   ; 'code field' - docolon
8407            DEFW    L19F3                   ; word to pad
8408            DEFW    L1A0E                   ; end-forth.
8409
8410            XOR     A                       ;
8411            LD      ($2301),A               ;
8412            LD      HL,$3C51                ;
8413            LD      ($230E),HL              ;
8414            EX      DE,HL                   ;
8415            LD      HL,(STKBOT)             ; STKBOT
8416            AND     A                       ;
8417            SBC     HL,DE
8418            LD      ($230C),HL
8419            LD      HL,($3C4C)
8420            LD      ($2310),HL
8421            LD      HL,CURRENT              ; CURRENT
8422            LD      DE,$2312
8423            LD      BC,$0008                ;
8424
8425            LDIR                            ;
8426
8427            JP      (IY)                    ; to 'next'.
8428
8429    ; ---
8430    ; ld_bytes
8431    ; ---
8432
8433    L1A74:  DEFW    L1A76
8434
8435    ; ---
8436    ; read header
8437    ; ---
8438
8439    L1A76:  LD      DE,$0019            ; header length
8440            LD      HL,$231A        ; address (pad)
8441            LD      C,D         ; clear C to indicate header
8442            SCF                 ; do not verify
8443            CALL    L18A7                   ; actual reading of bytes
8444            JR      C,L1AB6                 ; tape error
8445
8446            LD      DE,$231A
8447            LD      A,(DE)
8448            AND     A               ; zero for Dict
8449            JR      NZ,L1A95                ; otherwise Bytes
8450
8451            CALL    PR_INLINE               ; pr_inline
8452    L1A8D:  DEFB    $0D                     ; newline
8453            DEFM    "Dict"
8454            DEFB    ':' + $80               ;
8455
8456    L1A93:  JR      L1A9F                   ;
8457
8458    ; ---
8459
8460    L1A95:  CALL    PR_INLINE               ; pr_inline
8461    L1A98:  DEFB    $0D                     ; newline
8462            DEFM    "Bytes"
8463            DEFB    ':' + $80               ;
8464
8465    ; --- print name
8466
8467    L1A9F:  INC DE
```

```
8468            LD      B,$0A
8469    L1AA7:  LD      A,(DE)
8470            RST     08H                     ; print_ch
8471        INC DE
8472            DJNZ    L1AA7                   ; next char
8473        LD  A,$0D
8474        RST 08H
8475
8476            JP      (IY)                    ; to 'next'.
8477
8478    ; ---
8479    ; Tape error
8480    ; ---
8481
8482    L1AB6:  RST     20H                     ; Error 10
8483            DEFB    $0A                     ; Tape error
8484
8485    ; ---
8486    ; read data
8487    ; ---
8488
8489    L1AB8:  DEFW    L1ABA                   ; headerless 'code field'
8490    L1ABA:  LD      B,$FF
8491        LD      HL,($230C)          ; address
8492            LD      DE,($2325)              ; length
8493            LD      A,H
8494            OR      L
8495            JR      Z,L1ADF                 ; skip if zero
8496            SBC     HL,DE
8497            JR      C,L1AB6                 ; back to tape error
8498    L1ADF:  LD      HL,($230E)
8499            LD      A,H
8500            OR      L
8501            JR      NZ,L1AE9                ; skip if zero
8502            LD      HL,($2327)
8503    L1AE9:  LD      C,$FF
8504            RR      B               ; sets carry, do not verify
8505            CALL    L18A7                   ; actual reading of bytes
8506            JR      C,L1AB6                 ; back to report tape error
8507        JP  PRTDN
8508
8509    SNDCMD:
8510        LD  B,$04
8511            CALL    WRBLK
8512        EX  AF,AF'          ; get length of name
8513        LD  B,A
8514        EX  AF,AF'          ; save length again
8515        LD  HL,$2302
8516        JP  SEND
8517
8518    ;;; ---
8519
8520    L1A4F:  DEFW    L1A51
8521
8522    L1A51:  EX  AF,AF'              ; length of word in pad
8523            AND     A
8524            JP      Z,L1AB6                 ; forward if null.
8525        EX  AF,AF'          ; save length again
8526
8527            LD      HL,($230C)      ; get data length
8528            LD      A,H
8529            OR      L
8530            JP      Z,L1AB6                 ; don't save a file with zero data
8531        PUSH    HL          ; save length
8532
8533        LD  HL,TXTOPW       ; open file
8534        CALL    SNDCMD
8535        CALL    PRTC
8536
8537        LD  HL,TXTSEK
8538        LD  B,8
8539        CALL    SEND
8540        CALL    PRTC
```

```
8541
8542        LD  B,$04
8543        LD  HL,TXTWRF
8544        CALL    WRBLK
8545
8546        POP HL          ; get data length
8547            PUSH    HL          ; save data length again
8548            LD      DE,$001F        ; for total length
8549        ADD HL,DE           ; total length
8550        LD  E,$19                   ; header length
8551
8552        XOR A
8553        CALL    WRCHR
8554        XOR A
8555        CALL    WRCHR
8556        LD  A,H         ; send total length to Vdrive
8557        CALL    WRCHR
8558        LD  A,L
8559        CALL    WRCHR
8560        LD  A,$0D
8561        CALL    WRCHR
8562
8563            LD      HL,$2301                ; pad using ROM priority
8564
8565            CALL    L1820                   ; write data
8566
8567            POP     DE          ; get data length
8568            LD      HL,($230E)              ; data address
8569
8570            CALL    L1820                   ; write data
8571
8572        CALL    PRTC
8573
8574        LD  HL,TXTCLF
8575        CALL    SNDCMD
8576        JP  PRTDN           ; done
8577
8578    ; ---
8579
8580    L1820:  DI
8581
8582        INC DE
8583        LD  A,E         ; write length
8584        CALL    WRCHR
8585        LD  A,D
8586        CALL    WRCHR
8587        DEC DE
8588
8589        LD  B,0         ; initialize checksum
8590
8591    SVLP:   LD  C,(HL)
8592        LD  A,B
8593        XOR C
8594        LD  B,A
8595        LD  A,C
8596
8597        CALL    WRCHR           ; to vdrive
8598        INC HL
8599        DEC DE
8600        LD  A,D
8601        OR  E
8602        JR  NZ,SVLP
8603
8604        LD  A,B
8605        CALL    WRCHR           ; write calculated checksum
8606
8607        EI
8608        RET
8609
8610    ;;; ---
8611
8612    WRD2PAD:LD  E,$20
8613        RST 10H
```

```
8614        CALL    DO_FORTH
8615        DEFW    F_WORD
8616        DEFW    F_FORTH2ASM
8617        RST 18H
8618        EX  DE,HL
8619        LD  B,(HL)
8620        INC HL
8621        XOR A
8622        CP  B
8623        RET
8624
8625    ; --------------
8626    ; THE 'VER' WORD
8627    ; --------------
8628
8629    N_VER:
8630        .BYTE   "VE",'R' | INVERSE  ; Word Name (last letter inverse)
8631            .WORD   L_RDCHR                 ; Link Field
8632    L_VER:
8633        .BYTE   $ - N_VER - 2           ; Name Length Field
8634    ;;; --- code ---
8635            DEFW    C_VER
8636    C_VER:  CALL    L1808           ; pr_inline
8637        DEFM    "3."
8638        DEFB    '5' | INVERSE
8639        JP  (IY)
8640    ;;; ---
8641
8642    ; -----
8643    ; spare
8644    ; -----
8645
8646        DEFS    $1D7B - $, $F3      ; free space; DI to force stop if executed
8647
8648    ; -------------------
8649    ; THE 'CHARACTER SET'
8650    ; -------------------
8651    ; The 96 ASCII character bitmaps are copied to RAM during initialization and
8652    ; the 8x8 characters can afterwards be redefined by the user.
8653    ; Some ROM space is saved by supplying the blank top line of most characters
8654    ; and in case of the middle range (capitals with no descenders) the bottom
8655    ; line as well. Only the final copyright symbol is held in ROM as an 8x8
8656    ; character.
8657
8658
8659    ; $20 - Character: ' '          CHR$(32)
8660
8661    L1D7B:  DEFB    %00000000
8662            DEFB    %00000000
8663            DEFB    %00000000
8664            DEFB    %00000000
8665            DEFB    %00000000
8666            DEFB    %00000000
8667            DEFB    %00000000
8668
8669    ; $21 - Character: '!'          CHR$(33)
8670
8671            DEFB    %00010000
8672            DEFB    %00010000
8673            DEFB    %00010000
8674            DEFB    %00010000
8675            DEFB    %00000000
8676            DEFB    %00010000
8677            DEFB    %00000000
8678
8679    ; $22 - Character: '"'          CHR$(34)
8680
8681            DEFB    %00100100
8682            DEFB    %00100100
8683            DEFB    %00000000
8684            DEFB    %00000000
8685            DEFB    %00000000
8686            DEFB    %00000000
```

```
8687            DEFB    %00000000
8688
8689    ; $23 - Character: '#'          CHR$(35)
8690
8691            DEFB    %00100100
8692            DEFB    %01111110
8693            DEFB    %00100100
8694            DEFB    %00100100
8695            DEFB    %01111110
8696            DEFB    %00100100
8697            DEFB    %00000000
8698
8699    ; $24 - Character: '$'          CHR$(36)
8700
8701            DEFB    %00001000
8702            DEFB    %00111110
8703            DEFB    %00101000
8704            DEFB    %00111110
8705            DEFB    %00001010
8706            DEFB    %00111110
8707            DEFB    %00001000
8708
8709    ; $25 - Character: '%'          CHR$(37)
8710
8711            DEFB    %01100010
8712            DEFB    %01100100
8713            DEFB    %00001000
8714            DEFB    %00010000
8715            DEFB    %00100110
8716            DEFB    %01000110
8717            DEFB    %00000000
8718
8719    ; $26 - Character: '&'          CHR$(38)
8720
8721            DEFB    %00010000
8722            DEFB    %00101000
8723            DEFB    %00010000
8724            DEFB    %00101010
8725            DEFB    %01000100
8726            DEFB    %00111010
8727            DEFB    %00000000
8728
8729    ; $27 - Character: '''          CHR$(39)
8730
8731            DEFB    %00001000
8732            DEFB    %00010000
8733            DEFB    %00000000
8734            DEFB    %00000000
8735            DEFB    %00000000
8736            DEFB    %00000000
8737            DEFB    %00000000
8738
8739    ; $28 - Character: '('          CHR$(40)
8740
8741            DEFB    %00000100
8742            DEFB    %00001000
8743            DEFB    %00001000
8744            DEFB    %00001000
8745            DEFB    %00001000
8746            DEFB    %00000100
8747            DEFB    %00000000
8748
8749    ; $29 - Character: ')'          CHR$(42)
8750
8751            DEFB    %00100000
8752            DEFB    %00010000
8753            DEFB    %00010000
8754            DEFB    %00010000
8755            DEFB    %00010000
8756            DEFB    %00100000
8757            DEFB    %00000000
8758
8759    ; $2A - Character: '*'          CHR$(42)
```

```
8760
8761            DEFB    %00000000
8762            DEFB    %00010100
8763            DEFB    %00001000
8764            DEFB    %00111110
8765            DEFB    %00001000
8766            DEFB    %00010100
8767            DEFB    %00000000
8768
8769    ; $2B - Character: '+'          CHR$(43)
8770
8771            DEFB    %00000000
8772            DEFB    %00001000
8773            DEFB    %00001000
8774            DEFB    %00111110
8775            DEFB    %00001000
8776            DEFB    %00001000
8777            DEFB    %00000000
8778
8779    ; $2C - Character: ','          CHR$(44)
8780
8781            DEFB    %00000000
8782            DEFB    %00000000
8783            DEFB    %00000000
8784            DEFB    %00000000
8785            DEFB    %00001000
8786            DEFB    %00001000
8787            DEFB    %00010000
8788
8789    ; $2D - Character: '-'          CHR$(45)
8790
8791            DEFB    %00000000
8792            DEFB    %00000000
8793            DEFB    %00000000
8794            DEFB    %00111110
8795            DEFB    %00000000
8796            DEFB    %00000000
8797            DEFB    %00000000
8798
8799    ; $2E - Character: '.'          CHR$(46)
8800
8801            DEFB    %00000000
8802            DEFB    %00000000
8803            DEFB    %00000000
8804            DEFB    %00000000
8805            DEFB    %00011000
8806            DEFB    %00011000
8807            DEFB    %00000000
8808
8809    ; $2F - Character: '/'          CHR$(47)
8810
8811            DEFB    %00000000
8812            DEFB    %00000010
8813            DEFB    %00000100
8814            DEFB    %00001000
8815            DEFB    %00010000
8816            DEFB    %00100000
8817            DEFB    %00000000
8818
8819    ; $30 - Character: '0'          CHR$(48)
8820
8821            DEFB    %00111100
8822            DEFB    %01000110
8823            DEFB    %01001010
8824            DEFB    %01010010
8825            DEFB    %01100010
8826            DEFB    %00111100
8827            DEFB    %00000000
8828
8829    ; $31 - Character: '1'          CHR$(49)
8830
8831            DEFB    %00011000
8832            DEFB    %00101000
```

```
8833            DEFB    %00001000
8834            DEFB    %00001000
8835            DEFB    %00001000
8836            DEFB    %00111110
8837            DEFB    %00000000
8838
8839    ; $32 - Character: '2'          CHR$(50)
8840
8841            DEFB    %00111100
8842            DEFB    %01000010
8843            DEFB    %00000010
8844            DEFB    %00111100
8845            DEFB    %01000000
8846            DEFB    %01111110
8847            DEFB    %00000000
8848
8849    ; $33 - Character: '3'          CHR$(51)
8850
8851            DEFB    %00111100
8852            DEFB    %01000010
8853            DEFB    %00001100
8854            DEFB    %00000010
8855            DEFB    %01000010
8856            DEFB    %00111100
8857            DEFB    %00000000
8858
8859    ; $34 - Character: '4'          CHR$(52)
8860
8861            DEFB    %00001000
8862            DEFB    %00011000
8863            DEFB    %00101000
8864            DEFB    %01001000
8865            DEFB    %01111110
8866            DEFB    %00001000
8867            DEFB    %00000000
8868
8869    ; $35 - Character: '5'          CHR$(53)
8870
8871            DEFB    %01111110
8872            DEFB    %01000000
8873            DEFB    %01111100
8874            DEFB    %00000010
8875            DEFB    %01000010
8876            DEFB    %00111100
8877            DEFB    %00000000
8878
8879    ; $36 - Character: '6'          CHR$(54)
8880
8881            DEFB    %00111100
8882            DEFB    %01000000
8883            DEFB    %01111100
8884            DEFB    %01000010
8885            DEFB    %01000010
8886            DEFB    %00111100
8887            DEFB    %00000000
8888
8889    ; $37 - Character: '7'          CHR$(55)
8890
8891            DEFB    %01111110
8892            DEFB    %00000010
8893            DEFB    %00000100
8894            DEFB    %00001000
8895            DEFB    %00010000
8896            DEFB    %00010000
8897            DEFB    %00000000
8898
8899    ; $38 - Character: '8'          CHR$(56)
8900
8901            DEFB    %00111100
8902            DEFB    %01000010
8903            DEFB    %00111100
8904            DEFB    %01000010
8905            DEFB    %01000010
```

```
8906            DEFB    %00111100
8907            DEFB    %00000000
8908
8909    ; $39 - Character: '9'          CHR$(57)
8910
8911            DEFB    %00111100
8912            DEFB    %01000010
8913            DEFB    %01000010
8914            DEFB    %00111110
8915            DEFB    %00000010
8916            DEFB    %00111100
8917            DEFB    %00000000
8918
8919    ; $3A - Character: ':'          CHR$(58)
8920
8921            DEFB    %00000000
8922            DEFB    %00000000
8923            DEFB    %00010000
8924            DEFB    %00000000
8925            DEFB    %00000000
8926            DEFB    %00010000
8927            DEFB    %00000000
8928
8929    ; $3B - Character: ';'          CHR$(59)
8930
8931            DEFB    %00000000
8932            DEFB    %00010000
8933            DEFB    %00000000
8934            DEFB    %00000000
8935            DEFB    %00010000
8936            DEFB    %00010000
8937            DEFB    %00100000
8938
8939    ; $3C - Character: '<'          CHR$(60)
8940
8941            DEFB    %00000000
8942            DEFB    %00000100
8943            DEFB    %00001000
8944            DEFB    %00010000
8945            DEFB    %00001000
8946            DEFB    %00000100
8947            DEFB    %00000000
8948
8949    ; $3D - Character: '='          CHR$(61)
8950
8951            DEFB    %00000000
8952            DEFB    %00000000
8953            DEFB    %00111110
8954            DEFB    %00000000
8955            DEFB    %00111110
8956            DEFB    %00000000
8957            DEFB    %00000000
8958
8959    ; $3E - Character: '>'          CHR$(62)
8960
8961            DEFB    %00000000
8962            DEFB    %00010000
8963            DEFB    %00001000
8964            DEFB    %00000100
8965            DEFB    %00001000
8966            DEFB    %00010000
8967            DEFB    %00000000
8968
8969    ; $3F - Character: '?'          CHR$(63)
8970
8971            DEFB    %00111100
8972            DEFB    %01000010
8973            DEFB    %00000100
8974            DEFB    %00001000
8975            DEFB    %00000000
8976            DEFB    %00001000
8977
8978    ; $40 - Character: '@'          CHR$(64)
```

```
8979
8980            DEFB    %00111100
8981            DEFB    %01001010
8982            DEFB    %01010110
8983            DEFB    %01011110
8984            DEFB    %01000000
8985            DEFB    %00111100
8986
8987    ; $41 - Character: 'A'          CHR$(65)
8988
8989            DEFB    %00111100
8990            DEFB    %01000010
8991            DEFB    %01000010
8992            DEFB    %01111110
8993            DEFB    %01000010
8994            DEFB    %01000010
8995
8996    ; $42 - Character: 'B'          CHR$(66)
8997
8998            DEFB    %01111100
8999            DEFB    %01000010
9000            DEFB    %01111100
9001            DEFB    %01000010
9002            DEFB    %01000010
9003            DEFB    %01111100
9004
9005    ; $43 - Character: 'C'          CHR$(67)
9006
9007            DEFB    %00111100
9008            DEFB    %01000010
9009            DEFB    %01000000
9010            DEFB    %01000000
9011            DEFB    %01000010
9012            DEFB    %00111100
9013
9014    ; $44 - Character: 'D'          CHR$(68)
9015
9016            DEFB    %01111000
9017            DEFB    %01000100
9018            DEFB    %01000010
9019            DEFB    %01000010
9020            DEFB    %01000100
9021            DEFB    %01111000
9022
9023    ; $45 - Character: 'E'          CHR$(69)
9024
9025            DEFB    %01111110
9026            DEFB    %01000000
9027            DEFB    %01111100
9028            DEFB    %01000000
9029            DEFB    %01000000
9030            DEFB    %01111110
9031
9032    ; $46 - Character: 'F'          CHR$(70)
9033
9034            DEFB    %01111110
9035            DEFB    %01000000
9036            DEFB    %01111100
9037            DEFB    %01000000
9038            DEFB    %01000000
9039            DEFB    %01000000
9040
9041    ; $47 - Character: 'G'          CHR$(71)
9042
9043            DEFB    %00111100
9044            DEFB    %01000010
9045            DEFB    %01000000
9046            DEFB    %01001110
9047            DEFB    %01000010
9048            DEFB    %00111100
9049
9050    ; $48 - Character: 'H'          CHR$(72)
9051
```

```
9052            DEFB    %01000010
9053            DEFB    %01000010
9054            DEFB    %01111110
9055            DEFB    %01000010
9056            DEFB    %01000010
9057            DEFB    %01000010
9058
9059    ; $49 - Character: 'I'          CHR$(73)
9060
9061            DEFB    %00111110
9062            DEFB    %00001000
9063            DEFB    %00001000
9064            DEFB    %00001000
9065            DEFB    %00001000
9066            DEFB    %00111110
9067
9068    ; $4A - Character: 'J'          CHR$(74)
9069
9070            DEFB    %00000010
9071            DEFB    %00000010
9072            DEFB    %00000010
9073            DEFB    %01000010
9074            DEFB    %01000010
9075            DEFB    %00111100
9076
9077    ; $4B - Character: 'K'          CHR$(75)
9078
9079            DEFB    %01000100
9080            DEFB    %01001000
9081            DEFB    %01110000
9082            DEFB    %01001000
9083            DEFB    %01000100
9084            DEFB    %01000010
9085
9086    ; $4C - Character: 'L'          CHR$(76)
9087
9088            DEFB    %01000000
9089            DEFB    %01000000
9090            DEFB    %01000000
9091            DEFB    %01000000
9092            DEFB    %01000000
9093            DEFB    %01111110
9094
9095    ; $4D - Character: 'M'          CHR$(77)
9096
9097            DEFB    %01000010
9098            DEFB    %01100110
9099            DEFB    %01011010
9100            DEFB    %01000010
9101            DEFB    %01000010
9102            DEFB    %01000010
9103
9104    ; $4E - Character: 'N'          CHR$(78)
9105
9106            DEFB    %01000010
9107            DEFB    %01100010
9108            DEFB    %01010010
9109            DEFB    %01001010
9110            DEFB    %01000110
9111            DEFB    %01000010
9112
9113    ; $4F - Character: 'O'          CHR$(79)
9114
9115            DEFB    %00111100
9116            DEFB    %01000010
9117            DEFB    %01000010
9118            DEFB    %01000010
9119            DEFB    %01000010
9120            DEFB    %00111100
9121
9122    ; $50 - Character: 'P'          CHR$(80)
9123
9124            DEFB    %01111100
```

```
9125            DEFB    %01000010
9126            DEFB    %01000010
9127            DEFB    %01111100
9128            DEFB    %01000000
9129            DEFB    %01000000
9130
9131    ; $51 - Character: 'Q'          CHR$(81)
9132
9133            DEFB    %00111100
9134            DEFB    %01000010
9135            DEFB    %01000010
9136            DEFB    %01010010
9137            DEFB    %01001010
9138            DEFB    %00111100
9139
9140    ; $52 - Character: 'R'          CHR$(82)
9141
9142            DEFB    %01111100
9143            DEFB    %01000010
9144            DEFB    %01000010
9145            DEFB    %01111100
9146            DEFB    %01000100
9147            DEFB    %01000010
9148
9149    ; $53 - Character: 'S'          CHR$(83)
9150
9151            DEFB    %00111100
9152            DEFB    %01000000
9153            DEFB    %00111100
9154            DEFB    %00000010
9155            DEFB    %01000010
9156            DEFB    %00111100
9157
9158    ; $54 - Character: 'T'          CHR$(84)
9159
9160            DEFB    %11111110
9161            DEFB    %00010000
9162            DEFB    %00010000
9163            DEFB    %00010000
9164            DEFB    %00010000
9165            DEFB    %00010000
9166
9167    ; $55 - Character: 'U'          CHR$(85)
9168
9169            DEFB    %01000010
9170            DEFB    %01000010
9171            DEFB    %01000010
9172            DEFB    %01000010
9173            DEFB    %01000010
9174            DEFB    %00111110
9175
9176    ; $56 - Character: 'V'          CHR$(86)
9177
9178            DEFB    %01000010
9179            DEFB    %01000010
9180            DEFB    %01000010
9181            DEFB    %01000010
9182            DEFB    %00100100
9183            DEFB    %00011000
9184
9185    ; $57 - Character: 'W'          CHR$(87)
9186
9187            DEFB    %01000010
9188            DEFB    %01000010
9189            DEFB    %01000010
9190            DEFB    %01000010
9191            DEFB    %01011010
9192            DEFB    %00100100
9193
9194    ; $58 - Character: 'X'          CHR$(88)
9195
9196            DEFB    %01000010
9197            DEFB    %00100100
```

```
9198            DEFB    %00011000
9199            DEFB    %00011000
9200            DEFB    %00100100
9201            DEFB    %01000010
9202
9203    ; $59 - Character: 'Y'          CHR$(89)
9204
9205            DEFB    %10000010
9206            DEFB    %01000100
9207            DEFB    %00101000
9208            DEFB    %00010000
9209            DEFB    %00010000
9210            DEFB    %00010000
9211
9212    ; $5A - Character: 'Z'          CHR$(90)
9213
9214            DEFB    %01111110
9215            DEFB    %00000100
9216            DEFB    %00001000
9217            DEFB    %00010000
9218            DEFB    %00100000
9219            DEFB    %01111110
9220
9221    ; $5B - Character: '['          CHR$(91)
9222
9223            DEFB    %00001110
9224            DEFB    %00001000
9225            DEFB    %00001000
9226            DEFB    %00001000
9227            DEFB    %00001000
9228            DEFB    %00001110
9229
9230    ; $5C - Character: '\'          CHR$(92)
9231
9232            DEFB    %00000000
9233            DEFB    %01000000
9234            DEFB    %00100000
9235            DEFB    %00010000
9236            DEFB    %00001000
9237            DEFB    %00000100
9238
9239    ; $5D - Character: ']'          CHR$(93)
9240
9241            DEFB    %01110000
9242            DEFB    %00010000
9243            DEFB    %00010000
9244            DEFB    %00010000
9245            DEFB    %00010000
9246            DEFB    %01110000
9247
9248    ; $5E - Character: '^'          CHR$(94)
9249
9250            DEFB    %00010000
9251            DEFB    %00111000
9252            DEFB    %01010100
9253            DEFB    %00010000
9254            DEFB    %00010000
9255            DEFB    %00010000
9256
9257    ; $5F - Character: '_'          CHR$(95)
9258
9259            DEFB    %00000000
9260            DEFB    %00000000
9261            DEFB    %00000000
9262            DEFB    %00000000
9263            DEFB    %00000000
9264            DEFB    %00000000
9265            DEFB    %11111111
9266
9267    ; $60 - Character:  Â£           CHR$(96)
9268
9269            DEFB    %00011100
9270            DEFB    %00100010
```

```
9271            DEFB    %01111000
9272            DEFB    %00100000
9273            DEFB    %00100000
9274            DEFB    %01111110
9275            DEFB    %00000000
9276
9277    ; $61 - Character: 'a'          CHR$(97)
9278
9279            DEFB    %00000000
9280            DEFB    %00111000
9281            DEFB    %00000100
9282            DEFB    %00111100
9283            DEFB    %01000100
9284            DEFB    %00111110
9285            DEFB    %00000000
9286
9287    ; $62 - Character: 'b'          CHR$(98)
9288
9289            DEFB    %00100000
9290            DEFB    %00100000
9291            DEFB    %00111100
9292            DEFB    %00100010
9293            DEFB    %00100010
9294            DEFB    %00111100
9295            DEFB    %00000000
9296
9297    ; $63 - Character: 'c'          CHR$(99)
9298
9299            DEFB    %00000000
9300            DEFB    %00011100
9301            DEFB    %00100000
9302            DEFB    %00100000
9303            DEFB    %00100000
9304            DEFB    %00011100
9305            DEFB    %00000000
9306
9307    ; $64 - Character: 'd'          CHR$(100)
9308
9309            DEFB    %00000100
9310            DEFB    %00000100
9311            DEFB    %00111100
9312            DEFB    %01000100
9313            DEFB    %01000100
9314            DEFB    %00111110
9315            DEFB    %00000000
9316
9317    ; $65 - Character: 'e'          CHR$(101)
9318
9319            DEFB    %00000000
9320            DEFB    %00111000
9321            DEFB    %01000100
9322            DEFB    %01111000
9323            DEFB    %01000000
9324            DEFB    %00111100
9325            DEFB    %00000000
9326
9327    ; $66 - Character: 'f'          CHR$(102)
9328
9329            DEFB    %00001100
9330            DEFB    %00010000
9331            DEFB    %00011000
9332            DEFB    %00010000
9333            DEFB    %00010000
9334            DEFB    %00010000
9335            DEFB    %00000000
9336
9337    ; $67 - Character: 'g'          CHR$(103)
9338
9339            DEFB    %00000000
9340            DEFB    %00111100
9341            DEFB    %01000100
9342            DEFB    %01000100
9343            DEFB    %00111100
```

```
9344            DEFB    %00000100
9345            DEFB    %00111000
9346
9347    ; $68 - Character: 'h'          CHR$(104)
9348
9349            DEFB    %01000000
9350            DEFB    %01000000
9351            DEFB    %01111000
9352            DEFB    %01000100
9353            DEFB    %01000100
9354            DEFB    %01000100
9355            DEFB    %00000000
9356
9357    ; $69 - Character: 'i'          CHR$(105)
9358
9359            DEFB    %00010000
9360            DEFB    %00000000
9361            DEFB    %00110000
9362            DEFB    %00010000
9363            DEFB    %00010000
9364            DEFB    %00111000
9365            DEFB    %00000000
9366
9367    ; $6A - Character: 'j'          CHR$(106)
9368
9369            DEFB    %00000100
9370            DEFB    %00000000
9371            DEFB    %00000100
9372            DEFB    %00000100
9373            DEFB    %00000100
9374            DEFB    %00100100
9375            DEFB    %00011000
9376
9377    ; $6B - Character: 'k'          CHR$(107)
9378
9379            DEFB    %00100000
9380            DEFB    %00101000
9381            DEFB    %00110000
9382            DEFB    %00110000
9383            DEFB    %00101000
9384            DEFB    %00100100
9385            DEFB    %00000000
9386
9387    ; $6C - Character: 'l'          CHR$(108)
9388
9389            DEFB    %00010000
9390            DEFB    %00010000
9391            DEFB    %00010000
9392            DEFB    %00010000
9393            DEFB    %00010000
9394            DEFB    %00001100
9395            DEFB    %00000000
9396
9397    ; $6D - Character: 'm'          CHR$(109)
9398
9399            DEFB    %00000000
9400            DEFB    %01101000
9401            DEFB    %01010100
9402            DEFB    %01010100
9403            DEFB    %01010100
9404            DEFB    %01010100
9405            DEFB    %00000000
9406
9407    ; $6E - Character: 'n'          CHR$(110)
9408
9409            DEFB    %00000000
9410            DEFB    %01111000
9411            DEFB    %01000100
9412            DEFB    %01000100
9413            DEFB    %01000100
9414            DEFB    %01000100
9415            DEFB    %00000000
9416
```

```
9417   ; $6F - Character: 'o'          CHR$(111)
9418
9419           DEFB    %00000000
9420           DEFB    %00111000
9421           DEFB    %01000100
9422           DEFB    %01000100
9423           DEFB    %01000100
9424           DEFB    %00111000
9425           DEFB    %00000000
9426
9427   ; $70 - Character: 'p'          CHR$(112)
9428
9429           DEFB    %00000000
9430           DEFB    %01111000
9431           DEFB    %01000100
9432           DEFB    %01000100
9433           DEFB    %01111000
9434           DEFB    %01000000
9435           DEFB    %01000000
9436
9437   ; $71 - Character: 'q'          CHR$(113)
9438
9439           DEFB    %00000000
9440           DEFB    %00111100
9441           DEFB    %01000100
9442           DEFB    %01000100
9443           DEFB    %00111100
9444           DEFB    %00000100
9445           DEFB    %00000110
9446
9447   ; $72 - Character: 'r'          CHR$(114)
9448
9449           DEFB    %00000000
9450           DEFB    %00011100
9451           DEFB    %00100000
9452           DEFB    %00100000
9453           DEFB    %00100000
9454           DEFB    %00100000
9455           DEFB    %00000000
9456
9457   ; $73 - Character: 's'          CHR$(115)
9458
9459           DEFB    %00000000
9460           DEFB    %00111000
9461           DEFB    %01000000
9462           DEFB    %00111000
9463           DEFB    %00000100
9464           DEFB    %01111000
9465           DEFB    %00000000
9466
9467   ; $74 - Character: 't'          CHR$(116)
9468
9469           DEFB    %00010000
9470           DEFB    %00111000
9471           DEFB    %00010000
9472           DEFB    %00010000
9473           DEFB    %00010000
9474           DEFB    %00001100
9475           DEFB    %00000000
9476
9477   ; $75 - Character: 'u'          CHR$(117)
9478
9479           DEFB    %00000000
9480           DEFB    %01000100
9481           DEFB    %01000100
9482           DEFB    %01000100
9483           DEFB    %01000100
9484           DEFB    %00111100
9485           DEFB    %00000000
9486
9487   ; $76 - Character: 'v'          CHR$(118)
9488
9489           DEFB    %00000000
```

```
9490            DEFB    %01000100
9491            DEFB    %01000100
9492            DEFB    %00101000
9493            DEFB    %00101000
9494            DEFB    %00010000
9495            DEFB    %00000000
9496
9497    ; $77 - Character: 'w'          CHR$(119)
9498
9499            DEFB    %00000000
9500            DEFB    %01000100
9501            DEFB    %01010100
9502            DEFB    %01010100
9503            DEFB    %01010100
9504            DEFB    %00101000
9505            DEFB    %00000000
9506
9507    ; $78 - Character: 'x'          CHR$(120)
9508
9509            DEFB    %00000000
9510            DEFB    %01000100
9511            DEFB    %00101000
9512            DEFB    %00010000
9513            DEFB    %00101000
9514            DEFB    %01000100
9515            DEFB    %00000000
9516
9517    ; $79 - Character: 'y'          CHR$(121)
9518
9519            DEFB    %00000000
9520            DEFB    %01000100
9521            DEFB    %01000100
9522            DEFB    %01000100
9523            DEFB    %00111100
9524            DEFB    %00000100
9525            DEFB    %00111000
9526
9527    ; $7A - Character: 'z'          CHR$(122)
9528
9529            DEFB    %00000000
9530            DEFB    %01111100
9531            DEFB    %00001000
9532            DEFB    %00010000
9533            DEFB    %00100000
9534            DEFB    %01111100
9535            DEFB    %00000000
9536
9537    ; $7B - Character: '{'          CHR$(123)
9538
9539            DEFB    %00001110
9540            DEFB    %00001000
9541            DEFB    %00110000
9542            DEFB    %00110000
9543            DEFB    %00001000
9544            DEFB    %00001110
9545            DEFB    %00000000
9546
9547    ; $7C - Character: '|'          CHR$(124)
9548
9549            DEFB    %00001000
9550            DEFB    %00001000
9551            DEFB    %00001000
9552            DEFB    %00001000
9553            DEFB    %00001000
9554            DEFB    %00001000
9555            DEFB    %00000000
9556
9557    ; $7D - Character: '}'          CHR$(125)
9558
9559            DEFB    %01110000
9560            DEFB    %00010000
9561            DEFB    %00001100
9562            DEFB    %00001100
```

```
9563            DEFB    %00010000
9564            DEFB    %01110000
9565            DEFB    %00000000
9566
9567    ; $7E - Character: '~'          CHR$(126)
9568
9569            DEFB    %00110010
9570            DEFB    %01001100
9571            DEFB    %00000000
9572            DEFB    %00000000
9573            DEFB    %00000000
9574            DEFB    %00000000
9575            DEFB    %00000000
9576
9577    ; $7F - Character:  Â©           CHR$(127)
9578
9579            DEFB    %00111100
9580            DEFB    %01000010
9581            DEFB    %10011001
9582            DEFB    %10100001
9583            DEFB    %10100001
9584            DEFB    %10011001
9585            DEFB    %01000010
9586    L1FFB:  DEFB    %00111100
9587
9588
9589    ; ---------------
9590    ; THE 'SPARE' ROM
9591    ; ---------------
9592
9593    L1FFC:  DEFB    $FF                     ; unused
9594
9595    ; ----------
9596    ; THE 'LINK'
9597    ; ----------
9598
9599    ; The FORTH word copied to RAM links back to L1FFF
9600
9601    L1FFD:  DEFW    L_VER
9602    L1FFF:  DEFB    $00                     ; length of dummy word zero
9603
9604
9605    .END
9606
9607    ; -----------
9608    ;
9609    ; -----------
9610    ; ----------------------
9611    ; THE 'SYSTEM VARIABLES'
9612    ; ----------------------
9613    ; "Here is a list of system variables. We have given them all names, but that
9614    ; is just for ease of reference. The Ace will not recognize these names,
9615    ; except for a few, like 'BASE', that are FORTH words. I've written these
9616    ; FORTH words in bold type in the usual way."
9617    ;
9618    ;
9619    ; FP_WS         $3C00 (15360)   19 bytes used as work space for floating point
9620    ;                               arithmetic.
9621    ;
9622    ; LISTWS        $3C13 (15379)   5 bytes used as workspace by 'LIST' and 'EDIT'.
9623    ;
9624    ; RAMTOP        $3C18 (15384)   2 bytes - the first address past the last
9625    ;                               address in RAM.
9626    ;
9627    ; HLD           $3C1A (15386)   2 bytes. The address of the latest character
9628    ;                               held in the pad by formatted output.
9629    ;                               ('#', 'HOLD' and so on).
9630    ;
9631    ; SCRPOS        $3C1C (15388)   2 bytes. The address of the place in video RAM
9632    ;                               where the next character is to be printed
9633    ;                               (i.e. the 'print position').
9634    ;
9635    ; INSCRN        $3C1E (15390)   2 bytes. The address of the start of the
```

```
;                                current 'logical line' in the input buffer.
;
; CURSOR        $3C20 (15392)    2 bytes. The address of the cursor in the
;                                input buffer.
;
; ENDBUF        $3C22 (15394)    2 bytes. The address of the end of the current
;                                logical line in the input buffer.
;
; L_HALF        $3C24 (15396)    2 bytes. The address of the start of the the
;                                input buffer. The input buffer itself is stored
;                                in the video RAM, where you see it.
;
; KEYCOD        $3C26 (15398)    1 byte. The ASCII code of the last key pressed.
;
; KEYCNT        $3C27 (15399)    1 byte. Used by the routine that reads the
;                                keyboard.
;
; STATIN        $3C28 (15400)    1 byte. Used by the routine that reads the
;                                keyboard.
;
; EXWRCH        $3C29 (15401)    2 bytes. This is normally 0 but it can be
;                                changed to allow printing to be sent to some
;                                device other than the screen.
;
; FRAMES        $3C2B (15403)    4 bytes. These four bytes form a double length
;                                integer that counts the time since the Ace was
;                                switched on in 50ths of a second.
;
; XCOORD        $3C2F (15407)    1 byte. The x-coordinate last used by 'PLOT'.
;
; YCOORD        $3C30 (15408)    1 byte. The y-coordinate last used by 'PLOT'.
;
; CURRENT       $3C31 (15409)    2 bytes. The parameter field address for the
;                                vocabulary word of the current vocabulary.
;
; CONTEXT       $3C33 (15411)    2 bytes. The parameter field address for the
;                                vocabulary word of the context vocabulary.
;
; VOCLNK        $3C35 (15413)    2 bytes. The address of the fourth byte in the
;                                parameter field - the vocabulary linkage - of
;                                the vocabulary word of the most recently
;                                defined vocabulary.
;
; STKBOT        $3C37 (15415)    2 bytes. The address of the next byte into
;                                which anything will be enclosed in the
;                                dictionary, i.e. one byte past the present end
;                                of the dictionary.
;                                'HERE' is equivalent to 15415 @.
;
; DICT          $3C39 (15417)    2 bytes. The address of the length field in the
;                                newest word in the dictionary. If that length
;                                field is correctly filled in then DICT may
;                                be 0.
;
; SPARE         $3C3B (15419)    2 bytes. The address of the first byte past the
;                                top of the stack.
;
; ERR_NO        $3C3D (15421)    1 byte. This is usually 255, meaning "no error".
;                                If 'ABORT' is used, and ERR_NO is between 0 and
;                                127, then "ERROR" will be printed out, followed
;                                by the error number ERR_NO.
;
; FLAGS         $3C3E (15422)    1 byte. Shows the state of various parts of the
;                                system, each bit showing whether something
;                                particular is happening or not. Some of these
;                                may be useful.
;
;                                Bit 2, when 1, shows that there is an incomplete
;                                definition at the end of the dictionary.
;
;                                Bit 3, when 1, shows that output is to fed into
;                                the input buffer.
;
```

```
;                               Bit 4, when 1, shows that the Ace is in
;                               invisible mode.
;
;                               Bit 6, when 1, shows that the Ace is in compile
;                               mode.
;
; BASE          $3C3F (15423)   1 byte. The system number base.
;
;
;
; ------------------------------------------------------------------------
;                                    ---------
;                                  ---------------------------------------
; ------------                     ---------------------------------------
; ACE KEYBOARD                      ---------
; ------------                     ---------
;
;+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+
;|   ! | |   @ | |   # | |   $ | |   % | |   & | |   ' | |   ( | |   ) | |   _ |
;| 1 []| | 2 []| | 3 []| | 4 []| | 5 []| | 6 []| | 7 []| | 8   | | 9   | | 0 []|
;+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+
; DELETE   CAPS            INV    <=        ^       v        =>  GRAPHIC  DELETE
;  LINE    LOCK           VIDEO
;+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+
;|     | |     | |     | |   < | |   > | |   [ | |   ] | |   Â© | |   ; | |   " |
;| Q   | | W   | | E   | | R   | | T   | | Y   | | U   | | I   | | O   | | P   |
;+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+
;
;+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+
;|   ~ | |   | | |   \ | |   { | |   } | |   ^ | |   - | |   + | |   = | |     |
;| A   | | S   | | D   | | F   | | G   | | H   | | J   | | K   | | L   | |ENTER|
;+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+
;
;+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+
;|     | |   : | |   Â£ | |   ? | |   / | |   * | |   , | |   . | | SYM | |     |
;|SHIFT| | Z   | | X   | | C   | | V   | | B   | | N   | | M   | |SHIFT| |SPACE|
;+-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+
;
;
;                     [] mosaic graphic          Â£  currency symbol
;
; ------------------------------------------------------------------------
```