```
188                     *
189 D185 C1             LD11    POP   B
190 D186 C9                     RET
191                     *
192                     ***************************
193                     * RELINGUISH HEAP SPACE *
194                     ***************************
195                     *
196                     * Clears a string in the heap.
197                     *
198                     * Entry: HL points to 2nd length byte of string.
199                     * Exit:  HL points to 2nd length byte of cleared
200                     *        string. AFBCDE preserved.
201                     *
202 D187 2B             SHREL   DCX   H
203 D188 C336D2                 JMP   :D236     Clear Heap entry.
204                     *
205                     ****************************************
206                     * REQUEST SPACE IN HEAP FOR A STRING *
207                     ****************************************
208                     *
209                     * Finds a place in the heap for a new string.
210                     *
211                     * Entry: A: Required space.
212                     * Exit:  AFBCDE preserved.
213                     *        HL points to length of reserved area.
214                     *        Error if no space available.
215                     *
216 D18B D5             SHREQ   PUSH  D
217 D18C 1600                   MVI   D,:00     ) Required space in DE
218 D18E 5F                     MOV   E,A       )
219 D18F CDC5D1                 CALL  :D1C5     Run Heap request
220 D192 23                     INX   H
221 D193 D1                     POP   D
222 D194 C9                     RET
223                     *
224                     *
225                     *  ==============
226                     *** HEAP HANDLER ***
227                     *  ==============
228                     *
229                     *
230                     ***************************************
231                     * INIT. HEAP SPACE TO ALL AVAILABLE *
232                     ***************************************
233                     *
234                     * The pointers for textbuffer and symboltable
235                     * are updated correctly according to the requested
236                     * Heapsize. The Heap is cleared: It starts with
237                     * HSIZE-4 IOR #8000 and ends with #7FFF.
238                     *
239                     * Entry: DE: HEAP size (<32K).
240                     * Exit:  BC preserved. AFDEHL corrupted.
241                     *        Error if insufficient space.
242                     *
243 D195 2A9F02        HINIT   LHLD  :029F     Start text buffer in HL
244 D198 D5                   PUSH  D
245 D199 E5                   PUSH  H
246 D19A D5                   PUSH  D
247 D19B EB                   XCHG            Start textbuf in DE
248 D19C 2A9B02               LHLD  :029B     Start Heap in HL
        EB                    XCHG
```

```
250 D1A0 CD1ADE             CALL  :DE1A     Calc current heapsize
251 D1A3 EB                 XCHG            Store it in DE
252 D1A4 E1                 POP   H         Get new HEAP size
253 D1A5 CD1ADE             CALL  :DE1A     Calculate difference
254 D1A8 D1                 POP   D         Get old start textbuf
255 D1A9 CDD1C9             CALL  :C9D1     Move textbuf and symtab
256 D1AC 229F02             SHLD  :029F     Update start textbuf
257 D1AF D1                 POP   D         Get new HEAP size
258 D1B0 2A9B02             LHLD  :029B     Get startaddr HEAP
259 D1B3 1B                 DCX   D
260 D1B4 1B                 DCX   D
261 D1B5 1B                 DCX   D
262 D1B6 1B                 DCX   D         HSIZE-4
263 D1B7 7A                 MOV   A,D
264 D1B8 F680               ORI   :80       Free space available
265 D1BA 77                 MOV   M,A       ) Set start Heap to
266 D1BB 23                 INX   H         ) HSIZE-4 IOR #8000
267 D1BC 73                 MOV   M,E       )
268 D1BD 23                 INX   H
269 D1BE 19                 DAD   D         Get end HEAP -2
270 D1BF 367F               MVI   M,:7F     ) Set it to
271 D1C1 23                 INX   H         ) #7FFF
272 D1C2 36FF               MVI   M,:FF     ) (end marker)
273 D1C4 C9                 RET
274                 *
275                 ****************
276                 * HEAP REQUEST *
277                 ****************
278                 *
279 D1C5 CF         HREQU   RST   1         Run heap request
280 D1C6 0F                 DATA  :0F
281 D1C7 C9                 RET
282                 *
283 D1C8 FF                 DATA  :FF
284 D1C9 FF                 DATA  :FF
285 D1CA FF                 DATA  :FF
286 D1CB FF                 DATA  :FF
287 D1CC FF                 DATA  :FF
288 D1CD FF                 DATA  :FF
289 D1CE FF                 DATA  :FF
290 D1CF FF                 DATA  :FF
291 D1D0 FF                 DATA  :FF
292                 *
293                 ************************
294                 * part of EDIT (2EEC0) *
295                 ************************
296                 *
297                 * Prints a text line.
298                 *
299 D1D1 CAECEE     LD15    JZ    :EEEC     (2) If char=0
300 D1D4 F2DEEE             JP    :EEDE     (2) Cont for char #01-7F
301 D1D7 23                 INX   H         Next pos in textbuf
302 D1D8 C3D2EE             JMP   :EED2     (2) Ignore char >= #80
303                 *
304                 ******************************************
305                 * INPUT SCANNING: CHECK SOURCE AND INPUTS *
306                 ******************************************
307                 *
308                 * Part of D6BB.
309                 * Checks if input from keyboard or from DINC.
310                 * Checks also for any new inputs.
311                 *
```

```
312 D1DB 3A9602      MPT29   LDA   :0296     Get input direction
313 D1DE B7                  ORA   A
314 D1DF C2E002              JNZ   :02E0     Jump if input from DINC
315
316                  * If input from keyboard:
317
318 D1E2 CDA5D6      IN0     CALL  :D6A5     Check if new keys pressed
319 D1E5 C3C1D6              JMP   :D6C1     Into keyb.scanning
320                  *
321                  ********************************
322                  * part of FPT COMPARE (C079) *
323                  ********************************
324                  *
325                  * Entry: Contents MACC in ABCD, exp.byte in E.
326                  *         HL points to exp.byte MEM.
327                  *
328 D1E8 78           LD17    MOV   A,B
329 D1E9 23                   INX   H
330 D1EA A6                   ANA   M         AND 1st bytes both mantissas
331 D1EB 7E                   MOV   A,M
332 D1EC 2B                   DCX   H         Fnts to exp.byte MEM
333 D1ED FAF5D1               JM    :D1F5     Jump if both 1st mantissa
334                                            bits are '1'
335 D1F0 B8                   CMP   B         Comp both 1th mantissa bytes
336 D1F1 3F                   CMC             CY=0 if mantissa MACC >
337                                            mantissa MEM
338 D1F2 C39FC0               JMP   :C09F     Quit
339                  *
340 D1F5 7B           LD18    MOV   A,E       Get exp.byte MACC
341 D1F6 AE                   XRA   M         XOR both exponents
342 D1F7 E640                 ANI   :40       Check if only 1 exp. neg.
343 D1F9 7B                   MOV   A,E       Get exp.byte MACC
344 D1FA C387C0               JMP   :C087     Back to main routine
345                  *
346                  ************************
347                  * part of EDIT (2EE7B) *
348                  ************************
349                  *
350                  * Skip to B'th position on line.
351                  *
352 D1FD 7E           LD19    MOV   A,M       Get char in A
353 D1FE B7                   ORA   A         Set flags on it
354 D1FF FA94EE               JM    :EE94     (2) Ignore char >= #80
355 D202 78                   MOV   A,B       Get pos. required
356 D203 B9                   CMP   C         Reached ?
357 D204 7E                   MOV   A,M       Get char
358 D205 C382EE               JMP   :EE82     (2)
359                  *
360                  ********
361                  * DATA *
362                  ********
363                  *
364 D208 0D           MSG01   DATA  :0D       CR
365 D209 DB6F                 DBL   :6FDB     'SOME INPUT IGNORED'
366 D20B 00                   DATA  :00
367                  *
368                  ************************************
369                  * part of UNDERFLOW ERROR (C065) *
370                  ************************************
371                  *
372 D20C E7           LD21    RST   4         Copy operand to MACC
373 D20D 0C                   DATA  :0C
```

```
374 D20E 210400            LXI   H,:0004
375 D211 C34FC0            JMP   :C04F      Cont on (00D0/1)+4
376                  *
377                  ****************************************
378                  * part of RUN 'CLEAR' (cont. of 0E6B5) *
379                  ****************************************
380                  *
381 D214 229D02      MPT44 SHLD  :029D      Update HEAP size
382 D217 2A0001            LHLD  :0100      Get start current line
383 D21A 7C                MOV   A,H
384 D21B B5                ORA   L          Check if CURRNT=0
385 D21C D1                POP   D
386 D21D C8                RZ               Abort if immediate cmd
387 D21E CD01E4            CALL  :E401      (0) Run RESTORE
388 D221 09                DAD   B
389 D222 CD2DE9            CALL  :E92D      (0) Calc length of block
390                                         Result in BC
391 D225 B7                ORA   A
392 D226 C9                RET
393                  *
394                  ********************************
395                  * part of HEAP REQUEST (3E9AD) *
396                  ********************************
397                  *
398                  * Checks if end of Heap reached.
399                  *
400 D227 FE7F        HRQ80 CPI   :7F        End marker ?
401 D229 C2FAE9            JNZ   :E9FA      (3) Jump if not
402 D22C 7B                MOV   A,E        Get 2nd byte in A
403 D22D 3C                INR   A
404 D22E C2FAE9            JNZ   :E9FA      (3) Jump if it was <> FF
405 D231 3E07              MVI   A,:07      If end of Heap reached:
406 D233 C3F5D9            JMP   :D9F5      Run error 'OUT OF STRING
407                                         SPACE'
408                  *
409                  ***********************
410                  * CLEAR A HEAP ENTRY *
411                  ***********************
412                  *
413                  * A Heap entry is erased by setting the msb o
414                  * the first byte to 1.
415                  *
416                  * Entry: HL: Points to byte.
417                  * Exit:  All registers preserved.
418                  *
419 D236 F5          HREL  PUSH  PSW
420 D237 7E                MOV   A,M        Get byte
421 D238 F680              ORI   :80        Set msb=1
422 D23A 77                MOV   M,A        Store byte
423 D23B F1                POP   PSW
424 D23C C9                RET
425                  *
426                  *
427                  *
428 D23D                   END
```

```
****************************
* S Y M B O L   T A B L E *
****************************

HINIT  D195   HREL   D236   HREQU  D1C5   HRQ80  D227
INO    D1E2   LD10   D177   LD11   D185   LD15   D1D1
```

LD17	D1E8	LD18	D1F5	LD19	D1FD	LD21	D20C
LD3	D128	LD4	D13F	LD5	D145	LD9	D175
MPT29	D1DB	MPT44	D214	MSG01	D208	SCOPF	D16D
SCOPT	D173	SHAPP	D106	SHCOMP	D121	SHCOPY	D172
SHINIT	D101	SHM05	D151	SHM08	D15A	SHMID	D14F
SHREL	D187	SHREQ	D18B				