```
002                             ORG   :D101
003                     *
004                     *
005                     *
006                     *  ===============
007                     *** STRING HANDLER ***
008                     *  ===============
009                     *
010                     *
011                     ******************************
012                     * INITIALISE STRING HANDLER *
013                     ******************************
014                     *
015                     * Initialises a given size of string area.
016                     * This routine is used once by RESET (C719), but
017                     * without purpose. It belongs to another BASIC
018                     * package of the firm DAI.
019                     *
020                     * Entry: None.
021                     * Exit:  A=0, BCDEHL preserved. F corrupted.
022                     *
023 D101 AF             SHINIT  XRA   A
024 D102 320000                 STA   :0000     Zero 0000
025 D105 C9                     RET
026                     *
027                     ***********************
028                     * APPEND TWO STRINGS *
029                     ***********************
030                     *
031                     * Appends two strings together to one new string
032                     * by adding the 2nd string to the end of the 1st
033                     * string.
034                     *
035                     * Entry: DE: Startaddress 1st string.
036                     *            (1st byte is length byte).
037                     *        HL: Startaddress 2nd string.
038                     * Exit:  AFBCDE preserved.
039                     *        HL: Points to new string.
040                     *        Error if string too long.
041                     *
042 D106 F5             SHAPP   PUSH  PSW
043 D107 D5                     PUSH  D
044 D108 1A                     LDAX  D         Get length 1st string
045 D109 86                     ADD   M         Calc length new string
046 D10A DA38DA                 JC    :DA38     Run error 'STRING TOO LONG'
047                                             if length >255.
048 D10D E5                     PUSH  H         Save pntr 2nd string
049 D10E CD8BD1                 CALL  :D18B     Find place in Heap for
050                                             new string
051 D111 E5                     PUSH  H         Save pntr to new string
052 D112 23                     INX   H
053 D113 CD6DD1                 CALL  :D16D     Move 1st string to new loc.
054 D116 D1                     POP   D
055 D117 E3                     XTHL
056 D118 EB                     XCHG
057 D119 E3                     XTHL
058 D11A CD6DD1                 CALL  :D16D     Move 2nd string to new loc.
059 D11D E1                     POP   H         Get pntr to new string
060 D11E D1                     POP   D
061 D11F F1                     POP   PSW
062 D120 C9                     RET
063                     *
```

```
064                     ************************
065                     * COMPARE TWO STRINGS *
066                     ************************
067                     *
068                     * Compares two string character by character.
069                     * No characters >=#80 are allowed.
070                     *
071                     * Entry: DE: beginaddr. 1st string.
072                     *        HL: beginaddr. 2nd string.
073                     * Exit:  ABCDEHL preserved.
074                     *        Flags:     Z=1: Both strings ident.
075                     *                 S=1,Z=0: 2nd string longer.
076                     *                 S=0,Z=0: 1st string longer.
077                     *
078 D121 C5              SHCOMP  PUSH  B
079 D122 F5                      PUSH  PSW
080 D123 D5                      PUSH  D
081 D124 E5                      PUSH  H
082 D125 1A                      LDAX  D          Get length 1st string
083 D126 47                      MOV   B,A        Store it in B
084 D127 4E                      MOV   C,M        Length 2nd string in C
085 D128 13              LD3     INX   D
086 D129 23                      INX   H
087 D12A 78                      MOV   A,B
088 D12B D601                    SUI   :01        Check 1st string empty
089 D12D 47                      MOV   B,A        Save rest of bytes
090 D12E DA45D1                  JC    :D145      If 1st string empty
091 D131 79                      MOV   A,C
092 D132 D601                    SUI   :01        Check 2nd string empty
093 D134 4F                      MOV   C,A        Save rest of bytes
094 D135 3C                      INR   A
095 D136 3C                      INR   A
096 D137 DA3FD1                  JC    :D13F      If 2nd string empty
097 D13A 1A                      LDAX  D          Get byte 1st string
098 D13B BE                      CMP   M          and compare it with 2nd
099 D13C CA28D1                  JZ    :D128      If identical: cont. test
100 D13F E1              LD4     POP   H
101 D140 D1                      POP   D
102 D141 C1                      POP   B
103 D142 78                      MOV   A,B        Restore A
104 D143 C1                      POP   B
105 D144 C9                      RET
106
107                     * If 1st string empty:
108
109 D145 79              LD5     MOV   A,C        Get length 2nd string
110 D146 B7                      ORA   A
111 D147 CA3FD1                  JZ    :D13F      If also empty: abort, Z=1
112 D14A AF                      XRA   A
113 D14B 3D                      DCR   A          Z=0
114 D14C C33FD1                  JMP   :D13F      Quit
115                     *
116                     ******************************************
117                     * EXTRACT A SUBSTRING FROM THE MIDDLE OF *
118                     * ANOTHER STRING                         *
119                     ******************************************
120                     *
121                     * Entry: HL points to string.
122                     *        D  offset substring.
123                     *        E  length substring.
124                     * Exit:  If O.K.: HL points to substring.
125                     *        Else: Jump to error.
```

```
126                     *       AF corrupted. BCDE preserved.
127                     *
128 D14F C5             SHMID   PUSH  B
129 D150 D5                     PUSH  D
130 D151 7E             SHM05   MOV   A,M       Get length string
131 D152 92                     SUB   D         Minus offset substring
132 D153 DA15DA                 JC    :DA15     Run error 'NUMBER OUT OF
133                                               RANGE' if offset too big
134 D156 93                     SUB   E         Minus length substring
135 D157 DA15DA                 JC    :DA15     Run error 'NUMBER OUT OF
136                                               RANGE' if length too big
137 D15A 7A             SHM08   MOV   A,D       Get offset
138 D15B CD30DE                 CALL  :DE30     Calc beginaddr substring
139 D15E 23                     INX   H
140 D15F E5                     PUSH  H         Save begin substring
141 D160 7B                     MOV   A,E       Get length substring
142 D161 CD8BD1                 CALL  :D18B     Find place in Heap for
143                                               substring
144 D164 D1                     POP   D         Get begin substring
145 D165 E5                     PUSH  H
146 D166 CD73D1                 CALL  :D173     Move substring into Heap
147 D169 E1                     POP   H
148 D16A D1                     POP   D
149 D16B C1                     POP   B
150 D16C C9                     RET
151                     *
152                     ****************************
153                     * TRANSFER OF STRING DATA *
154                     ****************************
155                     *
156                     * Copies strings from one place to another.
157                     *
158                     * Start at SCOPF: Transfer known string from DE
159                     *                 to HL.
160                     * Start at SCOPT: Transfer string into limited
161                     *                 space of HL.
162                     *
163                     * Entry: DE: Startaddr. string to be transferred.
164                     *        HL: Destination address.
165                     *            1st bytes are length bytes.
166                     * Exit:  Both DE + HL point to byte after string.
167                     *        BC preserved; A=FF, CY=1.
168                     *
169 D16D 1A             SCOPF   LDAX  D         Get length
170 D16E 13                     INX   D         Pnt to 1st byte
171 D16F C375D1                 JMP   :D175
172                     *
173 D172 13             SHCOPY  INX   D         Pnt to 1st byte of string
174 D173 7E             SCOPT   MOV   A,M       Get available space
175 D174 23                     INX   H         Pnt to 1st place to store
176 D175 C5             LD9     PUSH  B
177 D176 47                     MOV   B,A       Available space/space reqd
178                                               in B
179 D177 78             LD10    MOV   A,B       Get space still available
180 D178 D601                   SUI   :01       Decr. space available
181 D17A 47                     MOV   B,A       and save it
182 D17B DA85D1                 JC    :D185     Jump if ready
183 D17E 1A                     LDAX  D         Get byte to be transferred
184 D17F 77                     MOV   M,A       and transfer it
185 D180 13                     INX   D         Pnt to next byte
186 D181 23                     INX   H         Pnt to next place
187 D182 C377D1                 JMP   :D177     Transfer next byte
```

```
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
```