TOOLKIT 1 appendix A

82—09

(c) DAInamic

TOOLKIT 1 appendix A

82-60

```
001                     ********************************************************
002                     *                                                      *
003                     *  AAAA     RRRR     RRRR     AAAA   Y      Y     >    *
004                     * A    A   R    R   R    R   A    A  Y      Y      >   *
005                     * A    A   R    R   R    R   A    A   Y    Y        >  *
006                     * AAAAAA   RRRRR    RRRRR    AAAAAA     Y Y    >>>>>>  *
007                     * A    A   R  R     R  R     A    A      Y           >  *
008                     * A    A   R   R    R   R    A    A      Y          >   *
009                     * A    A   R    R   R    R   A    A      Y         >    *
010                     *                                                      *
011                     * DDDDD     AAAA    TTTTTTT   AAAA                     *
012                     * D    D    A    A      T     A    A                    *
013                     * D    D    A    A      T     A    A                    *
014                     * D    D   AAAAAA      T     AAAAAA                    *
015                     * D    D   A    A      T     A    A                    *
016                     * D    D   A    A      T     A    A       Markus Sigg  *
017                     * DDDDD    A    A      T     A    A            8/82    *
018                     *                                                      *
019                     ********************************************************
020                     * This subroutine can be used to transfer data         *
021                     * from $-arrays to data lines. The call of the         *
022                     * routine looks as following:                          *
023                     * POKE #300,LINE MOD 256:POKE #301,LINE/256:           *
024                     * POKE #302,STEP MOD 256:POKE #303,STEP/256:           *
025                     * POKE #304,NUMBER:POKE #305,QMFLAG                    *
026                     * CALL #310,A$(FIRST)                                  *
027                     * LINE: line-number of first data-line                 *
028                     * STEP: step for line-numbering                        *
029                     * NUMBER: number of $-s' to transfer                   *
030                     * QMFLAG: if QMFLAG=34, the data is included           *
031                     *         by quotation-marks                           *
032                     * FIRST: index of first $                              *
033                     * The line from which the subroutine is called         *
034                     * must be smaller than the first data-line,            *
035                     * because the following lines are moved up.            *
036                     * Of course LINE+(NUMBER*STEP) should be less          *
037                     * than the next following line and less than           *
038                     * 65536.  The maximum-length of every $ of the         *
039                     * array is 121.  Supernumerary characters are          *
040                     * cutted of.       The subroutine should not be        *
041                     *      called by FOR var= loops, but by                *
042                     *            FOR var(x)= loops !                       *
043                     ********************************************************
044                     BSCBGN  EQU   :29F      start of basic-memory
045                     VARBGN  EQU   :2A1      start of variable-table
046                     VAREND  EQU   :2A3      end of variable-table
047                     MOVE    EQU   :DE4F     memory-move-routine
048                     MAXLEN  EQU   122       maximum-length+1
049                     *
050                             ORG   :300
051                     *
052 0300 E8FD           LINE    DBL   65000     first data-line
053 0302 0A00           STEP    DBL   10        line-step
```

```
054 0304 00          NUMBER  DATA  0          number of $-s to transfer
055 0305 00          QMFLAG  DATA  0          quotation-mark-flag,34=yes
056 0306 0000        VARPTR  DBL   0          adress of pointer to $
057 0308 0000        HLSAVE  DBL   0
058                  *
059                          ORG   :310       startadress
060                  ************************************************
061                  * The first part of the program calculates the *
062                  *        number of bytes to be inserted:       *
063                  ************************************************
064 0310 C5                  PUSH  B
065 0311 110000              LXI   D,0
066 0314 D5                  PUSH  D          number of bytes to insert
067 0315 220603              SHLD  VARPTR     save VARPTR-value
068 0318 3A0403              LDA   NUMBER
069 031B 4F                  MOV   C,A        number of $-s
070 031C 5E          GETLNG  MOV   E,M
071 031D 23                  INX   H
072 031E 56                  MOV   D,M        pointer to $ in DE
073 031F 23                  INX   H
074 0320 7A                  MOV   A,D
075 0321 B3                  ORA   E          null-$ ?
076 0322 CA4003              JZ    NULL$
077 0325 1A                  LDAX  D          length of $ in A
078 0326 FE7A                CPI   MAXLEN     $ too long ?
079 0328 DA2D03              JLS   OK1
080 032B 3E79                MVI   A,MAXLEN-1
081 032D 1600        OK1     MVI   D,0
082 032F 5F                  MOV   E,A
083 0330 E3                  XTHL
084 0331 19                  DAD   D          add length to insert-number
085 0332 110500              LXI   D,5        4 bytes for rest of line
086 0335 19                  DAD   D
087 0336 3A0503              LDA   QMFLAG
088 0339 B7                  ORA   A          quotation-marks ?
089 033A CA3F03              JZ    NOQT1
090 033D 23                  INX   H
091 033E 23                  INX   H
092 033F E3          NOQT1   XTHL
093 0340 0D          NULL$   DCR   C          last element ?
094 0341 C21C03              JNZ   GETLNG
095                  ************************************************
096                  * The number of bytes to insert is calculated now *
097                  * and saved on stack.  The following instructions *
098                  * search the location where the data-lines are to *
099                  * be set in, and move the following lines and the *
100                  *              variable-table up:                *
101                  ************************************************
102 0344 2A0003                LHLD  LINE
103 0347 EB                  XCHG             number of first data-line
104 0348 2A9F02              LHLD  BSCBGN     adress of first program-line
105 034B 0600                MVI   B,0
106 034D 7E          NXTLIN  MOV   A,M        length of whole line
107 034E B7                  ORA   A          end of program ?
108 034F CA6D03              JZ    GRT2
109 0352 4F                  MOV   C,A
110 0353 23                  INX   H
111 0354 7E                  MOV   A,M        left byte of line-number
112 0355 BA                  CMP   D          greater/equal than left byte
113 0356 D25D03              JGE   MSBGRT     .of first data line ?
114 0359 09                  DAD   B          add length
115 035A C34D03              JMP   NXTLIN     next line
116 035D C26C03      MSBGRT  JNZ   GRT1       greater than left byte
```

```
117 0360 23              INX   H
118 0361 0D              DCR   C
119 0362 7E              MOV   A,M       right byte of line-number
120 0363 BB              CMP   E         greater/eql. than right byte
121 0364 D26B03          JGE   GRT       .of first data line ?
122 0367 09              DAD   B         add length
123 0368 C34D03          JMP   NXTLIN    next line
124 036B 2B        GRT   DCX   H
125 036C 2B        GRT1  DCX   H
126 036D E5        GRT2  PUSH  H
127 036E D1              POP   D
128 036F C1              POP   B
129 0370 C5              PUSH  B
130 0371 09              DAD   B
131 0372 E5              PUSH  H
132 0373 C1              POP   B
133 0374 2AA302          LHLD  VAREND
134 0377 D5              PUSH  D
135 0378 CD4FDE          CALL  MOVE      move up rest of program
136                ************************************************
137                * Now there is enough place for the new data-lines.*
138                * They are generated by the last program-part:     *
139                ************************************************
140 037B 3A0403          LDA   NUMBER
141 037E 57              MOV   D,A       number of $-s
142 037F 2A0603          LHLD  VARPTR    adress of pointer to $
143 0382 4E        LINSET MOV   C,M
144 0383 23              INX   H
145 0384 46              MOV   B,M       adress of $ in BC
146 0385 23              INX   H
147 0386 78              MOV   A,B
148 0387 B1              ORA   C         null-$ ?
149 0388 CAF003          JZ    NULL$1
150 038B 0A              LDAX  B         length of $
151 038C 03              INX   B
152 038D FE7A            CPI   MAXLEN    $ too long ?
153 038F DA9403          JLS   OK2
154 0392 3E79            MVI   A,MAXLEN-1
155 0394 5F        OK2   MOV   E,A
156 0395 C604            ADI   4
157 0397 F5              PUSH  PSW
158 0398 3A0503          LDA   QMFLAG
159 039B B7              ORA   A
160 039C CAA303          JZ    NOQT2
161 039F F1              POP   PSW
162 03A0 3C              INR   A
163 03A1 3C              INR   A
164 03A2 F5              PUSH  PSW
165 03A3 F1        NOQT2 POP   PSW
166 03A4 CD0604          CALL  SET       set length of line
167 03A7 3A0103          LDA   LINE+1
168 03AA CD0604          CALL  SET
169 03AD 3A0003          LDA   LINE
170 03B0 CD0604          CALL  SET       set line-number
171 03B3 3EA2            MVI   A,:A2
172 03B5 CD0604          CALL  SET       set DATA-code
173 03B8 3A0503          LDA   QMFLAG
174 03BB B7              ORA   A
175 03BC CAC103          JZ    NOQT3
176 03BF 3E02            MVI   A,2
177 03C1 83        NOQT3 ADD   E
178 03C2 CD0604          CALL  SET       set length of data-part
179 03C5 3A0503          LDA   QMFLAG
```

```
180 03C8 B7                 ORA   A          quotation-marks ?
181 03C9 C40604             CNZ   SET
182 03CC 7B                 MOV   A,E
183 03CD B7                 ORA   A          no characters in DATA ?
184 03CE CADA03             JZ    ENDLIN
185 03D1 0A           DATSET LDAX  B
186 03D2 03                 INX   B
187 03D3 CD0604             CALL  SET        set data-character
188 03D6 1D                 DCR   E          last character of line ?
189 03D7 C2D103             JNZ   DATSET
190 03DA 3A0503       ENDLIN LDA   QMFLAG
191 03DD B7                 ORA   A          quotations-marks ?
192 03DE C40604             CNZ   SET
193 03E1 E5                 PUSH  H
194 03E2 D5                 PUSH  D
195 03E3 2A0203             LHLD  STEP
196 03E6 EB                 XCHG
197 03E7 2A0003             LHLD  LINE
198 03EA 19                 DAD   D          add step to line-number
199 03EB 220003             SHLD  LINE
200 03EE D1                 POP   D
201 03EF E1                 POP   H
202 03F0 15           NULL$1 DCR   D          last line ?
203 03F1 C28203             JNZ   LINSET
204 03F4 D1                 POP   D
205 03F5 C1                 POP   B
206 03F6 2AA102             LHLD  VARBGN
207 03F9 09                 DAD   B
208 03FA 22A102             SHLD  VARBGN
209 03FD 2AA302             LHLD  VAREND
210 0400 09                 DAD   B
211 0401 22A302             SHLD  VAREND     update pointers
212 0404 C1                 POP   B
213 0405 C9                 RET
214                   ********************************************************
215                   * Subroutine to set one byte of data-line; current     *
216                   * adress saved on stack, code in A.                    *
217                   ********************************************************
218 0406 220803       SET   SHLD  HLSAVE
219 0409 E1                 POP   H
220 040A E3                 XTHL
221 040B 77                 MOV   M,A
222 040C 23                 INX   H
223 040D E3                 XTHL
224 040E E5                 PUSH  H
225 040F 2A0803             LHLD  HLSAVE
226 0412 C9                 RET
227                   *
228 0413                    END
```

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

BSCBGN 029F   DATSET 03D1   ENDLIN 03DA   GETLNG 031C
GRT    036B   GRT1   036C   GRT2   036D   HLSAVE 0308
LINE   0300   LINSET 0382   MAXLEN 007A   MOVE   DE4F
MSBGRT 035D   NOQT1  033F   NOQT2  03A3   NOQT3  03C1
NULL$  0340   NULL$1 03F0   NUMBER 0304   NXTLIN 034D
OK1    032D   OK2    0394   QMFLAG 0305   SET    0406
STEP   0302   VARBGN 02A1   VAREND 02A3   VARPTR 0306
```

```
1     REM +++ MARKUS SIGG  8/82 +++
10    CLEAR 1000:MODE 0:PRINT CHR$(12);
20    PRINT "This programm demonstrates with a little example"
30    PRINT "the facilities of the subroutine ARRAYDATA.":PRINT
40    PRINT "ARRAYDATA is a utility-program and can be used if"
50    PRINT "the content of a string-array shall be transferred"
60    PRINT "into data-lines.":GOSUB 1100
70    PRINT "The ML-program can be called by the following"
80    PRINT "instructions:":PRINT
90    LIST 5000-5040:PRINT
100   PRINT "LINE:    line-number of first data-line"
110   PRINT "STP:     line-step"
120   PRINT "NUMBER: number of strings to transfer,0=256"
130   PRINT "FIRST:  index of first string to transfer"
140   PRINT "QMFLAG: set QMFLAG=34 (";CHR$(34);") when you want to transfer"
150   PRINT "        strings which contain ',' and QMFLAG=0 when you"
160   PRINT "        want to transfer numbers"
170   PRINT "ARRAY$: array containing the strings"
180   GOSUB 1100
190   PRINT "ATTENTION: The subroutine may not be called within normal"
200   PRINT "           loops, because the pointers to the loop-variables"
210   PRINT "           can't be updated !! Loops with subscripted"
220   PRINT "           variables must be used."
230   GOSUB 1100
240   PRINT "Here our example:"
250   PRINT
260   PRINT "We want to type in a list of 5 persons, using the"
270   PRINT "following characteristics:"
280   PRINT
290   PRINT "last name,first name,land,town,street,house-number"
310   PRINT
320   PRINT "Last name and first name shall be stored together in one"
330   PRINT "data-element in the first DATA-line."
340   PRINT "The second line shall contain land,town,street and house-"
350   PRINT "number, stored as seperated elements."
370   PRINT
380   PRINT "The DATA-lines will look as following:"
390   PRINT
400   PRINT "10000 DATA ";CHR$(34);"LAST NAME,FIRST NAME";CHR$(34)
410   PRINT "10010 DATA LAND,TOWN,STREET,HOUSENUMBER"
430   PRINT "11000 ..."
440   PRINT "11010 ..."
450   GOSUB 1100
460   PRINT "Here are the instructions to generate this list:"
470   PRINT
480   LIST 510-670
490   PRINT
500   PRINT "PRESS 'SPACE' TO RUN THESE INSTRUCTIONS !":CALLM #D6DA:PRINT CHR$(12);
510   DIM ARRAY$(1),I%(0)
520   LINE%=10000:NUMBER%=1
530   FOR I%(0)=1 TO 5:PRINT "Person no.";I%(0);":"
540   INPUT "First name,last name ";FIRSTNAME$,LASTNAME$:PRINT
550   ARRAY$(0)=LASTNAME$+","+FIRSTNAME$
```

```
560   INPUT "Land ";LAND$:PRINT
570   INPUT "Town ";TOWN$:PRINT
580   INPUT "Street ";STREET$:PRINT
590   INPUT "Housenumber ";HOUSENUMBER$:PRINT
600   ARRAY$(1)=LAND$+","+TOWN$+","+STREET$+","+HOUSENUMBER$
650   FIRST%=0:QMFLAG%=34:GOSUB 5000:LINE%=LINE%+10
660   FIRST%=1:QMFLAG%=0:GOSUB 5000
670   PRINT :LINE%=LINE%+990:NEXT
680   PRINT CHR$(12);"Here is the result:":LIST 10000-
690   GOSUB 1100
700   PRINT "We can execute a CLEAR:":TRON
710   CLEAR 1000
720   TROFF
730   PRINT "and our data is not lost:"
740   LIST 10000-
750   GOSUB 1100
760   PRINT "Now we want to lay out a second list which presents the"
770   PRINT "data sorted in names,lands,towns,..."
780   PRINT
790   PRINT "This list shall look as followiing:"
800   PRINT
810   PRINT "20000 DATA NAME 1,NAME 2,...,NAME 5"
820   PRINT "20010 DATA LAND 1,LAND 2,...,LAND 5"
830   PRINT "20020  .     .     .          ."
870   GOSUB 1100
880   PRINT "Following program solves our problem:"
890   PRINT
900   LIST 930-1010
910   PRINT
920   PRINT "PRESS 'SPACE' TO START THIS PROGRAM-PART !"
930   DIM ARRAY$(4)
940   LINE%=20000:STP%=10:NUMBER%=5:FIRST%=0:QMFLAG%=0
950   RESTORE
960   FOR I%=1 TO 5
970   FOR J%=0 TO 4
980   READ ELEMENT$:IF J%=0 THEN ELEMENT$=CHR$(34)+ELEMENT$+CHR$(34)
990   ARRAY$(J%)=ARRAY$(J%)+ELEMENT$:IF I%<>5 THEN ARRAY$(J%)=ARRAY$(J%)+","
1000  NEXT:NEXT
1010  GOSUB 5000
1020  GOSUB 1100
1030  PRINT "Here the new list:"
1040  PRINT
1050  LIST 20000-
1060  GOSUB 1100
1070  PRINT "And here the whole list:"
1080  PRINT
1090  LIST 10000-:END
1100  PRINT :PRINT "PRESS 'SPACE' TO CONTINUE !":CALLM #D6DA:PRINT CHR$(12);:RETURN
5000  REM Subroutine to call ARRAYDATA
5010  POKE #300,LINE% MOD 256:POKE #301,LINE%/256
5020  POKE #302,STP% MOD 256:POKE #303,STP%/256
5030  POKE #304,NUMBER%:POKE #305,QMFLAG%
5040  CALLM #310,ARRAY$(FIRST%):RETURN
```

```
001                 ************************************************
002                 *                                              *
003                 * EEEEEE  DDDDD      I   TTTTTTT     >         *
004                 * E       D    D     I      T         >        *
005                 * E       D    D     I      T          >       *
006                 * EEEE    D    D     I      T      >>>>>>      *
007                 * E       D    D     I      T         >        *
008                 * E       D    D     I      T        >         *
009                 * EEEEEE  DDDDD      I      T       >          *
010                 *                                              *
011                 * DDDDD    AAAA   TTTTTTT  AAAA                *
012                 * D    D  A    A     T    A    A               *
013                 * D    D  A    A     T    A    A               *
014                 * D    D  AAAAAA     T    AAAAAA               *
015                 * D    D  A    A     T    A    A               *
016                 * D    D  A    A     T    A    A     Markus Sigg *
017                 * DDDDD   A    A     T    A    A           8/82 *
018                 *                                              *
019                 ************************************************
020                 * This subroutine can be used to transfer data *
021                 * from the EDIT-buffer to data lines. The call *
022                 * of the routine looks as following:           *
023                 * POKE #300,LINE MOD 256:POKE #301,LINE/256    *
024                 * POKE #302,STEP MOD 256:POKE #303,STEP/256    *
025                 * POKE #304,QMFLAG:CALLM #310                  *
026                 * The whole EDIT-buffer is transferred to data *
027                 * lines, starting with the line-number given   *
028                 * by LINE and the line-step specified by STEP. *
029                 * QMFLAG=0:quotation-marks are not transferred *
030                 * QMFLAG<>0: the specified character is        *
031                 *            transferred if a quotation-mark   *
032                 *            occures. (example:39 for ')       *
033                 * The end of a line to be read out is marked   *
034                 * by the carriage-return character.            *
035                 * No line may contain more than 121 characters,*
036                 * otherwise errors will occure.                *
037                 * The line from which the subroutine is called *
038                 * must be smaller than the first data line.    *
039                 * The subroutine should not be called by       *
040                 * FOR var= loops, but by FOR var(x)= loops !   *
041                 ************************************************
042                 BUFBGN  EQU   :A2       start of edit-buffer
043                 BSCBGN  EQU   :29F      start of basic-memory
044                 VARBGN  EQU   :2A1      start of variable-table
045                 VAREND  EQU   :2A3      end of variable-table
046                 MOVE    EQU   :DE4F     memory-move-routine
047                 *
048                         ORG   :300
049                 *
050 0300 E8FD       LINE    DBL   65000     first data-line
051 0302 0A00       STEP    DBL   10        line-step
052 0304 00         QMFLAG  DATA  0         quotation-mark-flag
053 0305 0000       HLSAVE  DBL   0
```

```
054 0307 0000       ADRSVE  DBL   0
055                 *
056                         ORG   :310      startadress
057                 ***********************************************
058                 * The first part of the program calculates the *
059                 *       number of bytes to be inserted         *
060                 ***********************************************
061 0310 C5                 PUSH  B
062 0311 2AA200             LHLD  BUFBGN
063 0314 E5                 PUSH  H
064 0315 C1                 POP   B         start of buffer in BC
065 0316 210000             LXI   H,0       number of bytes to insert
066 0319 110600             LXI   D,6
067 031C 0A         GETCD1  LDAX  B
068 031D 03                 INX   B
069 031E B7                 ORA   A         end of buffer ?
070 031F C22F03             JNZ   NOEND
071 0322 0B                 DCX   B
072 0323 0B                 DCX   B
073 0324 0A                 LDAX  B
074 0325 FE0D               CPI   :D
075 0327 CA4503             JZ    BFEND1
076 032A 23                 INX   H
077 032B 19                 DAD   D
078 032C C34503             JMP   BFEND1
079 032F FE22       NOEND   CPI   34        quotation-mark ?
080 0331 C23B03             JNZ   NOQT1
081 0334 3A0403             LDA   QMFLAG
082 0337 B7                 ORA   A
083 0338 CA1C03             JZ    GETCD1
084 033B 23         NOQT1   INX   H
085 033C FE0D               CPI   :D        end of line ?
086 033E C21C03             JNZ   GETCD1
087 0341 19                 DAD   D
088 0342 C31C03             JMP   GETCD1
089 0345 E5         BFEND1  PUSH  H
090                 ***********************************************
091                 * The number of bytes to insert is calculated now *
092                 * and saved on stack.  The following instructions *
093                 * search the location where the data-lines are to *
094                 * be set in, and move the following lines and the *
095                 *              variable-table up:                 *
096                 ***********************************************
097 0346 2A0003                LHLD  LINE
098 0349 EB                    XCHG          number of first data-line
099 034A 2A9F02                LHLD  BSCBGN  adress of first program-line
100 034D 0600                  MVI   B,0
101 034F 7E         NXTLIN     MOV   A,M     length of whole line
102 0350 B7                    ORA   A       end of program ?
103 0351 CA6F03                JZ    GRT2
104 0354 4F                    MOV   C,A
105 0355 23                    INX   H
106 0356 7E                    MOV   A,M     left byte of line-number
107 0357 BA                    CMP   D       greater/equal than left byte
108 0358 D25F03                JGE   MSBGRT  .of first data line ?
109 035B 09                    DAD   B       add length
110 035C C34F03                JMP   NXTLIN  next line
111 035F C26E03     MSBGRT     JNZ   GRT1    greater than left byte
112 0362 23                    INX   H
113 0363 0D                    DCR   C
114 0364 7E                    MOV   A,M     right byte of line-number
115 0365 BB                    CMP   E       greater/eql. than right byte
116 0366 D26D03                JGE   GRT     .of first data line ?
```

```
117 0369 09              DAD   B          add length
118 036A C34F03          JMP   NXTLIN     next line
119 036D 2B        GRT   DCX   H
120 036E 2B        GRT1  DCX   H
121 036F E5        GRT2  PUSH  H
122 0370 D1              POP   D
123 0371 C1              POP   B
124 0372 C5              PUSH  B
125 0373 09              DAD   B
126 0374 E5              PUSH  H
127 0375 C1              POP   B
128 0376 2AA302          LHLD  VAREND
129 0379 D5              PUSH  D
130 037A CD4FDE          CALL  MOVE       move up rest of program
131                ********************************************************
132                * Now there is enough place for the new data-lines.*
133                * They are generated by the last program-part :    *
134                ********************************************************
135 037D 2AA200          LHLD  BUFBGN
136 0380 7E        SETLIN MOV   A,M
137 0381 B7              ORA   A          end of buffer ?
138 0382 CAF203          JZ    BFEND2
139 0385 E3              XTHL
140 0386 220703          SHLD  ADRSVE     adress of start of line
141 0389 E3              XTHL
142 038A 3E2A            MVI   A,'*'
143 038C CD0404          CALL  SET        skip length-byte of line
144 038F 3A0103          LDA   LINE+1
145 0392 CD0404          CALL  SET
146 0395 3A0003          LDA   LINE
147 0398 CD0404          CALL  SET        set line-number
148 039B 3EA2            MVI   A,:A2
149 039D CD0404          CALL  SET        set DATA-code
150 03A0 3E2A            MVI   A,'*'
151 03A2 CD0404          CALL  SET        skip length of data-part
152 03A5 3E22            MVI   A,'"'
153 03A7 CD0404          CALL  SET        set start-quotation-mark
154 03AA 0600            MVI   B,0        length of data-part
155 03AC 7E        GET   MOV   A,M
156 03AD 23              INX   H
157 03AE FE0D            CPI   :D         end of line ?
158 03B0 CACA03          JZ    ENDLIN
159 03B3 B7              ORA   A
160 03B4 CACA03          JZ    ENDLIN
161 03B7 FE22            CPI   34         quotation-mark ?
162 03B9 C2C303          JNZ   NOQT2
163 03BC 3A0403          LDA   QMFLAG
164 03BF B7              ORA   A
165 03C0 CAAC03          JZ    GET
166 03C3 CD0404    NOQT2 CALL  SET        set character
167 03C6 04              INR   B          increment data-length
168 03C7 C3AC03          JMP   GET
169 03CA F5        ENDLIN PUSH  PSW
170 03CB D1              POP   D
171 03CC 3E22            MVI   A,'"'
172 03CE CD0404          CALL  SET        set end-quotation-mark
173 03D1 D5              PUSH  D
174 03D2 E5              PUSH  H
175 03D3 2A0703          LHLD  ADRSVE
176 03D6 3E06            MVI   A,6
177 03D8 80              ADD   B
178 03D9 77              MOV   M,A
179 03DA 23              INX   H
```

```
180 03DB 23                 INX   H
181 03DC 23                 INX   H
182 03DD 23                 INX   H
183 03DE 04                 INR   B
184 03DF 04                 INR   B
185 03E0 70                 MOV   M,B
186 03E1 2A0203             LHLD  STEP
187 03E4 EB                 XCHG
188 03E5 2A0003             LHLD  LINE
189 03E8 19                 DAD   D
190 03E9 220003             SHLD  LINE
191 03EC E1                 POP   H
192 03ED F1                 POP   PSW
193 03EE B7                 ORA   A
194 03EF C28003             JNZ   SETLIN
195 03F2 D1         BFEND2  POP   D
196 03F3 C1                 POP   B
197 03F4 2AA102             LHLD  VARBGN
198 03F7 09                 DAD   B
199 03F8 22A102             SHLD  VARBGN
200 03FB 2AA302             LHLD  VAREND
201 03FE 09                 DAD   B
202 03FF 22A302             SHLD  VAREND
203 0402 C1                 POP   B
204 0403 C9                 RET
205                 ********************************************************
206                 * Subroutine to set one byte of data-line; current     *
207                 * adress saved on stack, code in A.                    *
208                 ********************************************************
209 0404 220503     SET     SHLD  HLSAVE
210 0407 E1                 POP   H
211 0408 E3                 XTHL
212 0409 77                 MOV   M,A
213 040A 23                 INX   H
214 040B E3                 XTHL
215 040C E5                 PUSH  H
216 040D 2A0503             LHLD  HLSAVE
217 0410 C9                 RET
218                 *
219 0411                    END
```

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

```
ADRSVE 0307   BFEND1 0345   BFEND2 03F2   BSCBGN 029F
BUFBGN 00A2   ENDLIN 03CA   GET    03AC   GETCD1 031C
GRT    036D   GRT1   036E   GRT2   036F   HLSAVE 0305
LINE   0300   MOVE   DE4F   MSBGRT 035F   NOEND  032F
NOQT1  033B   NOQT2  03C3   NXTLIN 034F   QMFLAG 0304
SET    0404   SETLIN 0380   STEP   0302   VARBGN 02A1
VAREND 02A3
```

```
1     REM +++ MARKUS SIGG  8/82 +++
10    MODE 0:PRINT CHR$(12);
20    PRINT "This program demonstrates the use of the subroutine"
30    PRINT "EDITDATA. EDITDATA can be used, when you want to transfer"
40    PRINT "the content of the EDIT-buffer into DATA-lines."
50    PRINT
60    PRINT "EDITDATA is called by the following instructions:"
70    LIST 5000-5030
80    PRINT "LINE:   line-number of first DATA-line"
90    PRINT "STP:    line-step"
100   PRINT "QMFLAG: 0=do not transfer quotation-marks"
110   PRINT "      <>0=use character(QMFLAG) instead of quot.mark"
120   PRINT
130   CLEAR 10000
140   PRINT "Now type EDIT,BREAK-BREAK,CONT !"
150   STOP
200   PRINT "Now the EDIT-buffer will be read out:"
210   TRON
220   LINE%=10000:STP%=5:QMFLAG%=39:GOSUB 5000
230   TROFF
240   PRINT
250   PRINT "PRESS 'SPACE' TO CONTINUE !"
260   CALLM #D6DA:PRINT CHR$(12);
270   LIST 10000-:END
5000  REM Subroutine to call EDITDATA
5010  POKE #300,LINE% MOD 256:POKE #301,LINE%/256
5020  POKE #302,STP% MOD 256:POKE #303,STP%/256
5030  POKE #304,QMFLAG%:CALLM #310:RETURN
```

TOOLKIT 1 appendix A

82—09

the programs on APPENDIX A-TOOLKIT 1

audio :

```
*     Obj ARRAY-DATA (H300-H41F)
*     ARRAY-DATA DEMO 1
*     ARRAY-DATA DEMO 2
*     ARRAY-DATA DEMO 3
*     Obj EDIT-DATA  (H300-H41F)
*     EDIT-DATA DEMO 1
*     EDIT-DATA DEMO 2
*     SOURCE ARRAY-DATA
*     SOURCE EDIT-DATA
```

dcr :

```
1-ASS - ARRAYDATA
2-OBJ - ARRAYDATA 29B-41F
3-BAS - ARRAY-DATA DEMO 1
4-BAS - ARRAY-DATA DEMO 2
5-BAS - ARRAY-DATA DEMO 3
6-ASS - EDITDATA
7-OBJ - EDITDATA 29B-410
8-BAS - EDIT-DATA DEMO 1
9-BAS - EDIT-DATA DEMO 2
```