

MSX BIOS

The Complete
MSX BASIC
I/O Listing



QEST PUBLISHING INC.

Edited: January 1985
by Steven M. Ting
Graphic design: Mervin Fong.

The information in this document is subject to change without notice. ASCII Corp. makes no warranty with regard to this manual, including but not limited to, implied warranties of merchantability and fitness for a particular purpose.

The parties above assume no responsibility for any errors which may appear in this document.

This document is not intended as "Consumer goods" under applicable federal or state law(s).

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of ASCII Corporation and Qest Publishing Inc.

MSX is a registered trademark of Microsoft Corporation, Bellevue, WA.

Z80 is a registered trademark of Zilog, Inc.

Printed in United States

MSX BIOS

Copyrighted © 1985 by ASCII Corporation of Japan

All rights Reserved

Published by

QUEST PUBLISHING INC.
39 W. 32nd Street Suite 800
New York, N. Y. 10001

(212) 564-0749
Telex: 650-190-8083 MCI

TABLE OF CONTENTS

BIOS LISTING	1 - 256
MSX BIOS CROSS REFERENCE.....	257 - 280
SYMBOL TABLE.....	281 - 285
APPENDIX A	
MSX USA & UK OVERLAY PATCHES.....	287 - 316
BIOS CALLS.....	317 - 324
APPENDIX B	
CHARACTER SET & KEYBOARD LAYOUT.....	325 - 338
HOOKS & RAM ROUTINES.....	339 - 356


```
1                    .list
2                    ;
3                    ;
4                    ;        (C) Copyright by ASCII Corp., 1983
5                    ;        Proprietary information. All rights reserved.
6                    ;
7                    ;        File:    BIOHDR.MAC
8                    ;        USE:    Restart calls and ROM entries table
9                    ;        Written by Jey Suzuki, Rick Yamashita
10                    ;        ASCII Corporation, Japan
11                    ;
12                    ;        Edit:    January, 1985
13                    ;        Reason: Zilog Z80 Mnemonic version and cleanup
14                    ;        Edited by:        Steven M. Ting
15                    ;
16                    ;
17                    ; Labels referenced in this listing, are the absolute locations
18                    ; within the MSX ROM. However, "ONLY" this BIOS entry point table,
19                    ; and RAM variables are guaranteed to be permanent.
20                    ;
21                    ; All other locations in the ROM, will be changed without notice.
22                    ;
23                    SUBTTL -BIOS header- BIOS calls (Basic Interpreter, Slot I/O)
```

```

24
25      ;
26      ; The following RST's (RST 0 thru RST 5) are reserved for BASIC
27      ; interpreter, RST 6 for inter-slot calls, and RST 7 for
28      ; hardware interrupt
29      ;
30      0000    F3      BEGIN:  DI          ;Fail safe
31      0001    C3 02D7      JP          CHKRAM      ;Finds all connected RAM
32                                     ;and cartridges
33      ;
34      ;
35      ; ** Special information for the VDP. **
36      ; Any program that accesses the VDP hardware directly
37      ; should read the I/O port address found here, to be certain
38      ; the software is compatible with future versions of the VDP.
39      ;
40      0004    1BBF      DW          CGTABL      ;Address of character generator table,
41                                     ;to allow use of other character ROM.
42      ;
43      0006    98      DB          98H      ;Current port address for VDP Data read
44      0007    98      DB          98H      ;  "  "  "  "  "  "  "  write
45      ;
46      0008    C3 2683      JP          SYNCHR      ;Check byte following the RST 8, see
47                                     ;if equal to the byte pointed by HL
48      000B    00      DB          0
49      000C    C3 01B6      JP          RDSLT      ;Read a byte from another slot
50      000F    00      DB          0
51      0010    C3 2686      JP          CHRGTTR      ;Fetch next char from BASIC text
52      0013    00      DB          0
53      0014    C3 01D1      JP          WRSLT      ;Write a byte to another slot
54      0017    00      DB          0

```

55	0018	C3 1B45	JP	OUTDO	;Output a char to the Console or printer
56	001B	00	DB	0	
57	001C	C3 0217	JP	CALSLT	;Perform Inter-slot call
58	001F	00	DB	0	
59	0020	C3 146A	JP	DCOMPR	;Compares [HL] to [DE]
60	0023	00	DB	0	
61	0024	C3 025E	JP	ENASLT	;Permanently enables a slot
62	0027	00	DB	0	
63	0028	C3 2689	JP	GETYPR	;Returns the [FAC] type
64	002B	00	DB	0	;ID Byte (1)
65					;Format:
66					; B7 B6 B5 B4 B3 B2 B1 B0
67					; + + + + + + + +
68					; + + + + - - - - - - Type of character
69					; + + + + generator.
70					; + + + + 0:Japanese
71					; + + + + 1:International
72					; + + + + 2:Korea
73					; + + - - - - - - - - Date format
74					; + + 0: Y-M-D 1: M-D-Y
75					; + + 2: D-M-Y
76					; - - - - - - - - - - Interrupt frequency
77					; 0: 60 Hz 1: 50 Hz
78	002C	00	DB	0	;ID Byte (2)
79					;Format:
80					; B7 B6 B5 B4 B3 B2 B1 B0
81					; + + + + + + + +
82					; + + + + - - - - - - Type of Keyboard
83					; + + + + 0:Japanese 2:French
84					; + + + + 1:Int 3:UK
85					; + + + + 4:DIN

```
86                                     ; - - - - - Version of BASIC
87                                     ;                                0: Japanese
88                                     ;                                1: International
89      002D      00 00 00              DB      0,0,0
90      0030      C3 0205              JP      CALLF      ;Performs Far-call (i.e., Inter-slot)
91      0033      00 00 00 00          DB      0,0,0,0,0
92      0037      00
93                                     ;
94                                     ;
95      ; Following are used for I/O initialization
96      ;
97      0038      C3 0C3C              JP      KEYINT      ;Handlers for hardware interrupt
98      003B      C3 049D              JP      INITIO      ;Do device initialization
99      003E      C3 139D              JP      INIFNK      ;Reset all function key's text
100                                     ;
101      SUBTTL -BIOS header- BIOS calls (Video display processor)
```

```

102
103      ;
104      ; The following entry points provides control of the
105      ; VDP's registers, screen mode settings, and memory block
106      ; move between DRAM and VRAM.
107      ;
108      0041      C3 0577      JP      DISSCR      ;Disables screen display
109      0044      C3 0570      JP      ENASCR      ;Enables screen display
110      0047      C3 057F      JP      WRTVDP      ;Write a byte to any VDP register
111      004A      C3 07D7      JP      RDVRM      ;Read VRAM addressed using [HL]
112      004D      C3 07CD      JP      WRTVRM      ;Write VRAM addressed using [HL]
113      0050      C3 07EC      JP      SETRD      ;Sets up VDP for read
114      0053      C3 07DF      JP      SETWRT      ;Sets up VDP for write
115      0056      C3 0815      JP      FILVRM      ;Fills VRAM with specified data
116      0059      C3 070F      JP      LDIRMV      ;Moves block of data from VRAM to memory
117      005C      C3 0744      JP      LDIRVM      ; " " " " " memory to VRAM
118      005F      C3 084F      JP      CHGMOD      ;Change screen mode of VDP to [SCRMOD]
119      0062      C3 07F7      JP      CHGCLR      ;change Foreground, background,
120                                     ;border, color
121      0065      00          DB      0
122      ;
123      ;
124      0066      C3 1398      JP      NMI          ;Handler for non-maskable interrupt
125      ;
126      0069      C3 06A8      JP      CLRSPR      ;Init sprite data
127      006C      C3 050E      JP      INITXT      ;Init VDP for 40 X 24 text mode (SCREEN 0)
128      006F      C3 0538      JP      INIT32      ; " " " 32 X 24 text mode (SCREEN 1)
129      0072      C3 05D2      JP      INIGRP      ; " " " High resolution mode (SCREEN 2)
130      0075      C3 061F      JP      INIMLT      ; " " " Multi color mode (SCREEN 3)
131      0078      C3 0594      JP      SETTXT      ;Sets VDP to display 40 X 24 text mode
132      007B      C3 05B4      JP      SETT32      ; " " " " 32 X 24 text mode
  
```

133	007E	C3 0602	JP	SETGRP	; " " " " High-res mode
134	0081	C3 0659	JP	SETMLT	; " " " " Multi color mode
135	0084	C3 06E4	JP	CALPAT	;Get address of sprite pattern table
136	0087	C3 06F9	JP	CALATR	; " " " " attribute table
137	008A	C3 0704	JP	GSPSIZ	;Returns current sprite size
138	008D	C3 1510	JP	GRPPRT	;Print a character on the graphic screen
139				;	
140				SUBTTL	-BIOS header- BIOS calls (Programmable Sound Generator control)

```
141
142      ;
143      ; Following entry points are used for PSG initialization,
144      ; read and write PSG registers, and PLAY statement execution.
145      ;
146      0090      C3 04BD      JP      GICINI      ;Init PSG, and static data for PLAY
147      0093      C3 1102      JP      WRTPSG     ;Write data to PSG
148      0096      C3 110E      JP      RDPSG      ;Read data from PSG
149      0099      C3 11C4      JP      STRTMS     ;Checks and start background task for PLAY
150      ;
151      SUBTTL -BIOS header- BIOS calls (Keyboard, CRT, and Printer)
```

-BIOS header- BIOS calls (Keyboard, CRT, and Printer)

```

152
153      ;
154      ; General INPUT and PRINT utilities.
155      ;
156      009C      C3 0D6A      JP      CHSNS      ;Checks status of keyboard status
157      009F      C3 10CB      JP      CHGET      ;Return char typed, with wait
158      00A2      C3 08BC      JP      CHPUT      ;Output character to console
159      00A5      C3 085D      JP      LPTOUT      ; " " to printer, if possible
160      00A8      C3 0884      JP      LPTSTT      ;Checks status of line printer
161      00AB      C3 089D      JP      CNVCHR      ;Checks for graphic header byte
162                                     ;and convert code
163      00AE      C3 23BF      JP      PINLIN      ;Read line from keyboard to buffer
164      00B1      C3 23D5      JP      INLIN       ;Same as above, except in case of
165                                     ;AUTFLG is set
166      00B4      C3 23CC      JP      QINLIN      ;Print a "?", then jump to INLIN
167      00B7      C3 046F      JP      BREAKX      ;[Control-STOP] pressed??
168      00BA      C3 03FB      JP      ISCNTC      ;[Shift-STOP] pressed??
169      00BD      C3 10F9      JP      CKCNTC      ;Same as ISCNTC, but used by BASIC
170      00C0      C3 1113      JP      BEEP        ;Buzz
171      00C3      C3 0848      JP      CLS         ;Clear screen
172      00C6      C3 088E      JP      POSIT      ;Place cursor at Column [H], Row [L]
173      00C9      C3 0B26      JP      FNKSB      ;Display Function key, if neccessary
174      00CC      C3 0B15      JP      ERAFNK     ;Stop displaying the Function keys
175      00CF      C3 0B2B      JP      DSPFNK     ;Enable Function key display
176      00D2      C3 083B      JP      TOTEXT     ;Force screen to text mode
177      ;
178      SUBTTL -BIOS header- BIOS calls (Game and Cassette I/O, Queue handler)

```



```
179
180      ;
181      ; Following are used to read the value from Joysticks,
182      ; Graphic pad (tablet), and Paddles.
183      ;
184      00D5      C3 11EE      JP      GTSTCK      ;Return status of joystick
185      00D8      C3 1253      JP      GTTRIG      ;Read joystick trigger button
186      00DB      C3 12AC      JP      GTPAD       ;Returns status of graphic pad
187      00DE      C3 1273      JP      GTPDL      ;Read paddle
188      ;
189      ;
190      ; Following are used to access the cassette tape,
191      ; data read/write, and motor on/off
192      ;
193      00E1      C3 1A63      JP      TAPION      ;Turn on motor and read tape header
194      00E4      C3 1ABC      JP      TAPIN       ;Read tape data
195      00E7      C3 19E9      JP      TAPIOF      ;Stops reading from tape
196      00EA      C3 19F1      JP      TAPOON      ;Turn on motor and write tape header
197      00ED      C3 1A19      JP      TAPOUT      ;Write data to tape
198      00F0      C3 19DD      JP      TAPOFF      ;Stops writing to tape
199      00F3      C3 1384      JP      STMOTR      ;Start, stop cassette motor, or
200      ; flip motor(on to off, off to on)
201      ;
202      ;
203      ; BASIC queues
204      ;
205      00F6      C3 14EB      JP      LFTQ       ;Bytes left in queue
206      00F9      C3 1492      JP      PUTQ       ;Send a byte to queue
207      ;
208      SUBTTL -BIOS header- BIOS calls (Generalized graphics)
```

```
209
210      ;
211      ; For BASIC interpreter's GENGRP and ADVGRP modules use
212      00FC      C3 16C5      JP      RIGHTC      ;Moves one pixel right
213      00FF      C3 16EE      JP      LEFTC       ; " " " left
214      0102      C3 175D      JP      UPC         ; " " " up
215      0105      C3 173C      JP      TUPC        ; " " " "
216      0108      C3 172A      JP      DOWNC       ; " " " down
217      010B      C3 170A      JP      TDOWNC      ; " " " "
218      010E      C3 1599      JP      SCALXY      ;Scales X Y coordinates
219      0111      C3 15DF      JP      MAPXYC      ;Maps coordinates to physical address
220      0114      C3 1639      JP      FETCHC      ;Get current physical address and
221                                     ;mask pattern
222      0117      C3 1640      JP      STOREC       ;Put current physical address and
223                                     ;mask pattern
224      011A      C3 1676      JP      SETATR       ;Sets the color attribute byte
225      011D      C3 1647      JP      READC        ;Reads attribute of current pixel
226      0120      C3 167E      JP      SETC         ;Sets current pixel to specified attribute
227      0123      C3 1809      JP      NSETCX      ;Sets pixel horizontally
228      0126      C3 18C7      JP      GTASPC      ;Returns aspect ratio
229      0129      C3 18CF      JP      PNTINI      ;Do paint initialization
230      012C      C3 18E4      JP      SCANR       ;Scan pixels to the right
231      012F      C3 197A      JP      SCANL       ; " " " " left
232      ;
233      SUBTTL -BIOS header- BIOS calls (Misc. Entries)
```

```
234
235      ;
236      ;
237      0132      C3 0F3D      JP      CHGCAP      ;Turn [CAPSLOCK] light, on/off
238      0135      C3 0F7A      JP      CHGSND      ;Change status of 1 bit sound port
239      0138      C3 144C      JP      RSLREG      ;Return output of primary slot register
240      013B      C3 144F      JP      WSLREG      ;Write to primary slot register
241      013E      C3 1449      JP      RDVDP       ;Read VDP status register
242      0141      C3 1452      JP      SNSMAT      ;Read a specified row in the
243                                     ;keyboard matrix
244      0144      C3 148A      JP      PHYDIO      ;Performs operation for mass storage
245                                     ;devices (such as disks)
246      0147      C3 148E      JP      FORMAT      ;Initialize mass storage device
247      014A      C3 145F      JP      ISFLIO      ;Are we doing device I/O
248      014D      C3 1B63      JP      OUTDLP      ;Output to line printer
249      0150      C3 1470      JP      GETVCP      ;Used by Music background tasking
250      0153      C3 1474      JP      GETVC2      ; " " " " "
251      0156      C3 0468      JP      KILBUF      ;Clear the keyboard buffer
252      0159      C3 01FF      JP      CALBAS      ;Performs far-call into BASIC
253      015C                                     DS      005AH      ;RESERVED FOR EXPANSION
254      ;
255      SUBTTL - SLOT - Slot handler stuff
```

```

256
257      00A8      PPI.AR EQU      0A8h      ;A8H      read from PPI Port A
258      00A8      PPI.AW EQU      0A8h      ;A8H      Write to PPI Port A
259      ;
260      ; Every cartridge located at 0000-3FFFH must contain codes in
261      ; this module which are entered via following addresses.
262      ;
263      ;      000CH RDSLT
264      ;      0014H WRSLT
265      ;      001CH CALSLT
266      ;      0024H ENASLT
267      ;
268      ;
269      ; ----- RDSLT -----
270      ;
271      ; Selects the appropriate slot according to the value given
272      ; through registers, and read the content of memory from the
273      ; slot.
274      ;
275      ; Input parameters:
276      ; A - FxxxSSPP
277      ;      |   |||
278      ;      |   ||+--+ primary slot # (0-3)
279      ;      |   ++---- secondary slot # (0-3)
280      ;      +----- 1 if secondary slot # specified
281      ;
282      ;                      HL - address of target memory
283      ; Returned value
284      ;                      A - content of memory
285      ;
286      ; Note: Interrupts are disabled automatically but never enabled

```

```
287             ;      by this routine.
288             ;
289 01B6          RDSLTL:
290 01B6          CD 027E          CALL    SELPRM          ;Calculate bit pattern and mask code
291 01B9          FA 01C6          JP      M,RDESLT        ;Expanded slot specified
292 01BC          DB A8           IN      A,(PPI.AR)
293 01BE          57             LD      D,A              ;Save current setting
294 01BF          A1             AND     C              ;Cancel current setting for target address
295 01C0          B0             OR      B              ;Add new setting
296 01C1          CD F380        CALL    RAMLOW          ;Call read primitive routine (in system area)
297 01C4          7B             LD      A,E              ;Return value via [Acc]
298 01C5          C9             RET
299 01C6          RDESLTL:
300 01C6          E5             PUSH    HL              ;Save target address
301 01C7          CD 02A3        CALL    SELEXP          ;Select secondary slot
302 01CA          E3             EX      (SP),HL         ;Restore target address and save [HL]
303 01CB          C5             PUSH    BC
304 01CC          CD 01B6        CALL    RDSLTL
305 01CF          18 1B          JR      WRESED          ;Restore old slot select register
306          SUBTTL -SLOT- Slot handler (Write slot)
```

```
307
308 ;
309 ; ----- WRSLOT -----
310 ;
311 ; Selects the appropriate slot according to the value given
312 ; through registers, and write to the memory in the specified
313 ; slot.
314 ;
315 ; Input parameters:
316 ; A - FxxxSSPP
317 ;      ||||
318 ;      ||++-- primary slot # (0-3)
319 ;      |++---- secondary slot # (0-3)
320 ;      +----- 1 if secondary slot # specified
321 ;
322 ;      HL - address of target memory
323 ;
324 ;      E - value to be written
325 ;
326 ; Note: Interrupts are disabled automatically but never enabled
327 ;      by this routine.
328 ;
329 01D1 WRSLOT:
330 01D1 D5      PUSH    DE      ;Save data to be written
331 01D2 CD 027E CALL    SELPRM  ;Calculate bit pattern and mask code
332 01D5 FA 01E1 JP      M,WRESLT ;Expanded slot specified
333 01D8 D1      POP     DE      ;Restore data to be written
334 01D9 DB A8   IN      A,(PPI.AR)
335 01DB 57      LD      D,A      ;Save current setting
336 01DC A1      AND     C      ;Cancel current setting for target address
337 01DD B0      OR      B      ;Add new setting
```

```

338      01DE      C3 F385      JP      WRPRIM      ;Call write primitive routine (in system area)
339      01E1      WRESLT:      EX      (SP),HL      ;Save target address, get data to be written
340      01E1      E3      PUSH      HL      ;Save data to be written
341      01E2      E5      CALL      SELEXP      ;Select secondary slot
342      01E3      CD 02A3      POP      DE      ;Restore data to be written
343      01E6      D1      EX      (SP),HL      ;Restore target address and save [HL]
344      01E7      E3      PUSH      BC
345      01E8      C5      CALL      WRSLT
346      01E9      CD 01D1      WRESED:
347      01EC      POP      BC
348      01EC      C1      EX      (SP),HL      ;Save target address and get old [HL]
349      01ED      E3      PUSH      AF      ;Save value returned by RDSLT
350      01EE      F5      LD      A,B      ;Get current setting
351      01EF      78      AND      00111111B      ;Cancel current setting for 0C000H..0FFFFH
352      01F0      E6 3F      OR      C
353      01F2      B1      OUT      (PPI.AW),A      ;Enable 0C000H..0FFFFH of target bank
354      01F3      D3 A8      LD      A,L      ;Restore old setting of slot register
355      01F5      7D      LD      (0FFFFH),A
356      01F6      32 FFFF      LD      A,B
357      01F9      78      LD      A,B      ;Finally restore old primary slot register
358      01FA      D3 A8      OUT      (PPI.AW),A
359      01FC      F1      POP      AF      ;Restore value returned by RDSLT
360      01FD      E1      POP      HL      ;Restore target address
361      01FE      C9      RET

```

```
362
363      01FF          CALBAS:
364      01FF      FD 2A FCC0      LD      IY,(EXPTBL-1)
365      0203      18 12          JR      CALSLT
366      0205          CALLF:
367      0205      E3          EX      (SP),HL          ;Get return address, save [HL]
368      0206      F5          PUSH   AF              ;Save working registers
369      0207      D5          PUSH   DE
370      0208      7E          LD      A,(HL)          ;Get destination slot
371      0209      F5          PUSH   AF
372      020A      FD E1      POP     IY              ;Move it to IYH
373      020C      23          INC     HL
374      020D      5E          LD      E,(HL)          ;Get destination address
375      020E      23          INC     HL
376      020F      56          LD      D,(HL)
377      0210      23          INC     HL              ;Prepare true return address
378      0211      D5          PUSH   DE
379      0212      DD E1      POP     IX              ;Move it to IX
380      0214      D1          POP     DE              ;Restore working registers
381      0215      F1          POP     AF
382      0216      E3          EX      (SP),HL          ;Resture [HL], save true return address
383          SUBTTL -SLOT-
```



```

384
385
386 ; ----- CALSLT -----
387 ;
388 ; Performs inter-slot call to specified address.
389 ;
390 ; Input parameters:
391 ; IY - FxxxSSPP
392 ;      |   |||
393 ;      |   ||++-- primary slot # (0-3)
394 ;      |   ++---- secondary slot # (0-3)
395 ;      +----- 1 if secondary slot # specified
396 ;
397 ; IX - address to call
398 ;
399 ; Note: Interrupts are disabled automatically but never enabled
400 ;      by this routine.
401 ;      You can never pass arguments via alternate registers
402 ;      of Z80.
403 ;
404 0217 CALSLT:
405 0217 D9          EXX          ;Save environments
406 0218 08          EX          AF,AF'
407 0219 FD E5      PUSH        IY
408 021B F1          POP         AF          ;Get target slot information
409 021C DD E5      PUSH        IX
410 021E E1          POP         HL          ;Get target address
411 021F CD 027E    CALL        SELPRM
412 0222 FA 022E    JP          M,CAESL
413 0225 DB A8      IN          A,(PPI.AR)
414 0227 F5          PUSH        AF          ;Save current value of primary slot register

```

415	0228	A1	AND	C	;Cancel current setting for target address
416	0229	B0	OR	B	;Add new setting
417	022A	D9	EXX		;Restore environments except PSW
418	022B	C3 F38C	JP	CLPRIM	;Jump to primitive routine (in system area)
419	022E		CALESL:		
420	022E	CD 02A3	CALL	SELEXP	;Select secondary slot register
421	0231	F5	PUSH	AF	;Move primary slot # in [IYH]
422	0232	FD E1	POP	IY	
423	0234	E5	PUSH	HL	;Save [B,C,L] which contain information
424	0235	C5	PUSH	BC	;for restoring slot environments
425	0236	4F	LD	C,A	;Move primary slot # to [BC]
426	0237	06 00	LD	B,0	
427	0239	7D	LD	A,L	;Re-calculate what is currently output
428	023A	A4	AND	H	;to expansion slot register
429	023B	B2	OR	D	
430	023C	21 FCC5	LD	HL,SLTTBL	;Calculate address into SLTTBL
431	023F	09	ADD	HL,BC	
432	0240	77	LD	(HL),A	;Set current value output to expansion
433					;slot register
434	0241	E5	PUSH	HL	;Remember this address
435	0242	08	EX	AF,AF'	;Restore possible arguments passed
436	0243	D9	EXX		;via registers
437	0244	CD 0217	CALL	CALSLT	;Call by primary slot #
438	0247	D9	EXX		;Save possible values returned via
439	0248	08	EX	AF,AF'	;registers
440	0249	E1	POP	HL	;Restore address into SLTTBL
441	024A	C1	POP	BC	;Restore information about old slots
442	024B	D1	POP	DE	
443	024C	78	LD	A,B	;Get current setting
444	024D	E6 3F	AND	00111111B	;Cancel current setting for 0C000H..0FFFFH
445	024F	B1	OR	C	

446	0250	F3	DI		
447	0251	D3 A8	OUT	(PPI.AW),A	;Enable 0C000H..0FFFFH of target bank
448	0253	7B	LD	A,E	;Restore old setting of slot register
449	0254	32 FFFF	LD	(0FFFFH),A	
450	0257	78	LD	A,B	;Finally restore old primary slot register
451	0258	D3 A8	OUT	(PPI.AW),A	
452	025A	73	LD	(HL),E	;And change SLTTBL also
453	025B	08	EX	AF,AF'	;Restore possible returned values
454	025C	D9	EXX		
455	025D	C9	RET		

-SLOT-

```

456
457 ;
458 ; ----- ENASLT -----
459 ;
460 ; Selects the appropriate slot according to the value given
461 ; through registers, and permanently enables the slot.
462 ;
463 ; Input parameters:
464 ;
465 ; A - FxxxSSPP
466 ;      ||||
467 ;      ||+--- primary slot # (0-3)
468 ;      |++--- secondary slot # (0-3)
469 ;      +----- 1 if secondary slot # specified
470 ;
471 ; HL - address of target memory
472 ;
473 ; Note: Interrupts are disabled automatically but never enabled
474 ;       by this routine.
475 ;
476 025E ENASLT:
477 025E CD 027E      CALL    SELPRM      ;Calculate bit pattern and mask code
478 0261 FA 026B      JP      M,ENESLT    ;Expanded slot specified
479 0264 DB A8        IN        A,(PPI.AR)
480 0266 A1           AND        C        ;Cancel current setting for target address
481 0267 B0           OR         B        ;Add new setting
482 0268 D3 A8        OUT        (PPI.AW),A
483 026A C9           RET
484 026B ENESLT:
485 026B E5           PUSH     HL          ;Save target address
486 026C CD 02A3      CALL     SELEXP     ;Select secondary slot

```

487	026F	4F	LD	C,A	;Move primary slot # to [BC]
488	0270	06 00	LD	B,0	
489	0272	7D	LD	A,L	;Re-calculate what is currently output
490	0273	A4	AND	H	;to expansion slot register
491	0274	B2	OR	D	
492	0275	21 FCC5	LD	HL,SLTTBL	;Calculate address into SLTTBL
493	0278	09	ADD	HL,BC	
494	0279	77	LD	(HL),A	;Set current value output to expansion
495					;slot register
496	027A	E1	POP	HL	;Restore target address
497	027B	79	LD	A,C	;Restore primary slot # to [Acc]
498	027C	18 E0	JR	ENASLT	;Enable by primary slot register

```
499
500      027E      SELPRM:
501      027E      F3      DI
502      027F      F5      PUSH      AF      ;Save slot address
503      0280      7C      LD      A,H      ;Extract upper 2 bits
504      0281      07      RLCA
505      0282      07      RLCA
506      0283      E6 03      AND      00000011B
507      0285      5F      LD      E,A
508      0286      3E C0      LD      A,0C0H      ;Format mask pat. correspond to address
509      0288      SLPRM1:
510      0288      07      RLCA
511      0289      07      RLCA
512      028A      1D      DEC      E
513      028B      F2 0288      JP      P,SLPRM1
514      028E      5F      LD      E,A      ;Save mask pattern
515                                  ;      00000011      0000-3FFF
516                                  ;      00001100      4000-7FFF
517                                  ;      00110000      8000-BFFF
518                                  ;      11000000      C000-FFFF
519      028F      2F      CPL
520      0290      4F      LD      C,A      ;Save mask pattern
521                                  ;      11111100      0000-3FFF
522                                  ;      11110011      4000-7FFF
523                                  ;      11001111      8000-BFFF
524                                  ;      00111111      C000-FFFF
525      0291      F1      POP      AF      ;Restore slot address
526      0292      F5      PUSH      AF
527      0293      E6 03      AND      00000011B      ;Extract primary slot #
528      0295      3C      INC      A
529      0296      47      LD      B,A
```

530	0297	3E AB	LD	A,10101011B	;Convert slot # to proper bit pattern
531	0299		SLPRM2:		
532	0299	C6 55	ADD	A,01010101B	
533	029B	10 FC	DJNZ	SLPRM2	
534	029D	57	LD	D,A	;Save bit pattern for primary slot #
535					; 00000000 slot #0
536					; 01010101 slot #1
537					; 10101010 slot #2
538					; 11111111 slot #3
539	029E	A3	AND	E	;Extract significant bits
540	029F	47	LD	B,A	;Set it to [B]
541	02A0	F1	POP	AF	;Expanded slot specified?
542	02A1	A7	AND	A	;Set sign flag if so
543	C2A2	C9	RET		
544	02A3		SELEXP:		
545	02A3	F5	PUSH	AF	;Save target slot
546	02A4	7A	LD	A,D	;Get bit pattern for primary slot
547	02A5	E6 C0	AND	11000000B	;Extract slot # for 0C000H..0FFFFH
548	02A7	4F	LD	C,A	;Save it
549	02A8	F1	POP	AF	;Restore target slot
550	02A9	F5	PUSH	AF	;Save target slot
551	02AA	57	LD	D,A	;Load [D] with specified slot address
552	02AB	DB A8	IN	A,(PPI.AR)	
553	02AD	47	LD	B,A	;Save current setting
554	02AE	E6 3F	AND	00111111B	;Cancel current setting for 0C000H..0FFFFH
555	02B0	B1	OR	C	
556	02B1	D3 A8	OUT	(PPI.AW),A	;Enable 0C000H..0FFFFH or target bank
557	02B3	7A	LD	A,D	;Load slot information
558	02B4	0F	RRCA		
559	02B5	0F	RRCA		
560	02B6	E6 03	AND	00000011B	;Extract secondary slot #

561	02B8	57	LD	D,A	
562	02B9	3E AB	LD	A,10101011B	;Convert secondary slot # to proper
563	02BB		SLEXP1:		
564	02BB	C6 55	ADD	A,01010101B	;bit pattern
565	02BD	15	DEC	D	
566	02BE	F2 02BB	JP	P,SLEXP1	; 00000000 slot #0
567					; 01010101 slot #1
568					; 10101010 slot #2
569					; 11111111 slot #3
570	02C1	A3	AND	E	;Make bit pattern to be added
571	02C2	57	LD	D,A	;Save this
572	02C3	7B	LD	A,E	;Make bit pattern to strip off old value
573	02C4	2F	CPL		
574	02C5	67	LD	H,A	;Save this
575	02C6	3A FFFF	LD	A,(0FFFFH)	;Read expanded slot register
576	02C9	2F	CPL		
577	02CA	6F	LD	L,A	;Save current setting
578	02CB	A4	AND	H	;Strip off old bits
579	02CC	B2	OR	D	;And set new bits
580	02CD	32 FFFF	LD	(0FFFFH),A	;Set secondary slot register
581	02D0	78	LD	A,B	
582	02D1	D3 A8	OUT	(PPI.AW),A	;Restore original primary port
583	02D3	F1	POP	AF	;Restore target slot
584	02D4	E6 03	AND	00000011B	;Fake read from primary slot
585	02D6	C9	RET		
586			SUBTTL	- MSXIO - I/O Module	


```
587
588                ;;;;;;;;;;;;;;;;;;;;;;;;;;
589                ;                                ;
590                ;      Port definition      ;
591                ;                                ;
592                ;;;;;;;;;;;;;;;;;;;;;;;;;;
593                ;
594                ;      VDP address definition
595                ;
596    0098    VDP.DRW EQU    10011000B    ;98H    Read/write data VDP
597    0099    VDP.CW  EQU    10011001B    ;99H    write command to VDP
598    0099    VDP.SR  EQU    10011001B    ;99H    read status from VDP
599                ;
600    0007    V.COLR  EQU    7                ;In text mode, foreground and background color
601                ;                          ;Otherwise background color
602                ;
603                ;      PSG address definition
604                ;
605    00A0    PSG.LW  EQU    10100000B    ;A0H    latch address for PSG
606    00A1    PSG.DW  EQU    10100001B    ;A1H    write data to PSG
607    00A2    PSG.DR  EQU    10100010B    ;A2H    read data from PSG
608                ;
609    000E    PSG.PA  EQU    14                ;Port A of PSG
610    000F    PSG.PB  EQU    15                ;Port B of PSG
611                ;
612                ;      PPI address definition
613                ;
614    00A8    PPI.AR  EQU    10101000B    ;A8H    read from PPI Port A
615    00A9    PPI.BR  EQU    10101001B    ;A9H    read from PPI Port B
616    00AA    PPI.CR  EQU    10101010B    ;AAH    read from PPI Port C
617    00A8    PPI.AW  EQU    10101000B    ;A8H    Write to PPI Port A
```

```

618      00AA      PPI.CW EQU      10101010B      ;AAH      write to PPI Port C
619      00AB      PPI.CM EQU      10101011B      ;ABH      write to PPI command register
620      ;
621      ;          Printer port definition
622      ;
623      0091      LPT.DW EQU      10010001B      ;Data port
624      0090      LPT.SB EQU      10010000B      ;Strobe output
625      0090      LPT.ST EQU      10010000B      ;Printer status
626      ;
627      ;          Text mode (40*24)                                SCREEN 0
628      ;
629      ;          TXTNAM,TXTCGP
630      ;
631      ;          Text mode (graphics 1)                            SCREEN 1
632      ;
633      ;          T32NAM,T32COL,T32CGP,T32ATR,T32PAT
634      ;
635      ;          Hires mode                                SCREEN 2
636      ;
637      ;          GRPNAM,GRPCOL,GRPCGP,GRPATR,GRPPAT
638      ;
639      ;          Multi-color mode                                SCREEN 3
640      ;
641      ;          MLTNAM,MLTCGP,MLTATR,MLTPAT
642      ;
643      ;          Screen size
644      ;
645      ;          LINLEN,CRTCNT,LINL32,LINL40
646      ;
647      ;          External constants
648      ;

```

```
649          ;          CGTABL          Character generator table
650          ;
651          ;          External variables
652          ;
653          ;          FORCLR          Foreground color
654          ;          BAKCLR          Background color
655          ;          BDRCLR          Border color for PAINT
656          ;          SCRMOD          Current screen mode
657          ;          0 - 40*24 text
658          ;          1 - 32*24 text
659          ;          2 - hiresolution graphics
660          ;          3 - Multicolor graphics
661          ;          OLDSCR
662          ;          NAMBAS          Base of current name table
663          ;          CGPBAS          Base of current cgen table
664          ;          PATBAS          Base of current sprite pattern table
665          ;          ATRBAS          Base of current sprite attribute table
666          ;          JIFFY          Jiffy count
667          ;          CLIKSW          Click switch
668          ;          CLIKFL          Click flag to suppress multiple key clicks
669          ;          RG0SAV          VDP register #0 save area
670          ;          RG1SAV          VDP register #1 save area
671          ;          STATFL          VDP status register
672          ;          PATWRK          Work area for pattern converter
673          ;
674          ;          External routines
675          ;
676          ;          GETQ
677          ;          PUTQ
678          ;          INITQ
679          SUBTTL - MSXIO - Find available RAM
```

```
680
681      02D7      CHKRAM:
682                ;
683                ; ----- CHKRAM -----
684                ;
685                ; Look into every slot from 0FFFFH to C000H, and set system work
686                ; area. Note that we cannot use RAM as work area nor perform
687                ; subroutine calls 'cause we do not yet know where the available
688                ; RAM exits. Everything has to be done inside ROM and CPU's
689                ; register until the RAM is found.
690                ;
691      02D7      3E 82          LD      A,82H          ;Port A - output (mode 0)
692      02D9      D3 AB          OUT     (PPI.CM),A      ;Port B - input (mode 0)
693      02DB      AF            XOR      A              ;Port C - output (mode 0)
694      02DC      D3 A8          OUT     (PPI.AW),A      ;Select slot 0 for all addresses
695      02DE      3E 50          LD      A,'P'          ;Disable all cassette related outputs
696      02E0      D3 AA          OUT     (PPI.CW),A      ;Motor off
697                ;
698                ; Start searching
699                ;
700                ; Register usage:
701                ; B - non 0 if we're now checking secondary slot
702                ; SPH - slot # of the biggest RAM block
703                ; SPL - secondary slot # of the biggest RAM block (if any)
704                ; DE - lowest address of the biggest RAM block ever found
705                ; C - 'slot-expanded' flag
706                ;
707                ; 0000xxxx
708                ;    ||||
709                ;    |||+- slot #3 expanded
710                ;    ||+-- slot #2 expanded
```

```

711      ;      |+--- slot #1 expanded
712      ;      +----- slot #0 expanded
713      ;
714      02E2    11 FFFF      LD      DE,0FFFFH      ;Initialize lowest address ever found
715      02E5    AF          XOR      A              ;Start from slot #0
716      02E6    4F          LD      C,A             ;Clear bit pattern
717      02E7                                CKRM05:
718      02E7    D3 A8      OUT      (PPI.AW),A      ;Select the slot
719      02E9    CB 21      SLA      C              ;Shift bit pattern
720      02EB    06 00      LD      B,0             ;Assume this slot is not expanded
721      02ED    21 FFFF      LD      HL,0FFFFH      ;Read from possible expansion slot register
722      02F0    36 F0      LD      (HL),0F0H      ;Write a binary 11110000
723      02F2    7E          LD      A,(HL)
724      02F3    D6 0F      SUB      0FH            ;Read back as 00001111?
725      02F5    20 0B      JR      NZ,CKRM15      ;Nop, this is not an expanded slot
726      02F7    77          LD      (HL),A        ;Write 00000000
727      02F8    7E          LD      A,(HL)
728      02F9    3C          INC      A              ;Read back as 11111111?
729      02FA    20 06      JR      NZ,CKRM15      ;Nop, not expanded slot
730      02FC    04          INC      B              ;We're checking expanded slot
731      02FD    CB C1      SET      0,C            ;Say this slot is expanded
732      02FF                                CKRM10:
733      ;
734      ;Start from expansion slot #0
735      ;
736      02FF    32 FFFF      LD      (0FFFFH),A    ;Select the expanded slot
737      0302                                CKRM15:
738      0302    21 BF00      LD      HL,0BF00H      ;Start checking from 0BF00H to 8000H
739      0305                                CKRM20:
740      0305    7E          LD      A,(HL)
741      0306    2F          CPL

```

MSX ROM BASIC BIOS) Macro-80
MSXIO - Find available RAM

3.44 01-Jan-85

PAGE 16-2

30

742	0307	77	LD	(HL),A	
743	0308	BE	CP	(HL)	
744	0309	2F	CPL		
745	030A	77	LD	(HL),A	
746	030B	20 07	JR	NZ,CKRM25	;RAM not equipped in this page
747	030D	2C	INC	L	;Make sure it's not a coincidence
748	030E	20 F5	JR	NZ,CKRM20	;Check more
749	0310	25	DEC	H	
750	0311	FA 0305	JP	M,CKRM20	;Check next page
751	0314				
752	0314	2E 00	LD	L,0	
753	0316	24	INC	H	
754	0317	7D	LD	A,L	;Below the one ever found
755	0318	93	SUB	E	
756	0319	7C	LD	A,H	
757	031A	9A	SBC	A,D	
758	031B	30 0A	JR	NC,CKRM30	;No
759	031D	EB	EX	DE,HL	;Register this address as the lowest
760	031E	3A FFFF	LD	A,(0FFFFH)	;Set possible secondary slot #
761	0321	2F	CPL		
762	0322	6F	LD	L,A	
763	0323	DB A8	IN	A,(PPI.AR)	;Set primary slot #
764	0325	67	LD	H,A	
765	0326	F9	LD	SP,HL	;Register these slot #'s
766	0327				
767	0327	78	LD	A,B	
768	0328	A7	AND	A	;Are we checking secondary slot
769	0329	28 0A	JR	Z,CKRM35	;No
770	032B	3A FFFF	LD	A,(0FFFFH)	
771	032E	2F	CPL		
772	032F	C6 10	ADD	A,10H	;Prepare to select next secondary slot

(MSX ROM BASIC BIOS) Macro-80
- MSXIO - Find available RAM

3.44

01-Jan-85

PAGE 16-3

31

773	0331	FE 40	CP	01000000B	
774	0333	38 CA	JR	C,CKRM10	;Continue if more secondary slots remain
775	0335		CKRM35:		
776	0335	DB A8	IN	A,(PPI.AR)	
777	0337	C6 50	ADD	A,01010000B	;Prepare to select next slot
778	0339	30 AC	JR	NC,CKRM05	;Continue if more primary slots remain

```

779
780      ;
781      ; Check is done, select the biggest one
782      ;
783      033B    21 0000      LD      HL,0
784      033E    39          ADD     HL,SP
785      033F    7C          LD      A,H
786      0340    D3 A8      OUT     (PPI.AW),A      ;Set primary slot register
787      0342    7D          LD      A,L
788      0343    32 FFFF      LD      (0FFFFH),A      ;Set possible secondary slot register
789      ;
790      ; Next, check 0C000H..0FFFFH
791      ;
792      0346    79          LD      A,C
793      0347    07          RLCA
794      0348    07          RLCA
795      0349    07          RLCA
796      034A    07          RLCA
797      034B    4F          LD      C,A
798      034C    11 FFFF      LD      DE,0FFFFH      ;Initialize lowest address ever found
799      034F    DB A8      IN      A,(PPI.AR)      ;Start from slot #0
800      0351    E6 3F      AND     00111111B
801      0353      CKRM50:
802      0353    D3 A8      OUT     (PPI.AW),A      ;Select the slot
803      0355    06 00      LD      B,0      ;Assume this slot is not expanded
804      0357    CB 01      RLC      C      ;Shift bit pattern
805      0359    30 0A      JR      NC,CKRM60      ;This slot is not expanded
806      035B    04          INC     B      ;We're checking expanded slot
807      035C    3A FFFF      LD      A,(0FFFFH)
808      035F    2F          CPL
809      0360    E6 3F      AND     00111111B

```


810	0362		CKRM55:		
811	0362	32 FFFF	LD	(0FFFFH),A	;Select the expanded slot
812	0365		CKRM60:		
813	0365	21 FE00	LD	HL,0FE00H	;Start checking from 0FE00H to 0C000H
814	0368		CKRM65:		
815	0368	7E	LD	A,(HL)	
816	0369	2F	CPL		
817	036A	77	LD	(HL),A	
818	036B	BE	CP	(HL)	
819	036C	2F	CPL		
820	036D	77	LD	(HL),A	
821	036E	20 09	JR	NZ,CKRM70	;RAM not equipped in this page
822	0370	2C	INC	L	;Make sure it's not a coincidence
823	0371	20 F5	JR	NZ,CKRM65	;Check more
824	0373	25	DEC	H	
825	0374	7C	LD	A,H	
826	0375	FE C0	CP	0C0H	
827	0377	30 EF	JR	NC,CKRM65	;Check next page
828	0379		CKRM70:		
829	0379	2E 00	LD	L,0	
830	037B	24	INC	H	
831	037C	7D	LD	A,L	;Below the one ever found
832	037D	93	SUB	E	
833	037E	7C	LD	A,H	
834	037F	9A	SBC	A,D	
835	0380	30 0A	JR	NC,CKRM75	;No
836	0382	EB	EX	DE,HL	;Register this address as the lowest
837	0383	3A FFFF	LD	A,(0FFFFH)	;Set possible secondary slot #
838	0386	2F	CPL		
839	0387	6F	LD	L,A	
840	0388	DB A8	IN	A,(PPI.AR)	;Set primary slot #

841	038A	67	LD	H,A	
842	038B	F9	LD	SP,HL	;Register these slot #'s
843	038C		CKRM75:		
844	038C	78	LD	A,B	
845	038D	A7	AND	A	;Are we checking secondary slot
846	038E	28 08	JR	Z,CKRM80	;No
847	0390	3A FFFF	LD	A,(0FFFFH)	
848	0393	2F	CPL		
849	0394	C6 40	ADD	A,01000000B	;Prepare to select next secondary slot
850	0396	30 CA	JR	NC,CKRM55	;Continue if more secondary slots remain
851	0398		CKRM80:		
852	0398	DB A8	IN	A,(PPI.AR)	
853	039A	C6 40	ADD	A,01000000B	;Prepare to select next slot
854	039C	30 B5	JR	NC,CKRM50	;Continue if more primary slots remain
855			SUBTTTL	- MSXIO - Slot attribute setup	

```

856
857
858      ; Check is done, select the biggest one
859      ;
860      039E    21 0000      LD      HL,0
861      03A1    39          ADD     HL,SP
862      03A2    7C          LD      A,H
863      03A3    D3 A8      OUT     (PPI.AW),A      ;Set primary slot register
864      03A5    7D          LD      A,L
865      03A6    32 FFFF      LD      (0FFFFH),A      ;Set possible secondary slot register
866      03A9    79          LD      A,C      ;Set 'slot expanded' flag
867
868      ;
869      ; Clear work area with zero
870      ;
871      03AA    01 0C49      LD      BC,0C49H      ;length of work area
872      03AD    11 F381      LD      DE,RAMLOW+1
873      03B0    21 F380      LD      HL,RAMLOW      ;beginning of work
874      03B3    36 00      LD      (HL),0      ;init first byte
875      03B5    ED B0      LDIR      ;transfer it to rest of area
876
877      ;
878      ; Set EXPTBL
879      ;
880      03B7    4F          LD      C,A      ;Get 'slot-expanded' flag
881      03B8    06 04      LD      B,4      ;Loop 4 times
882      03BA    21 FCC4      LD      HL,EXPTBL+3
883      03BD      SSLTLP:
884      03BD    CB 19      RR      C      ;Set carry if LSB is set
885      03BF    9F          SBC      A,A      ;[Acc]=255 if expanded, 0 if not expanded
886      03C0    E6 80      AND      80H      ;Affects only MSB
887      03C2    77          LD      (HL),A      ;Set table for each slot
888      03C3    2B          DEC     HL

```

```

887      03C4      10 F7              DJNZ      SSLTLP
888                                     ;
889                                     ; Set SLTTBL
890                                     ;
891      03C6      DB A8              IN        A,(PPI.AR)      ;Remember primary slot register's content
892      03C8      4F                  LD        C,A
893      03C9      AF                  XOR       A              ;Read from slot #0
894      03CA      D3 A8              OUT       (PPI.AW),A
895      03CC      3A FFFF            LD        A,(0FFFFH)
896      03CF      2F                  CPL
897      03D0      6F                  LD        L,A
898      03D1      3E 40              LD        A,01000000B      ;Read from slot #1
899      03D3      D3 A8              OUT       (PPI.AW),A
900      03D5      3A FFFF            LD        A,(0FFFFH)
901      03D8      2F                  CPL
902      03D9      67                  LD        H,A
903      03DA      3E 80              LD        A,80H          ;Read from slot #2
904      03DC      D3 A8              OUT       (PPI.AW),A
905      03DE      3A FFFF            LD        A,(0FFFFH)
906      03E1      2F                  CPL
907      03E2      5F                  LD        E,A
908      03E3      3E C0              LD        A,0C0H          ;Read from slot #3
909      03E5      D3 A8              OUT       (PPI.AW),A
910      03E7      3A FFFF            LD        A,(0FFFFH)
911      03EA      2F                  CPL
912      03EB      57                  LD        D,A
913      03EC      79                  LD        A,C          ;Restore primary slot register
914      03ED      D3 A8              OUT       (PPI.AW),A
915      03EF      22 FCC5            LD        (SLTTBL),HL      ;Set SLTTBL
916      03F2      EB                  EX        DE,HL
917      03F3      22 FCC7            LD        (SLTTBL+2),HL
  
```

(MSX ROM BASIC BIOS) Macro-80
-- MSXIO - Slot attribute setup

3.44

01-Jan-85

PAGE 18-2

37

918 03F6 ED 56 IM 1 ;IM 1

919 03F8 C3 2680 JP INIT

920 SUBTTL - MSXIO - Control-[C] processing

```

921
922      03FB      ISCNTC:
923      03FB      3A FBB1      LD      A,(BASROM)      ;Is BASIC text in ROM
924      03FE      A7          AND      A
925      03FF      C0          RET      NZ              ;Yes
926      0400      E5          PUSH     HL
927      0401      21 FC9B     LD      HL,INTFLG      ;Seen any interesting key
928      0404      F3          DI
929      0405      7E          LD      A,(HL)
930      0406      36 00      LD      (HL),0
931      0408      E1          POP      HL
932      0409      FB          EI
933      040A      A7          AND      A
934      040B      C8          RET      Z              ;No
935      040C      FE 03      CP      3              ;Is it ctrl-stop?
936      040E      28 1C      JR      Z,EXCABO      ;Yes, execution aborted
937      ;
938      ; Pause until next STOP is pressed
939      ;
940      0410      E5          PUSH     HL              ;STOP pressed (pause)
941      0411      D5          PUSH     DE
942      0412      C5          PUSH     BC
943      0413      CD 09DA     CALL     CKDPC0      ;Display cursor if disabled
944      0416      21 FC9B     LD      HL,INTFLG      ;Wait for next interesting key
945      0419      WATINT:
946      0419      F3          DI
947      041A      7E          LD      A,(HL)
948      041B      36 00      LD      (HL),0
949      041D      FB          EI              ;Wait for character if SELECT pressed
950      041E      A7          AND      A              ;Seen?
951      041F      28 F8      JR      Z,WATINT      ;Not yet

```

```

952      0421      F5          PUSH      AF
953      0422      CD 0A27     CALL      CKERC0          ;Erase cursor if disabled
954      0425      F1          POP       AF
955      0426      C1          POP       BC
956      0427      D1          POP       DE
957      0428      E1          POP       HL
958      0429      FE 03       CP         3          ;Abort?
959      042B      C0          RET        NZ          ;No
960      042C                                EXCABO:
961      042C      E5          PUSH      HL          ;Save text pointer
962      042D      CD 0468     CALL      KILBUF       ;Cancel any input
963      0430      CD 0454     CALL      CKSTTP       ;Is STOP trap ON
964      0433      30 0A       JR         NC,EXABOL   ;No, accept this break
965      0435      21 FC6A     LD         HL,REQSTP    ;Request STOP trap
966      0438      F3          DI          ;Since REQTRP does not change interrupt mask,
967      0439      CD 0EF1     CALL      REQTRP       ;this must be enclosed by 'DI' and 'EI'
968      043C      FB          EI
969      043D      E1          POP       HL          ;Restore text pointer
970      043E      C9          RET
971      043F                                EXABOL:
972      ;
973      043F      CD 083B     CALL      TOTEXT       ;Make sure we're in text mode
974      0442      3A FCC1     LD         A,(EXPTBL)   ;Make sure BASIC is enabled
975      0445      26 40       LD         H,01000000B
976      0447      CD 025E     CALL      ENASLT
977      044A      E1          POP       HL          ;Restore text pointer
978      044B      AF          XOR        A          ;Must return with carry cleared, zero set
979      044C      ED 7B F6B1   LD         SP,(SAVSTK) ;LSPD
980      0450      C5          PUSH      BC
981      0451      C3 63E6     JP         STOP
982

```

```
983      0454      CKSTTP:
984      ;
985      ; Check for STOP trap
986      ;
987      ;
988      0454      3A FC6A      LD      A,(REQSTP)      ;Is STOP trap ON
989      0457      0F          RRCA
990      0458      D0          RET      NC              ;No, accept this break
991      0459      2A FC6B      LD      HL,(REQSTP+1)    ;Is STOP trap specified
992      045C      7C          LD      A,H
993      045D      B5          OR      L
994      045E      C8          RET      Z              ;No, accept this break
995      045F      2A F41C      LD      HL,(CURLIN)      ;Are we in direct mode
996      0462      23          INC     HL
997      0463      7C          LD      A,H
998      0464      B5          OR      L
999      0465      C8          RET      Z              ;Yes, treat as break
1000     0466      37          SCF              ;Set flag to indicate STOP trap active
1001     0467      C9          RET
1002     0468      KILBUF:
1003     ;
1004     0468      2A F3F8      LD      HL,(PUTPNT)      ;Empties ring buffer
1005     046B      22 F3FA      LD      (GETPNT),HL
1006     046E      C9          RET
```



```
1007
1008      046F      BREAKX:
1009      ;
1010      ; Check if stop key pressed. If pressed, return with carry set.
1011      ;
1012      046F      DB AA          IN      A,(PPI.CR)
1013      0471      E6 F0          AND      0F0H          ;Leave others unaffected
1014      0473      F6 07          OR       7            ;Select 6th row
1015      0475      D3 AA          OUT      (PPI.CW),A
1016      0477      DB A9          IN      A,(PPI.BR)
1017      0479      E6 10          AND      10H          ;STOP key is assigned to bit 4
1018      047B      C0            RET      NZ            ;0 when pressed
1019      047C      DB AA          IN      A,(PPI.CR)
1020      047E      3D            DEC      A
1021      047F      D3 AA          OUT      (PPI.CW),A
1022      0481      DB A9          IN      A,(PPI.BR)
1023      0483      E6 02          AND      2
1024      0485      C0            RET      NZ
1025      0486      E5            PUSH     HL
1026      0487      2A F3F8        LD      HL,(PUTPNT)    ;Cancel any input
1027      048A      22 F3FA        LD      (GETPNT),HL
1028      048D      E1            POP      HL
1029      048E      3A FBEL        LD      A,(OLDKEY+7)    ;STOP pressed, mark as pressed to prevent
1030      0491      E6 EF          AND      0EFH          ; to be doubly recognized
1031      0493      32 FBEL        LD      (OLDKEY+7),A
1032      0496      3E 0D          LD      A,0DH
1033      0498      32 F3F7        LD      (REPCNT),A
1034      049B      37            SCF
1035      049C      C9            RET
1036      SUBTTL - MSXIO - PSG Initialization
```

```
1037
1038      049D          INITIO:
1039                      ;
1040                      ; Initialize I O
1041                      ;
1042      049D      3E 07          LD      A,7
1043      049F      1E 80          LD      E,80H
1044      04A1      CD 1102        CALL    WRTPSG          ;Set Port A to input mode
1045      04A4      3E 0F          LD      A,0FH          ;Port B to output mode
1046      04A6      1E CF          LD      E,0CFH
1047      04A8      CD 1102        CALL    WRTPSG
1048      04AB      3E 0B          LD      A,0BH          ;Dummy write cycle to wake up the PSG
1049      04AD      5F             LD      E,A            ;envelope register
1050      04AE      CD 1102        CALL    WRTPSG          ;Any value is OK!
1051      04B1      CD 110C        CALL    INGI
1052      04B4      E6 40          AND     01000000B
1053      04B6      32 FCAD        LD      (KANAMD),A
1054      04B9      3E FF          LD      A,0FFH
1055      04BB      D3 90          OUT     (LPT.SB),A
1056      04BD          GICINI:
1057                      ;
1058                      ; Initialize GI sound chip, queues, and static data.
1059                      ;
1060                      ; Entry - Interrupts must be disabled
1061                      ; Exit - All registers preserved.
1062                      ;
1063      04BD      E5             PUSH     HL              ;save caller's registers
1064      04BE      D5             PUSH     DE
1065      04BF      C5             PUSH     BC
1066      04C0      F5             PUSH     AF
1067                      ;
```

```

1068                ; First, clear all static data
1069                ;
1070    04C1    21 FB3F                LD    HL,MUSICF
1071    04C4    06 71                LD    B,71H                ;=VCBC + VCBSIZ + MUSCIF
1072    04C6    AF                  XOR    A
1073    04C7                MUSCLL:
1074    04C7    77                  LD    (HL),A
1075    04C8    23                  INC    HL
1076    04C9    10 FC                DJNZ  MUSCLL
1077                ;
1078                ; Then clear music dynamic queue
1079                ;
1080    04CB    11 F975                LD    DE,VOICAQ                ;Address of music queue
1081    04CE    06 7F                LD    B,7FH                ;Mask pattern, 7F = Music queue len - 1
1082    04D0    21 0080                LD    HL,80H                ;Queue length
1083    04D3                GICIN1:
1084    04D3    E5                  PUSH   HL                ;Save length of queue
1085    04D4    D5                  PUSH   DE                ;Save address of queue
1086    04D5    C5                  PUSH   BC                ;Save mask pattern
1087    04D6    F5                  PUSH   AF                ;Save queue ID
1088    04D7    CD 14DA                CALL  INITQ                ;Initialize a queue by [Acc],[B],[DE]
1089    04DA    F1                  POP     AF
1090    04DB    C6 08                ADD    A,8                ;write to regs 8,9,10
1091    04DD    1E 00                LD     E,0
1092    04DF    CD 1102                CALL  WRTPSG                ;0 out amplitude (turn voice off)
1093    04E2    D6 08                SUB    8                ;Restore [Acc]
1094    04E4    F5                  PUSH   AF                ;Save queue ID
1095    04E5    2E 0F                LD     L,0FH                ;OctaveX
1096    04E7    CD 1477                CALL  GETVC1                ;[HL] points to octave for voice [A]
1097    04EA    EB                  EX     DE,HL
1098    04EB    21 0508                LD     HL,MUSITB                ;[HL] points to default value table

```

```

1099      04EE      01 0006      LD      BC,6      ;EMSITB - MUSITB
1100      04F1      ED B0      LDIR      ;default variables for this voice
1101      04F3      F1      POP      AF      ;Restore queue ID
1102      04F4      C1      POP      BC      ;Restore mask
1103      04F5      E1      POP      HL      ;Restore queue address
1104      04F6      D1      POP      DE      ;Restore queue length
1105      04F7      19      ADD      HL,DE      ;Update queue address
1106      04F8      EB      EX      DE,HL
1107      04F9      3C      INC      A      ;Next channel
1108      04FA      FE 03      CP      3
1109      04FC      38 D5      JR      C,GICIN1      ;Loop till done all three voices
1110      04FE      3E 07      LD      A,7      ;write to reg 7 mixer control
1111      0500      1E B8      LD      E,0B8H      ;input port A, output port B
1112      0502      CD 1102      CALL     WRTPSG      ;disable noise, enable all 3 tones
1113      0505      C3 08DA      JP      POPALL      ;Restore environments
1114      0508
1115      ;
1116      ; table of default values for music variables
1117      ;
1118      0508      04      DB      04H      ;default octave
1119      0509      04      DB      04H      ;default note length
1120      050A      78      DB      78H      ;default tempo
1121      050B      88      DB      88H      ;default volume
1122      050C      FF      DB      0FFH      ;default envelope period
1123      050D      00      DB      00H
1124      050E      EMSITB:      ;end of music table
1125      SUBTTL - MSXIO - Utility routines for VDP

```

```
1126
1127      050E      INITXT:
1128                ;
1129                ; Initialize VDP for text mode (40 by 24)
1130                ;
1131      050E      CD 0577      CALL      DISSCR
1132      0511      AF          XOR      A
1133      0512      32 FCAF      LD        (SCRMOD),A
1134      0515      32 FCB0      LD        (OLDSCR),A
1135      0518      3A F3AE      LD        A,(LINL40)
1136      051B      32 F3B0      LD        (LINLEN),A
1137      051E      2A F3B3      LD        HL,(TXTNAM)
1138      0521      22 F922      LD        (NAMBAS),HL
1139      0524      2A F3B7      LD        HL,(TXTCGP)
1140      0527      22 F924      LD        (CGPBAS),HL
1141      052A      CD 07F7      CALL      CHGCLR      ;Set border/foreground/background color
1142      052D      CD 077E      CALL      CLRTXT
1143      0530      CD 071E      CALL      INIPAT      ;Initialize character pattern
1144      0533      CD 0594      CALL      SETTXT      ;Actually set VDP registers
1145      0536      18 38      JR        ENASCR
1146      0538      INIT32:
1147                ;
1148                ; Initialize VDP for text mode (graphics 1)
1149                ;
1150      0538      CD 0577      CALL      DISSCR
1151      053B      3E 01      LD        A,1
1152      053D      32 FCAF      LD        (SCRMOD),A
1153      0540      32 FCB0      LD        (OLDSCR),A
1154      0543      3A F3AF      LD        A,(LINL32)
1155      0546      32 F3B0      LD        (LINLEN),A
1156      0549      2A F3BD      LD        HL,(T32NAM)
```

1157	054C	22 F922	LD	(NAMBAS),HL	
1158	054F	2A F3C1	LD	HL,(T32CGP)	
1159	0552	22 F924	LD	(CGPBAS),HL	
1160	0555	2A F3C5	LD	HL,(T32PAT)	
1161	0558	22 F926	LD	(PATBAS),HL	
1162	055B	2A F3C3	LD	HL,(T32ATR)	
1163	055E	22 F928	LD	(ATRBAS),HL	
1164	0561	CD 07F7	CALL	CHGCLR	;Set border foreground background color
1165	0564	CD 077E	CALL	CLRTXT	
1166	0567	CD 071E	CALL	INIPAT	;Initialize character pattern
1167	056A	CD 06BB	CALL	ERASPR	;Clear sprites
1168	056D	CD 05B4	CALL	SETT32	;Actually set VDP registers
1169	0570			ENASCR:	
1170				;	
1171				; Enable screen display	
1172				;	
1173	0570	3A F3E0	LD	A,(RGLSAV)	
1174	0573	F6 40	OR	01000000B	
1175	0575	18 05	JR	DISSC1	
1176	0577			DISSCR:	
1177				;	
1178				; Disable screen display	
1179				;	
1180	0577	3A F3E0	LD	A,(RGLSAV)	
1181	057A	E6 BF	AND	0BFH	
1182	057C			DISSC1:	
1183	057C	47	LD	B,A	
1184	057D	0E 01	LD	C,1	

```
1185
1186      057F      WRTVDP:
1187                ;
1188                ; Write data to VDP
1189                ;
1190                ; C = register #
1191                ; B = value to be set
1192                ;
1193                ; Register save area for the register is properly set
1194                ;
1195      057F      78          LD      A,B          ;Get data to set
1196      0580      F3          DI
1197      0581      D3 99      OUT      (VDP.CW),A
1198      0583      79          LD      A,C          ;Get register #
1199      0584      F6 80      OR       80H
1200      0586      D3 99      OUT      (VDP.CW),A
1201      0588      FB          EI
1202      0589      E5          PUSH    HL
1203      058A      78          LD      A,B          ;Remember this value 'cause this is
1204      058B      06 00      LD      B,0          ;a write-only register
1205      058D      21 F3DF     LD      HL,RG0SAV
1206      0590      09          ADD     HL,BC
1207      0591      77          LD      (HL),A
1208      0592      E1          POP     HL
1209      0593      C9          RET
1210      0594      SETTXT:
1211                ;
1212                ; Set VDP for text mode (40 by 32)
1213                ;
1214      0594      3A F3DF     LD      A,(RG0SAV)    ;Set register #0
1215      0597      E6 01      AND     1
```

```

1216      0599      47          LD      B,A
1217      059A      0E 00      LD      C,0
1218      059C      CD 057F    CALL    WRTVDP
1219      059F      3A F3E0    LD      A,(RG1SAV)      ;Set register #1
1220      05A2      E6 E7      AND      0E7H
1221      05A4      F6 10      OR       10H
1222      05A6      47          LD      B,A
1223      05A7      0C          INC      C
1224      05A8      CD 057F    CALL    WRTVDP
1225      05AB      21 F3B3    LD      HL,TXTNAM
1226      05AE      11 0000    LD      DE,0          ;Set mask pattern
1227      05B1      C3 0677    JP       SETSCM        ;Set screen mode
1228      05B4          SETT32:
1229                      ;
1230                      ; Set VDP for text mode (graphics 1)
1231                      ;
1232      05B4      3A F3DF    LD      A,(RG0SAV)      ;Set register #0
1233      05B7      E6 01      AND      1
1234      05B9      47          LD      B,A
1235      05BA      0E 00      LD      C,0
1236      05BC      CD 057F    CALL    WRTVDP
1237      05BF      3A F3E0    LD      A,(RG1SAV)      ;Set register #1
1238      05C2      E6 E7      AND      0E7H
1239      05C4      47          LD      B,A
1240      05C5      0C          INC      C
1241      05C6      CD 057F    CALL    WRTVDP
1242      05C9      21 F3BD    LD      HL,T32NAM
1243      05CC      11 0000    LD      DE,0          ;Set mask pattern
1244      05CF      C3 0677    JP       SETSCM        ;Set screen mode
1245      05D2          INIGRP:
1246                      ;

```



```
1247                ; Initialize VDP for graphics mode
1248                ;
1249    05D2    CD 0577                CALL    DISSCR
1250    05D5    3E 02                LD      A,2
1251    05D7    32 FCAF                LD      (SCRMOD),A
1252    05DA    2A F3CF                LD      HL,(GRPPAT)
1253    05DD    22 F926                LD      (PATBAS),HL
1254    05E0    2A F3CD                LD      HL,(GRPATR)
1255    05E3    22 F928                LD      (ATRBAS),HL
1256    05E6    2A F3C7                LD      HL,(GRPNAM)    ;Initialize name table
1257    05E9    CD 07DF                CALL    SETWRT
1258    05EC    AF                    XOR     A
1259    05ED    06 03                LD      B,3
1260    05EF                INIGR1:
1261    05EF    D3 98                OUT     (VDP.DRW),A
1262    05F1    3C                    INC     A
1263    05F2    20 FB                JR      NZ,INIGR1
1264    05F4    10 F9                DJNZ   INIGR1
1265    05F6    CD 07A1                CALL    CLSHRS    ;Clear pattern and color table
1266    05F9    CD 06BB                CALL    ERASPR
1267    05FC    CD 0602                CALL    SETGRP    ;Actually set VDP mode
1268    05FF    C3 0570                JP      ENASCR
1269    0602                SETGRP:
1270                ;
1271                ; Set VDP for graphics mode (graphics 2)
1272                ;
1273    0602    3A F3DF                LD      A,(RG0SAV)    ;Set register #0
1274    0605    F6 02                OR      2
1275    0607    47                    LD      B,A
1276    0608    0E 00                LD      C,0
1277    060A    CD 057F                CALL    WRTVDP
```

```
1278 060D 3A F3E0 LD A,(RGLSAV) ;Set register #1
1279 0610 E6 E7 AND 0E7H
1280 0612 47 LD B,A
1281 0613 0C INC C
1282 0614 CD 057F CALL WRTVDP
1283 0617 21 F3C7 LD HL,GRPNAM
1284 061A 11 7F03 LD DE,7F03H
1285 061D 18 58 JR SETSCM
1286 061F INIMLT:
1287 ;
1288 ; Initialize VDP for multi-color mode
1289 ;
1290 061F CD 0577 CALL DISSCR
1291 0622 3E 03 LD A,3
1292 0624 32 FCAF LD (SCRMOD),A
1293 0627 2A F3D9 LD HL,(MLTPAT)
1294 062A 22 F926 LD (PATBAS),HL
1295 062D 2A F3D7 LD HL,(MLTATR)
1296 0630 22 F928 LD (ATRBAS),HL
1297 0633 2A F3D1 LD HL,(MLTNAM) ;Initialize name table
1298 0636 CD 07DF CALL SETWRT
1299 0639 11 0006 LD DE,6
1300 063C INIML1:
1301 063C 0E 04 LD C,4
1302 063E INIML2:
1303 063E 7A LD A,D
1304 063F 06 20 LD B,' '
1305 0641 INIML3:
1306 0641 D3 98 OUT (VDP.DRW),A
1307 0643 3C INC A
1308 0644 10 FB DJNZ INIML3
```

1309	0646	0D	DEC	C	
1310	0647	20 F5	JR	NZ,INIML2	
1311	0649	57	LD	D,A	
1312	064A	1D	DEC	E	
1313	064B	20 EF	JR	NZ,INIML1	
1314	064D	CD 07B9	CALL	CLSMLT	;Clear pattern table
1315	0650	CD 06BB	CALL	ERASPR	
1316	0653	CD 0659	CALL	SETMLT	;Actually set VDP mode
1317	0656	C3 0570	JP	ENASCR	
1318	0659		SETMLT:		
1319			;		
1320			;	Set VDP for multicolor mode	
1321			;		
1322	0659	3A F3DF	LD	A,(RG0SAV)	;Set register #0
1323	065C	E6 01	AND	1	
1324	065E	47	LD	B,A	
1325	065F	0E 00	LD	C,0	
1326	0661	CD 057F	CALL	WRTVDP	
1327	0664	3A F3E0	LD	A,(RG1SAV)	;Set register #1
1328	0667	E6 E7	AND	0E7H	
1329	0669	F6 08	OR	8	
1330	066B	47	LD	B,A	
1331	066C	0E 01	LD	C,1	
1332	066E	CD 057F	CALL	WRTVDP	
1333	0671	21 F3D1	LD	HL,MLTNAM	
1334	0674	11 0000	LD	DE,0	;Set mask pattern
1335	0677		SETSCM:		
1336	0677	01 0602	LD	BC,SETGRP	
1337	067A	CD 0690	CALL	SETREG	;Set name table
1338	067D	06 0A	LD	B,0AH	
1339	067F	7A	LD	A,D	

1340	0680	CD 0691	CALL	SETRG1	;Set color table
1341	0683	06 05	LD	B,5	
1342	0685	7B	LD	A,E	
1343	0686	CD 0691	CALL	SETRG1	;Set pattern table
1344	0689	06 09	LD	B,9	
1345	068B	CD 0690	CALL	SETREG	;Set sprite attribute table
1346	068E	06 05	LD	B,5	;Set sprite pattern table
1347	0690		SETREG:		
1348	0690	AF	XOR	A	
1349	0691		SETRG1:		
1350	0691	E5	PUSH	HL	
1351	0692	F5	PUSH	AF	
1352	0693	7E	LD	A,(HL)	
1353	0694	23	INC	HL	
1354	0695	66	LD	H,(HL)	
1355	0696	6F	LD	L,A	
1356	0697	AF	XOR	A	
1357	0698		SETRG2:		
1358	0698	29	ADD	HL,HL	
1359	0699	8F	ADC	A,A	
1360	069A	10 FC	DJNZ	SETRG2	
1361	069C	6F	LD	L,A	
1362	069D	F1	POP	AF	
1363	069E	B5	OR	L	
1364	069F	47	LD	B,A	
1365	06A0	CD 057F	CALL	WRTVDP	
1366	06A3	E1	POP	HL	
1367	06A4	23	INC	HL	
1368	06A5	23	INC	HL	
1369	06A6	0C	INC	C	
1370	06A7	C9	RET		

```

1371
1372      06A8          CLRSPR:
1373                  ;
1374                  ; Clear all sprites
1375                  ;
1376      06A8      3A F3E0          LD      A,(RGLSAV)          ;Set register #1
1377      06AB      47              LD      B,A
1378      06AC      0E 01          LD      C,1
1379      06AE      CD 057F        CALL    WRTVDP
1380      06B1      2A F926        LD      HL,(PATBAS)          ;Clear sprite pattern table
1381      06B4      01 0800        LD      BC,0800H            ;Length of sprite pattern table
1382      06B7      AF              XOR      A
1383      06B8      CD 0815        CALL    FILVRM
1384      06BB          ERASPR:
1385      06BB      3A F3E9        LD      A,(FORCLR)          ;Load foreground color (default) to [E]
1386      06BE      5F              LD      E,A
1387      06BF      2A F928        LD      HL,(ATRBAS)
1388      06C2      01 2000        LD      BC,2000H            ;Set number of sprite plane to [B]
1389      06C5          CLSPR2:
1390                  ; default sprite name to [C]
1391                  ;
1392      06C5      3E D1          LD      A,0D1H              ;Erase code (i.e. vertical position)
1393      06C7      CD 07CD        CALL    WRTVRM              ;Set vertical position
1394      06CA      23              INC      HL
1395      06CB      23              INC      HL
1396      06CC      79              LD      A,C                ;Load default sprite name
1397      06CD      CD 07CD        CALL    WRTVRM
1398      06D0      23              INC      HL
1399      06D1      0C              INC      C                ;Prepare for next
1400      06D2      3A F3E0        LD      A,(RGLSAV)
1401      06D5      0F              RRCA
  
```

1402	06D6	0F	RRCA		;16*16?
1403	06D7	30 03	JR	NC,CLSPR3	;No
1404	06D9	0C	INC	C	;Yes, C=C+4
1405	06DA	0C	INC	C	
1406	06DB	0C	INC	C	
1407	06DC		CLSPR3:		
1408	06DC	7B	LD	A,E	;Load default color
1409	06DD	CD 07CD	CALL	WRTVRM	
1410	06E0	23	INC	HL	
1411	06E1	10 E2	DJNZ	CLSPR2	
1412	06E3	C9	RET		
1413	06E4		CALPAT:		
1414			;		
1415	06E4	6F	LD	L,A	
1416	06E5	26 00	LD	H,0	
1417	06E7	29	ADD	HL,HL	;Assume 8 byte long
1418	06E8	29	ADD	HL,HL	
1419	06E9	29	ADD	HL,HL	
1420	06EA	CD 0704	CALL	GSPSIZ	;Check size of sprite
1421	06ED	FE 08	CP	8	
1422	06EF	28 02	JR	Z,GSPAD1	;Good assumption
1423	06F1	29	ADD	HL,HL	;32 byte long sprite
1424	06F2	29	ADD	HL,HL	
1425	06F3		GSPAD1:		
1426	06F3	EB	EX	DE,HL	
1427	06F4	2A F926	LD	HL,(PATBAS)	;Get base address of sprite pattern table
1428	06F7	19	ADD	HL,DE	;Form destination/source address
1429	06F8	C9	RET		
1430	06F9		CALATR:		
1431			;		
1432	06F9	6F	LD	L,A	;Get plane number to [L]

```
1433      06FA      26 00          LD      H,0
1434      06FC      29          ADD     HL,HL          ;Sprite attribute consists of 4 bytes
1435      06FD      29          ADD     HL,HL
1436      06FE      EB          EX      DE,HL
1437      06FF      2A F928      LD      HL,(ATRBAS)      ;Load base address
1438      0702      19          ADD     HL,DE          ;Calculate target address
1439      0703      C9          RET
1440      0704          GSPSIZ:
1441          ;
1442          ; Get sprite size
1443          ;
1444      0704      3A F3E0      LD      A,(RGLSAV)
1445      0707      0F          RRCA
1446      0708      0F          RRCA
1447      0709      3E 08      LD      A,8          ;Assume 8 byte long
1448      070B      D0          RET      NC          ;Good assumption
1449      070C      3E 20      LD      A,32          ;32 byte long sprite
1450      070E      C9          RET
```

```
1451
1452      070F      LDIRMV:
1453      ;
1454      070F      CD 07EC      CALL      SETRD
1455      0712      E3          EX        (SP),HL
1456      0713      E3          EX        (SP),HL
1457      0714      LDIMV1:
1458      0714      DB 98      IN        A,(VDP.DRW)
1459      0716      12      LD        (DE),A
1460      0717      13      INC       DE
1461      0718      0B      DEC       BC
1462      0719      79      LD        A,C
1463      071A      B0      OR        B
1464      071B      20 F7     JR        NZ,LDIMV1
1465      071D      C9      RET
1466      071E      INIPAT:
1467      ;
1468      ; Set default character pattern
1469      ;
1470      071E      CD FDC7      CALL      H.INIP
1471      0721      2A F924      LD        HL,(CGPBAS)      ;Get target address of VRAM
1472      0724      CD 07DF      CALL      SETWRT          ;Set VDP for write operation
1473      0727      3A F91F      LD        A,(CGPNT)        ;Get slot # of character genarator table
1474      072A      2A F920      LD        HL,(CGPNT+1)     ;Get address of character genarator table
1475      072D      01 0800      LD        BC,0800H        ;Load total length
1476      0730      F5          PUSH     AF                ;Save source slot
1477      0731      INIPT1:
1478      0731      F1          POP      AF                ;Restore source slot
1479      0732      F5          PUSH     AF                ;Save source slot
1480      0733      C5          PUSH     BC                ;Save counter
1481      0734      F3          DI
```



```
1482      0735      CD 01B6          CALL      RDSLT          ;Read from specified slot
1483      0738      FB              EI
1484      0739      C1              POP       BC              ;Restore counter
1485      073A      D3 98          OUT       (VDP.DRW),A
1486      073C      23              INC       HL              ;Bump character source pointer
1487      073D      0B              DEC       BC
1488      073E      79              LD        A,C
1489      073F      B0              OR        B
1490      0740      20 EF          JR         NZ,INIPT1
1491      0742      F1              POP       AF              ;Discard stack
1492      0743      C9              RET
1493      0744          LDIRVM:
1494          ;
1495      0744      EB              EX        DE,HL
1496      0745      CD 07DF          CALL      SETWRT
1497      0748          LDIVM1:
1498      0748      1A              LD        A,(DE)
1499      0749      D3 98          OUT       (VDP.DRW),A
1500      074B      13              INC       DE
1501      074C      0B              DEC       BC
1502      074D      79              LD        A,C
1503      074E      B0              OR        B
1504      074F      20 F7          JR         NZ,LDIVM1
1505      0751      C9              RET
1506      0752          GETPAT:
1507          ;
1508          ; Get pattern corresponding to ASCII code in [A]
1509          ;
1510          ; Pattern is returned to 8 byte work area (PATWRK). Entered
1511          ; by GRPPRT (print a character to graphic screen) subroutine.
1512          ;
```

Address	Op Code	Op	Op2	Op3	Op4	Comments
1513						; All registers are completely destroyed
1514						;
1515	0752	26	00	LD	H,0	;Prepare for calculation
1516	0754	6F		LD	L,A	
1517	0755	29		ADD	HL,HL	
1518	0756	29		ADD	HL,HL	
1519	0757	29		ADD	HL,HL	
1520	0758	EB		EX	DE,HL	
1521	0759	2A	F920	LD	HL,(CGPNT+1)	
1522	075C	19		ADD	HL,DE	;[HL]:=source address
1523	075D	11	FC40	LD	DE,PATWRK	;Load destination address
1524	0760	06	C8	LD	B,8	;Load total length
1525	0762	3A	F91F	LD	A,(CGPNT)	;Get slot # of character genarator table
1526	0765					GTPAT1:
1527	0765	F5		PUSH	AF	;Save source slot
1528	0766	E5		PUSH	HL	;Save source address
1529	0767	D5		PUSH	DE	;Save destination address
1530	0768	C5		PUSH	BC	;Save counter
1531	0769	CD	01B6	CALL	RDSLTL	;Read from specified slot
1532	076C	FB		EI		
1533	076D	C1		POP	BC	;Restore counter
1534	076E	D1		POP	DE	;Restore destination address
1535	076F	E1		POP	HL	;Restore source address
1536	0770	12		LD	(DE),A	
1537	0771	13		INC	DE	;Bump destination pointer
1538	0772	23		INC	HL	;Bump character source pointer
1539	0773	F1		POP	AF	;Restore source slot
1540	0774	10	EF	DJNZ	GTPAT1	
1541	0776	C9		RET		
1542	0777					CLSSUB:
1543						;

1544	0777	CD 0B9F	CALL	CHKSCR	;Check current screen mode
1545	077A	28 25	JR	Z,CLSHRS	;Hires
1546	077C	30 3B	JR	NC,CLSMLT	;Multi-color
1547	077E			CLRTXT:	
1548				;	
1549				; Clear screen (text mode)	
1550				;	
1551	077E	3A FCAF	LD	A,(SCRMOD)	
1552	0781	A7	AND	A	
1553	0782	2A F922	LD	HL,(NAMBAS)	;Set address for write
1554	0785	01 03C0	LD	BC,03C0H	;40 * 24
1555	0788	28 03	JR	Z,CLRTX1	
1556	078A	01 0300	LD	BC,0300H	;32 * 24
1557	078D			CLRTX1:	
1558	078D	3E 20	LD	A,' '	;Fill space character code
1559	078F	CD 0815	CALL	FILVRM	
1560	0792	CD 0A7F	CALL	CSHOME	;Set cursor at home position
1561	0795	21 FBB2	LD	HL,LINTTB	;Say all lines are terminated
1562	0798	06 18	LD	B,18H	
1563	079A			CLRTX2:	
1564	079A	70	LD	(HL),B	;Load non 0 value
1565	079B	23	INC	HL	
1566	079C	10 FC	DJNZ	CLRTX2	
1567	079E	C3 0B26	JP	FNKSB	
1568	07A1			CLSHRS:	
1569				;	
1570	07A1	CD 0832	CALL	CHGBDR	;Set border color
1571	07A4	01 1800	LD	BC,1800H	;Initialize color
1572	07A7	C5	PUSH	BC	;Save this for future use
1573	07A8	2A F3C9	LD	HL,(GRPCOL)	
1574	07AB	3A F3EA	LD	A,(BAKCLR)	;Load background color

1575	07AE	CD 0815	CALL	FILVRM	
1576	07B1	2A F3CB	LD	HL,(GRPCGP)	
1577	07B4	C1	POP	BC	;Load 6144
1578	07B5	AF	XOR	A	
1579	07B6		JFLVRM:		
1580	07B6	C3 0815	JP	FILVRM	
1581	07B9		CLSMLT:		
1582			;		
1583	07B9	CD 0832	CALL	CHGBDR	;Set border color
1584	07BC	21 F3EA	LD	HL,BAKCLR	;Set all pixels to background color
1585	07BF	7E	LD	A,(HL)	
1586	07C0	87	ADD	A,A	
1587	07C1	87	ADD	A,A	
1588	07C2	87	ADD	A,A	
1589	07C3	87	ADD	A,A	
1590	07C4	B6	OR	(HL)	
1591	07C5	2A F3D5	LD	HL,(MLTCGP)	;Set up address for write
1592	07C8	01 0600	LD	BC,0600H	
1593	07CB	18 E9	JR	JFLVRM	;Clear sprites (except sprite pattern)

```
1594
1595      07CD      WRTVRM:
1596      ;
1597      ; Write a byte to VRAM
1598      ;
1599      07CD      F5      PUSH      AF      ;Save data to be written
1600      07CE      CD 07DF      CALL      SETWRT
1601      07D1      E3      EX      (SP),HL
1602      07D2      E3      EX      (SP),HL
1603      07D3      F1      POP      AF
1604      07D4      D3 98      OUT      (VDP.DRW),A
1605      07D6      C9      RET
1606      07D7      RDVRM:
1607      ;
1608      ; Read a byte from VRAM
1609      ;
1610      07D7      CD 07EC      CALL      SETRD
1611      07DA      E3      EX      (SP),HL
1612      07DB      E3      EX      (SP),HL
1613      07DC      DB 98      IN      A,(VDP.DRW)
1614      07DE      C9      RET
1615      07DF      SETWRT:
1616      ;
1617      ; Set address for write to VDP
1618      ;
1619      ; Address is passed to HL
1620      ;
1621      07DF      7D      LD      A,L
1622      07E0      F3      DI
1623      07E1      D3 99      OUT      (VDP.CW),A
1624      07E3      7C      LD      A,H
```

```

1625      07E4      E6 3F          AND      00111111B
1626      07E6      F6 40          OR       01000000B      ;For write, set bit 6 high
1627      07E8      D3 99          OUT      (VDP.CW),A
1628      07EA      FB            EI
1629      07EB      C9            RET
1630      07EC
1631                      SETRD:
1632                      ;
1633                      ; Set address for read from VDP
1634                      ;
1635                      ; Address is passed to HL
1636                      ;
1636      07EC      7D            LD        A,L
1637      07ED      F3            DI
1638      07EE      D3 99          OUT      (VDP.CW),A
1639      07F0      7C            LD        A,H
1640      07F1      E6 3F          AND      00111111B
1641      07F3      D3 99          OUT      (VDP.CW),A
1642      07F5      FB            EI
1643      07F6      C9            RET
1644      07F7
1645                      CHGCLR:
1646                      ;
1647                      ; CHGCLR - changes foreground, background, and border color
1648                      ;
1648      07F7      3A FCAF          LD        A,(SCRMOD)      ;Are we in text mode
1649      07FA      3D            DEC        A
1650      07FB      FA 0824          JP        M,CHCLTX      ;Yes, change color in 40*24 text mode
1651      07FE      F5            PUSH     AF
1652      07FF      CD 0832          CALL    CHGBDR      ;Change border color for all
1653      0802      F1            POP       AF
1654      0803      C0            RET      NZ      ;No
1655      0804      3A F3E9          LD        A,(FORCLR)      ;We're in 32*24 text mode

```

1656	0807	87		ADD	A,A
1657	0808	87		ADD	A,A
1658	0809	87		ADD	A,A
1659	080A	87		ADD	A,A
1660	080B	21	F3EA	LD	HL,BAKCLR
1661	080E	B6		OR	(HL)
1662	080F	2A	F3BF	LD	HL,(T32COL)
1663	0812	01	0020	LD	BC,20H
1664	0815			FILVRM:	
1665	0815	F5		PUSH	AF
1666	0816	CD	07DF	CALL	SETWRT
1667	0819			FLVRM1:	
1668	0819	F1		POP	AF
1669	081A	D3	98	OUT	(VDP.DRW),A
1670	081C	F5		PUSH	AF
1671	081D	0B		DEC	BC
1672	081E	79		LD	A,C
1673	081F	B0		OR	B
1674	0820	20	F7	JR	NZ,FLVRM1
1675	0822	F1		POP	AF
1676	0823	C9		RET	
1677	0824			CHCLTX:	
1678				;	
1679	0824	3A	F3E9	LD	A,(FORCLR)
1680	0827	87		ADD	A,A
1681	0828	87		ADD	A,A
1682	0829	87		ADD	A,A
1683	082A	87		ADD	A,A
1684	082B	21	F3EA	LD	HL,BAKCLR
1685	082E	B6		OR	(HL)
1686	082F	47		LD	B,A

1687	0830	18 03	JR	CHGBD1	
1688	0832		CHGBDR:		
1689			;		
1690	0832	3A F3EB	LD	A,(BDRCLR)	;Get border color
1691	0835		CHGBD1:		
1692	0835	47	LD	B,A	
1693	0836	0E 07	LD	C,7	
1694	0838	C3 057F	JP	WRTVDP	


```
1695
1696      083B      TOTEXT:
1697      ;
1698      ; TOTEXT - Force screen to text mode
1699      ;
1700      083B      CD 0B9F      CALL      CHKSCR      ;Check current screen mode
1701      083E      D8          RET        C          ;We're in text mode
1702      083F      3A FCB0      LD         A,(OLDSCR)
1703      0842      CD FDBD      CALL      H.TOTE
1704      0845      C3 084F      JP         CHGMOD      ;No, change to text mode then
1705      0848      CLS:
1706      ;
1707      ; CLS - clears screen
1708      ;
1709      0848      C0          RET        NZ          ;Statement not ending
1710      0849      E5          PUSH       HL          ;Save text pointer
1711      084A      CD 0777      CALL      CLSSUB
1712      084D      E1          POP        HL          ;Restore text pointer
1713      084E      C9          RET
1714      084F      CHGMOD:
1715      ;
1716      ; CHGMOD - changes mode of screen
1717      ;
1718      084F      3D          DEC        A          ;Change to what mode
1719      0850      FA 050E      JP         M,INITXT      ;To text mode
1720      0853      CA 0538      JP         Z,INIT32
1721      0856      3D          DEC        A
1722      0857      CA 05D2      JP         Z,INIGRP      ;To hires mode
1723      085A      C3 061F      JP         INIMLT      ;To multicolor mode
1724      SUBTTL - MSXIO - Some entry points
```

```

1725
1726      085D      LPTOUT:
1727      ;
1728      ; Output a character to printer
1729      ;
1730      085D      CD FFB6      CALL      H.LPTO
1731      0860      F5          PUSH      AF          ;Save character to output
1732      0861      CHPLP1:
1733      0861      CD 046F      CALL      BREAKX      ;Check if aborted
1734      0864      38 12      JR        C,LPTABO
1735      0866      CD 0884      CALL      LPTSTT
1736      0869      28 F6      JR        Z,CHPLP1      ;No
1737      086B      F1          POP       AF          ;Restore character
1738      086C      CHPLP2:
1739      086C      F5          PUSH      AF          ;Save it again
1740      086D      D3 91      OUT       (LPT.DW),A      ;Send to output port
1741      086F      AF          XOR       A          ;Generate strobe
1742      0870      D3 90      OUT       (LPT.SB),A
1743      0872      3D          DEC       A
1744      0873      D3 90      OUT       (LPT.SB),A
1745      0875      F1          POP       AF          ;Restore data output
1746      0876      A7          AND       A
1747      0877      C9          RET
1748      0878      LPTABO:
1749      ;
1750      0878      AF          XOR       A          ;Reset carriage position
1751      0879      32 F415      LD        (LPTPOS),A
1752      087C      3E 0D      LD        A,0DH      ;Send CR even if LPT not active
1753      087E      CD 086C      CALL      CHPLP2
1754      0881      F1          POP       AF
1755      0882      37          SCF
  
```

```

1756      0883      C9                      RET
1757      0884                      LPTSTT:
1758                                ;
1759      0884      CD FFBB                  CALL      H.LPTS
1760      0887      DB 90                    IN        A,(90H)          ;LSB is 0 if ready
1761      0889      0F                      RRCA
1762      088A      0F                      RRCA
1763      088B      3F                      CCF
1764      088C      9F                      SBC      A,A
1765      088D      C9                      RET                          ;No
1766      088E                      POSIT:
1767                                ;
1768                                ; Position cursor to specified position
1769                                ;
1770      088E      3E 1B                    LD        A,1BH
1771      0890      DF                      RST      18H                ;OUTCHR
1772      0891      3E 59                    LD        A,'Y'
1773      0893      DF                      RST      18H
1774      0894      7D                      LD        A,L
1775      0895      C6 1F                    ADD      A,1FH            ;= ' ' - 1
1776      0897      DF                      RST      18H
1777      0898      7C                      LD        A,H
1778      0899      C6 1F                    ADD      A,1FH
1779      089B      DF                      RST      18H
1780      089C      C9                      RET
1781      089D                      CNVCHR:
1782                                ;
1783                                ; Convert character code
1784                                ;
1785      089D      E5                      PUSH      HL
1786      089E      F5                      PUSH      AF

```

1787	089F	21 FCA6	LD	HL,GRPHED	;Preceeded by a header byte
1788	08A2	AF	XOR	A	
1789	08A3	BE	CP	(HL)	
1790	08A4	77	LD	(HL),A	;Clear this since seen
1791	08A5	28 0D	JR	Z,CNVCH3	;No
1792	08A7	F1	POP	AF	
1793	08A8	D6 40	SUB	01000000B	;Get rid of offset
1794	08AA	FE 20	CP	' '	;Valid range
1795	08AC	38 04	JR	C,CNVCH2	;Yes
1796	08AE	C6 40	ADD	A,01000000B	;Compensate value
1797	08B0		CNVCH1:		
1798	08B0	BF	CP	A	;Set Z flag
1799	08B1	37	SCF		;Make sure carry is cleared
1800	08B2		CNVCH2:		
1801	08B2	E1	POP	HL	
1802	08B3	C9	RET		
1803	08B4		CNVCH3:		
1804			;		
1805	08B4	F1	POP	AF	
1806	08B5	FE 01	CP	1	;Graphic header
1807	08B7	20 F7	JR	NZ,CNVCH1	;No, do not modify
1808	08B9	77	LD	(HL),A	;Set GRPHED flag
1809	08BA	E1	POP	HL	;Carry is clear indicating one more byte is
1810	08BB	C9	RET		;required
1811			SUBTTL	- MSXIO -	Output a character to CRT

```
1812
1813      08BC      CHPUT:
1814      ;
1815      08BC      E5      PUSH      HL
1816      08BD      D5      PUSH      DE
1817      08BE      C5      PUSH      BC
1818      08BF      F5      PUSH      AF
1819      08C0      CD FDA4      CALL      H.CHPU
1820      08C3      CD 0B9F      CALL      CHKSCR      ;Are we in text mode
1821      08C6      30 12      JR      NC,POPALL      ;No, ignore this
1822      08C8      CD 0A2E      CALL      CKERCS      ;Erase old cursor if cursor enabled
1823      08CB      F1      POP      AF
1824      08CC      F5      PUSH      AF
1825      08CD      CD 08DF      CALL      CHPUT1
1826      08D0      CD 09E1      CALL      CKDPCS      ;Display new cursor if cursor enabled
1827      08D3      3A F3DD      LD      A,(CSRX)
1828      08D6      3D      DEC      A
1829      08D7      32 F661      LD      (TTYPOS),A
1830      08DA      POPALL:
1831      08DA      F1      POP      AF
1832      08DB      PBDHRT:
1833      08DB      C1      POP      BC
1834      08DC      D1      POP      DE
1835      08DD      E1      POP      HL
1836      08DE      C9      RET
1837      08DF      CHPUT1:
1838      ;
1839      08DF      CD 089D      CALL      CNVCHR      ;Convert character code
1840      08E2      D0      RET      NC      ;Was a graphic header, wait for next
1841      08E3      4F      LD      C,A      ;Save character code in [C]
1842      08E4      20 0D      JR      NZ,CHPUT3      ;Converted code, send as is
```

```

1843    08E6    21 FCA7          LD      HL,ESCCNT
1844    08E9    7E              LD      A,(HL)          ;Are we executing escape sequence
1845    08EA    A7              AND     A              ;
1846    08EB    C2 098F         JP      NZ,INESC       ;Yes
1847    08EE    79              LD      A,C            ;Restore character
1848    08EF    FE 20           CP      ' '            ;Control code
1849    08F1    38 21           JR      C,CNTPUT        ;Yes
1850    08F3                                CHPUT3:
1851    08F3    2A F3DC         LD      HL,(CSRY)
1852    08F6    FE 7F           CP      7FH            ;Rubout
1853    08F8    CA 0AE3         JP      Z,RUBOUT        ;Yes
1854    08FB    CD 0BE6         CALL    PUTVRM         ;Convert to raw code and write to VRAM
1855    08FE    CD 0A44         CALL    RIGHT         ;Advance cursor
1856    0901    C0              RET     NZ             ;All done if not wrapped to next line
1857    0902    AF              XOR     A
1858    0903    CD 0C2B         CALL    SETTRM        ;Underminate this line
1859    0906    26 01           LD      H,l            ;Go to start of the next line
1860    0908                                LF:
1861                                ;
1862                                ; Line feed
1863                                ;
1864    0908    CD 0A61         CALL    DOWN            ;Down cursor
1865    090B    C0              RET     NZ             ;Exit if not at bottom
1866    090C    CD 0A69         CALL    STOCSSR
1867    090F    2E 01           LD      L,l            ;L:=window top line
1868    0911    C3 0A88         JP      DELLNO         ;Scroll up by deleting the first line
1869    0914                                CNTPUT:
1870                                ;
1871                                ; Following control codes are supported
1872                                ;
1873                                ; 7 Bell

```

```
1874                ; 8 Back space
1875                ; 9 Tab
1876                ; 10 Line feed
1877                ; 11 Cursor home
1878                ; 12 Clear screen
1879                ; 13 Carriage return
1880                ;
1881                ; 27 Enter escape sequence
1882                ; 28 Cursor right
1883                ; 29 Cursor left
1884                ; 30 Cursor up
1885                ; 31 Cursor down
1886                ;
1887    0914    21 092D                LD    HL,JMPBC
1888    0917    0E 0C                LD    C,0CH
1889    0919                INDJMP:
1890    0919    23                INC    HL
1891    091A    23                INC    HL
1892    091B    A7                AND    A                ;Make sure carry is cleared
1893    091C    0D                DEC    C
1894    091D    F8                RET    M                ;Undefined function
1895    091E    BE                CP     (HL)            ;Found?
1896    091F    23                INC    HL
1897    0920    20 F7            JR     NZ,INDJMP        ;No
1898    0922    4E                LD     C,(HL)          ;Get routine address in BC
1899    0923    23                INC    HL
1900    0924    46                LD     B,(HL)          ;
1901    0925    2A F3DC            LD     HL,(CSRY)       ;Jump to each routine with cursor pos
1902    0928    CD 092D            CALL  JMPBC
1903    092B    AF                XOR    A                ;Tell screen editor not to echo this character
1904    092C    C9                RET
```

```
1905      092D      JMPBC:
1906      ;
1907      092D      C5      PUSH      BC
1908      092E      C9      RET
1909      ;
1910      ;      Function dispatch table
1911      ;
1912      092F      CNTTBL:
1913      092F      07      DB      7      ;Beep
1914      0930      1113    DW      BEEP
1915      0932      08      DB      8      ;Back space
1916      0933      0A4C    DW      BS
1917      0935      09      DB      9      ;Tabulation
1918      0936      0A71    DW      TAB
1919      0938      0A      DB      10     ;Line feed
1920      0939      0908    DW      LF
1921      093B      0B      DB      11     ;Home
1922      093C      0A7F    DW      CSHOME
1923      093E      0C      DB      12     ;Clear
1924      093F      077E    DW      CLRTXT
1925      0941      0D      DB      13     ;Carriage return
1926      0942      0A81    DW      CR
1927      0944      1B      DB      27     ;Enter escape sequence
1928      0945      0989    DW      ENTESC
1929      0947      1C      DB      28     ;Cursor right
1930      0948      0A5B    DW      ADVCUR
1931      094A      1D      DB      29     ;Cursor left
1932      094B      0A4C    DW      BS
1933      094D      1E      DB      30     ;Cursor up
1934      094E      0A57    DW      UP
1935      0950      1F      DB      31     ;Cursor down
```


(MSX ROM BASIC BIOS) Macro-80
- MSXIO - Output a character to CRT

3.44

01-Jan-85

PAGE 29-4

73

1936 0951 0A61
1937

DW DOWN

SUBTTL - MSXIO - Escape sequence handler

1938					
1939	0953		ESCTBL:		
1940	0953	6A	DB	"j"	;Clear screen
1941	0954	077E	DW	CLRTXT	
1942	0956	45	DB	"E"	;Clear screen
1943	0957	077E	DW	CLRTXT	; To maintain compatibility with VT52
1944	0959	4B	DB	"K"	;Erase to end-of-line
1945	095A	0AEE	DW	EOL	
1946	095C	4A	DB	"J"	;Erase to end-of-page
1947	095D	0B05	DW	EOP	
1948	095F	6C	DB	"l"	;Erase entire line
1949	0960	0AEC	DW	ELN	
1950	0962	4C	DB	"L"	;Insert a line
1951	0963	0AB4	DW	ILN	
1952	0965	4D	DB	"M"	;Delete a line
1953	0966	0A85	DW	DLN	
1954	0968	59	DB	"Y"	;Locate cursor
1955	0969	0986	DW	LOC	
1956	096B	41	DB	"A"	;Cursor up
1957	096C	0A57	DW	UP	
1958	096E	42	DB	"B"	;Cursor down
1959	096F	0A61	DW	DOWN	
1960	0971	43	DB	"C"	;Cursor right
1961	0972	0A44	DW	RIGHT	
1962	0974	44	DB	"D"	;Cursor left
1963	0975	0A55	DW	LEFT	
1964	0977	48	DB	"H"	;Cursor home
1965	0978	0A7F	DW	CSHOME	
1966	097A	78	DB	"x"	;Set modes
1967	097B	0980	DW	SETMOD	
1968	097D	79	DB	"y"	;Reset modes

```
1969      097E      0983              DW      RSTMOD
1970      0980              SETMOD:
1971                          ;
1972                          ; Function dispatch table
1973                          ;
1974      0980      3E 01              LD      A,1
1975      0982      01              DB      1
1976      0983              RSTMOD:
1977      0983      3E 02              LD      A,2
1978      0985      01              DB      1
1979      0986              LOC:
1980      0986      3E 04              LD      A,4
1981      0988      01              DB      1
1982      0989              ENTESC:
1983      0989      3E FF              LD      A,0FFH
1984      098B      32 FCA7           LD      (ESCCNT),A
1985      098E      C9              RET

; Say row is expected next
; 'LXI B' instruction

; Tell him we're in escape sequence
```

```

1986
1987      098F      INESC:
1988      ;
1989      098F      F2 099D      JP      P,INESC1      ;Arguments expected
1990      0992      36 00      LD      (HL),0      ;Exit from escape sequence
1991      0994      79      LD      A,C      ;Restore character
1992      0995      21 0951      LD      HL,ESCTBL-2
1993      0998      0E 0F      LD      C,0FH      ;Number of ESC handler entries
1994      099A      C3 0919      JP      INDJMP
1995      099D      INESC1:
1996      ;
1997      099D      3D      DEC      A      ;Set modes?
1998      099E      28 1E      JR      Z,GOSET      ;Yes
1999      09A0      3D      DEC      A      ;Reset modes?
2000      09A1      28 25      JR      Z,GORSET
2001      09A3      3D      DEC      A
2002      09A4      77      LD      (HL),A      ;Update ESCCNT
2003      09A5      3A F3B0      LD      A,(LINLEN)      ;Assume column expected
2004      09A8      11 F3DD      LD      DE,CSRX      ;
2005      09AB      28 06      JR      Z,INESC2      ;Column expected
2006      09AD      36 03      LD      (HL),3
2007      09AF      CD 0C32      CALL   GETLEN      ;Row expected
2008      09B2      1B      DEC      DE      ;Point CSRY
2009      09B3      INESC2:
2010      09B3      47      LD      B,A      ;Get max limit in B
2011      09B4      79      LD      A,C      ;Restore character
2012      09B5      D6 20      SUB     ' '      ;0-xx
2013      09B7      B8      CP      B
2014      09B8      3C      INC      A
2015      09B9      12      LD      (DE),A
2016      09BA      D8      RET      C      ;Legal value

```

```
2017      09BB      78              LD      A,B              ;Substitute by possible largest value
2018      09BC      12              LD      (DE),A
2019      09BD      C9              RET
2020      09BE              GOSET:
2021              ;
2022              ; Set various modes
2023              ;
2024      09BE      77              LD      (HL),A              ;Exit from escape sequence
2025      09BF      79              LD      A,C              ;Restore character
2026      09C0      D6 34          SUB      '4'              ;Block cursor?
2027      09C2      28 0B          JR      Z,STSTYL              ;Yes
2028      09C4      3D              DEC      A              ;Cursor off?
2029      09C5      28 0F          JR      Z,STCSSW              ;Yes, reset cursor-enable switch
2030      09C7      C9              RET              ;Unimplemented feature
2031      09C8              GORSET:
2032              ;
2033              ; Reset various modes
2034              ;
2035      09C8      77              LD      (HL),A              ;Exit from escape sequence
2036      09C9      79              LD      A,C              ;Restore character
2037      09CA      D6 34          SUB      '4'              ;Underscore cursor?
2038      09CC      20 05          JR      NZ,RSET10              ;No, try next
2039      09CE      3C              INC      A
2040      09CF              STSTYL:
2041      09CF      32 FCAA          LD      (CSTYLE),A
2042      09D2      C9              RET
2043      09D3              RSET10:
2044              ;
2045      09D3      3D              DEC      A              ;Cursor on?
2046      09D4      C0              RET      NZ              ;No, unimplemented feature
2047      09D5      3C              INC      A
```

```
2048      09D6      STCSSW:
2049      09D6      32 FCA9      LD      (CSRSW),A
2050      09D9      C9          RET
2051      09DA      CKDPC0:
2052              ;
2053              ; Display cursor if disabled
2054              ;
2055      09DA      3A FCA9      LD      A,(CSRSW)
2056      09DD      A7          AND      A
2057      09DE      C0          RET      NZ
2058      09DF      18 05      JR      DSPCSR
2059      09E1      CKDPCS:
2060              ;
2061              ; Display cursor if enabled
2062              ;
2063      09E1      3A FCA9      LD      A,(CSRSW)
2064      09E4      A7          AND      A
2065      09E5      C8          RET      Z
2066      09E6      DSPCSR:
2067              ;
2068              ; Display a cursor
2069              ;
2070      09E6      CD FDA9      CALL     H.DSPC
2071      09E9      CD 0B9F      CALL     CHKSCR
2072      09EC      D0          RET      NC
2073      09ED      2A F3DC      LD      HL,(CSRY)      ;Get current cursor position
2074      09F0      E5          PUSH     HL              ;Save it for future use
2075      09F1      CD 0BD8      CALL     GETVRM        ;Get a raw character at cursor
2076      09F4      32 FBCC      LD      (CODSAV),A    ;Remember this code
2077      09F7      6F          LD      L,A            ;Then read pattern for this code
2078      09F8      26 00      LD      H,0
```

```

2079      09FA      29      ADD      HL,HL      ; [A] * 8
2080      09FB      29      ADD      HL,HL
2081      09FC      29      ADD      HL,HL
2082      09FD      EB      EX      DE,HL
2083      09FE      2A F924  LD      HL,(CGPBAS)
2084      0A01      E5      PUSH     HL
2085      0A02      19      ADD      HL,DE
2086      0A03      CD 0BA5  CALL     GET8B
2087      0A06      21 FC1F  LD      HL,BUFEND+7      ;Make a complement of this pattern
2088      0A09      06 08      LD      B,8      ;Assume full reverse cursor
2089      0A0B      3A FCAA  LD      A,(CSTYLE)
2090      0A0E      A7      AND      A
2091      0A0F      28 02      JR      Z,DSPCS1      ;Good assumption
2092      0A11      06 03      LD      B,3      ;No, reverse bottom 3 lines only
2093      0A13      DSPCS1:
2094      0A13      7E      LD      A,(HL)
2095      0A14      2F      CPL
2096      0A15      77      LD      (HL),A
2097      0A16      2B      DEC      HL
2098      0A17      10 FA      DJNZ     DSPCS1
2099      0A19      E1      POP      HL      ;Assign this pattern to 255
2100      0A1A      01 07F8  LD      BC,07F8H
2101      0A1D      09      ADD      HL,BC
2102      0A1E      CD 0BBE  CALL     PUT8B
2103      0A21      E1      POP      HL      ;Restore cursor position
2104      0A22      0E FF      LD      C,0FFH      ;Get code for cursor
2105      0A24      C3 0BE6  JP      PUTVRM      ;Set it at cursor position
2106      0A27      CKERC0:
2107      ;
2108      ; Erase cursor if disabled
2109      ;

```

```
2110      0A27      3A FCA9          LD      A,(CSRSW)
2111      0A2A      A7              AND      A
2112      0A2B      C0              RET      NZ
2113      0A2C      18 05          JR      ERACSR
2114      0A2E                      CKERCS:
2115                                ;
2116                                ; Erase a cursor if enabled
2117                                ;
2118      0A2E      3A FCA9          LD      A,(CSRSW)
2119      0A31      A7              AND      A
2120      0A32      C8              RET      Z
2121      0A33                      ERACSR:
2122                                ;
2123                                ; Erase cursor
2124                                ;
2125      0A33      CD FDAE          CALL     H.ERAC
2126      0A36      CD 0B9F          CALL     CHKSCR
2127      0A39      D0              RET      NC
2128      0A3A      2A F3DC          LD      HL,(CSRY)
2129      0A3D      3A FBCC          LD      A,(CODSAV)      ;Get old code
2130      0A40      4F              LD      C,A
2131      0A41      C3 0BE6          JP      PUTVRM          ;Restore old code
2132                                ;
2133      SUBTTL - MSXIO - Cursor movements
```



```
2134
2135      0A44      RIGHT:
2136      ;
2137      ; Cursor right
2138      ;
2139      0A44      3A F3B0      LD      A,(LINLEN)
2140      0A47      BC          CP      H          ;Are we at the right-end of line?
2141      0A48      C8          RET      Z          ;Yes, return with Z flag
2142      0A49      24          INC      H          ;Go to next column
2143      0A4A      18 1D      JR      STOCSR
2144      0A4C      BS:
2145      ;
2146      ; Back space
2147      ;
2148      0A4C      CD 0A55      CALL     LEFT
2149      0A4F      C0          RET      NZ          ;Not at left-end
2150      0A50      3A F3B0      LD      A,(LINLEN)
2151      0A53      67          LD      H,A
2152      0A54      11          DB      11H          ;'LXI D,' instruction
2153      0A55      LEFT:
2154      ;
2155      ; Cursor left
2156      ;
2157      0A55      25          DEC      H          ;Are we at the left-end of line?
2158      0A56      3E          DB      3EH          ;'MVI A,' instruction
2159      0A57      UP:
2160      ;
2161      ; Cursor up
2162      ;
2163      0A57      2D          DEC      L          ;Are we at the top of any window?
2164      0A58      C8          RET      Z          ;Yes, return with Z flag
```

```
2165      0A59      18 0E              JR      STOCSR
2166      0A5B              ADVCUR:
2167              ;
2168              ; Advance cursor
2169              ;
2170      0A5B      CD 0A44              CALL    RIGHT
2171      0A5E      C0                  RET      NZ
2172      0A5F      26 01              LD      H,1
2173      0A61              DOWN:
2174              ;
2175              ; Cursor down
2176              ;
2177      0A61      CD 0C32              CALL    GETLEN      ;Get an actual bottom of screen
2178      0A64      BD                  CP      L              ;Are we at the bottom of screen?
2179      0A65      C8                  RET      Z              ;Yes, return with Z flag
2180      0A66      38 05              JR      C,DOWN1      ;We're below screen bottom
2181      0A68      2C                  INC     L              ;Go to next line
2182      0A69              STOCSR:
2183      0A69      22 F3DC              LD      (CSRY),HL
2184      0A6C      C9                  RET
2185      0A6D              DOWN1:
2186              ;
2187      0A6D      2D                  DEC     L
2188      0A6E      AF                  XOR     A
2189      0A6F      18 F8              JR      STOCSR
2190      0A71              TAB:
2191              ;
2192              ; Tabulation
2193              ;
2194      0A71      3E 20              LD      A,' '
2195      0A73      CD 08DF              CALL    CHPUT1
```

```
2196      0A76      3A F3DD          LD      A,(CSRX)
2197      0A79      3D              DEC      A
2198      0A7A      E6 07          AND      7
2199      0A7C      20 F3          JR       NZ,TAB
2200      0A7E      C9              RET
2201      0A7F                      CSHOME:
2202                                ;
2203                                ; Cursor home
2204                                ;
2205      0A7F      2E 01          LD      L,1
2206      0A81                      CR:
2207                                ;
2208                                ; Carriage return
2209                                ;
2210      0A81      26 01          LD      H,1          ;CR only, not new-line
2211      0A83      18 E4          JR       STOCSR
2212                                ;
2213      SUBTTL - MSXIO - Line insert and delete of CRT
```

```

2214
2215      0A85      DLN:
2216                ;
2217                ; Delete a line specified by [L]
2218                ;
2219                ; Cursor should be set at the top of line
2220                ;
2221      0A85      CD 0A81      CALL      CR
2222      0A88      DELLN0:
2223      0A88      CD 0C32      CALL      GETLEN      ;Get an actual height of screen
2224      0A8B      95          SUB       L
2225      0A8C      D8          RET       C            ;Something is wrong
2226      0A8D      CA 0AEC      JP       Z,ELN      ;Delete the bottom line only
2227      0A90      E5          PUSH      HL          ;Save row
2228      0A91      F5          PUSH      AF          ;Save counter (# of lines to be moved upward)
2229      0A92      4F          LD        C,A
2230      0A93      06 00      LD        B,0
2231      0A95      CD 0C1D      CALL      GETTRM      ;Get address of [LINTTB] in [DE]
2232      0A98      6B          LD        L,E
2233      0A99      62          LD        H,D
2234      0A9A      23          INC       HL
2235      0A9B      ED B0      LDIR
2236      0A9D      21 FBCA      LD        HL,FSTPOS
2237      0AA0      35          DEC       (HL)
2238      0AA1      F1          POP       AF
2239      0AA2      E1          POP       HL
2240      0AA3      DELLN1:
2241      0AA3      F5          PUSH      AF          ;Save counter
2242      0AA4      2C          INC       L
2243      0AA5      CD 0BAA      CALL      GET1LN      ;Get 1 line specified by L
2244      0AA8      2D          DEC       L
  
```

- MSXIO - Line insert and delete of CRT

```

2245      0AA9      CD 0BC3          CALL      PUT1LN          ;Put 1 line specified by L
2246      0AAC      2C              INC        L
2247      0AAD      F1              POP        AF              ;Restore counter
2248      0AAE      3D              DEC        A
2249      0AAF      20 F2           JR         NZ,DELLN1
2250      0AB1      C3 0AEC         JP         ELN              ;Blank bottom line
2251      0AB4
2252
2253      ;
2254      ; Insert a line
2255      ;
2256      ; Cursor should be set at the top of line
2257      0AB4      CD 0A81          CALL      CR
2258      0AB7
2259      0AB7      CD 0C32          CALL      GETLEN          ;Get an actual height of screen
2260      0ABA      67              LD         H,A
2261      0ABB      95              SUB        L
2262      0ABC      D8              RET        C              ;Something is wrong!!
2263      0ABD      CA 0AEC         JP         Z,ELN
2264      0AC0      6C              LD         L,H
2265      0AC1      E5              PUSH       HL              ;Save row to be inserted
2266      0AC2      F5              PUSH       AF              ;Save # of lines to be moved downward
2267      0AC3      4F              LD         C,A
2268      0AC4      06 00           LD         B,0
2269      0AC6      CD 0C1D          CALL      GETTRM
2270      0AC9      6B              LD         L,E
2271      0ACA      62              LD         H,D
2272      0ACB      E5              PUSH       HL              ;Save pointer to [LINTTB] for the bottom line
2273      0ACC      2B              DEC        HL              ;Form source address
2274      0ACD      ED B8           LDDR
2275      0ACF      E1              POP        HL

```

```
2276      0AD0      74          LD      (HL),H          ;Make sure the bottom line is terminated
2277      0AD1      F1          POP     AF
2278      0AD2      E1          POP     HL
2279      0AD3                      INSLN1:
2280      0AD3      F5          PUSH    AF          ;Save counter
2281      0AD4      2D          DEC     L
2282      0AD5      CD 0BAA      CALL    GET1LN
2283      0AD8      2C          INC     L
2284      0AD9      CD 0BC3      CALL    PUT1LN
2285      0ADC      2D          DEC     L
2286      0ADD      F1          POP     AF          ;Restore counter
2287      0ADE      3D          DEC     A
2288      0ADF      20 F2        JR      NZ,INSLN1
2289      0AE1      18 09        JR      ELN
2290
2291      SUBTTL - MSXIO - Character(s) erase
```

```
2292
2293     0AE3          RUBOUT:
2294          ;
2295          ; Erase previous character
2296          ;
2297     0AE3     CD 0A4C          CALL     BS          ;Back space
2298     0AE6     C8              RET      Z          ;We're at the top of screen
2299     0AE7     0E 20          LD       C,' '      ;Overstrike with a space
2300     0AE9     C3 0BE6          JP      PUTVRM
2301     0AEC          ELN:
2302          ;
2303          ; Erase entire line
2304          ;
2305          ; Cursor should remain unchanged
2306          ;
2307     0AEC     26 01          LD       H,l
2308     0AEE          EOL:
2309          ;
2310          ; Erase to end-of-line
2311          ;
2312          ; Cursor should remain unchanged
2313          ;
2314     0AEE     CD 0C29          CALL     TERMIN
2315     0AF1     E5              PUSH     HL          ;Save current position (column)
2316     0AF2     CD 0BF2          CALL     VADDR
2317     0AF5     CD 07DF          CALL     SETWRT
2318     0AF8     E1              POP      HL          ;Restore current position
2319     0AF9          EREOL1:
2320     0AF9     3E 20          LD       A,' '      ;Overstrike with a space
2321     0AFB     D3 98          OUT      (VDP.DRW),A
2322     0AFD     24              INC      H
```

```
2323      0AFE      3A F3B0          LD      A,(LINLEN)
2324      0B01      BC              CP      H
2325      0B02      30 F5          JR      NC,EREOL1
2326      0B04      C9              RET
2327      0B05          EOP:
2328                      ;
2329                      ; Erase to end-of-page
2330                      ;
2331                      ; Cursor should remain unchanged
2332                      ;
2333      0B05      E5          PUSH     HL          ;Save current position
2334      0B06      CD 0AEE      CALL    EOL          ;Erase to end-of-line
2335      0B09      EI          POP      HL          ;Restore current position
2336      0B0A      CD 0C32      CALL    GETLEN        ;Get an actual height of CRT
2337      0B0D      BD          CP      L
2338      0B0E      D8          RET      C          ;Something is wrong
2339      0B0F      C8          RET      Z          ;All done
2340      0B10      26 01      LD      H,1
2341      0B12      2C          INC      L
2342      0B13      18 F0      JR      EOP
2343                      ;
2344                      SUBTTL - MSXIO - Function keys display/erase.
```



```
2345
2346      0B15      ERAFNK:
2347      ;
2348      ; Erase function key
2349      ;
2350      0B15      CD FDB8      CALL      H.ERAF
2351      0B18      AF          XOR      A          ;Say no function key is displayed
2352      0B19      CD 0B9C      CALL      SETCHK
2353      0B1C      D0          RET      NC          ;We're not in text mode, just set flag
2354      0B1D      E5          PUSH     HL          ;Save possible text pointer
2355      0B1E      2A F3B1      LD       HL,(CRTCNT) ;Erase last line
2356      0B21      CD 0AEC      CALL     ELN
2357      0B24      E1          POP      HL          ;Restore possible text pointer
2358      0B25      C9          RET
2359      0B26      FNKSB:
2360      ;
2361      ; Display function key if enabled
2362      ;
2363      0B26      3A F3DE      LD       A,(CNSDFG) ;Now being displayed?
2364      0B29      A7          AND      A
2365      0B2A      C8          RET      Z          ;No
2366      0B2B      DSPFNK:
2367      ;
2368      ; Display function key
2369      ;
2370      0B2B      CD FDB3      CALL     H.DSPF
2371      0B2E      3E FF      LD       A,0FFH      ;Say function key is displayed
2372      0B30      CD 0B9C      CALL     SETCHK
2373      0B33      D0          RET      NC          ;We're not in text mode, just set flag
2374      0B34      E5          PUSH     HL          ;Save possible text pointer
2375      0B35      3A F3DC      LD       A,(CSRY)
```

```

2376 0B38 21 F3B1      LD      HL,CRTCNT
2377 0B3B BE          CP      (HL)
2378 0B3C 3E 0A      LD      A,0AH      ;Scroll up if we're at the bottom of screen
2379 0B3E 20 01      JR      NZ,NTBOTM
2380 0B40 DF          RST      18H
2381 0B41          NTBOTM:
2382 0B41 3A FBEB      LD      A,(SFTKEY)      ;Get current shift status
2383 0B44 0F          RRCA
2384 0B45 21 F87F      LD      HL,FNKSTR      ;Assume shift not pressed
2385 0B48 3E 01      LD      A,1
2386 0B4A 38 04      JR      C,DSPFK1      ;Good assumption
2387 0B4C 21 F8CF      LD      HL,FNKSTR+80      ;Shift is being pressed
2388 0B4F AF          XOR      A
2389 0B50          DSPFK1:
2390 0B50 32 FBCD      LD      (FNKSWI),A      ;Mark which part of function key is displayed
2391 0B53 11 FC18      LD      DE,BUFEND      ;Set temporary destination
2392 0B56 D5          PUSH     DE
2393 0B57 06 28      LD      B,'('      ;=40
2394 0B59 3E 20      LD      A,' '
2395 0B5B          DSFKCL:
2396 0B5B 12          LD      (DE),A
2397 0B5C 13          INC      DE
2398 0B5D 10 FC      DJNZ     DSFKCL
2399 0B5F D1          POP      DE      ;Restore temporary destination in [DE]
2400 0B60 0E 05      LD      C,5      ;Total number of keys
2401 0B62 3A F3B0      LD      A,(LINLEN)      ;Calculate (LINLEN-4) / 5
2402 0B65 D6 04      SUB      4
2403 0B67 38 2B      JR      C,DSPFKE      ;Not enough room for function keys
2404 0B69 06 FF      LD      B,0FFH
2405 0B6B          DSPFK4:
2406 0B6B 04          INC      B

```

2407	0B6C	D6 05	SUB	5	
2408	0B6E	30 FB	JR	NC,DSPFK4	
2409	0B70	78	LD	A,B	
2410	0B71	A7	AND	A	
2411	0B72	28 20	JR	Z,DSPFKE	;No enough room
2412	0B74	3E	DB	3EH	;Skip next byte
2413	0B75		DSPFK2:		
2414	0B75	13	INC	DE	;Put separator space
2415	0B76	C5	PUSH	BC	;Save key counter
2416	0B77	0E 00	LD	C,0	;Reset # of characters actually fetched
2417	0B79		DSPFK5:		
2418	0B79	7E	LD	A,(HL)	;Get from function key string
2419	0B7A	23	INC	HL	;Prepare for next fetch
2420	0B7B	0C	INC	C	
2421	0B7C	CD 089D	CALL	CNVCHR	
2422	0B7F	30 F8	JR	NC,DSPFK5	;This is a graphic header, fetch more
2423	0B81	20 04	JR	NZ,DSPFK8	;Converted graphics character, store this
2424	0B83	FE 20	CP	' '	;Printable?
2425	0B85	38 01	JR	C,DSPFK6	;No, ignore this
2426	0B87		DSPFK8:		
2427	0B87	12	LD	(DE),A	
2428	0B88		DSPFK6:		
2429	0B88	13	INC	DE	
2430	0B89	10 EE	DJNZ	DSPFK5	
2431	0B8B	3E 10	LD	A,10H	
2432	0B8D	91	SUB	C	
2433	0B8E	4F	LD	C,A	;Skip rest
2434	0B8F	09	ADD	HL,BC	
2435	0B90	C1	POP	BC	;Restore counter
2436	0B91	0D	DEC	C	
2437	0B92	20 E1	JR	NZ,DSPFK2	

```
2438      0B94          DSPFKE:
2439      0B94      2A F3B1          LD      HL,(CRTCNT)      ;Display at the lowest line
2440      0B97      CD 0BC3          CALL   PUT1LN
2441      0B9A      E1              POP     HL                ;Restore possible text pointer
2442      0B9B      C9              RET
2443
2444          SUBTTL - MSXIO - Low level routines
```

```

2445
2446      0B9C          SETCHK:
2447                      ;
2448                      ; Set CNSDFG and check current screen mode
2449                      ;
2450      0B9C      32 F3DE          LD      (CNSDFG),A
2451      0B9F          CHKSCR:
2452                      ;
2453                      ; Check current screen mode
2454                      ;
2455      0B9F      3A FCAF          LD      A,(SCRMOD)
2456      0BA2      FE 02          CP      2
2457      0BA4      C9              RET
2458      0BA5                      ;Return with the status
2459      GET8B:
2460                      ;
2461                      ; Get 8 bytes from HL
2462                      ;
2462      0BA5      E5              PUSH    HL
2463      0BA6      0E 08          LD      C,8
2464      0BA8      18 0A          JR      GET1L1
2465      0BAA          GET1LN:
2466                      ;
2467                      ; Get character and attribute of position specified by H,L
2468                      ;
2469                      ; Character returned in C
2470                      ;
2471      0BAA      E5              PUSH    HL
2472      0BAB      26 01          LD      H,1
2473      0BAD      CD 0BF2          CALL    VADDR
2474      0BB0      3A F3B0          LD      A,(LINLEN)
2475      0BB3      4F              LD      C,A
  
```

```
2476      0BB4          GET1L1:
2477      0BB4      06 00          LD      B,0
2478      0BB6      11 FC18        LD      DE,BUFEND      ;Storage for 1 line
2479      0BB9      CD 070F        CALL   LDIRMV
2480      0BBC      E1             POP     HL
2481      0BBD      C9             RET
2482      0BBE          PUT8B:
2483      ;
2484      0BBE      E5             PUSH    HL
2485      0BBF      0E 08          LD      C,8
2486      0BC1      18 0A          JR      PUT1L1
2487      0BC3          PUT1LN:
2488      ;
2489      0BC3      E5             PUSH    HL
2490      0BC4      26 01          LD      H,1
2491      0BC6      CD 0BF2        CALL   VADDR
2492      0BC9      3A F3B        LD      A,(LINLEN)
2493      0BCC      4F             LD      C,A
2494      0BCD          PUT1L1:
2495      0BCD      06 00          LD      B,0
2496      0BCF      EB             EX      DE,HL
2497      0BD0      21 FC18        LD      HL,BUFEND
2498      0BD3      CD 0744        CALL   LDIRVM
2499      0BD6      E1             POP     HL
2500      0BD7      C9             RET
2501      0BD8          GETVRM:
2502      ;
2503      0BD8      E5             PUSH    HL      ;Save coordinate
2504      0BD9      CD 0BF2        CALL   VADDR      ;Calculate VRAM address
2505      0BDC      CD 07EC        CALL   SETRD      ;Set up VDP for read
2506      0BDF      E3             EX      (SP),HL
```

2507	0BE0	E3	EX	(SP),HL	
2508	0BE1	DB 98	IN	A,(VDP.DRW)	;Get character code in C
2509	0BE3	4F	LD	C,A	
2510	0BE4	E1	POP	HL	;Restore coordinate
2511	0BE5	C9	RET		
2512	0BE6		PUTVRM:		
2513			;		
2514	0BE6	E5	PUSH	HL	
2515	0BE7	CD 0BF2	CALL	VADDR	
2516	0BEA	CD 07DF	CALL	SETWRT	
2517	0BED	79	LD	A,C	
2518	0BEE	D3 98	OUT	(VDP.DRW),A	
2519	0BF0	E1	POP	HL	
2520	0BF1	C9	RET		
2521	0BF2		VADDR:		
2522			;		
2523			; Calculate buffer address out of H,L (column,row)		
2524			;		
2525			; address returned in HL		
2526			;		
2527	0BF2	C5	PUSH	BC	
2528	0BF3	5C	LD	E,H	;Get column in L
2529	0BF4	26 00	LD	H,0	
2530	0BF6	54	LD	D,H	
2531	0BF7	2D	DEC	L	
2532	0BF8	29	ADD	HL,HL	
2533	0BF9	29	ADD	HL,HL	
2534	0BFA	29	ADD	HL,HL	
2535	0BFB	4D	LD	C,L	
2536	0BFC	44	LD	B,H	
2537	0BFD	29	ADD	HL,HL	

Address	Hex	Symbol	Instruction	Comment
2538	0BFE	29	ADD	HL,HL
2539	0BFF	19	ADD	HL,DE
2540	0C00	3A FCAF	LD	A,(SCRMOD)
2541	0C03	A7	AND	A
2542	0C04	3A F3B0	LD	A,(LINLEN)
2543	0C07	28 04	JR	Z,VADDR1
2544	0C09	D6 22	SUB	'"'
2545	0C0B	18 03	JR	VADDR2
2546	0C0D		VADDR1:	
2547			;	
2548	0C0D	09	ADD	HL,BC
2549	0C0E	D6 2A	SUB	41+1
2550	0C10		VADDR2:	
2551	0C10	2F	CPL	
2552	0C11	A7	AND	A
2553	0C12	1F	RRA	
2554	0C13	5F	LD	E,A
2555	0C14	19	ADD	HL,DE
2556	0C15	EB	EX	DE,HL
2557	0C16	2A F922	LD	HL,(NAMBAS)
2558	0C19	19	ADD	HL,DE
2559	0C1A	2B	DEC	HL
2560	0C1B	C1	POP	BC
2561	0C1C	C9	RET	
2562	0C1D		GETTRM:	
2563			;	
2564			;	Get value of line-terminator-table and affect flags
2565			;	
2566			;	Entry: L has the line #
2567			;	Exit: DE has the address of corresponding terminator byte.
2568			;	Z flag is affected.


```

2569      ;
2570      0C1D      E5      PUSH      HL      ;Save HL
2571      0C1E      11      FBB1      LD      DE,BASROM
2572      0C21      26      00      LD      H,0
2573      0C23      19      ADD      HL,DE      ;Get address of table
2574      0C24      7E      LD      A,(HL)
2575      0C25      EB      EX      DE,HL      ;Move address to DE
2576      0C26      E1      POP      HL      ;Restore HL
2577      0C27      A7      AND      A      ;Affect flags
2578      0C28      C9      RET
2579      0C29      TERMIN:
2580      ;
2581      0C29      3E      DB      3EH      ;Load non 0 value in Acc
2582      0C2A      UNTERM:
2583      0C2A      AF      XOR      A
2584      0C2B      SETTRM:
2585      0C2B      F5      PUSH      AF
2586      0C2C      CD      0C1D      CALL      GETTRM      ;Get address of terminator byte in DE
2587      0C2F      F1      POP      AF
2588      0C30      12      LD      (DE),A      ;Change table
2589      0C31      C9      RET
2590      0C32      GETLEN:
2591      ;
2592      ; Get an actual height of screen
2593      ;
2594      0C32      3A      F3DE      LD      A,(CNSDFG)      ;0 or -1
2595      0C35      E5      PUSH      HL
2596      0C36      21      F3B1      LD      HL,CRTCNT
2597      0C39      86      ADD      A,(HL)
2598      0C3A      E1      POP      HL
2599      0C3B      C9      RET

```

(MSX ROM BASIC BIOS) Macro-80
- MSXIO - Low level routines

3.44

01-Jan-85

PAGE

36-5

98

2600

;

2601

SUBTTL - MSXIO - Keyboard encoding routines

```
2602
2603      0C3C      KEYINT:
2604      ;
2605      ; Encode keyboard
2606      ;
2607      ; Timer interrupt routine
2608      ;
2609      0C3C      E5      PUSH      HL      ;Save all registers
2610      0C3D      D5      PUSH      DE
2611      0C3E      C5      PUSH      BC
2612      0C3F      F5      PUSH      AF
2613      0C40      D9      EXX
2614      0C41      08      EX      AF,AF'
2615      0C42      E5      PUSH      HL
2616      0C43      D5      PUSH      DE
2617      0C44      C5      PUSH      BC
2618      0C45      F5      PUSH      AF
2619      0C46      FD E5    PUSH      IY
2620      0C48      DD E5    PUSH      IX
2621      0C4A      CD FD9A    CALL     H.KEYI      ;To allow other interrupts than 60Hz timer
2622      0C4D      DB 99      IN       A,(VDP.SR)  ;Clear possible interrupt request
2623      0C4F      A7      AND      A      ;Interrupt requested by VDP?
2624      0C50      F2 0D02    JP       P,INTRET    ;No, skip the rest
2625      0C53      CD FD9F    CALL     H.TIMI      ;To allow timer interrupt to be
2626      ;used elsewhere.
2627      0C56      FB      EI      ;Now that it became obvious that VDP
2628      ;generated the interrupt, we re-enable
2629      ;interrupt here to allow RS232C's
2630      ;interrupt or something like that.
2631      0C57      32 F3E7    LD       (STATFL),A  ;Store this new status
2632      0C5A      E6 20      AND      ' '      ;Collision detected?
```

```

2633      0C5C      21 FC6D          LD      HL,TRPTBL+33      ;Assume so
2634      0C5F      C4 0EF1          CALL     NZ,REQTRP        ;Request trap if so
2635
2636          ; Check interval trap
2637          ;
2638      0C62      2A FCA2          LD      HL,(INTCNT)        ;Count down interval count
2639      0C65      2B              DEC      HL
2640      0C66      7C              LD      A,H
2641      0C67      B5              OR      L
2642      0C68      20 09          JR      NZ,NTINTT            ;Not yet reached 0
2643      0C6A      21 FC7F          LD      HL,TRPTBL+3*17      ;Request trap
2644      0C6D      CD 0EF1          CALL     REQTRP
2645      0C70      2A FCA0          LD      HL,(INTVAL)        ;Load initial value
2646      0C73
2647      0C73      22 FCA2          LD      (INTCNT),HL        ;Update interval count
2648          ;
2649          ; Increment jiffy count
2650          ;
2651      0C76      2A FC9E          LD      HL,(JIFFY)
2652      0C79      23              INC      HL
2653      0C7A      22 FC9E          LD      (JIFFY),HL
2654          ;
2655          ; Check music queue
2656          ;
2657      0C7D      3A FB3F          LD      A,(MUSICF)          ;Check music flag
2658      0C80      4F              LD      C,A
2659      0C81      AF              XOR      A                  ;Start with queue 0
2660      0C82
2661      0C82      CB 19          RR      C                  ;C7=carry, carry=C0, [C]=[C]/2
2662      0C84      F5              PUSH     AF                ;Save queue ID
2663      0C85      C5              PUSH     BC                ;Save MUSICF

```

```
2664      0C86      DC 113B          CALL      C,ACTION
2665      0C89      C1                POP       BC
2666      0C8A      F1                POP       AF
2667      0C8B      3C                INC       A           ;Next queue
2668      0C8C      FE 03             CP        3           ;All done?
2669      0C8E      38 F2             JR        C,MUSINT     ;Not yet
2670      0C90      21 F3F6           LD        HL,SCNCNT
2671      0C93      35                DEC       (HL)         ;Need to scan?
2672      0C94      20 6C             JR        NZ,INTRET    ;No, return soon
2673      0C96      36 03             LD        (HL),3       ;Time delay of first repeat
2674
2675          ;
2676          ; Check trigger button of joy sticks
2677          ;
2677      0C98      AF                XOR       A
2678      0C99      CD 120C           CALL      SLSTCK       ;Read joystick A
2679      0C9C      E6 30             AND       00110000B
2680      0C9E      F5                PUSH      AF
2681      0C9F      3E 01             LD        A,1
2682      0CA1      CD 120C           CALL      SLSTCK
2683      0CA4      E6 30             AND       '0'
2684      0CA6      07                RLCA
2685      0CA7      07                RLCA
2686      0CA8      C1                POP       BC
2687      0CA9      B0                OR        B
2688      0CAA      F5                PUSH      AF
2689      0CAB      CD 1226           CALL      GTROW8
2690      0CAE      E6 01             AND       1
2691      0CB0      C1                POP       BC
2692      0CB1      B0                OR        B
2693      0CB2      4F                LD        C,A           ;Save this
2694      0CB3      21 F3E8           LD        HL,TRGFLG
```

2695	0CB6	AE	XOR	(HL)	;Any transition?
2696	0CB7	A6	AND	(HL)	;Is this transition negative
2697	0CB8	71	LD	(HL),C	;Update trigger status
2698	0CB9	4F	LD	C,A	
2699	0CBA	0F	RRCA		;Check space key trigger
2700	0CBB	21 FC70	LD	HL,TRPTBL+3*12	
2701	0CBE	DC 0EF1	CALL	C,REQTRP	
2702	0CC1	CB 11	RL	C	;Check trigger 4
2703	0CC3	21 FC7C	LD	HL,TRPTBL+3*16	
2704	0CC6	DC 0EF1	CALL	C,REQTRP	
2705	0CC9	CB 11	RL	C	;Check trigger 2
2706	0CCB	21 FC76	LD	HL,TRPTBL+3*14	
2707	0CCE	DC 0EF1	CALL	C,REQTRP	
2708	0CD1	CB 11	RL	C	;Check trigger 3
2709	0CD3	21 FC79	LD	HL,TRPTBL+3*15	
2710	0CD6	DC 0EF1	CALL	C,REQTRP	
2711	0CD9	CB 11	RL	C	;Check trigger 1
2712	0CDB	21 FC73	LD	HL,TRPTBL+3*13	
2713	0CDE	DC 0EF1	CALL	C,REQTRP	
2714					
2715					; Scan keyboard
2716					
2717	0CE1	AF	XOR	A	;Enable first key click
2718	0CE2	32 FBD9	LD	(CLIKFL),A	
2719	0CE5	CD 0D12	CALL	KEYCHK	;Detect valid key transition and check buffer
2720	0CE8	20 18	JR	NZ,INTRET	;Some characters still remain, don't repeat
2721	0CEA	21 F3F7	LD	HL,REPCNT	
2722	0CED	35	DEC	(HL)	;Need to enter repeat mode
2723	0CEE	20 12	JR	NZ,INTRET	;No
2724	0CF0	36 01	LD	(HL),1	;Set short time repeat
2725	0CF2	21 FBDA	LD	HL,OLDKEY	;Clear OLDKEY status

2726	0CF5	11 FBDB	LD	DE,OLDKEY+1	
2727	0CF8	01 000A	LD	BC,0AH	
2728	0CFB	36 FF	LD	(HL),0FFH	
2729	0CFD	ED B0	LDIR		
2730	0CFF	CD 0D4E	CALL	KEYCK4	;Check if currently pressed key is valid
2731	0D02				
2732	0D02	DD E1	POP	IX	;Restore all registers
2733	0D04	FD E1	POP	IY	
2734	0D06	F1	POP	AF	
2735	0D07	C1	POP	BC	
2736	0D08	D1	POP	DE	
2737	0D09	E1	POP	HL	
2738	0D0A	08	EX	AF,AF'	
2739	0D0B	D9	EXX		
2740	0D0C	F1	POP	AF	
2741	0D0D	C1	POP	BC	
2742	0D0E	D1	POP	DE	
2743	0D0F	E1	POP	HL	
2744	0D10	FB	EI		
2745	0D11	C9	RET		
2746	0D12				
2747					
2748	0D12	DB AA	IN	A,(PPI.CR)	;Get what is currently output to Port C
2749	0D14	E6 F0	AND	0F0H	;Leave higher 4 bits unaffected
2750	0D16	4F	LD	C,A	
2751	0D17	06 0B	LD	B,0BH	
2752	0D19	21 FBE5	LD	HL,NEWKEY	;Move current key status to NEWKEY
2753	0D1C				
2754	0D1C	79	LD	A,C	
2755	0D1D	D3 AA	OUT	(PPI.CW),A	;Select row
2756	0D1F	DB A9	IN	A,(PPI.BR)	;Get column information of selected row

```

2757      0D21      77              LD      (HL),A          ;Move it
2758      0D22      0C              INC      C              ;Select next row
2759      0D23      23              INC      HL
2760      0D24      10 F6           DJNZ     KEYCK1         ;Loop until all rows are sensed
2761      0D26      3A FBB0         LD      A,(ENSTOP)       ;Warm start enabled?
2762      0D29      A7              AND      A
2763      0D2A      28 0E           JR      Z,NOSTOP         ;No
2764      0D2C      3A FBEB         LD      A,(SFTKEY)       ;Get current status of the 6th row
2765      0D2F      FE E8           CP      0E8H            ;Check if KANA, GRAPH, CTRL and SHIFT
2766      0D31      20 07           JR      NZ,NOSTOP        ;are pressed simultaneously
2767      0D33      DD 21 409B       LD      IX,READYR
2768      0D37      C3 01FF         JP      CALBAS
2769      0D3A                                NOSTOP:
2770                                ;
2771      0D3A      11 FBE5         LD      DE,NEWKEY         ;[OLDKEY] + 11
2772      0D3D      06 0B         LD      B,0BH
2773      0D3F                                KEYCK2:
2774      0D3F      1B              DEC      DE
2775      0D40      2B              DEC      HL
2776      0D41      1A              LD      A,(DE)           ;Get OLDKEY status
2777      0D42      BE              CP      (HL)             ;Compare with NEWKEY status
2778      0D43      20 04           JR      NZ,KEYCK3        ;Changed, set long repeat interval
2779      0D45      10 F8           DJNZ     KEYCK2
2780      0D47      18 05           JR      KEYCK4           ;No change
2781      0D49                                KEYCK3:
2782                                ;
2783      0D49      3E 0D           LD      A,0DH
2784      0D4B      32 F3F7         LD      (REPCNT),A
2785      0D4E                                KEYCK4:
2786      0D4E      06 0B         LD      B,0BH             ;Set number of rows
2787      0D50      21 FBDA         LD      HL,OLDKEY

```



```

2788      0D53      11 FBE5          LD      DE,NEWKEY
2789      0D56          KEYCK5:
2790      0D56      1A          LD      A,(DE)          ;Get current key status
2791      0D57      4F          LD      C,A
2792      0D58      AE          XOR      (HL)          ;See if any bit changed
2793      0D59      A6          AND      (HL)          ;See if this change is negative transition
2794      0D5A      71          LD      (HL),C          ;Update old status
2795      0D5B      C4 0D89      CALL     NZ,KEYANY      ;Active transition, go find it
2796      0D5E      13          INC      DE
2797      0D5F      23          INC      HL
2798      0D60      10 F4      DJNZ     KEYCK5
2799      0D62          CHKBUF:
2800          ;
2801          ; Check if buffer is empty or not
2802          ;
2803      0D62      2A F3FA      LD      HL,(GETPNT)      ;Load GETPNT
2804      0D65      3A F3F8      LD      A,(PUTPNT)      ;Load lower 8 bit of PUTPNT
2805      0D68      95          SUB      L              ;Check if same
2806      0D69      C9          RET
2807      0D6A          CHSNS:
2808          ;
2809      0D6A      FB          EI              ;Make sure interrupts are enabled
2810      0D6B      E5          PUSH     HL              ;Save environments
2811      0D6C      D5          PUSH     DE
2812      0D6D      C5          PUSH     BC
2813      0D6E      CD 0B9F      CALL     CHKSCR          ;Are we in text mode?
2814      0D71      30 0F      JR      NC,CHSNS1        ;No, do not flip function keys
2815      0D73      3A FBCE      LD      A,(FNKSWI)      ;Get current shift status
2816      0D76      21 FBEB      LD      HL,SFTKEY      ;Get current function key display
2817      0D79      AE          XOR      (HL)          ;Are they different
2818      0D7A      21 F3DE      LD      HL,CNSDFG        ;Function key displayed at all?

```

```

2819 0D7D A6                AND    (HL)
2820 0D7E 0F                RRCA
2821 0D7F DC 0B2B          CALL    C,DSPFNK      ;Update display
2822 0D82                CHSNS1:
2823 0D82 CD 0D62          CALL    CHKBUF
2824 0D85 C1                POP     BC            ;Restore environments
2825 0D86 D1                POP     DE
2826 0D87 E1                POP     HL
2827 0D88 C9                RET
2828 0D89                KEYANY:
2829                        ;
2830                        ; [[[ SUBROUTINE 'KEYANY' ]]]
2831                        ;
2832 0D89 E5                PUSH    HL            ;Save environments
2833 0D8A D5                PUSH    DE
2834 0D8B C5                PUSH    BC
2835 0D8C F5                PUSH    AF            ;Save pressed bit
2836 0D8D 3E 0B          LD      A,0BH
2837 0D8F 90                SUB     B            ;Calculate base code
2838 0D90 87                ADD     A,A
2839 0D91 87                ADD     A,A
2840 0D92 87                ADD     A,A
2841 0D93 4F                LD      C,A
2842 0D94 06 08          LD      B,8
2843 0D96 F1                POP     AF            ;Set up counter for 8 bit
2844 0D97                KYANY1:                ;Restore pressed bit
2845 0D97 1F                RRA
2846 0D98 C5                PUSH    BC
2847 0D99 F5                PUSH    AF
2848 0D9A DC 0E3B          CALL    C,KEYCOD      ;If pressed bit, call key coder.
2849 0D9D F1                POP     AF

```

```
2850 0D9E C1 POP BC
2851 0D9F 0C INC C ;Try next code
2852 0DA0 10 F5 DJNZ KYANYL ;Loop until all bits are checked
2853 0DA2 C3 08DB JP PBDHRT ;Restore environments
2854 ;
2855 ; [[[ SUBROUTINE 'KEYCOD' ]]]
2856 ;
2857 ; Return key-code in buffer if valid
2858 ;
2859 0DA5 KYJTAB:
2860 0DA5 0A DB 10
2861 0DA6 0E67 DW KYNUM ;0..9
2862 0DA8 16 DB 22
2863 0DA9 0EA1 DW KYCOD1
2864 0DAB 30 DB 48
2865 0DAC 0E7E DW KYALP ;A..Z
2866 0DAE 33 DB 51
2867 0DAF 0F10 DW KYEASY
2868 0DB1 34 DB 52
2869 0DB2 0F36 DW KYLOCK ;Capital lock
2870 0DB4 35 DB 53
2871 0DB5 0F1F DW KYKLOK ;Kana lock
2872 0DB7 3A DB 58
2873 0DB8 0EBB DW KYFUNC ;Function key
2874 0DBA 3C DB 60
2875 0DBB 0F10 DW KYEASY
2876 0DBD 3D DB 61
2877 0DBE 0F46 DW KYSTOP ;Stop key
2878 0DC0 41 DB 65
2879 0DC1 0F10 DW KYEASY
2880 0DC3 42 DB 66
```

2881	0DC4	0F06	DW	KYCLS	;CLS/HOME key
2882	0DC6	FF	DB	255	
2883	0DC7	0F10	DW	KYEASY	
2884					
2885	0DC9				
2886	0DC9	FF	DB	255	
2887	0DCA	21	DB	"'"	
2888	0DCB	22	DB	34	;Double quote
2889	0DCC	23 24 25 26	DB	"#\$%&'()"	
2890	0DD0	27 28 29			
2891					
2892	0DD3				
2893	0DD3	0F55	DW	PUTCHR	;CTRL+shift
2894	0DD5	0F55	DW	PUTCHR	;CTRL
2895	0DD7	0E93	DW	KEYSFT	; SHIFT
2896	0DD9	0E95	DW	KEYNOM	;
2897					
2898	0ddb				
2899	0ddb	0dfd	DW	KY1SFC-10	;CTRL+SHIFT
2900	0ddd	0df1	DW	KY1CNT-10	;CTRL
2901	0ddf	0de5	DW	KY1SFT-10	; SHIFT
2902	0del	0dd9	DW	KY1NOM-10	;
2903	0de3				
2904	0de3	2D 5E 5C 40	DB	"-^\@[;:],./"	
2905	0de7	5B 3B 3A 5D			
2906	0deb	2C 2E 2F			
2907	0dee	FF	DB	255	
2908	0def				
2909	0def	3D 7E 7C 60	DB	"=~ `{+*}"	
2910	0df3	7B 2B 2A 7D			
2911	0df7	3C	DB	00111100B	;Less than sign

2912	0DF8	3E	DB	00111110B	;Greater than sign
2913	0DF9	3F 5F	DB	"?_"	
2914	0DFB				
			KYLCNT:		
2915	0DFB	2D	DB	"_"	
2916	0DFC	1E	DB	"^"_"@"	
2917	0DFD	1C	DB	"\"_"@"	
2918	0DFE	00	DB	"@"_"@"	
2919	0DFF	1B	DB	"["_"@"	
2920	0E00	3B 3A	DB	";:"	
2921	0E02	1D	DB	"]"_"@"	
2922	0E03	2C 2E 2F	DB	","./"	
2923	0E06	FF	DB	255	
2924	0E07				
			KYLSFC:		
2925	0E07	3D	DB	"="	
2926	0E08	1E	DB	"^"_"@"	
2927	0E09	1C	DB	"\"_"@"	
2928	0E0A	00	DB	"@"_"@"	
2929	0E0B	1B	DB	"["_"@"	
2930	0E0C	2B 2A	DB	"+"*	
2931	0E0E	1D	DB	"]"_"@"	
2932	0E0F	3C	DB	001111100B	;Less than sign
2933	0E10	3E	DB	001111110B	;Greater than sign
2934	0E11	3F	DB	"?"	
2935	0E12	1F	DB	"_"_"@"	
2936					
			;		
2937	0E13				
			EASYTB:		
2938	0E13	00	DB	0	;Shift (48)
2939	0E14	00	DB	0	;Control (49)
2940	0E15	00	DB	0	;Graph (50)
2941	0E16	00	DB	0	;Cap lock (51)
2942	0E17	00	DB	0	;Kana lock (52)

2943	0E18	00	DB	0	;F1	(53)
2944	0E19	00	DB	0	;F2	(54)
2945	0E1A	00	DB	0	;F3	(55)
2946	0E1B	00	DB	0	;F4	(56)
2947	0E1C	00	DB	0	;F5	(57)
2948	0E1D	1B	DB	27	;Escape	(58)
2949	0E1E	09	DB	9	;Tab	(59)
2950	0E1F	00	DB	0	;Stop	(60)
2951	0E20	08	DB	8	;Back space	(61)
2952	0E21	18	DB	"X"-"@"	;Select	(62)
2953	0E22	0D	DB	13	;Enter	(63)
2954	0E23	20	DB	32	;Space	(64)
2955	0E24	0C	DB	12	;Clear	(65)
2956	0E25	12	DB	"R"-"@"	;Insert	(66)
2957	0E26	7F	DB	127	;Rubout	(67)
2958	0E27	1D	DB	29	;Left	(68)
2959	0E28	1E	DB	30	;Up	(69)
2960	0E29	1F	DB	31	;Down	(70)
2961	0E2A	1C	DB	28	;Right	(71)
2962						
2963						
2964						
					For additional key matrix	
2965	0E2B	01	DB	"A"-"@"	;	(72)
2966	0E2C	04	DB	"D"-"@"	;	(73)
2967	0E2D	0F	DB	"O"-"@"	;	(74)
2968	0E2E	10	DB	"P"-"@"	;	(75)
2969	0E2F	11	DB	"Q"-"@"	;	(76)
2970	0E30	12	DB	"R"-"@"	;	(77)
2971	0E31	13	DB	"S"-"@"	;	(78)
2972	0E32	14	DB	"T"-"@"	;	(79)
2973	0E33	00	DB	0	;	(80)

MSX ROM BASIC BIOS) Macro-80
- MSXIO - Keyboard encoding routines

3.44

01-Jan-85

PAGE

37-12

111

2974	0E34	00	DB	0	;	(81)
2975	0E35	00	DB	0	;	(82)
2976	0E36	00	DB	0	;	(83)
2977	0E37	00	DB	0	;	(84)
2978	0E38	00	DB	0	;	(85)
2979	0E39	00	DB	0	;	(86)
2980	0E3A	00	DB	0	;	(87)

```

1981
1982
1983      0E3B      ;
1984      ; KEYCOD:
1985      ;
1986      ; [[[ SUBROUTINE 'KEYCOD' ]]]
1987      ;
1988      ; Return key-code in buffer if valid
1989      ;
1989      0E3B      79      LD      A,C      ;Get raw code
1990      0E3C      FE FF   CP      0FFH    ;Just for fail safe
1991      0E3E      C8      RET      Z
1992      0E3F      21 0DA5  LD      HL,KYJTAB
1993      0E42      CD FDCC  CALL     H.KEYC
1994      0E45      FE 30    CP      48      ;Possibly a KANA or graphic character
1995      0E47      30 13    JR      NC,KYCLAS ;No
1996      0E49      3A FBEB  LD      A,(SFTKEY) ;Get shift key status
1997      0E4C      0F      RRCA      ;Control pressed?
1998      0E4D      0F      RRCA
1999      0E4E      30 0B    JR      NC,KYCLA0 ;Yes, this supersedes everything
2000      0E50      0F      RRCA      ;How about graphic shift
2001      0E51      D2 107D  JP      NC,KYGRAP ;Yes, this has the 2nd priority
2002      0E54      3A FCAC  LD      A,(KANAST) ;KANA lock active
2003      0E57      A7      AND      A
2004      0E58      C2 0F83  JP      NZ,KYKANA ;Yes
2005      0E5B      KYCLA0:
2006      0E5B      79      LD      A,C
2007      0E5C      KYCLAS:
2008      0E5C      BE      CP      (HL)    ;Compare range
2009      0E5D      23      INC      HL
2010      0E5E      5E      LD      E,(HL)  ;Get jump address in [DE]
2011      0E5F      23      INC      HL

```



```

3012      0E60      56                      LD      D,(HL)
3013      0E61      23                      INC     HL
3014      0E62      D5                      PUSH    DE                ;Assume matched
3015      0E63      D8                      RET     C                ;Good assumption
3016      0E64      D1                      POP     DE                ;Discard stack
3017      0E65      18 F5                  JR      KYCLAS            ;Check next possibility
3018      0E67                      KYNUM:
3019                      ;
3020      0E67      C6 30                  ADD     A,'0'                ;Assume no shift
3021      0E69      47                      LD      B,A                ;Save code
3022      0E6A      3A FBEB                LD      A,(SFTKEY)          ;Check shift status
3023      0E6D      0F                      RRCA
3024      0E6E      78                      LD      A,B                ;Restore code
3025      0E6F      38 0A                  JR      C,JPUTCH            ;Good assumption
3026      0E71      06 00                  LD      B,0
3027      0E73      21 0DC9                LD      HL,NMSFTB
3028      0E76      09                      ADD     HL,BC                ;This must not be 'DADF'
3029      0E77      7E                      LD      A,(HL)              ;Get code for shift-number
3030      0E78      FE FF                  CP      0FFH                ;Shift '0'?
3031      0E7A      C8                      RET     Z                ;Yes, ignore this
3032      0E7B                      JPUTCH:
3033      0E7B      C3 0F55                  JP      PUTCHR              ;Put this in buffer
3034      0E7E                      KYALP:
3035                      ;
3036      0E7E      3A FBEB                LD      A,(SFTKEY)
3037      0E81      E6 03                  AND     3
3038      0E83      87                      ADD     A,A
3039      0E84      5F                      LD      E,A
3040      0E85      16 00                  LD      D,0
3041      0E87      21 0DD3                LD      HL,ALPJMP
3042      0E8A      19                      ADD     HL,DE

```

```

3043      0E8B      7E          LD      A,(HL)          ;Get jump address
3044      0E8C      23          INC     HL
3045      0E8D      66          LD      H,(HL)
3046      0E8E      6F          LD      L,A
3047      0E8F      79          LD      A,C          ;Get code
3048      0E90      D6 15       SUB     15H          ;Make it a control character (1 - 26)
3049      0E92      E9          JP      (HL)
3050      0E93          KEYSFT:
3051          ;
3052      0E93      C6 20       ADD     A,' '
3053      0E95          KEYNOM:
3054      0E95      47          LD      B,A          ;Save code
3055      0E96      3A FCAB     LD      A,(CAPST)
3056      0E99      2F          CPL
3057      0E9A      E6 20       AND     00100000B      ;Bit 5 is on if CAP lock not active
3058      0E9C      A8          XOR     B
3059      0E9D      C6 40       ADD     A,01000000B
3060      0E9F      18 DA       JR      JPUTCH
3061      0EA1          KYCOD1:
3062          ;
3063      0EA1      21 0DDB     LD      HL,KYC1TB
3064      0EA4      3A FBEB     LD      A,(SFTKEY)
3065      0EA7      E6 03       AND     3          ;Extract shift and control status
3066      0EA9      87          ADD     A,A
3067      0EAA      5F          LD      E,A
3068      0EAB      16 00       LD      D,0
3069      0EAD      19          ADD     HL,DE
3070      0EAE      7E          LD      A,(HL)
3071      0EAF      23          INC     HL
3072      0EB0      66          LD      H,(HL)
3073      0EB1      6F          LD      L,A

```

```
3074      0EB2      59              LD      E,C
3075      0EB3      19              ADD     HL,DE
3076      0EB4      7E              LD      A,(HL)
3077      0EB5      FE FF          CP      0FFH          ;Should generate some code?
3078      0EB7      C2 0F55        JP      NZ,PUTCHR      ;Yes
3079      0EBA      C9              RET                ;No code should be generated
3080      0EBB                      KYFUNC:
3081                      ;
3082                      ; Function keys
3083                      ;
3084      0EBB      3A FBEB          LD      A,(SFTKEY)      ;Is shift pressed?
3085      0EBE      0F              RRCA
3086      0EBF      38 04          JR      C,KYFNC1        ;No
3087      0EC1      79              LD      A,C
3088      0EC2      C6 05          ADD     A,5
3089      0EC4      4F              LD      C,A
3090      0EC5                      KYFNC1:
3091      0EC5      59              LD      E,C          ;[DE] is (56..65)
3092      0EC6      16 00          LD      D,0
3093      0EC8      21 FB99        LD      HL,FNKFLG-53      ;Check if this function key is an event device
3094      0ECB      19              ADD     HL,DE
3095      0ECC      7E              LD      A,(HL)
3096      0ECD      A7              AND     A
3097      0ECE      20 13          JR      NZ,FNKINT        ;Request trap if not in direct mode
3098      0ED0                      KYFNC2:
3099      0ED0      EB              EX      DE,HL
3100      0ED1      29              ADD     HL,HL
3101      0ED2      29              ADD     HL,HL
3102      0ED3      29              ADD     HL,HL
3103      0ED4      29              ADD     HL,HL
3104      0ED5      11 F52F        LD      DE,FNKSTR-53*16
```

```

3105      0ED8      19              ADD      HL,DE          ;Get function key string address
3106      0ED9      EB              EX        DE,HL          ;Move address to DE
3107      0EDA              KYFNC3:
3108      0EDA      1A              LD        A,(DE)          ;Get from function key string
3109      0EDB      A7              AND        A              ;End of string
3110      0EDC      C8              RET        Z              ;Yes
3111      0EDD      CD 0F55          CALL     PUTCHR          ;Put this character in buffer
3112      0EE0      13              INC        DE              ;Check next character
3113      0EE1      18 F7              JR        KYFNC3
3114      0EE3              FNKINT:
3115      0EE3              ;
3116      0EE3      2A F41C          LD        HL,(CURLIN)      ;Are we in direct mode (CURLIN=65535)
3117      0EE6      23              INC        HL
3118      0EE7      7C              LD        A,H
3119      0EE8      B5              OR         L
3120      0EE9      28 E5              JR        Z,KYFNC2          ;Yes, treat as normal function key
3121      0EEB      21 FBAD          LD        HL,TRPTBL-53*3
3122      0EEE      19              ADD        HL,DE
3123      0EEF      19              ADD        HL,DE
3124      0EF0      19              ADD        HL,DE

```

```

3125
3126      ;
3127      0EF1      REQTRP:
3128      ;
3129      ; Request trap (called to request trap for event devices)
3130      ;
3131      ;
3132      ; Since REQTRP is mostly called from within an interrupt routine,
3133      ; don't touch the interrupt mask through DI or EI.
3134      ;
3135      0EF1      7E          LD      A,(HL)
3136      0EF2      E6 01      AND     1          ;Trap on?
3137      0EF4      C8          RET     Z          ;TRAP NOT ON
3138      0EF5      7E          LD      A,(HL)
3139      0EF6      F6 04      OR      4          ;Trap request
3140      0EF8      BE          CP      (HL)
3141      0EF9      C8          RET     Z          ;No change
3142      0EFA      77          LD      (HL),A
3143      0EFB      EE 05      XOR     5          ;Trap on + Trap request
3144      0EFD      C0          RET     NZ
3145      0EFE      3A FBD8     LD      A,(ONGSBF)
3146      0F01      3C          INC     A
3147      0F02      32 FBD8     LD      (ONGSBF),A
3148      0F05      C9          RET
3149      ;
3150      0F06      KYCLS:
3151      0F06      3A FBEB     LD      A,(SFTKEY)      ;Set carry if shift not pressed
3152      0F09      0F          RRCA
3153      0F0A      3E 0C      LD      A,0CH          ;Load code for CLS
3154      0F0C      DE 00      SBC     A,0          ;Change to HOME if shift not pressed
3155      0F0E      18 45      JR      PUTCHR
  
```

```

3156      0F10      KYEASY:
3157      ;
3158      ; Easily converted keys
3159      ;
3160      0F10      CD FDD1      CALL      H.KYEA      ;For CCP (Cut, copy, paste) editor rom
3161      0F13      5F          LD        E,A          ;These character are simply taken from table
3162      0F14      16 00      LD        D,0
3163      0F16      21 0DE3      LD        HL,EASYTB-48
3164      0F19      19          ADD       HL,DE
3165      0F1A      7E          LD        A,(HL)
3166      0F1B      A7          AND       A          ;Should this key generate some code
3167      0F1C      C8          RET       Z          ;No
3168      0F1D      18 36      JR        PUTCHR      ;Yes
3169      0F1F      KYKLOK:
3170      ;
3171      ; Kana lock key
3172      ;
3173      0F1F      21 FCAC      LD        HL,KANAST
3174      0F22      7E          LD        A,(HL)
3175      0F23      2F          CPL
3176      0F24      77          LD        (HL),A
3177      0F25      3E 0F      LD        A,0FH
3178      0F27      D3 A0      OUT       (PSG.LW),A
3179      0F29      DB A2      IN        A,(PSG.DR)
3180      0F2B      E6 7F      AND       7FH
3181      0F2D      47          LD        B,A
3182      0F2E      7E          LD        A,(HL)
3183      0F2F      2F          CPL
3184      0F30      E6 80      AND       80H
3185      0F32      B0          OR        B
3186      0F33      D3 A1      OUT       (PSG.DW),A

```

```

3187      0F35          NOKEY:
3188      0F35      C9          RET
3189      0F36          KYLOCK:
3190                      ;
3191                      ; Capital lock key
3192                      ;
3193      0F36      21 FCAB          LD      HL,CAPST
3194      0F39      7E          LD      A,(HL)          ;Toggle capital status
3195      0F3A      2F          CPL
3196      0F3B      77          LD      (HL),A          ;Update capital status
3197      0F3C      2F          CPL
3198      0F3D          CHGCAP:
3199      0F3D      A7          AND      A
3200      0F3E      3E 0C          LD      A,0CH          ;Assume 'turn off'
3201      0F40      28 01          JR      Z,CGCAP1          ;Good assumption
3202      0F42      3C          INC      A          ;Change to 'turn on'
3203      0F43          CGCAP1:
3204      0F43      D3 AB          OUT      (PPI.CM),A
3205      0F45      C9          RET
3206      0F46          KYSTOP:
3207                      ;
3208                      ; STOP key
3209                      ;
3210      0F46      3A FBEB          LD      A,(SFTKEY)
3211      0F49      0F          RRCA          ;Move CTRL status to carry
3212      0F4A      0F          RRCA
3213      0F4B      3E 03          LD      A,3          ;Assume CTRL pressed also
3214      0F4D      30 01          JR      NC,KYSTP1          ;Good assumption
3215      0F4F      3C          INC      A          ;CTRL not pressed, just treat as pause
3216      0F50          KYSTP1:
3217      0F50      32 FC9B          LD      (INTFLG),A
  
```

```

3218      0F53      38 0F              JR      C,GENCLK      ;Only generate click if pause
3219      0F55              PUTCHR:
3220              ;
3221              ; Put one character in key buffer.
3222              ;
3223      0F55      2A F3F8              LD      HL,(PUTPNT)      ;Load PUTPNT in [HL]
3224      0F58      77              LD      (HL),A              ;Save the character to buffer
3225      0F59      CD 10C2              CALL   UPDATE          ;Increment PUTPNT
3226      0F5C      3A F3FA              LD      A,(GETPNT)      ;Load lower 8bit of GETPNT
3227      0F5F      BD              CP      L                  ;Compare it with new PUTPNT
3228      0F60      C8              RET      Z                  ;If same skip next step
3229      0F61      22 F3F8              LD      (PUTPNT),HL      ;Save HL in PUTPNT
3230      0F64              GENCLK:
3231      0F64      3A F3DB              LD      A,(CLIKSW)      ;Key click enabled?
3232      0F67      A7              AND      A
3233      0F68      C8              RET      Z                  ;No
3234      0F69      3A FBD9              LD      A,(CLIKFL)      ;Already generated?
3235      0F6C      A7              AND      A
3236      0F6D      C0              RET      NZ                  ;Yes, don't click any more
3237      0F6E      3E 0F              LD      A,0FH
3238      0F70      32 FBD9              LD      (CLIKFL),A      ;Set flag to disable more clicks
3239      0F73      D3 AB              OUT     (PPI.CM),A
3240      0F75      3E 0A              LD      A,0AH
3241      0F77              CLICKW:
3242      0F77      3D              DEC      A
3243      0F78      20 FD              JR      NZ,CLICKW
3244      0F7A              CHGSND:
3245      0F7A      A7              AND      A
3246      0F7B      3E 0E              LD      A,0EH          ;Assume 'turn off'
3247      0F7D      28 01              JR      Z,CGSND1      ;Good assumption
3248      0F7F      3C              INC      A                  ;Change to 'turn on'

```



```

3249      0F80          CGSND1:
3250      0F80      D3 AB          OUT      (PPI.CM),A
3251      0F82      C9          RET
3252      0F83          KYKANA:
3253          ;
3254          ; KANA key pressed while KANA lock is active
3255          ;
3256      0F83      3A FCAD          LD      A,(KANAMD)          ;JIS or AIUEO?
3257      0F86      A7          AND      A          ;Affect Z flag
3258      0F87      3A FBEB          LD      A,(SFTKEY)          ;Check shift key
3259      0F8A      0F          RRCA          ;Affect Carry flag
3260      0F8B      28 0A          JR      Z,KAIUEO          ;AIUEO order
3261      0F8D      21 101D          LD      HL,KANJNO
3262      0F90      38 0D          JR      C,KYKAN1
3263      0F92      21 104D          LD      HL,KANJSF
3264      0F95      18 08          JR      KYKAN1
3265      0F97          KAIUEO:
3266          ;
3267      0F97      21 0FBD          LD      HL,KANANO          ;Assume shift not pressed
3268      0F9A      38 03          JR      C,KYKAN1          ;Good assumption
3269      0F9C      21 0FED          LD      HL,KANASF
3270      0F9F          KYKAN1:
3271      0F9F      06 00          LD      B,0
3272      0FA1      09          ADD      HL,BC
3273      0FA2      01 0F55          LD      BC,PUTCHR          ;Push jump address
3274      0FA5      C5          PUSH     BC
3275      0FA6      3A FCAB          LD      A,(CAPST)          ;Capital lock (katakana) active?
3276      0FA9      A7          AND      A
3277      0FAA      7E          LD      A,(HL)
3278      0FAB      C0          RET      NZ          ;active
3279      0FAC      FE A6          CP      165+1          ;Special characters?

```

```
3280 0FAE D8 RET C ;Yes, no conversion necessary
3281 0FAF FE B0 CP 0B0H
3282 0FB1 C8 RET Z
3283 0FB2 FE DE CP 0DEH
3284 0FB4 D0 RET NC
3285 0FB5 D6 20 SUB ' ' ;Assume first half
3286 0FB7 FE A0 CP 191-32+1 ;Really first half
3287 0FB9 D8 RET C ;Good assumption
3288 0FBA C6 40 ADD A,32+32 ;Compensate
3289 0FBC C9 RET
3290 0FBD KANANO:
3291 ; Kana table (AIUEO order, un-shifted)
3292 ;
3293 0FBD C9 B1 B2 B3 DB 0C9H,0B1H,0B2H,0B3H,0B4H,0B5H,0C5H
3294 0FC1 B4 B5 C5
3295 0FC4 C6 C7 C8 D7 DB 0C6H,0C7H,0C8H,0D7H,0D8H,0D9H,0DAH
3296 0FC8 D8 D9 DA
3297 0FCB DB D3 DE DF DB 0DBH,0D3H,0DEH,0DFH,0D6H,0DCH,0A6H
3298 0FCF D6 DC A6
3299 0FD2 DD BB C4 C2 DB 0DDH,0BBH,0C4H,0C2H,0BDH,0B8H,0BEH
3300 0FD6 BD B8 BE
3301 0FD9 BF CF CC D0 DB 0BFH,0CFH,0CCH,0D0H,0D1H,0D2H,0D5H
3302 0FDD D1 D2 D5
3303 0FE0 D4 CD CE B6 DB 0D4H,0CDH,0CEH,0B6H,0B9H,0BCH,0BAH
3304 0FE4 B9 BC BA
3305 0FE7 CB C3 B7 C1 DB 0CBH,0C3H,0B7H,0C1H,0CAH,0C0H
3306 0FEB CA C0
3307 0FED KANASF:
3308 ; Shifted
3309 ;
3310 0FED C9 A7 A8 A9 DB 0C9H,0A7H,0A8H,0A9H,0AAH,0ABH,0C5H
```

3311	0FF1	AA AB C5		
3312	0FF4	C6 C7 C8 D7	DB	0C6H,0C7H,0C8H,0D7H,0D8H,0D9H,0DAH
3313	0FF8	D8 D9 DA		
3314	0FFB	A2 D3 B0 A3	DB	0A2H,0D3H,0B0H,0A3H,0AEH,0A4H,0A1H
3315	0FFF	AE A4 A1		
3316	1002	A5 BB C4 AF	DB	0A5H,0BBH,0C4H,0AFH,0BDH,0B8H,0BEH
3317	1006	BD B8 BE		
3318	1009	BF CF CC D0	DB	0BFH,0CFH,0CCH,0D0H,0D1H,0D2H,0ADH
3319	100D	D1 D2 AD		
3320	1010	AC CD CE B6	DB	0ACH,0CDH,0CEH,0B6H,0B9H,0BCH,0BAH
3321	1014	B9 BC BA		
3322	1017	CB C3 B7 C1	DB	0CBH,0C3H,0B7H,0C1H,0CAH,0C0H
3323	101B	CA C0		
3324	101D			
3325			KANJNO:	
3326			;	Kana table JIS order, un-shifted
3327	101D	DC C7 CC B1	DB	0DCH,0C7H,0CCH,0B1H,0B3H,0B4H,0B5H
3328	1021	B3 B4 B5		
3329	1024	D4 D5 D6 CE	DB	0D4H,0D5H,0D6H,0CEH,0CDH,0B0H,0DEH
3330	1028	CD B0 DE		
3331	102B	DF DA B9 D1	DB	0DFH,0DAH,0B9H,0D1H,0C8H,0D9H,0D2H
3332	102F	C8 D9 D2		
3333	1032	DB C1 BA BF	DB	0DBH,0C1H,0BAH,0BFH,0BCH,0B2H,0CAH
3334	1036	BC B2 CA		
3335	1039	B7 B8 C6 CF	DB	0B7H,0B8H,0C6H,0CFH,0C9H,0D8H,0D3H
3336	103D	C9 D8 D3		
3337	1040	D0 D7 BE C0	DB	0D0H,0D7H,0BEH,0C0H,0BDH,0C4H,0B6H
3338	1044	BD C4 B6		
3339	1047	C5 CB C3 BB	DB	0C5H,0CBH,0C3H,0BBH,0DDH,0C2H
3340	104B	DD C2		
3341	104D		KANJSF:	

3342					;	Shifted	
3343					;		
3344	104D	A6	C7	CC	A7	DB	0A6H,0C7H,0CCH,0A7H,0A9H,0AAH,0ABH
3345	1051	A9	AA	AB			
3346	1054	AC	AD	AE	CE	DB	0ACH,0ADH,0AEH,0CEH,0CDH,0B0H,0DEH
3347	1058	CD	B0	DE			
3348	105B	A2	DA	B9	A3	DB	0A2H,0DAH,0B9H,0A3H,0A4H,0A1H,0A5H
3349	105F	A4	A1	A5			
3350	1062	DB	C1	BA	BF	DB	0DBH,0C1H,0BAH,0BFH,0BCH,0A8H,0CAH
3351	1066	BC	A8	CA			
3352	1069	B7	B8	C6	CF	DB	0B7H,0B8H,0C6H,0CFH,0C9H,0D8H,0D3H
3353	106D	C9	D8	D3			
3354	1070	D0	D7	BE	C0	DB	0D0H,0D7H,0BEH,0C0H,0BDH,0C4H,0B6H
3355	1074	BD	C4	B6			
3356	1077	C5	CB	C3	BB	DB	0C5H,0CBH,0C3H,0BBH,0DDH,0AFH
3357	107B	DD	AF				

```
3358
3359                ;
3360    107D          KYGRAP:
3361                ;
3362                ; Graphic characters
3363                ;
3364    107D    06 00          LD      B,0
3365    107F    21 1092        LD      HL,GRPTAB
3366    1082    09            ADD     HL,BC
3367    1083    7E            LD      A,(HL)          ;Get from graphic key table
3368    1084    A7            AND     A              ;Should generate some code
3369    1085    C8            RET     Z              ;No
3370    1086    FE 80        CP      80H           ;1 byte code?
3371    1088    F5            PUSH    AF
3372    1089    3E 01        LD      A,1           ;Assume not
3373    108B    DC 0F55        CALL    C,PUTCHR      ;Was 2 byte code, put header byte
3374    108E    F1            POP     AF
3375    108F    C3 0F55        JP      PUTCHR
3376                ;
3377    1092          GRPTAB:
3378    1092    4F 47 41 42    DB      4FH,47H,41H,42H,43H,44H,45H
3379    1096    43 44 45
3380    1099    46 4D 4E 57    DB      46H,4DH,4EH,57H,00H,49H,00H
3381    109D    00 49 00
3382    10A0    84 82 81 85    DB      84H,82H,81H,85H,5FH,5DH,80H
3383    10A4    5F 5D 80
3384    10A7    83 00 5B 5A    DB      83H,00H,5BH,5AH,54H,58H,55H
3385    10AB    54 58 55
3386    10AE    53 4A 56 00    DB      53H,4AH,56H,00H,00H,5EH,4BH
3387    10B2    00 5E 4B
3388    10B5    00 00 50 00    DB      00H,00H,50H,00H,52H,4CH,59H
```

```

3389      10B9      52 4C 59
3390      10BC      00 51 00 5C          DB      00H,51H,00H,5CH,48H,00H
3391      10C0      48 00
3392                                ;
3393      10C2                                UPDATE:
3394                                ;
3395                                ; Update pointer
3396                                ;
3397      10C2      23                                INC      HL
3398      10C3      7D                                LD        A,L
3399      10C4      FE 18                            CP        18H          ;Check buffer boundary
3400      10C6      C0                                RET        NZ
3401      10C7      21 FBF0                          LD        HL,KEYBUF
3402      10CA      C9                                RET
3403      10CB                                CHGET:
3404                                ;
3405                                ; Get one character from keyboard
3406                                ;
3407      10CB      E5                                PUSH     HL
3408      10CC      D5                                PUSH     DE
3409      10CD      C5                                PUSH     BC
3410      10CE      CD FDC2                          CALL     H.CHGE
3411      10D1      CD 0D6A                          CALL     CHSNS          ;Character already there?
3412      10D4      20 0B                            JR        NZ,CHGET2     ;Yes, do not touch cursor
3413      10D6      CD 09DA                          CALL     CKDPC0         ;Display cursor if disabled
3414      10D9                                CHGET1:
3415      10D9      CD 0D6A                          CALL     CHSNS          ;Any character in buffer?
3416      10DC      28 FB                            JR        Z,CHGET1     ;No, wait
3417      10DE      CD 0A27                          CALL     CKERC0         ;Erase cursor if disabled
3418      10E1                                CHGET2:
3419      10E1      21 FC9B                          LD        HL,INTFLG

```

3420	10E4	7E	LD	A,(HL)	
3421	10E5	FE 04	CP	4	;Code for pause?
3422	10E7	20 02	JR	NZ,CHGET3	;No
3423	10E9	36 00	LD	(HL),0	;Clear this
3424	10EB		CHGET3:		
3425	10EB	2A F3FA	LD	HL,(GETPNT)	
3426	10EE	4E	LD	C,(HL)	;Save pressed key
3427	10EF	CD 10C2	CALL	UPDATE	;Update [GETPNT]
3428	10F2	22 F3FA	LD	(GETPNT),HL	;Set new [GETPNT]
3429	10F5	79	LD	A,C	;Pass result to Acc
3430	10F6	C3 08DB	JP	PBDHRT	
3431	10F9		CKCNTC:		
3432			;		
3433			;	Check ctl-C	
3434			;		
3435	10F9	E5	PUSH	HL	
3436	10FA	21 0000	LD	HL,0	;To disable CONTinuing
3437	10FD	CD 03FB	CALL	ISCNTC	
3438	1100	E1	POP	HL	
3439	1101	C9	RET		
3440			;		
3441			SUBTTL	- MSXIO - Music routines	

```
3442
3443      1102      WRTPSG:
3444      ;
3445      ; Write data to specified register of GI sound chip
3446      ; Entry - (E)=data,(A)=register number
3447      ; Exit - All regs preserved
3448      ;
3449      ; GI Reg# - usage
3450      ;
3451      ; 0      voice A fine tune
3452      ; 1      voice A coarse tune
3453      ; 2      voice B fine tune
3454      ; 3      voice B coarse tune
3455      ; 4      voice C fine tune
3456      ; 5      voice C coarse tune
3457      ; 7 B7,B6      = Reg 14,15 Input Output flags
3458      ;   B5,B4,B3 = voice C,B,A noise enable (0=enabled)
3459      ;   B2,B1,B0 = voice C,B,A tone enable (0=enabled)
3460      ; 8      voice A volume (0..15 = volume, 16=use envelope)
3461      ; 9      voice B volume (0..15 = volume, 16=use envelope)
3462      ; 10     voice C volume (0..15 = volume, 16=use envelope)
3463      ; 11-12 envelope period
3464      ; 13     envelope shape (0..15)
3465      ; 14     joystick 1 port
3466      ; 15     joystick 2 port
3467      ;
3468      1102      F3      DI
3469      1103      D3 A0      OUT      (PSG.LW),A      ;LATCH ADDRESS
3470      1105      F5      PUSH     AF
3471      1106      7B      LD       A,E
3472      1107      D3 A1      OUT      (PSG.DW),A      ;OUTPUT DATA
```



```
3473      1109      FB              EI
3474      110A      F1              POP      AF
3475      110B      C9              RET
3476      110C              INGI:
3477                      ;
3478                      ; Input data from PAD
3479                      ;
3480      110C      3E 0E              LD      A,PSG.PA
3481      110E              RDPSPG:
3482      110E      D3 A0              OUT      (PSG.LW),A
3483      1110      DB A2              IN      A,(PSG.DR)
3484      1112      C9              RET
3485      1113              BEEP:
3486                      ;
3487                      ; BEEP causes a 'bell' sound
3488                      ;
3489                      ; Exit - all registers are destroyed
3490                      ;
3491      1113      AF              XOR      A              ;[A]=fine tune register for voice A
3492      1114      1E 55              LD      E,01010101B      ;data to be written on R0
3493      1116      CD 1102              CALL    WRTPSG
3494      1119      5F              LD      E,A              ;0 to coarse tune register
3495      111A      3C              INC      A
3496      111B      CD 1102              CALL    WRTPSG      ;R1 coarse
3497      111E      1E BE              LD      E,10111110B      ;enable voice [A] tone
3498      1120      3E 07              LD      A,7              ;[A]=voice enable register
3499      1122      CD 1102              CALL    WRTPSG      ;R7
3500      1125      5F              LD      E,A              ;set volume to 7
3501      1126      3C              INC      A              ;[A]=voice A volume register
3502      1127      CD 1102              CALL    WRTPSG      ;R8
3503      112A      01 07D0              LD      BC,07D0H
```

```

3504      112D      CD 1133      CALL      CSDLY1
3505      1130      C3 04BD      JP        GICINI      ;reset GI sound chip
3506      1133      CSDLY1:
3507      ;
3508      ; Delay by [BC]
3509      ;
3510      1133      0B      DEC      BC
3511      1134      E3      EX      (SP),HL
3512      1135      E3      EX      (SP),HL
3513      1136      78      LD      A,B
3514      1137      B1      OR      C
3515      1138      20 F9      JR      NZ,CSDLY1
3516      113A      C9      RET
3517      ;
3518      113B      ACTION:
3519      ;
3520      ; Get action information from specified music queue. Perform
3521      ; action with synchronization. Called by interrupt routine
3522      ; in time.
3523      ;
3524      ; - Action information -
3525      ;
3526      ;      ITEM 1 - 2  BYTES
3527      ;
3528      ;      + Number of bytes that follow this item
3529      ;      |
3530      ;      NNNTTTTTTTTTTTTTT
3531      ;      |
3532      ;      +Period of time
3533      ;
3534      ; ITEM 2, 3, 4 - FROM 1 TO 5 BYTES

```

```

3535      ;
3536      ; IF HO 2 BITS = 0 then this is the HO byte of the tone period.
3537      ; IF HO 2 BITS = 2 then this is just a volume control byte.
3538      ; IF BIT 4 IS ON, envelope control is in effect, and bits
3539      ; 0-3 give shape number of envelope.
3540      ; IF BIT 4 IS OFF, BITS 0-3 give amplitude number.
3541      ; IF HO 2 BITS = 3 THEN this byte will be followed by a 2 byte
3542      ; envelope period, HO first.
3543      ;
3544      ; ENTRY - (A)=Channel count number (0..2)
3545      ;
3546      113B 47      LD      B,A          ;Save channel number
3547      113C CD 1470 CALL    GETVCP      ;Get pointer into vcb of channel
3548      113F 2B      DEC     HL
3549      1140 56      LD      D,(HL)
3550      1141 2B      DEC     HL
3551      1142 5E      LD      E,(HL)      ;[DE]=countdown timer for voice
3552      1143 1B      DEC     DE          ;Decrement timer
3553      1144 73      LD      (HL),E      ;Put it back to first
3554      1145 23      INC     HL
3555      1146 72      LD      (HL),D
3556      1147 7A      LD      A,D
3557      1148 B3      OR      E
3558      1149 C0      RET     NZ          ;No action if not zero
3559      114A 78      LD      A,B          ;Voice 0 uses queue 0
3560      114B 32 FB3E LD      (QUEUEEN),A ;Set queue ID for further 'CALL XGETQ'
3561      114E CD 11E2 CALL    XGETQ
3562      1151 FE FF   CP      0FFH
3563      1153 28 5B   JR      Z,VOICOF    ;branch if EOF marker
3564      1155 57      LD      D,A          ;SAVE IN [D]
3565      1156 E6 E0   AND     0E0H        ;Get number of following items

```

3566	1158	07	RLCA		
3567	1159	07	RLCA		
3568	115A	07	RLCA		
3569	115B	4F	LD	C,A	;Save in [C]
3570	115C	7A	LD	A,D	
3571	115D	E6 1F	AND	1FH	;GET LO 5 BITS OF [D]
3572	115F	77	LD	(HL),A	;Set MSB of new countdown
3573	1160	CD 11E2	CALL	XGETQ	;Get LSB of new countdown
3574	1163	2B	DEC	HL	
3575	1164	77	LD	(HL),A	;Set it
3576	1165	0C	INC	C	
3577	1166		MORACT:		
3578	1166	0D	DEC	C	;Done all items?
3579	1167	C8	RET	Z	;Yes
3580	1168	CD 11E2	CALL	XGETQ	;Get next item from queue
3581	116B	57	LD	D,A	;Save this to [D]
3582	116C	E6 C0	AND	0C0H	;Get H0 2 bits
3583	116E	20 11	JR	NZ,XVOL	;Execute volume action
3584					
3585					; Set tone
3586					
3587	1170	CD 11E2	CALL	XGETQ	;Get low byte for tone
3588	1173	5F	LD	E,A	
3589	1174	78	LD	A,B	;Get back voice number
3590	1175	07	RLCA		;X 2
3591	1176	CD 1102	CALL	WRTPSG	;Output fine tune register
3592	1179	3C	INC	A	;Point to coarse tune register
3593	117A	5A	LD	E,D	;Restore saved value
3594	117B	CD 1102	CALL	WRTPSG	;Output coarse tune reg
3595	117E	0D	DEC	C	;Decrement since we took 2 bytes from queue
3596	117F	18 E5	JR	MORACT	

```
3597 1181          XVOL:
3598                ;
3599 1181 67          LD      H,A          ;save it in [H]
3600 1182 E6 80      AND     80H        ;BIT 7 SET?
3601 1184 28 0F      JR      Z,XEPER
3602                ;
3603                ; Set volume
3604                ;
3605 1186 5A          LD      E,D        ;[A] has junk in ho which shouldn't matter
3606 1187 78          LD      A,B        ;Get back voice number
3607 1188 C6 08      ADD     A,8         ;Regs 8,9,10
3608 118A CD 1102    CALL    WRTPSG      ;Output amplitude reg
3609 118D 7B          LD      A,E
3610 118E E6 10      AND     10H        ;Check envelope generate bit
3611 1190 3E 0D      LD      A,0DH      ;Reg 13 for shape
3612 1192 C4 1102    CALL    NZ,WRTPSG   ;Set envelope shape if enabled
3613 1195          XEPER:
3614                ;
3615                ; Set envelope period
3616                ;
3617 1195 7C          LD      A,H
3618 1196 E6 40      AND     01000000B   ;See if set envelope period
3619 1198 28 CC      JR      Z,MORACT    ;No
3620 119A CD 11E2    CALL    XGETQ      ;Get ho byte of envelope period
3621 119D 57          LD      D,A
3622 119E CD 11E2    CALL    XGETQ      ;Get low byte of envelope period
3623 11A1 5F          LD      E,A
3624 11A2 3E 0B      LD      A,0BH      ;Register 11 for fine tune
3625 11A4 CD 1102    CALL    WRTPSG
3626 11A7 3C          INC     A          ;Point to coarse tune
3627 11A8 5A          LD      E,D
```

```
3628 11A9 CD 1102          CALL  WRTPSG
3629 11AC 0D              DEC    C
3630 11AD 0D              DEC    C
3631 11AE 18 B6          JR      MORACT
3632 11B0                VOICOF:
3633                      ;
3634                      ; Comes here when an EOF mark has been found for a specified
3635                      ; channel
3636                      ;
3637 11B0 78              LD      A,B
3638 11B1 C6 08          ADD     A,8           ;Set appropriate reg #
3639 11B3 1E 00          LD      E,0
3640 11B5 CD 1102          CALL  WRTPSG           ;Turn off volume
3641 11B8 04              INC     B
3642 11B9 21 FB3F          LD      HL,MUSICF
3643 11BC AF              XOR     A
3644 11BD 37              SCF
3645 11BE                RSTFL1:
3646 11BE 17              RLA
3647 11BF 10 FD          DJNZ    RSTFL1
3648 11C1 A6              AND     (HL)           ;Get that bit
3649 11C2 AE              XOR     (HL)           ;Turn it off
3650 11C3 77              LD      (HL),A
3651 11C4                STRTMS:
3652                      ;
3653                      ; STRTMS starts the background music task if:
3654                      ; 1) - it is currently idle (MUSICF=0) and
3655                      ; 2) - there is work queued for it (PLYCNT .GTR. 0)
3656                      ;
3657 11C4 3A FB3F          LD      A,(MUSICF)
3658 11C7 B7              OR      A
```

```
3659 11C8 C0 RET NZ ;return if background task is active
3660 11C9 21 FB40 LD HL,PLYCNT
3661 11CC 7E LD A,(HL)
3662 11CD B7 OR A
3663 11CE C8 RET Z ;return if nothing for it to do
3664 11CF 35 DEC (HL) ;1 less thing for it to do
3665 11D0 21 0001 LD HL,1
3666 11D3 22 FB41 LD (VCBA),HL ;start it playing now
3667 11D6 22 FB66 LD (VCBB),HL
3668 11D9 22 FB8B LD (VCBC),HL
3669 11DC 3E 07 LD A,0111B ;Trigger!
3670 11DE 32 FB3F LD (MUSICF),A
3671 11E1 C9 RET
3672 11E2 XGETQ:
3673 ;
3674 11E2 3A FB3E LD A,(QUEUEN) ;Get queue ID
3675 11E5 E5 PUSH HL
3676 11E6 D5 PUSH DE
3677 11E7 C5 PUSH BC
3678 11E8 CD 14AD CALL GETQ ;Get a byte from a specified queue
3679 11EB C3 08DB JP PBDHRT ;pop H, D, B and return
3680 ;
3681 SUBTTL - MSXIO - Joystick and Paddle interface
```

```
3682
3683     11EE                GTSTCK:
3684                ;
3685     11EE    3D          DEC    A
3686     11EF    FA 1200      JP     M,KYSTCK      ;STICK(0) - read cursor keys
3687     11F2    CD 120C      CALL   SLSTCK      ;Read joystick
3688     11F5    21 1233      LD     HL,STKTBL
3689     11F8                STICK1:
3690     11F8    E6 0F        AND    0FH
3691     11FA    5F          LD     E,A
3692     11FB    16 00        LD     D,0
3693     11FD    19          ADD    HL,DE
3694     11FE    7E          LD     A,(HL)
3695     11FF    C9          RET
3696     1200                KYSTCK:
3697                ;
3698     1200    CD 1226      CALL   GTROW8      ;Read keyboard
3699     1203    0F          RRCA                ;Move cursor status to lower four bits
3700     1204    0F          RRCA
3701     1205    0F          RRCA
3702     1206    0F          RRCA
3703     1207    21 1243      LD     HL,KSTKTBL
3704     120A    18 EC        JR     STICK1
3705     120C                SLSTCK:
3706                ;
3707                ; Select proper joystick and read from it
3708                ;
3709     120C    47          LD     B,A
3710     120D    3E 0F        LD     A,PSG.PB
3711     120F    F3          DI
3712     1210    CD 110E      CALL   RDPSG      ;Read what is currently output to port B
```


- MSXIO - Joystick and Paddle interface

```

3713      1213      10 06                DJNZ      SLSTC1          ;STICK(1)
3714      1215      E6 DF                AND        0DFH          ;Make sure P8 is low state
3715      1217      F6 4C                OR          4CH           ;Select joystick 2, enable P6,P7
3716      1219      18 04                JR          SLSTC2
3717      121B                      SLSTC1:
3718                                ;
3719      121B      E6 AF                AND        0AFH          ;Select joystick 1, make sure P8 is low state
3720      121D      F6 03                OR          3             ;Enable P6,P7
3721      121F                      SLSTC2:
3722      121F      D3 A1                OUT        (PSG.DW),A
3723      1221      CD 110C              CALL       INGI          ;Read status of joystick port
3724      1224      FB                    EI
3725      1225      C9                    RET
3726      1226                      GTROW8:
3727                                ;
3728                                ; Get keyboard's 8th row, bit assignments are as follows.
3729                                ;
3730                                ; RDULxxxS
3731                                ; |||| |
3732                                ; |||| +- space
3733                                ; |||+----- left
3734                                ; ||+----- up
3735                                ; |+----- down
3736                                ; +----- right
3737                                ;
3738      1226      F3                    DI
3739      1227      DB AA                IN          A,(PPI.CR)
3740      1229      E6 F0                AND        0F0H
3741      122B      C6 08                ADD         A,8
3742      122D      D3 AA                OUT        (PPI.CW),A
3743      122F      DB A9                IN          A,(PPI.BR)

```

- MSXIO - Joystick and Paddle interface

3744	1231	FB	EI		
3745	1232	C9	RET		
3746					
3747	1233		STKTB:		
3748	1233	00	DB	0	;RLBF
3749	1234	05	DB	5	;RLB
3750	1235	01	DB	1	;RL F
3751	1236	00	DB	0	;RL
3752	1237	03	DB	3	;R BF
3753	1238	04	DB	4	;R B
3754	1239	02	DB	2	;R F
3755	123A	03	DB	3	;R
3756	123B	07	DB	7	; LBF
3757	123C	06	DB	6	; LB
3758	123D	08	DB	8	; L F
3759	123E	07	DB	7	; L
3760	123F	00	DB	0	; BF
3761	1240	05	DB	5	; B
3762	1241	01	DB	1	; F
3763	1242	00	DB	0	;
3764					
3765	1243		STKTB:		
3766	1243	00	DB	0	;RBF
3767	1244	03	DB	3	;RBF
3768	1245	05	DB	5	;RB L
3769	1246	04	DB	4	;RB
3770	1247	01	DB	1	;R FL
3771	1248	02	DB	2	;R F
3772	1249	00	DB	0	;R L
3773	124A	03	DB	3	;R
3774	124B	07	DB	7	; BFL

- MSXIO - Joystick and Paddle interface

```

3775      124C      00                      DB      0          ; BF
3776      124D      06                      DB      6          ; B L
3777      124E      05                      DB      5          ; B
3778      124F      08                      DB      8          ; FL
3779      1250      01                      DB      1          ; F
3780      1251      07                      DB      7          ; L
3781      1252      00                      DB      0          ;
3782                                     ;
3783      1253          GTTRIG:
3784                                     ;
3785      1253      3D                      DEC      A
3786      1254      FA 126C                JP      M,KEYTRG      ;STRIG(0), use keyboard
3787      1257      F5                      PUSH     AF
3788      1258      E6 01                  AND      1
3789      125A      CD 120C                CALL     SLSTCK      ;Read joystick
3790      125D      C1                      POP      BC
3791      125E      05                      DEC      B
3792      125F      05                      DEC      B
3793      1260      06 10                  LD      B,10H      ;Prepare mask pattern for trigger A
3794      1262      FA 1267                JP      M,TRIG1
3795      1265      06 20                  LD      B,' '      ;Prepare mask pattern for trigger B
3796      1267          TRIG1:
3797      1267      A0                      AND      B          ;Extract trigger status
3798      1268          TRIG2:
3799      1268      D6 01                  SUB      1          ;Return 255 if [Acc]=0, 0 if non-0
3800      126A      9F                      SBC      A,A
3801      126B      C9                      RET
3802      126C          KEYTRG:
3803                                     ;
3804      126C      CD 1226                CALL     GTROW8      ;Read keyboard
3805      126F      E6 01                  AND      1          ;Extract space status

```

```

3806      1271      18 F5          JR      TRIG2
3807      1273          GTPDL:
3808          ;
3809          ; Get value of paddle
3810          ;
3811          ; Input parameters (passed via [Acc])
3812          ;
3813          ; 1 - Paddle A connected to joystick port 1
3814          ; 2 - Paddle A connected to joystick port 2
3815          ; 3 - Paddle B connected to joystick port 1
3816          ; 4 - Paddle B connected to joystick port 2
3817          ; 5 - Paddle C connected to joystick port 1
3818          ; 6 - Paddle C connected to joystick port 2
3819          ; 7 - Paddle D connected to joystick port 1
3820          ; 8 - Paddle D connected to joystick port 2
3821          ; 9 - Paddle E connected to joystick port 1
3822          ; 10 - Paddle E connected to joystick port 2
3823          ; 11 - Paddle F connected to joystick port 1
3824          ; 12 - Paddle F connected to joystick port 2
3825          ;
3826      1273      3C          INC      A          ;Force parameter 2 based
3827      1274      A7          AND      A
3828      1275      1F          RRA
3829      1276      F5          PUSH     AF          ;Save port # (carry reset if port 1)
3830      1277      47          LD       B,A
3831      1278      AF          XOR      A
3832      1279      37          SCF
3833      127A          PDL1:
3834      127A      17          RLA          ;Form mask pattern
3835      127B      10 FD      DJNZ     PDL1
3836      127D      47          LD       B,A          ;Set mask pattern
  
```

```

3837 127E F1 POP AF
3838 127F 0E 10 LD C,10H ;Assume port 1
3839 1281 11 03AF LD DE,03AFH
3840 1284 30 05 JR NC,PDLPl ;Good assumption
3841 1286 0E 20 LD C,' '
3842 1288 11 4C9F LD DE,4C9FH
3843 128B PDLPl:
3844 128B 3E 0F LD A,PSG.PB
3845 128D F3 DI
3846 128E CD 110E CALL RDPSG ;Get current port B content
3847 1291 A3 AND E
3848 1292 B2 OR D
3849 1293 B1 OR C
3850 1294 D3 A1 OUT (PSG.DW),A ;Set trigger high
3851 1296 A9 XOR C
3852 1297 D3 A1 OUT (PSG.DW),A ;Set trigger low again
3853 1299 3E 0E LD A,0EH
3854 129B D3 A0 OUT (PSG.LW),A
3855 129D 0E 00 LD C,0 ;Initialize counter
3856 129F PDL2:
3857 129F DB A2 IN A,(PSG.DR)
3858 12A1 A0 AND B ;End of pulse?
3859 12A2 28 05 JR Z,PDL3 ;Yes
3860 12A4 0C INC C ;Bump counter
3861 12A5 C2 129F JP NZ,PDL2 ;No overflow yet
3862 12A8 0D DEC C ;Make it 255
3863 12A9 PDL3:
3864 12A9 FB EI
3865 12AA 79 LD A,C ;Return counted value
3866 12AB C9 RET
3867 12AC GTPAD:

```

MSXIO - Joystick and Paddle interface

```

3868      ;
3869      ; Read touch pad (NEC PC-6051 compatible)
3870      ;
3871      ; Input parameter (passed via [Acc])
3872      ;
3873      ; 0 - sense touch pad status ---
3874      ; 1 - return X coordinate      |for touch pad connected
3875      ; 2 - return Y coordinate      |to joystick port 1
3876      ; 3 - return switch status ----
3877      ;
3878      ; 4 - sense touch pad status ---
3879      ; 5 - return X coordinate      |for touch pad connected
3880      ; 6 - return Y coordinate      |to joystick port 2
3881      ; 7 - return switch status ----
3882      ;
3883      ; Result is returned via [Acc]. As for status, 255 is returned
3884      ; if true, 0 if false.
3885      ;
3886      12AC      FE 04      CP      4      ;Read pad connected to port 1
3887      12AE      11 0CEC      LD      DE,0CECH      ;Assume so
3888      12B1      38 05      JR      C,GTPDP1      ;Good assumption
3889      12B3      11 03D3      LD      DE,03D3H      ;Connected to port 2
3890      12B6      D6 04      SUB      4
3891      12B8      GTPDP1:
3892      12B8      3D      DEC      A      ;Argument=0?
3893      12B9      FA 12C5      JP      M,GTPAD0      ;If so, read pad and return status
3894      12BC      3D      DEC      A
3895      12BD      3A FC9D      LD      A,(PADX)      ;Assume PAD(1) - X coordinate
3896      12C0      F8      RET      M      ;Good assumption
3897      12C1      3A FC9C      LD      A,(PADY)      ;Return Y coordinate
3898      12C4      C8      RET      Z

```

```
3899      12C5          GTPAD0:
3900      12C5      F5          PUSH      AF          ;Save status (minus if PAD(0) specified)
3901      12C6      EB          EX          DE,HL      ;[L]=bits that are not to be modified
3902      12C7      22 F866      LD          (RUNFLG),HL ;[H]=bits that are to be added
3903      12CA      9F          SBC          A,A
3904      12CB      2F          CPL
3905      12CC      E6 40          AND          01000000B
3906      12CE      4F          LD          C,A          ;0 if port 1 specified, 100 octal if port 2
3907      12CF      3E 0F          LD          A,PSG.PB
3908      12D1      F3          DI          ;disable interrupt till done
3909      12D2      CD 110E      CALL         RDPSG
3910      12D5      E6 BF          AND          0BFH
3911      12D7      B1          OR          C
3912      12D8      D3 A1          OUT         (PSG.DW),A ;Select proper port
3913      12DA      F1          POP          AF
3914      12DB      FA 12E8      JP          M,TRYAGN    ;PAD(0) specified
3915      12DE      CD 110C      CALL         INGI
3916      12E1      FB          EI
3917      12E2      E6 08          AND          8
3918      12E4      D6 01          SUB          1
3919      12E6      9F          SBC          A,A
3920      12E7      C9          RET
3921      12E8          TRYAGN:
3922      ;
3923      12E8      0E 00          LD          C,0          ;
3924      12EA      CD 1332      CALL         REDPAD      ;inz
3925      12ED      CD 1332      CALL         REDPAD      ;sense Panel input and select X
3926      12F0      38 28          JR          C,PADX1    ;branch if no input
3927      12F2      CD 1320      CALL         REDCOD      ;read first coordinate
3928      12F5      38 23          JR          C,PADX1    ;branch if input released
3929      12F7      D5          PUSH         DE          ;save for comparison
```

3930	12F8	CD 1320	CALL	REDCOD	;read another input
3931	12FB	C1	POP	BC	;restore previos coord
3932	12FC	38 1C	JR	C,PADX1	;branch if input released
3933	12FE	78	LD	A,B	
3934	12FF	92	SUB	D	;[A]=ABS(X0-X1)
3935	1300	30 02	JR	NC,NONEG1	
3936	1302	2F	CPL		
3937	1303	3C	INC	A	
3938	1304		NONEG1:		
3939	1304	FE 05	CP	5	;less than 5?
3940	1306	30 E0	JR	NC,TRYAGN	;no, try again
3941	1308	79	LD	A,C	
3942	1309	93	SUB	E	;[A]=ABS(Y0-Y1)
3943	130A	30 02	JR	NC,NONEG2	
3944	130C	2F	CPL		
3945	130D	3C	INC	A	
3946	130E		NONEG2:		
3947	130E	FE 05	CP	5	;less than 5
3948	1310	30 D6	JR	NC,TRYAGN	;no, try again
3949	1312	7A	LD	A,D	
3950	1313	32 FC9D	LD	(PADX),A	;update coordinate [X]
3951	1316	7B	LD	A,E	
3952	1317	32 FC9C	LD	(PADY),A	;update coordinate [Y]
3953	131A		PADX1:		
3954	131A	FB	EI		;finally enable interrupt
3955	131B	7C	LD	A,H	;get SENSE input value
3956	131C	D6 01	SUB	1	
3957	131E	9F	SBC	A,A	
3958	131F	C9	RET		;return value
3959	1320		REDCOD:		
3960					;


```
3961                ; Read X,Y coordinate into [D,E]
3962                ;
3963    1320    0E 0A                LD    C,0AH                ;change to channel to [Y] when done
3964    1322    CD 1332            CALL    REDPAD                ;read [X]
3965    1325    D8                RET    C                ;return if input released
3966    1326    55                LD    D,L
3967    1327    D5                PUSH   DE
3968    1328    0E 00                LD    C,0                ;change to [X] after read
3969    132A    CD 1332            CALL    REDPAD                ;read [Y]
3970    132D    D1                POP    DE
3971    132E    5D                LD    E,L                ;store Y read out
3972    132F    AF                XOR    A                ;clear carry
3973    1330    67                LD    H,A                ;force input is OK
3974    1331    C9                RET
3975    1332                REDPAD:
3976                ;
3977                ; Read touch panel input into [L]
3978                ; Carry set if input released during read
3979                ;
3980    1332    CD 135B            CALL    CHKEOC                ;make sure AD completed
3981    1335    06 08                LD    B,8                ;input 8 bits
3982    1337    51                LD    D,C                ;input channel# after done
3983    1338                REDLOP:
3984    1338    CB 82                RES    0,D                ;serial clock(SCK)=1
3985    133A    CB 92                RES    2,D
3986    133C    CD 136D            CALL    OUTGI
3987    133F    CD 110C            CALL    INGI                ;read PAD
3988    1342    67                LD    H,A                ;save SENSE status
3989    1343    1F                RRA
3990    1344    1F                RRA
3991    1345    1F                RRA
```

- MSXIO - Joystick and Paddle interface

```

3992      1346      CB 15              RL      L              ;bit 2 to LSB of [L]
3993      1348      CB C2              SET     0,D           ;SCK=0
3994      134A      CB D2              SET     2,D
3995      134C      CD 136D            CALL    OUTGI
3996      134F      10 E7              DJNZ    REDLOP
3997      1351      CB E2              SET     4,D
3998      1353      CB EA              SET     5,D
3999      1355      CD 136D            CALL    OUTGI          ;initiate another AD
4000      1358      7C                  LD      A,H          ;LSB=SENSE status
4001      1359      1F                  RRA              ;SENSE status to carry
4002      135A      C9                  RET                ;OK if no carry
4003      135B                                CHKEOC:
4004                                ;
4005                                ; Check and wait for EOC
4006                                ;
4007      135B      3E 35              LD      A,00110101B
4008      135D      B1                  OR      C
4009      135E      57                  LD      D,A
4010      135F      CD 136D            CALL    OUTGI          ;reset CS
4011      1362                                EOCCHK:
4012      1362      CD 110C            CALL    INGI
4013      1365      E6 02              AND     2              ;test EOC
4014      1367      28 F9              JR      Z,EOCCHK
4015      1369      CB A2              RES     4,D          ;set CS and return
4016      136B      CB AA              RES     5,D
4017      136D                                OUTGI:
4018                                ;
4019                                ; Output [D] to PAD
4020                                ;
4021      136D      E5                  PUSH    HL
4022      136E      D5                  PUSH    DE

```

```
4023      136F      2A F866          LD      HL,(RUNFLG)      ;Also known as [PADWRK]
4024      1372      7D              LD      A,L
4025      1373      2F              CPL
4026      1374      A2              AND      D
4027      1375      57              LD      D,A
4028      1376      3E 0F           LD      A,PSG.PB
4029      1378      D3 A0           OUT     (PSG.LW),A
4030      137A      DB A2           IN      A,(PSG.DR)
4031      137C      A5              AND      L
4032      137D      B2              OR       D
4033      137E      B4              OR       H
4034      137F      D3 A1           OUT     (PSG.DW),A
4035      1381      D1              POP      DE
4036      1382      E1              POP      HL
4037      1383      C9              RET
4038      ;
4039      SUBTTL - MSXIO - Misc. routines for MSXIO
```

```
4040
4041      1384          STMOTR:
4042      1384      A7          AND      A
4043      1385      FA 1392      JP      M,FLPMOT      ;Flip motor switch
4044      1388          STMOT1:
4045      1388      20 03          JR      NZ,MOTRON
4046      138A      3E 09          LD      A,00001001B      ;Stop motor
4047      138C      C2          DB      0C2H      ;Skip next 2 bytes ('JNZ' instruction)
4048      138D          MOTRON:
4049      138D      3E 08          LD      A,8
4050      138F      D3 AB          OUT     (PPI.CM),A
4051      1391      C9          RET
4052      1392          FLPMOT:
4053      ;
4054      1392      DB AA          IN      A,(PPI.CR)
4055      1394      E6 10          AND     10H
4056      1396      18 F0          JR      STMOT1
4057      1398          NMI:
4058      ;
4059      ; NMI handler
4060      ;
4061      1398      CD FDD6          CALL   H.NMI
4062      139B      ED 45          RETN      ;RETN
```

```
4063
4064                ;
4065    139D          INIFNK:
4066                ;
4067                ; Initialize function key strings
4068                ;
4069    139D    01 00A0          LD      BC,0A0H
4070    13A0    11 F87F          LD      DE,FNKSTR
4071    13A3    21 13A9          LD      HL,FKTABL
4072    13A6    ED B0           LDIR
4073    13A8    C9             RET
4074                ;
4075    13A9          FNKDEF:
4076    13A9    63 6F 6C 6F      DB      "color "
4077    13AD    72 20
4078    13AF          DS      10
4079    13B9    61 75 74 6F      DB      "auto "
4080    13BD    20
4081    13BE          DS      11
4082    13C9    67 6F 74 6F      DB      "goto "
4083    13CD    20
4084    13CE          DS      11
4085    13D9    6C 69 73 74      DB      "list "
4086    13DD    20
4087    13DE          DS      11
4088    13E9    72 75 6E          DB      "run"
4089    13EC    0D              DB      13
4090    13ED          DS      12
4091    13F9    63 6F 6C 6F      DB      "color 15,4,7"
4092    13FD    72 20 31 35
4093    1401    2C 34 2C 37
```

4094	1405	0D	DB	13
4095	1406		DS	3
4096	1409	63 6C 6F 61	DB	"cload"
4097	140D	64		
4098	140E	22	DB	34
4099	140F		DS	10
4100	1419	63 6F 6E 74	DB	"cont"
4101	141D	0D	DB	13
4102	141E		DS	11
4103	1429	6C 69 73 74	DB	"list."
4104	142D	2E		
4105	142E	0D 1E 1E	DB	13,30,30
4106	1431		DS	8
4107	1439	0C	DB	12
4108	143A	72 75 6E	DB	"run"
4109	143D	0D	DB	13
4110	143E		DS	11
4111				
4112	1449		RDVDP:	
4113				
4114	1449	DB 99	IN	A,(VDP.SR)
4115	144B	C9	RET	
4116	144C		RSLREG:	
4117				
4118	144C	DB A8	IN	A,(PPI.AR)
4119	144E	C9	RET	
4120	144F		WSLREG:	
4121				
4122	144F	D3 A8	OUT	(PPI.AW),A
4123	1451	C9	RET	
4124	1452		SNSMAT:	

```
4125      ;
4126      1452      4F      LD      C,A
4127      1453      F3      DI
4128      1454      DB AA      IN      A,(PPI.CR)      ;Get what is currently output to Port C
4129      1456      E6 F0      AND      0F0H      ;Leave higher 4 bits unaffected
4130      1458      81      ADD      A,C
4131      1459      D3 AA      OUT      (PPI.CW),A      ;Select row
4132      145B      DB A9      IN      A,(PPI.BR)      ;Get column information of selected row
4133      145D      FB      EI
4134      145E      C9      RET
4135      145F      ISFLIO:
4136      ;
4137      ; Check if we're doing device I O
4138      ;
4139      145F      CD FEDF      CALL      H.ISFL
4140      1462      E5      PUSH      HL      ;Save [H,L]
4141      1463      2A F864      LD      HL,(PTRFIL)      ;Get file pointer
4142      1466      7D      LD      A,L
4143      1467      B4      OR      H      ;No zero?
4144      1468      E1      POP      HL      ;Restore [H,L]
4145      1469      C9      RET
4146      146A      DCOMPR:
4147      ;
4148      ; COMPAR compares [H,L] with [D,E] unsigned
4149      ;
4150      ; [H,L] less than [D,E] set carry
4151      ; [H,L] = [D,E] set zero
4152      ;
4153      ; [A] is the only register used
4154      ;
4155      146A      7C      LD      A,H
```

4156	146B	92	SUB	D
4157	146C	C0	RET	NZ
4158	146D	7D	LD	A,L
4159	146E	93	SUB	E
4160	146F	C9	RET	
4161	1470		GETVCP:	
4162			;	
4163			; Entry - [A] = voice id (0..2)	
4164			; Exit - [HL] = pointer to QLENGX for voice (within static var buf)	
4165			; [A] = 0. All other registers preserved.	
4166			;	
4167	1470	2E 02	LD	L,2
4168	1472	18 03	JR	GETVC1
4169	1474		GETVC2:	
4170			;	
4171			; Entry - [L] = desired displacement into voice buffer	
4172			; Exit - [HL] = pointer to desired variable for voice VOICEN	
4173			; [A] = 0. All other registers preserved.	
4174			;	
4175	1474	3A FB38	LD	A,(VOICEN)
4176	1477		GETVC1:	
4177			;	
4178			; Entry - [A] = voice id (0..2)	
4179			; [L] = desired displacement into voice buffer	
4180			; Exit - [HL] = pointer to desired variable for voice VOICEN	
4181			; [A] = 0. All other registers preserved.	
4182			;	
4183	1477	D5	PUSH	DE
4184	1478	11 FB41	LD	DE,VCBA
4185	147B	26 00	LD	H,0
4186	147D	19	ADD	HL,DE


```

4187      147E      B7              OR      A
4188      147F      28 07          JR      Z,GETVCX
4189      1481      11 0025        LD      DE,25H          ;VCB size
4190      1484              GETVCL:
4191      1484      19              ADD     HL,DE
4192      1485      3D              DEC     A
4193      1486      20 FC          JR      NZ,GETVCL
4194      1488              GETVCX:
4195      1488      D1              POP     DE
4196      1489      C9              RET
4197      148A              PHYDIO:
4198              ;
4199      148A      CD FFA7          CALL    H.PHYD
4200      148D      C9              RET
4201      148E              FORMAT:
4202              ;
4203      148E      CD FFAC          CALL    H.FORM
4204      1491      C9              RET
4205              SUBTTL - QUEUTL - Queue utility routines
  
```

```
4206
4207      ;      Copyright (C) 1980 by Microsoft Corporation
4208      ;      Written by Marc Wilson
4209      ;
4210      ; This utility provides for multiple queues with the following
4211      ; capabilities:
4212      ;
4213      ; Queues of varying length - 1,3,7,15,31,63,127,255
4214      ;
4215      ; Each queue can be any of the possible lengths
4216      ; The queues can be initialized at any time and be
4217      ; located anywhere a single pointer (QUEUES) provides
4218      ; the address of the queue table.
4219      ;
4220      ; The queue table has all information for each queue,
4221      ; 6 bytes per queue. A single non-zero character can
4222      ; be pushed back on top of the queue.
4223      ;
4224      ; The entry for each queue is as follows:
4225      ;      +0      PUT OFFSET
4226      ;      +1      GET OFFSET
4227      ;      +2      BACK CHARACTER
4228      ;      +3      QUEUE LENGTH
4229      ;      +4,+5    QUEUE ADDRESS
4230      ;
4231      ; The utility assumes that the queue table is
4232      ; valid for all queue numbers passed to the routines
4233      ;
4234      ;ROUTINES:
4235      ; All routines assume that [A] equals the queue number,
4236      ; [QUEUES] contains the address of the queue table.
```

```
4237             ; Other requirements follow.
4238             ; GETQ    - Returns current top of queue in [A],
4239             ;           zero flag set if queue empty
4240             ; PUTQ    - Puts byte in [E] reg on end of queue,
4241             ;           zero set if queue is full
4242             ;
4243             ;NOTE:
4244             ; The routines are designed to be reentrant, however
4245             ; there are some restrictions for cases involving a
4246             ; single queue (in any case operating on different
4247             ; queues is alright). The first restriction is that
4248             ; the same routine cannot be reentered. The second
4249             ; is that INITQ and POPQ do not allow PUTQ,
4250             ; GETQ or BCKQ to be entered.
4251             ;
4252             ; LFTQ    - Returns unused number of bytes in queue in [A] reg
4253             ; INITQ    - Initialize queue to empty state,
4254             ;           B reg=length, (DE)=ADDR
4255             ; *** All routines destroy the registers ***
4256             ;
4257             SUBTTL  - QUEUTL - Queue routines
```

```
4258
4259      1492      PUTQ:
4260      ;
4261      ; Put data on queue
4262      ;
4263      1492      CD 14FA      CALL      GETPTR      ;Get queue pointers
4264      1495      78          LD        A,B
4265      1496      3C          INC        A            ;Bump PUT
4266      1497      23          INC        HL
4267      1498      A6          AND        (HL)         ;Wrap around
4268      1499      B9          CP         C
4269      149A      C8          RET         Z            ;QUEUE full
4270      149B      E5          PUSH      HL
4271      149C      2B          DEC        HL
4272      149D      2B          DEC        HL
4273      149E      2B          DEC        HL
4274      149F      E3          EX         (SP),HL      ;Save place to put new pointer
4275      14A0      23          INC        HL
4276      14A1      4F          LD        C,A          ;Pointer in C
4277      14A2      7E          LD        A,(HL)
4278      14A3      23          INC        HL
4279      14A4      66          LD        H,(HL)
4280      14A5      6F          LD        L,A          ;(HL) = QUEUE address
4281      14A6      06 00      LD        B,0
4282      14A8      09          ADD        HL,BC        ;(HL) = Address to put char
4283      14A9      73          LD        (HL),E
4284      14AA      E1          POP        HL
4285      14AB      71          LD        (HL),C        ;set new pointer
4286      14AC      C9          RET
4287      14AD      GETQ:
4288      ;
```

```

4289                ; Get data from QUEUE
4290                ;
4291    14AD    CD 14FA    CALL    GETPTR    ;Get queue pointers
4292    14B0    36 00    LD      (HL),0    ;zero back character
4293    14B2    20 1D    JR      NZ,GETBAK
4294    14B4    79      LD      A,C
4295    14B5    B8      CP      B
4296    14B6    C8      RET      Z        ;QUEUE empty!
4297    14B7    23      INC     HL
4298    14B8    3C      INC     A        ;Bump GET offset
4299    14B9    A6      AND     (HL)    ;wrap around
4300    14BA    2B      DEC     HL
4301    14BB    2B      DEC     HL
4302    14BC    E5      PUSH    HL        ;Save place to store pointer
4303    14BD    23      INC     HL
4304    14BE    23      INC     HL
4305    14BF    23      INC     HL
4306    14C0    4F      LD      C,A      ;offset in C
4307    14C1    7E      LD      A,(HL)
4308    14C2    23      INC     HL
4309    14C3    66      LD      H,(HL)
4310    14C4    6F      LD      L,A      ;[HL] = QUEUE address
4311    14C5    06 00    LD      B,0
4312    14C7    09      ADD     HL,BC
4313    14C8    7E      LD      A,(HL)    ;get char from QUEUE
4314    14C9    E1      POP     HL
4315    14CA    71      LD      (HL),C
4316    14CB    B7      OR      A
4317    14CC    C0      RET     NZ
4318    14CD    3C      INC     A
4319    14CE    3E 00    LD      A,0
  
```

- QUEUTL - Queue routines

```
4320      14D0      C9                RET
4321      14D1                GETBAK:
4322      14D1      4F                LD      C,A
4323      14D2      06 00            LD      B,0
4324      14D4      21 F970          LD      HL,QUEBAK-1
4325      14D7      09                ADD     HL,BC
4326      14D8      7E                LD      A,(HL)
4327      14D9      C9                RET
4328      14DA                INITQ:
4329                        ;
4330                        ; INITQ - Initialize QUEUE
4331                        ;
4332      14DA      C5                PUSH    BC                ;Save queue length
4333      14DB      CD 1504          CALL     QSTART            ;Get addr of start of QUEUE table entry
4334      14DE      70                LD      (HL),B            ;Clear PUT offset
4335      14DF      23                INC     HL
4336      14E0      70                LD      (HL),B            ;Clear GET offset
4337      14E1      23                INC     HL
4338      14E2      70                LD      (HL),B            ;Clear back character
4339      14E3      23                INC     HL
4340      14E4      F1                POP     AF
4341      14E5      77                LD      (HL),A            ;Set QUEUE length
4342      14E6      23                INC     HL
4343      14E7      73                LD      (HL),E
4344      14E8      23                INC     HL
4345      14E9      72                LD      (HL),D            ;Set QUEUE address
4346      14EA      C9                RET
4347      14EB                LFTQ:
4348                        ;
4349                        ; LFTQ - Returns number of bytes remaining in QUEUE
4350                        ;
```

4351	14EB	CD 14FA	CALL	GETPTR	;Get QUEUE ptrs
4352	14EE	78	LD	A,B	
4353	14EF	3C	INC	A	
4354	14F0	23	INC	HL	
4355	14F1	A6	AND	(HL)	
4356	14F2	47	LD	B,A	;B=PUT PTR+1
4357	14F3	79	LD	A,C	
4358	14F4	90	SUB	B	;subtract PUT from GET
4359	14F5	A6	AND	(HL)	;make it positive UNSIGNED INTEGER
4360	14F6	6F	LD	L,A	
4361	14F7	26 00	LD	H,0	
4362	14F9	C9	RET		
4363					
4364	14FA		GETPTR:		
4365			;		
4366			; QUEUE general routines		
4367			;		
4368	14FA	CD 1504	CALL	QSTART	;Get start of QUEUE TABLE entry
4369	14FD	46	LD	B,(HL)	;B = PUT OFFSET
4370	14FE	23	INC	HL	
4371	14FF	4E	LD	C,(HL)	;C = GET OFFSET
4372	1500	23	INC	HL	
4373	1501	7E	LD	A,(HL)	;A = BACK CHARACTER
4374	1502	B7	OR	A	
4375	1503	C9	RET		
4376			;		
4377	1504		QSTART:		
4378	1504	07	RLCA		;*2
4379	1505	47	LD	B,A	
4380	1506	07	RLCA		;*4
4381	1507	80	ADD	A,B	;*6

MSX ROM BASIC BIOS) Macro-80
QUEUTL - Queue routines

3.44

01-Jan-85

PAGE

46-4

160

4382	1508	4F	LD	C,A
4383	1509	06 00	LD	B,0
4384	150B	2A F3F3	LD	HL,(QUEUES)
4385	150E	09	ADD	HL,BC
4386	150F	C9	RET	
4387				

SUBTTL - MSXGRP - Graphic driver (Print a character on GRP screen)


```

4388
4389      1510      GRPPRT:
4390      ;
4391      ; Print a character on the graphic screen
4392      ;
4393      1510      E5      PUSH      HL
4394      1511      D5      PUSH      DE
4395      1512      C5      PUSH      BC
4396      1513      F5      PUSH      AF
4397      1514      CD 089D      CALL      CNVCHR      ;Convert code
4398      1517      30 62      JR      NC,JPPPAL      ;Graphic header byte, return soon
4399      1519      20 08      JR      NZ,GPRT05      ;Converted graphic code
4400      151B      FE 0D      CP      0DH      ;CR?
4401      151D      28 5F      JR      Z,GRPCR      ;Do not ignore CR even on graphic screen
4402      151F      FE 20      CP      ' '      ;Control character?
4403      1521      38 58      JR      C,JPPPAL      ;Yes, ignore this
4404      1523      GPRT05:
4405      1523      CD 0752      CALL      GETPAT      ;Get character pattern in PATWRK
4406      1526      3A F3E9      LD      A,(FORCLR)      ;Set color of character
4407      1529      32 F3F2      LD      (ATRBYT),A
4408      152C      2A FCB9      LD      HL,(GRPACY)
4409      152F      EB      EX      DE,HL      ;Current Y coordinate in [DE]
4410      1530      ED 4B FCB7      LD      BC,(GRPACX)      ;Current X coordinate in [BC]
4411      1534      CD 1599      CALL      SCALXY      ;Do the scaling
4412      1537      30 42      JR      NC,JPPPAL      ;Do not print if already out of screen
4413      1539      CD 15DF      CALL      MAPXYC      ;Map to CLOC and CMASK
4414      153C      11 FC40      LD      DE,PATWRK
4415      153F      0E 08      LD      C,8      ;Row counter
4416      1541      GPRT10:
4417      1541      06 08      LD      B,8      ;Column counter
4418      1543      CD 1639      CALL      FETCHC      ;Get current CLOC and CMASK

```

4419	1546	E5	PUSH	HL	;Save these
4420	1547	F5	PUSH	AF	
4421	1548	1A	LD	A,(DE)	;Get pattern for a row
4422	1549		GPRT20:		
4423	1549	87	ADD	A,A	;Check each bit
4424	154A	F5	PUSH	AF	
4425	154B	DC 167E	CALL	C,SETC	;Set it if 1
4426	154E	CD 16AC	CALL	TRIGHT	;Move 1 pixel right
4427	1551	E1	POP	HL	;Assume out of screen
4428	1552	38 04	JR	C,GPRT30	;Good assumption, skip the rest
4429	1554	E5	PUSH	HL	
4430	1555	F1	POP	AF	
4431	1556	10 F1	DJNZ	GPRT20	;Loop till done all columns
4432	1558		GPRT30:		
4433	1558	F1	POP	AF	;Restore CLOC and CMASK
4434	1559	E1	POP	HL	
4435	155A	CD 1640	CALL	STOREC	;Set these
4436	155D	CD 170A	CALL	TDOWNC	;Move 1 pixel down
4437	1560	38 04	JR	C,GPRT40	;Out of screen, skip rest and return
4438	1562	13	INC	DE	;Point to next row
4439	1563	0D	DEC	C	
4440	1564	20 DB	JR	NZ,GPRT10	;Loop till done all rows
4441	1566		GPRT40:		
4442	1566	CD 15D9	CALL	CHKMOD	;Check current screen mode
4443	1569	3A FCB7	LD	A,(GRPACX)	
4444	156C	28 06	JR	Z,GPRT50	;We're in high-resolution mode
4445	156E	C6 20	ADD	A,' '	
4446	1570	38 0C	JR	C,GRPCR	;We're going out of screen
4447	1572	18 04	JR	GPRT60	
4448	1574		GPRT50:		
4449					;

```

4450      1574      C6 08              ADD      A,8
4451      1576      38 06              JR        C,GRPCR
4452      1578              GPRT60:
4453      1578      32 FCB7            LD        (GRPACX),A      ;Update cursor position
4454      157B              JPPPAL:
4455      157B      C3 08DA            JP        POPALL
4456      157E              GRPCR:
4457              ;
4458      157E      AF                  XOR      A              ;Reset X position
4459      157F      32 FCB7            LD        (GRPACX),A
4460      1582      CD 15D9            CALL     CHKMOD
4461      1585      3A FCB9            LD        A,(GRPACY)
4462      1588      28 03              JR        Z,GPRT70
4463      158A      C6 20              ADD      A,4*8
4464      158C      01                  DB        1
4465      158D              GPRT70:
4466      158D      C6 08              ADD      A,8
4467      158F      FE C0              CP        0C0H
4468      1591      38 01              JR        C,GPRT80
4469      1593      AF                  XOR      A              ;Reset Y position also
4470      1594              GPRT80:
4471      1594      32 FCB9            LD        (GRPACY),A
4472      1597      18 E2              JR        JPPPAL
4473              SUBTTL - MSXGRP - (Routines for general graphics)

```

```

4474
4475      1599      SCALXY:
4476      ;
4477      ; SCALXY - Clips X,Y to max values in physical size and flags out
4478      ; of range values.
4479      ;
4480      ; ENTRY [BC] := X (0 ... max X), [DE] = Y (0 ... max Y)
4481      ; EXIT [BC] = X clipped, [DE] = Y clipped
4482      ; CARRY is reset if one of the value was out of bound
4483      ;
4484      1599      E5      PUSH      HL      ;save [HL]
4485      159A      C5      PUSH      BC      ;save [BC] - X coordinate
4486      159B      06 01      LD      B,1      ;no-error flag
4487      159D      EB      EX      DE,HL      ;Y coordinate to [HL]
4488      159E      7C      LD      A,H      ;Is Y coordinate negative?
4489      159F      87      ADD      A,A
4490      15A0      30 05      JR      NC,YPOSTV      ;No, positive
4491      15A2      21 0000      LD      HL,0      ;Substitute by 0 is negative
4492      15A5      18 08      JR      YNEGTV      ;And set out of bound flag
4493      15A7      YPOSTV:
4494      ;
4495      15A7      11 00C0      LD      DE,0C0H      ;Maximum Y+1
4496      15AA      E7      RST      20H      ;Test [HL] with [DE]
4497      15AB      38 04      JR      C,SCLYOK      ;if carry, not out of bound
4498      15AD      EB      EX      DE,HL      ;[HL] = 192
4499      15AE      2B      DEC      HL      ;Y = 191 ,maximum Y coordinate
4500      15AF      YNEGTV:
4501      15AF      06 00      LD      B,0      ;set out of bound flag
4502      15B1      SCLYOK:
4503      15B1      E3      EX      (SP),HL      ;save Y and get X to [HL]
4504      15B2      7C      LD      A,H      ;Is X coordinate negative?

```

```
4505      15B3      87              ADD      A,A
4506      15B4      30 05          JR        NC,XPOSTV      ;No, positive
4507      15B6      21 0000        LD        HL,0            ;Substitute by 0 if negative
4508      15B9      18 08          JR        XNEGTV          ;And set out of bound flag
4509      15BB                      XPOSTV:
4510                      ;
4511      15BB      11 0100        LD        DE,0100H        ;max X +1
4512      15BE      E7              RST        20H           ;Test [HL] with [DE]
4513      15BF      38 04          JR        C,SCLXOK
4514      15C1      EB              EX        DE,HL           ;[HL] = 256
4515      15C2      2B              DEC        HL            ;[HL] = 255 - max X coordinate
4516      15C3                      XNEGTV:
4517      15C3      06 00        LD        B,0               ;error flag
4518      15C5                      SCLXOK:
4519      15C5      D1              POP        DE             ;restore [DE] = Y
4520      15C6      CD 15D9        CALL      CHKMOD
4521      15C9      28 08          JR        Z,HRSSCL
4522      15CB      CB 3D          SRL        L              ;We're in high-resolution mode
4523      15CD      CB 3D          SRL        L              ;Divide both X and Y by 4 because we're
4524      15CF      CB 3B          SRL        E              ;in multi-color mode
4525      15D1      CB 3B          SRL        E
4526      15D3                      HRSSCL:
4527      15D3      78              LD        A,B
4528      15D4      0F              RRCA
4529      15D5      44              LD        B,H            ;set carry if no error
4530      15D6      4D              LD        C,L            ;[BC] = X
4531      15D7      E1              POP        HL
4532      15D8      C9              RET
4533      15D9                      CHKMOD:
4534                      ;
4535                      ; Check current screen mode
```

```

4536      ;
4537      15D9      3A FCAF      LD      A,(SCRMOD)
4538      15DC      D6 02      SUB      2      ;In what mode are we now?
4539      15DE      C9      RET      ;Return with the condition flag
4540      15DF      MAPXYC:
4541      ;
4542      ; MAPXYC - Maps X,Y coordinates to "C" (address, mask)
4543      ;
4544      ; Entry: [BC] = X, [DE] = Y
4545      ;
4546      ; Exit: CLOC = [HL] -- Video Ram address
4547      ; CMASK = [A] -- Bit Mask
4548      ;
4549      ; [ High-resolution mode ]
4550      ;
4551      ; X coord - XXXXXXXX ( 8 bits, max=255)
4552      ; 76543210
4553      ;
4554      ; Y coord - YYYYYYYY ( 8 bits, max=191)
4555      ; 76543210
4556      ;
4557      ; CLOC = YYYYYYYYYYYY
4558      ; 7654376543210
4559      ; XXX
4560      ; 210
4561      ;-----
4562      ; CMASK = 10000000 000
4563      ; 01000000 001
4564      ; 00100000 010
4565      ; 00010000 011
4566      ; 00001000 100
  
```

```

4567          ;          00000100 101
4568          ;          00000010 110
4569          ;          00000001 111
4570          ;
4571          ; [ Multi-color mode ]
4572          ;
4573          ;      X coord - XXXXXX ( 6 bits , max=63 )
4574          ;          543210
4575          ;
4576          ;      Y coord - YYYYYY ( 6 bits , max=47 )
4577          ;          543210
4578          ;
4579          ;      CLOC = YYYYYXXXXYYY
4580          ;          54354321210
4581          ;
4582          ; CMASK = 11110000 if X0=0 (even)
4583          ; CMASK = 00001111 if X0=1 (odd)
4584          ;
4585          ; Note: The boundary check has already been done by a call
4586          ; to SCALXY, so no range checking is needed.
4587          ;
4588 15DF      C5          PUSH      BC          ;Save X
4589 15E0      CD 15D9     CALL      CHKMOD     ;Check current screen mode
4590 15E3      20 2E      JR        NZ,MMPXYC ;Multi-color mode
4591 15E5      51         LD        D,C       ;Save X to D also
4592 15E6      79         LD        A,C
4593 15E7      E6 07      AND       7
4594 15E9      4F         LD        C,A
4595 15EA      21 160B     LD        HL,TWOPWR ;Table of power of two
4596 15ED      09         ADD       HL,BC
4597 15EE      7E         LD        A,(HL)     ;read bit mask CMASK

```

```
4598      15EF      32 F92C          LD      (CMASK),A
4599      15F2      7B              LD      A,E          ;Get Y coordinate
4600      15F3      0F              RRCA
4601      15F4      0F              RRCA
4602      15F5      0F              RRCA
4603      15F6      E6 1F          AND      00011111B
4604      15F8      47              LD      B,A
4605      15F9      7A              LD      A,D          ;Get X coordinate
4606      15FA      E6 F8          AND      11111000B
4607      15FC      4F              LD      C,A
4608      15FD      7B              LD      A,E          ;Get Y coordinate
4609      15FE      E6 07          AND      00000111B
4610      1600      B1              OR      C
4611      1601      4F              LD      C,A
4612      1602      2A F3CB        LD      HL,(GRPCGP)
4613      1605      09              ADD     HL,BC
4614      1606      22 F92A        LD      (CLOC),HL      ;Set pattern generator address
4615      1609      C1              POP     BC
4616      160A      C9              RET
4617      160B
4618
4619      ;
4620      ; Table of power of two
4621      ;
4621      160B      80 40 20 10      DB      80H,40H,20H,10H
4622      160F      08 04 02 01      DB      08H,04H,02H,01H
4623      ;
4624      1613      MMPXYC:
4625      ;
4626      ; Map XY for multi-color mode
4627      ;
4628      1613      79              LD      A,C          ;Get X position
```


4629	1614	0F	RRCA		;Even or odd?
4630	1615	3E F0	LD	A,11110000B	;Assume even
4631	1617	30 02	JR	NC,MMPXY1	;Good assumption
4632	1619	3E 0F	LD	A,00001111B	;Odd
4633	161B		MMPXY1:		
4634	161B	32 F92C	LD	(CMASK),A	;Set up mask pattern
4635	161E	79	LD	A,C	
4636	161F	87	ADD	A,A	
4637	1620	87	ADD	A,A	
4638	1621	E6 F8	AND	11111000B	
4639	1623	4F	LD	C,A	;Get lower byte
4640	1624	7B	LD	A,E	
4641	1625	E6 07	AND	0111B	
4642	1627	B1	OR	C	
4643	1628	4F	LD	C,A	
4644	1629	7B	LD	A,E	
4645	162A	0F	RRCA		
4646	162B	0F	RRCA		
4647	162C	0F	RRCA		
4648	162D	E6 07	AND	0111B	
4649	162F	47	LD	B,A	;Get higher byte
4650	1630	2A F3D5	LD	HL,(MLTCGP)	;Load start address of pattern table
4651	1633	09	ADD	HL,BC	
4652	1634	22 F92A	LD	(CLOC),HL	
4653	1637	C1	POP	BC	
4654	1638	C9	RET		

```

4655
4656      1639      FETCHC:
4657      ;
4658      ; FETCHC - Reads the value of the graphics accumulator
4659      ;
4660      ; Exit: [HL] = CLOC, [A] = CMASK
4661      ;
4662      1639      3A F92C      LD      A,(CMASK)
4663      163C      2A F92A      LD      HL,(CLOC)
4664      163F      C9          RET
4665      1640      STOREC:
4666      ;
4667      ; STOREC - Sets the graphics accumulator
4668      ;
4669      ; Entry: [HL] = CLOC, [A] = CMASK
4670      ;
4671      1640      32 F92C      LD      (CMASK),A
4672      1643      22 F92A      LD      (CLOC),HL
4673      1646      C9          RET
4674      1647      READC:
4675      ;
4676      ; READC - Get the attribute of the current graphics accumulator
4677      ; position
4678      ;
4679      1647      C5          PUSH    BC
4680      1648      E5          PUSH    HL
4681      1649      CD 1639      CALL    FETCHC      ;Get CLOC and CMASK
4682      164C      47          LD      B,A          ;Save CMASK
4683      164D      CD 15D9      CALL    CHKMOD      ;Check current screen mode
4684      1650      20 1A        JR      NZ,MREADC   ;Multi-color mode
4685      1652      CD 07D7      CALL    RDVRM      ;Read VDP's VRAM (pattern)

```

4686	1655	A0	AND	B	;Extract specified pixel
4687	1656	F5	PUSH	AF	;Save whether the pixel is on or off
4688	1657	01 2000	LD	BC,GRPDIF	
4689	165A	09	ADD	HL,BC	
4690	165B	CD 07D7	CALL	RDVRM	;Read VDP's VRAM (color)
4691	165E	47	LD	B,A	;Save this to B
4692	165F	F1	POP	AF	;Restore condition
4693	1660	78	LD	A,B	;Restore color
4694	1661	28 04	JR	Z,READC1	;Specified dot is off, return
4695					;background color
4696	1663		READC0:		
4697	1663	0F	RRCA		;Specified dot is on, return foreground color
4698	1664	0F	RRCA		
4699	1665	0F	RRCA		
4700	1666	0F	RRCA		
4701	1667		READC1:		
4702	1667	E6 0F	AND	0FH	;Make it a legal value
4703	1669	E1	POP	HL	
4704	166A	C1	POP	BC	
4705	166B	C9	RET		
4706	166C		MREADC:		
4707			;		
4708	166C	CD 07D7	CALL	RDVRM	;Read VRAM
4709	166F	04	INC	B	;Check if specified pixel is even or odd
4710	1670	05	DEC	B	
4711	1671	F2 1667	JP	P,READC1	;Odd, return lower nibble
4712	1674	18 ED	JR	READC0	;Even, return upper nibble

```
4713
4714      1676      SETATR:
4715      ;
4716      ; SETATR - Sets the attribute (color, reverse, etc..) to be
4717      ; used in future actions.
4718      ;
4719      ; Entry - [A] = Attribute
4720      ; Exit - carry set if illegal value
4721      ;
4722      1676      FE 10      CP      16      ;Must be less than 16
4723      1678      3F      CCF
4724      1679      D8      RET      C
4725      167A      32 F3F2      LD      (ATRBYT),A
4726      167D      C9      RET
4727      167E      SETC:
4728      ;
4729      ; SETC - Sets the point indicated by the graphics accumulator
4730      ; to ATTRBYT
4731      ;
4732      ; All registers except AF must be preserved.
4733      ;
4734      167E      E5      PUSH      HL
4735      167F      C5      PUSH      BC
4736      1680      CD 15D9      CALL      CHKMOD      ;Check current screen mode
4737      1683      CD 1639      CALL      FETCHC
4738      1686      20 08      JR      NZ,MSETC      ;Multi-color mode
4739      1688      D5      PUSH      DE
4740      1689      CD 186C      CALL      PATWRT
4741      168C      D1      POP      DE
4742      168D      C1      POP      BC
4743      168E      E1      POP      HL
```

```

4744 168F C9 RET
4745 1690 MSETC:
4746 ;
4747 ; Set a pixel in multi-color mode
4748 ;
4749 1690 47 LD B,A ;Save CMASK in [B]
4750 1691 CD 07D7 CALL RDVRM ;Read VRAM
4751 1694 4F LD C,A
4752 1695 78 LD A,B
4753 1696 2F CPL ;Leave another unaffected
4754 1697 A1 AND C
4755 1698 4F LD C,A
4756 1699 3A F3F2 LD A,(ATRBYT) ;Get specified color
4757 169C 04 INC B ;Check if even or odd
4758 169D 05 DEC B
4759 169E F2 16A5 JP P,MSETC1 ;Odd
4760 16A1 87 ADD A,A
4761 16A2 87 ADD A,A
4762 16A3 87 ADD A,A
4763 16A4 87 ADD A,A
4764 16A5 MSETC1:
4765 16A5 B1 OR C ;Form new color
4766 16A6 CD 07CD CALL WRTVRM ;Write new pattern
4767 16A9 C1 POP BC
4768 16AA E1 POP HL
4769 16AB C9 RET
4770 SUBTTL - MSXGRP - (Graphic cursor movements)

```

```
4771
4772      ;
4773      ; UPC, DOWNC, RIGHTC, LEFTC
4774      ;
4775      ; These are the C relative movement routines. They
4776      ; adjust the current graphics accumulator in the indicated
4777      ; direction without checking boundary conditions.
4778      ;
4779      ;-----
4780      ;
4781      16AC      TRIGHT:
4782      ;
4783      ; TRIGHT - move 1 pixel right
4784      ; Return carry set if already on border
4785      ;
4786      16AC      E5          PUSH    HL
4787      16AD      CD 15D9     CALL    CHKMOD
4788      16B0      C2 1779     JP      NZ,MTRGT
4789      16B3      CD 1639     CALL    FETCHC      ;Get CLOC,CMASK
4790      16B6      0F          RRCA          ;Move 1 pixel right
4791      16B7      30 4B       JR      NC,HRZMV1   ;Within byte, just change CMASK
4792      16B9      7D          LD      A,L        ;Get low byte of CLOC
4793      16BA      E6 F8       AND     0F8H
4794      16BC      FE F8       CP      0F8H       ;On right edge?
4795      16BE      3E 80       LD      A,80H      ;Assume not
4796      16C0      20 10       JR      NZ,RGHTC1   ;Goot assumption
4797      16C2      C3 175A     JP      ONBRD1     ;On border, set carry and return
4798      16C5      RIGHTC:
4799      ;
4800      ; RIGHTC - move 1 pixel right
4801      ;
```

```
4802      16C5      E5                PUSH    HL
4803      16C6      CD 15D9           CALL    CHKMOD
4804      16C9      C2 178B           JP      NZ,MRGTC
4805      16CC      CD 1639           CALL    FETCHC
4806      16CF      0F                RRCA
4807      16D0      30 32             JR      NC,HRZMV1      ;move right 1 pixel
4808      16D2                        RGHTC1:                ;within byte, just change CMASK
4809      16D2      D5                PUSH    DE
4810      16D3      11 0008           LD      DE,8          ;Load offset to new position
4811      16D6      18 27             JR      HRZMOV         ;Change CLOC also
4812      16D8                        TLEFT:
4813                        ;
4814                        ; TLEFT - move 1 pixel left
4815                        ; Return carry set if already on border
4816                        ;
4817      16D8      E5                PUSH    HL
4818      16D9      CD 15D9           CALL    CHKMOD
4819      16DC      C2 179C           JP      NZ,MTLEFT
4820      16DF      CD 1639           CALL    FETCHC         ;Get CLOC and CMASK
4821      16E2      07                RLCA                    ;Move 1 pixel left
4822      16E3      30 1F             JR      NC,HRZMV1      ;Within byte boundary, just change CMASK
4823      16E5      7D                LD      A,L            ;Check if we're on left edge
4824      16E6      E6 F8             AND     0F8H
4825      16E8      3E 01             LD      A,l          ;Assume not
4826      16EA      20 0F             JR      NZ,LEFTC1     ;Good assumption
4827      16EC      18 6C             JR      ONBRD1      ;We're on border, set carry and return
4828      16EE                        LEFTC:
4829                        ;
4830                        ; LEFTC - move 1 pixel left
4831                        ;
4832      16EE      E5                PUSH    HL
```

```
4833      16EF      CD 15D9          CALL    CHKMOD
4834      16F2      C2 17AC          JP      NZ,MLFTC
4835      16F5      CD 1639          CALL    FETCHC
4836      16F8      07                RLCA
4837      16F9      30 09          JR      NC,HRZMV1      ;move left 1 pixel
                                         ;within byte boundary, just change CMASK
4838      16FB                      LEFTC1:
4839      16FB      D5                PUSH    DE
4840      16FC      11 FFF8          LD      DE,0FFF8H      ;Load offset to new position
4841      16FF                      HRZMOV:
4842      16FF      19                ADD     HL,DE          ;Add offset to new position
4843      1700      22 F92A          LD      (CLOC),HL      ;Update pattern address
4844      1703      D1                POP     DE
4845      1704                      HRZMV1:
4846      1704      32 F92C          LD      (CMASK),A      ;Update CMASK
4847      1707      A7                AND     A              ;Clear carry
4848      1708      E1                POP     HL
4849      1709      C9                RET
4850      170A                      TDOWNC:
4851                      ;
4852                      ; TDOWNC - move 1 pixel down.
4853                      ;
4854                      ; Return carry set if already on screen border.
4855                      ;
4856      170A      E5                PUSH    HL
4857      170B      D5                PUSH    DE
4858      170C      2A F92A          LD      HL,(CLOC)
4859      170F      CD 15D9          CALL    CHKMOD
4860      1712      C2 17C6          JP      NZ,MTDNC
4861      1715      E5                PUSH    HL
4862      1716      2A F3CB          LD      HL,(GRPCGP)
4863      1719      11 1700          LD      DE,1700H
```


4864	171C	19	ADD	HL,DE	
4865	171D	EB	EX	DE,HL	
4866	171E	E1	POP	HL	
4867	171F	E7	RST	20H	;Test [HL] with [DE]
4868					;Looks like on border?
4869	1720	38 13	JR	C,DWNC10	;No
4870	1722	7D	LD	A,L	;Possibly on border
4871	1723	3C	INC	A	
4872	1724	E6 07	AND	7	;Really?
4873	1726	20 0D	JR	NZ,DWNC10	;No
4874	1728	18 2F	JR	ONBRDR	;Yes, set carry and return
4875					;
4876	172A		DOWNC:		
4877			;		
4878			; DOWNC - move 1 pixel down		
4879			;		
4880	172A	E5	PUSH	HL	
4881	172B	D5	PUSH	DE	
4882	172C	2A F92A	LD	HL,(CLOC)	
4883	172F	CD 15D9	CALL	CHKMOD	
4884	1732	C2 17DC	JP	NZ,MDNC	
4885	1735		DWNC10:		
4886	1735	23	INC	HL	;move down 1 pixel
4887	1736	7D	LD	A,L	;Prepare for boundary check
4888	1737	11 00F8	LD	DE,0F8H	;Load possible offset to new location
4889	173A	18 31	JR	VRTMOV	;Check
4890	173C		TUPC:		
4891			;		
4892			; TUPC - move 1 pixel up.		
4893			; Return carry set if already on screen border.		
4894			;		

```
4895      173C      E5          PUSH      HL
4896      173D      D5          PUSH      DE
4897      173E      2A F92A      LD        HL,(CLOC)
4898      1741      CD 15D9      CALL      CHKMOD
4899      1744      C2 17E3      JP        NZ,MTUPC
4900      1747      E5          PUSH      HL
4901      1748      2A F3CB      LD        HL,(GRPCGP)
4902      174B      11 0100      LD        DE,0100H
4903      174E      19          ADD       HL,DE
4904      174F      EB          EX        DE,HL
4905      1750      E1          POP       HL
4906      1751      E7          RST       20H          ;Test [HL] with [DE]
4907                                         ;Looks like on border?
4908      1752      30 14          JR        NC,UPC10      ;No
4909      1754      7D          LD        A,L          ;Possibly on border
4910      1755      E6 07          AND      7          ;Really?
4911      1757      20 0F          JR        NZ,UPC10      ;No
4912      1759                                         ONBRDR:
4913      1759      D1          POP       DE
4914      175A                                         ONBRD1:
4915      175A      37          SCF          ;Set carry indicating we're on border
4916      175B      E1          POP       HL
4917      175C      C9          RET
4918      175D                                         UPC:
4919                                         ;
4920                                         ; UPC - move 1 pixel up
4921                                         ;
4922      175D      E5          PUSH      HL
4923      175E      D5          PUSH      DE
4924      175F      2A F92A      LD        HL,(CLOC)      ;get current position
4925      1762      CD 15D9      CALL      CHKMOD
```

```
4926      1765      C2 17F8                JP      NZ,MUPC
4927      1768                UPC10:
4928      1768      7D                LD      A,L          ;Prepare for boundary check
4929      1769      2B                DEC      HL          ;move up 1 pixel
4930      176A      11 FF08            LD      DE,0FF08H    ;Load possible offset to new location
4931      176D                VRTMOV:
4932      176D      E6 07                AND      7          ;Crossed boundary?
4933      176F      20 01                JR      NZ,VRTMV1   ;No, it's okay
4934      1771      19                ADD      HL,DE        ;Get new location
4935      1772                VRTMV1:
4936      1772      22 F92A            LD      (CLOC),HL    ;Update pattern address
4937      1775      A7                AND      A          ;Clear carry
4938      1776      D1                POP      DE
4939      1777      E1                POP      HL
4940      1778      C9                RET
4941      1779                MTRGT:
4942                ;
4943                ; Graphics cursor movement in multi-color mode
4944                ; [ Horizontal movements ]
4945                ;
4946      1779      CD 1639            CALL     FETCHC
4947      177C      A7                AND      A
4948      177D      3E 0F            LD      A,0FH          ;Assume CMASK is even
4949      177F      FA 17C0          JP      M,MHZMV1        ;Within byte, just change CMASK
4950      1782      7D                LD      A,L
4951      1783      E6 F8                AND      0F8H
4952      1785      FE F8            CP      0F8H          ;On right edge?
4953      1787      20 0B            JR      NZ,MRGTC1       ;No, move to next pixel
4954      1789      18 CF            JR      ONBRD1         ;We're on right edge, set carry and return
4955      178B                MRGTC:
4956                ;
```

```

4957 178B CD 1639 CALL FETCHC
4958 178E A7 AND A
4959 178F 3E 0F LD A,0FH ;Assume CMASK is even
4960 1791 FA 17C0 JP M,MHZMV1 ;Good assumption
4961 1794 MRGTCL:
4962 1794 D5 PUSH DE
4963 1795 11 0008 LD DE,8 ;Next pixel is 8 byte far
4964 ;from the current position
4965 1798 3E F0 LD A,0F0H
4966 179A 18 1F JR MHCMOV
4967 179C MTLFT:
4968 ;
4969 179C CD 1639 CALL FETCHC
4970 179F A7 AND A
4971 17A0 3E F0 LD A,0F0H ;Assume CMASK is odd
4972 17A2 F2 17C0 JP P,MHZMV1 ;Good assumption, just change CMASK
4973 17A5 7D LD A,L
4974 17A6 E6 F8 AND 0F8H ;On left edge?
4975 17A8 20 0B JR NZ,MLFTCL ;No
4976 17AA 18 AE JR ONBRD1 ;We're on left edge, set carry and return
4977 17AC MLFTCL:
4978 ;
4979 17AC CD 1639 CALL FETCHC
4980 17AF A7 AND A
4981 17B0 3E F0 LD A,0F0H ;Assume CMASK is odd
4982 17B2 F2 17C0 JP P,MHZMV1 ;Good assumption, just change CMASK
4983 17B5 MLFTCL:
4984 17B5 D5 PUSH DE
4985 17B6 11 FFF8 LD DE,0FFF8H
4986 17B9 3E 0F LD A,0FH
4987 17BB MHCMOV:

```

```

4988      17BB      19                      ADD      HL,DE
4989      17BC      22 F92A                  LD        (CLOC),HL
4990      17BF      D1                      POP       DE
4991      17C0                      MHZMV1:
4992      17C0      32 F92C                  LD        (CMASK),A
4993      17C3      A7                      AND        A
4994      17C4      E1                      POP       HL
4995      17C5      C9                      RET
4996      17C6                      MTDNC:
4997                      ;
4998                      ; [ Vertical movements ]
4999                      ;
5000      17C6      E5                      PUSH      HL
5001      17C7      2A F3D5                  LD        HL,(MLTCGP)
5002      17CA      11 0500                  LD        DE,0500H
5003      17CD      19                      ADD      HL,DE
5004      17CE      E1                      POP       HL
5005      17CF      E7                      RST       20H
5006      17D0      38 0A                  JR        C,MDNC
5007      17D2      7D                      LD        A,L
5008      17D3      3C                      INC        A
5009      17D4      E6 07                  AND        7
5010      17D6      20 04                  JR        NZ,MDNC
5011      17D8      37                      SCF
5012                      ;Possibly on border?
5013                      ;No
5014                      ;Check if least 3 bits are all 1's
5015      17D9      D1                      POP       DE
5016      17DA      E1                      POP       HL
5017      17DB      C9                      RET
5018      17DC                      MDNC:
5019                      ;
5020      17DC      23                      INC        HL
5021                      ;Move down 1 byte

```

5019	17DD	7D		LD	A,L	
5020	17DE	11 00F8		LD	DE,0F8H	;Load possible offset to next block
5021	17E1	18 1A		JR	MVTMOV	;Check
5022	17E3		MTUPC:			
5023			;			
5024	17E3	E5		PUSH	HL	
5025	17E4	2A F3D5		LD	HL,(MLTCGP)	
5026	17E7	11 0100		LD	DE,0100H	;Possibly on border?
5027	17EA	19		ADD	HL,DE	
5028	17EB	E1		POP	HL	
5029	17EC	E7		RST	20H	;Test [HL] with [DE]
5030	17ED	30 09		JR	NC,MUPC	;No
5031	17EF	7D		LD	A,L	;Check if we're top of a block
5032	17F0	E6 07		AND	7	
5033	17F2	20 04		JR	NZ,MUPC	;No
5034	17F4	37		SCF		;We're on top border, set carry and return
5035	17F5	D1		POP	DE	
5036	17F6	E1		POP	HL	
5037	17F7	C9		RET		
5038	17F8		MUPC:			
5039			;			
5040	17F8	7D		LD	A,L	
5041	17F9	2B		DEC	HL	;Move up 1 byte
5042	17FA	11 FF08		LD	DE,0FF08H	;Load possible offset to next block
5043	17FD		MVTMOV:			
5044	17FD	E6 07		AND	7	;Wrapped to next block?
5045	17FF	20 01		JR	NZ,MVTMV1	;No
5046	1801	19		ADD	HL,DE	;Yes, add up offset to next block
5047	1802		MVTMV1:			
5048	1802	22 F92A		LD	(CLOC),HL	
5049	1805	A7		AND	A	;Clear carry

(MSX ROM BASIC BIOS) Macro-80
- MSXGRP - (Graphic cursor movements)

3.44

01-Jan-85

PAGE

51-9

183

5050 1806 D1

POP DE

5051 1807 E1

POP HL

5052 1808 C9

RET

5053

SUBTTL -MSXGRP- (Box fill and Misc.)

```
5054
5055      1809      NSETCX:
5056                ;
5057                ; NSETCX - Performs SETC, RIGHTC [HL] times
5058                ;
5059                ; In fact, SETC and RIGHTC are never called to increase speed,
5060                ; and for the reason described below.
5061                ;
5062                ; Since only 2 colors can be displayed in a byte, some special
5063                ; handling is required when a full-byte is set when writing left
5064                ; or right extras. In this case, we can completely ignore the
5065                ; background color for that byte, allowing 2 colors displayed
5066                ; in a byte.
5067                ;
5068                ; All registers may be destroyed.
5069                ;
5070      1809      CD 15D9      CALL      CHKMOD
5071      180C      C2 18BB      JP        NZ,MNSTCX      ;Multi-color mode
5072      180F      E5          PUSH     HL              ;Save count
5073      1810      CD 1639      CALL     FETCHC         ;Get CLOC and CMASK
5074      1813      E3          EX       (SP),HL        ;Reget count, save CLOC
5075      1814      87          ADD      A,A            ;Beginig at leftmost position?
5076      1815      38 18      JR       C,NSTC20        ;Yes, no extra dots at the left
5077      1817      F5          PUSH     AF             ;Save mask pattern*2
5078      1818      01 FFFF      LD      BC,0FFFFH
5079      181B      0F          RRCA
5080      181C      NSTC10:
5081      181C      09          ADD      HL,BC          ;Decrement pixel count
5082      181D      30 45      JR       NC,NSTCSP       ;The whole dots are within a byte
5083      181F      0F          RRCA
5084      1820      30 FA      JR       NC,NSTC10
```


5085	1822	F1	POP	AF	;Restore mask pattern*2
5086	1823	3D	DEC	A	;Form left-extra pattern
5087	1824	E3	EX	(SP),HL	;Reget CLOC, save count
5088	1825	E5	PUSH	HL	;Save CLOC
5089	1826	CD 186C	CALL	PATWRT	;Write to VRAM (pattern and color)
5090	1829	E1	POP	HL	;Restore CLOC
5091	182A	11 0008	LD	DE,8	;Load an offset to next byte
5092	182D	19	ADD	HL,DE	;Update pattern address
5093	182E	E3	EX	(SP),HL	;Reget count, save CLOC
5094	182F				
5095	182F	7D	LD	A,L	;Get low byte of count
5096	1830	E6 07	AND	7	;[A]=count mod 8
5097	1832	4F	LD	C,A	;save count after byte boundary
5098	1833	7C	LD	A,H	
5099	1834	0F	RRCA		
5100	1835	7D	LD	A,L	
5101	1836	1F	RRA		
5102	1837	0F	RRCA		
5103	1838	0F	RRCA		
5104	1839	E6 3F	AND	00111111B	;[HL]=[HL]/8
5105	183B	E1	POP	HL	;Reget CLOC
5106	183C	47	LD	B,A	;[B]=counter
5107	183D	28 14	JR	Z,NSTC40	;No dots in this part
5108	183F				
5109	183F	AF	XOR	A	;Make specified color a background color
5110	1840	CD 07CD	CALL	WRTVRM	;Write to VRAM (pattern)
5111	1843	11 2000	LD	DE,GRPDIF	
5112	1846	19	ADD	HL,DE	;Calculate address of color table
5113	1847	3A F3F2	LD	A,(ATRBYT)	;Get specified color
5114	184A	CD 07CD	CALL	WRTVRM	;Write to VRAM (color)
5115	184D	11 2008	LD	DE,GRPDIF+8	;Load an offset to next byte

5116	1850	19	ADD	HL,DE	;Bump CLOC
5117	1851	10 EC	DJNZ	NSTC30	;Loop until done
5118	1853		NSTC40:		
5119	1853	0D	DEC	C	;dot count in char boundary
5120	1854	F8	RET	M	;No dots in right extra
5121	1855	E5	PUSH	HL	;Save CLOC
5122	1856	21 185D	LD	HL,RGTEXT	;Load address for 'right-extra' pattern table
5123	1859	09	ADD	HL,BC	
5124	185A	7E	LD	A,(HL)	;Get pattern
5125	185B	18 0E	JR	NSTC50	
5126	185D		RGTEXT:		
5127			;		
5128	185D	80 C0 E0 F0	DB	80H,0C0H,0E0H,0F0H	
5129	1861	F8 FC FE	DB	0F8H,0FCH,0FEH	
5130	1864		NSTCSP:		
5131			;		
5132	1864	87	ADD	A,A	;Get mask pattern for the right (11111100)
5133	1865	3D	DEC	A	
5134	1866	2F	CPL		
5135	1867	47	LD	B,A	;Save it
5136	1868	F1	POP	AF	;Get mask pattern for the left (00011111)
5137	1869	3D	DEC	A	
5138	186A	A0	AND	B	;Make a pattern to write (00011100)
5139	186B		NSTC50:		
5140	186B	E1	POP	HL	;Restore CLOC ex.

```
5141
5142      186C      PATWRT:
5143                ;
5144                ; PATWRT - Write a pattern to high-resolution screen
5145                ;
5146                ; Entry: A - Pattern to be written
5147                ;       HL - Address of pattern table
5148                ;       ATRBYT - Color of this pattern
5149                ;
5150      186C      47      LD      B,A      ;Save pattern to be added
5151      186D      CD 07D7      CALL   RDVRM      ;Read VRAM (pattern)
5152      1870      4F      LD      C,A      ;Save current pattern
5153      1871      11 2000      LD      DE,GRPDIF
5154      1874      19      ADD     HL,DE      ;Form address of color table
5155      1875      CD 07D7      CALL   RDVRM      ;Read from VRAM (color)
5156      1878      F5      PUSH    AF
5157      1879      E6 0F      AND     0FH      ;Extract background color
5158      187B      5F      LD      E,A      ;Save background color
5159      187C      F1      POP     AF      ;Restore foreground and background color
5160      187D      93      SUB     E
5161      187E      57      LD      D,A      ;Set foreground color in the upper 4 bit
5162                                ;[B] has the specified pattern,
5163                                ;[C] has the current pattern,
5164                                ;[D] has the current foreground color
5165                                ; shifted left 4 times,
5166                                ;[E] has the current background color,
5167                                ;[HL] has the address of color table.
5168      187F      3A F3F2      LD      A,(ATRBYT) ;Get specified color
5169      1882      BB      CP      E      ;Same with current background?
5170      1883      28 19      JR      Z,SAMEBG ;Yes
5171      1885      87      ADD     A,A
```

5172	1886	87	ADD	A,A	
5173	1887	87	ADD	A,A	
5174	1888	87	ADD	A,A	
5175	1889	BA	CP	D	;Same with current foreground?
5176	188A	28 16	JR	Z,SAMEFG	;Yes
5177	188C	F5	PUSH	AF	;Save new foreground color
5178	188D	78	LD	A,B	
5179	188E	B1	OR	C	
5180	188F	FE FF	CP	OFFH	;All pixels are going to be set?
5181	1891	28 17	JR	Z,PATWRL	;Yes, Spock will use a new repair technique
5182					;logically...
5183	1893	E5	PUSH	HL	;Save address of color table
5184	1894	D5	PUSH	DE	;Save current background color
5185	1895	CD 18A2	CALL	SAMEFG	;Write to VRAM (pattern)
5186	1898	D1	POP	DE	;Restore current background in [E]
5187	1899	E1	POP	HL	;Restore color table address
5188	189A	F1	POP	AF	;Restore new foreground color in upper
5189					;4 bits of [Acc]
5190	189B	B3	OR	E	;Form new foreground and background color
5191	189C	18 1A	JR	JMPWRT	;Write to color table
5192	189E				
5193					
5194	189E	78	LD	A,B	
5195	189F	2F	CPL		
5196	18A0	A1	AND	C	
5197	18A1	11	DB	11H	;Skip next 2 bytes (LXI D)
5198	18A2				
5199	18A2	78	LD	A,B	
5200	18A3	B1	OR	C	
5201	18A4				
5202	18A4	11 2000	LD	DE,GRPDIF	

5203	18A7	19		ADD	HL,DE	
5204	18A8	18 0E		JR	JMPWRT	
5205	18AA		PATWRT:			;Write to pattern table
5206			;			
5207	18AA	F1		POP	AF	;Discard new foreground color
5208	18AB	78		LD	A,B	;Reget specified pattern
5209	18AC	2F		CPL		;Forget current background color, 'cause
5210	18AD	E5		PUSH	HL	;there's no background, we display
5211	18AE	D5		PUSH	DE	;new pattern as background color.
5212	18AF	CD 18A4		CALL	WTPTAB	;Write to pattern table
5213	18B2	D1		POP	DE	
5214	18B3	E1		POP	HL	
5215	18B4	3A F3F2		LD	A,(ATRBYT)	;Get new color (this will be the
5216						;background color)
5217	18B7	B2		OR	D	;Add current foreground color
5218	18B8		JMPWRT:			
5219	18B8	C3 07CD		JP	WRTVRM	;Write to VRAM (color)

```
5220
5221      18BB      MNSTCX:
5222      ;
5223      ; NSETCX for multicolor screen
5224      ;
5225      18BB      E5          PUSH      HL          ;Save counter
5226      18BC      CD 167E    CALL      SETC        ;Set pixel
5227      18BF      CD 16C5    CALL      RIGHTC      ;Move to right
5228      18C2      E1          POP       HL          ;Restore counter
5229      18C3      2D          DEC       L
5230      18C4      20 F5      JR        NZ,MNSTCX
5231      18C6      C9          RET
5232      18C7      GTASPC:
5233      ;
5234      ; GTASPC - load aspect ratio for CIRCLE
5235      ;
5236      18C7      2A F40B     LD         HL,(ASCPCT1)
5237      18CA      EB          EX         DE,HL
5238      18CB      2A F40D     LD         HL,(ASCPCT2)
5239      18CE      C9          RET
5240      SUBTTL -MSXGRP - (Routines for paint)
```

```
5241
5242      18CF          PNTINI:
5243                      ;
5244                      ; PNTINI - Initialize border color
5245                      ;
5246      18CF      F5          PUSH      AF          ;Save specified color
5247      18D0      CD 15D9     CALL      CHKMOD     ;In what mode are we now?
5248      18D3      28 06      JR         Z,PNTHRS   ;High-resolution mode
5249      18D5      F1          POP         AF
5250      18D6      FE 10      CP          10H       ;Legal value?
5251      18D8      3F          CCF              ;Carry means illegal
5252      18D9      18 05      JR         PNTIRT
5253      18DB          PNTHRS:
5254                      ;
5255      18DB      F1          POP         AF          ;Discard specified color
5256      18DC      3A F3F2     LD          A,(ATRBYT) ;Always ignore specified border
5257      18DF      A7          AND         A          ;Always legal
5258      18E0          PNTIRT:
5259      18E0      32 FCB2     LD          (BRDATR),A ;Set border color
5260      18E3      C9          RET              ;Return with the condition
5261      18E4          SCANR:
5262                      ;
5263                      ; SCANR - scan pixels to right
5264                      ; Maximum number of pixels to test is passed in [DE].
5265                      ;
5266      18E4      21 0000     LD          HL,0       ;Initialize PNTCNT
5267      18E7      4D          LD          C,L        ;Initialize PNTDFL
5268      18E8      CD 15D9     CALL      CHKMOD     ;Check current screen mode
5269      18EB      20 64      JR         NZ,MSCANR  ;Multi-color mode
5270                      ;
5271                      ; Scan to right in high-resolution mode
```

```
5272                ; [B] set to 0 is need to suspend painting, 1 otherwise.
5273                ;
5274                ;      Work1 = Temporary storage for 'suspend painting'
5275                ;      Work2 = Save area for pixel count to draw right
5276                ;      Work3 = Save area for 'pixel changed' flag
5277                ;
5278    18ED    78                LD      A,B
5279    18EE    32 F866          LD      (RUNFLG),A      ;Remember to suspend or not
5280    18F1    AF              XOR      A              ;Clear 'pixel changed' flag
5281    18F2    32 F869          LD      (WORK3),A
5282    18F5    3A FCB2          LD      A,(BRDATR)
5283    18F8    47              LD      B,A              ;Set border color to [B] for comparison
5284    18F9                                SCANR1:
5285    18F9    CD 1647          CALL     READC          ;Read current color
5286    18FC    B8              CP      B              ;Still on border?
5287    18FD    20 0D           JR      NZ,SCANR2       ;No, start painting
5288    18FF    1B              DEC     DE              ;All pixels tested?
5289    1900    7A              LD      A,D
5290    1901    B3              OR      E
5291    1902    C8              RET      Z              ;Yes
5292    1903    CD 16AC          CALL     TRIGHT         ;Advance to right, and check if out of screen
5293    1906    30 F1           JR      NC,SCANR1       ;Not yet out of screen, continue
5294    1908    11 0000         LD      DE,0           ;All pixels has border attribute on
5295    190B    C9              RET                     ;this row, let BRDCNT be 0, and return
5296    190C                                SCANR2:
5297                ;
5298                ; A pixel with non-border attribute is found. Start painting
5299                ;
5300    190C    CD 19AE          CALL     CHKCHG         ;Check if pixel changed
5301    190F    D5              PUSH     DE              ;Save BRDCNT
5302    1910    CD 1639          CALL     FETCHC         ;Get current CLOC, CMASK
```


5303	1913	22 F942	LD	(CSAVEA),HL	;Set first non-border pixel encountered
5304	1916	32 F944	LD	(CSAVEM),A	
5305	1919	11 0000	LD	DE,0	;Initialize # of painted pixels (PNTCNT)
5306	191C		SCANR3:		
5307	191C	13	INC	DE	;Update PNTCNT
5308	191D	CD 16AC	CALL	TRIGHT	;Move 1 pixel right
5309	1920	38 0B	JR	C,SCANR4	;Out of screen
5310	1922	CD 1647	CALL	READC	;Read color of current pixel
5311	1925	B8	CP	B	;Reached border?
5312	1926	28 05	JR	Z,SCANR4	;Yes
5313	1928	CD 19AE	CALL	CHKCHG	;Check if pixel changed
5314	192B	18 EF	JR	SCANR3	;Keep on scanning
5315	192D		SCANR4:		
5316			;		
5317	192D	D5	PUSH	DE	;Save PNTCNT
5318	192E	CD 1639	CALL	FETCHC	;Since NSETCX does not update 'C', these value
5319	1931	E5	PUSH	HL	; must be saved
5320	1932	F5	PUSH	AF	
5321	1933	2A F942	LD	HL,(CSAVEA)	;Set where to start painting
5322	1936	3A F944	LD	A,(CSAVEM)	
5323	1939	CD 1640	CALL	STOREC	;Set CLOC and CMASK
5324	193C	EB	EX	DE,HL	;Set length of line to [HL] (PNTCNT)
5325	193D	22 F867	LD	(WORK2),HL	
5326	1940	3A F866	LD	A,(WORK1)	;Same as [RUNFLG]
5327	1943	A7	AND	A	
5328	1944	C4 1809	CALL	NZ,NSETCX	;Draw [HL] pixels to the right if not suspend
5329	1947	F1	POP	AF	;Restore 'last-examined-pixel' information
5330	1948	E1	POP	HL	
5331	1949	CD 1640	CALL	STOREC	
5332	194C	E1	POP	HL	;Restore PNTCNT
5333	194D	D1	POP	DE	;Restore BRDCNT

(MSX ROM BASIC BIOS) Macro-80
-MSXGRP - (Routines for paint)

3.44

01-Jan-85

PAGE

55-3

194

5334 194E C3 19A9

JP

SCANL4

```
5335
5336      1951      MSCANR:
5337      ;
5338      ; Scan to right in multi-color mode
5339      ;
5340      1951      CD 19C7      CALL      MTSBRD      ;Is it border color?
5341      1954      30 0D      JR      NC,MSCNR1      ;No, start painting
5342      1956      1B      DEC      DE      ;All pixels tested?
5343      1957      7A      LD      A,D
5344      1958      B3      OR      E
5345      1959      C8      RET      Z      ;Yes
5346      195A      CD 16AC      CALL      TRIGHT      ;Advance to right, and check if out of screen
5347      195D      30 F2      JR      NC,MSCANR      ;Not yet out of screen, continue
5348      195F      11 0000      LD      DE,0      ;Out of screen, let BRDCNT be 0, and return
5349      1962      C9      RET
5350      1963      MSCNR1:
5351      ;
5352      1963      CD 1639      CALL      FETCHC      ;Get CLOC,CMASK
5353      1966      22 F942      LD      (CSAVEA),HL      ;Save VRAM address
5354      1969      32 F944      LD      (CSAVEM),A      ;Save mask pattern
5355      196C      21 0000      LD      HL,0      ;Initialize PNTCNT
5356      196F      MSCNR2:
5357      196F      23      INC      HL      ;Increment PNTCNT
5358      1970      CD 16AC      CALL      TRIGHT      ;Advance to right, and check if out of screen
5359      1973      D8      RET      C      ;Going out of screen
5360      1974      CD 19C7      CALL      MTSBRD      ;Reached border color?
5361      1977      30 F6      JR      NC,MSCNR2      ;Not yet, continue
5362      1979      C9      RET
```

```
5363
5364      197A      SCANL:
5365                ;
5366                ; SCANL - Scan pixels to left
5367                ;
5368      197A      21 0000      LD      HL,0      ;Initialize PNTCNT
5369      197D      4D          LD      C,L      ;Initialize PNTDFL
5370      197E      CD 15D9     CALL    CHKMOD    ;Check current screen mode
5371      1981      20 37      JR      NZ,MSCANL  ;Multi-color mode
5372                ;
5373                ; Scan to left in high-resolution mode
5374                ;
5375      1983      AF          XOR      A          ;Clear 'pixel changed' flag
5376      1984      32 F869     LD      (WORK3),A
5377      1987      3A FCB2     LD      A,(BRDATR)
5378      198A      47          LD      B,A          ;Set border color to [B] for comparison
5379      198B      SCANL1:
5380      198B      CD 16D8     CALL    TLEFT      ;Advance to left, and check if out of screen
5381      198E      38 0F      JR      C,SCANL3    ;On left edge
5382      1990      CD 1647     CALL    READC      ;Read color of target pixel
5383      1993      B8          CP      B          ;Reached border?
5384      1994      28 06      JR      Z,SCANL2    ;Yes
5385      1996      CD 19AE     CALL    CHKCHG     ;Check if pixel changed
5386      1999      23          INC      HL        ;Update PNTCNT
5387      199A      18 EF      JR      SCANL1
5388      199C      SCANL2:
5389                ;
5390      199C      CD 16C5     CALL    RIGHTC     ;'C' must specify 'last pixel painted'
5391      199F      SCANL3:
5392      199F      E5          PUSH     HL        ;Save PNTCNT
5393      19A0      ED 5B F867  LD      DE,(WORK2) ;Load suspended pixels which remain
```

```
5394      19A4      19                      ADD      HL,DE           ;to the right
5395      19A5      CD 1809                  CALL     NSETCX          ;Draw [HL] pixel from current 'C'
5396      19A8      E1                      POP      HL           ;Restore PNTCNT
5397      19A9                      SCANL4:
5398      19A9      3A F869                  LD       A,(WORK3)       ;Non 0 if pixels changed attribute
5399      19AC      4F                      LD       C,A
5400      19AD      C9                      RET
5401      19AE                      CHKCHG:
5402      ;
5403      19AE      E5                      PUSH     HL
5404      19AF      21 F3F2                  LD       HL,ATRBYT       ;Get specified paint attribute
5405      19B2      BE                      CP       (HL)          ;Same?
5406      19B3      E1                      POP      HL
5407      19B4      C8                      RET      Z           ;Yes, no change of attribute
5408      19B5      3C                      INC      A           ;Load non 0 to [Acc]
5409      19B6      32 F869                  LD       (WORK3),A      ;Remember this temporarily
5410      19B9      C9                      RET
5411      19BA                      MSCANL:
5412      ;
5413      ; Scan to left in multi-color mode
5414      ;
5415      19BA      CD 16D8                  CALL     TLEFT          ;Advance to left, and check if out of screen
5416      19BD      D8                      RET      C           ;going out of screen
5417      19BE      CD 19C7                  CALL     MTSBRD         ;Reached border color?
5418      19C1      DA 16C5                  JP       C,RIGHTC       ;Yes, adjust CLOC, CMASK and return
5419      19C4      23                      INC      HL           ;Increment PNTCNT
5420      19C5      18 F3                  JR       MSCANL         ;Continue
5421      19C7                      MTSBRD:
5422      ;
5423      ; Test border subroutine for multi-color mode
5424      ;
```

5425	19C7	CD 1647	CALL	READC	;Get the color of target pixel
5426	19CA	47	LD	B,A	
5427	19CB	3A FCB2	LD	A,(BRDATR)	;Load specified border color
5428	19CE	90	SUB	B	;Reached border?
5429	19CF	37	SCF		;Assume so
5430	19D0	C8	RET	Z	;Yes, return with carry flag set
5431	19D1	3A F3F2	LD	A,(ATRBYT)	;Is current pixel same as ATRBYT?
5432	19D4	B8	CP	B	
5433	19D5	C8	RET	Z	;Yes, no changes made.
5434					;Return with carry reset
5435	19D6	CD 167E	CALL	SETC	;Set this pixel to ATRBYT
5436	19D9	0E 01	LD	C,1	;Set 'pixel-changed' flag
5437	19DB	A7	AND	A	;Tell caller that we plot a dot
5438	19DC	C9	RET		
5439			SUBTTL -CASET- Cassette drivers stuff		

```
5440
5441           ; Cassette read/write stuff
5442           ;
5443           ; Following driver assumes that T cycle is 279.365 ns
5444           ;
5445           ; Variables referenced
5446           ;     PPI.CM           To write to cassette
5447           ;     PSG.DR           To read from cassette
5448           ;     BREAKX          Routine to check for [STOP] key pressed
5449           ;
5450 19DD      TAPOFF:
5451           ;
5452 19DD      C5           PUSH     BC
5453 19DE      F5           PUSH     AF
5454 19DF      01 0000     LD       BC,0
5455 19E2      CTWOF1:
5456 19E2      0B           DEC     BC
5457 19E3      78           LD      A,B           ;Test BC
5458 19E4      B1           OR      C
5459 19E5      20 FB       JR      NZ,CTWOF1
5460 19E7      F1           POP     AF
5461 19E8      C1           POP     BC
5462 19E9      TAPIOF:
5463 19E9      F5           PUSH     AF
5464 19EA      3E 09       LD      A,00001001B   ;Stop motor
5465 19EC      D3 AB       OUT     (PPI.CM),A
5466 19EE      F1           POP     AF
5467 19EF      FB           EI
5468 19F0      C9           RET
5469 19F1      TAPOON:
5470           ;
```

```
5471          ; Write out header, if [A]=0 then write short header
5472          ; otherwise write long header ( 5sec)
5473          ;
5474      19F1      B7          OR      A          ;set flag for length of header
5475      19F2      F5          PUSH     AF          ;save flag
5476      19F3      3E 08      LD      A,8          ;Motor on
5477      19F5      D3 AB      OUT     (PPI.CM),A
5478      19F7      21 0000    LD      HL,0
5479      19FA      MOTRWT:
5480      19FA      2B          DEC     HL
5481      19FB      7C          LD      A,H
5482      19FC      B5          OR      L
5483      19FD      20 FB      JR      NZ,MOTRWT    ;wait till motor starts
5484      19FF      F1          POP     AF          ;get back header length flag
5485      1A00      3A F40A    LD      A,(HEADER)   ;get length of header
5486      1A03      28 02      JR      Z,SYNCW1    ;short header
5487      1A05      87          ADC     A,A
5488      1A06      87          ADL     A,A
5489      1A07      SYNCW1:
5490      1A07      47          LD      B,A
5491      1A08      0E 00      LD      C,0          ;set up counter
5492      1A0A      F3          DI              ;Don't disturb during writing to cassette
5493      1A0B      SYNLP1:
5494      1A0B      CD 1A4D    CALL     BITLOT      ;Write enough marks
5495      1A0E      CD 1A3F    CALL     RETRET      ;compensate overhead
5496      1A11      0B          DEC     BC
5497      1A12      78          LD      A,B
5498      1A13      B1          OR      C
5499      1A14      20 F5      JR      NZ,SYNLP1    ;loop till counter exhausts
5500      1A16      C3 046F    JP      BREAKX      ;check control-stop and return
5501      1A19      TAPOUT:
```



```

5502      1A19          DATAW:
5503          ;
5504          ; Output a byte
5505          ;
5506      1A19      2A F406      LD      HL,(LOW)      ;get time constants for space
5507      1A1C      F5          PUSH     AF
5508      1A1D      7D          LD      A,L
5509      1A1E      D6 0E      SUB      0EH      ;compensate loss time since last stop bit
5510      1A20      6F          LD      L,A
5511      1A21      CD 1A50      CALL     BITOUT      ;output start bit
5512      1A24      F1          POP      AF
5513      1A25      06 08      LD      B,8      ;Initialize counter
5514      1A27          DATAWL:
5515      1A27      0F          RRCA          ;next bit to carry
5516      1A28      DC 1A40      CALL     C,BIT1      ;output mark if the bit is 1
5517      1A2B      D4 1A39      CALL     NC,BIT0      ;Output space
5518      1A2E      10 F7      DJNZ     DATAWL      ;Loop until 8 bits sent
5519      1A30      CD 1A40      CALL     BIT1      ;Output stop bit
5520      1A33      CD 1A40      CALL     BIT1
5521      1A36      C3 046F      JP      BREAKX      ;Check if break pressed and return
  
```

```

5522                                     BIT0:
5523     1A39                                     ;
5524                                     ;
5525                                     ; Output a bit to cassette
5526                                     ;
5527                                     ; Absolute jumps are used to improve accuracy
5528                                     ;
5529     1A39    2A F406                LD      HL,(LOW)        ;Output 0 (space)          (17 T)
5530     1A3C    CD 1A50                CALL   BITOUT           ;                      (18 T)
5531     1A3F                                RETRET:
5532     1A3F    C9                     RET                    ;                      (11 T)
5533     1A40                                BIT1:
5534                                     ;
5535     1A40    CD 1A4D                CALL   BITLOT         ;                      (18 T)
5536     1A43    E3                     EX      (SP),HL        ;                      (20 T)
5537     1A44    E3                     EX      (SP),HL        ;compensate overhead (20 T)
5538     1A45    00                     NOP                   ;(Total 60 state)   ( 5 T)
5539     1A46    00                     NOP                   ;                      ( 5 T)
5540     1A47    00                     NOP                   ;                      ( 5 T)
5541     1A48    00                     NOP                   ;                      ( 5 T)
5542     1A49    CD 1A4D                CALL   BITLOT         ;To compensate time (18 T)
5543     1A4C    C9                     RET                    ;Don't change this (11 T)
5544     1A4D                                BITLOT:
5545                                     ;
5546                                     ; output a single cycle
5547                                     ;
5548                                     ; Total number of states =16 x [L] + 16 x [H] + 71
5549                                     ;                               =4.47uS x [L] + 4.47uS x [H] + 19.8usec
5550                                     ;
5551     1A4D    2A F408                LD      HL,(HIGH)       ;                      (17 T)
5552     1A50                                BITOUT:

```

```

5553      1A50      F5                PUSH    AF                ;                (12 T)
5554                ;
5555      1A51                KEEPL:
5556      1A51      2D                DEC     L                ;Keep low level      ( 5 T)
5557      1A52      C2 1A51          JP      NZ,KEEPL          ;                (11 T)
5558      1A55      3E 0B          LD      A,0BH              ;                ( 8 T)
5559      1A57      D3 AB          OUT     (PPI.CM),A          ;Output high level  (11 T)
5560      1A59                KEEPH:
5561      1A59      25                DEC     H                ;keep high level    ( 5 T)
5562      1A5A      C2 1A59          JP      NZ,KEEPH          ;                (11 T)
5563      1A5D      3E 0A          LD      A,0AH              ;                ( 8 T)
5564      1A5F      D3 AB          OUT     (PPI.CM),A          ;Output low level   (11 T)
5565      1A61      F1                POP     AF                ;Restore data       (12 T)
5566                ;
5567      1A62      C9                RET                     ;                (11 T)
5568      1A63                TAPION:
5569                ;
5570                ; Detect header block
5571                ;
5572      1A63      3E 08          LD      A,8                ;Motor on
5573      1A65      D3 AB          OUT     (PPI.CM),A
5574      1A67      F3                DI
5575      1A68      3E 0E          LD      A,0EH              ;Select PSG port A
5576      1A6A      D3 A0          OUT     (PSG.LW),A
5577      1A6C                SYN05:
5578                ;
5579                ; First, wait until a series of good pulses are found.
5580                ;
5581      1A6C      21 0457          LD      HL,0457H          ;Initialize counter
5582                ;Number of pulse to detect header
5583      1A6F                SYN10:

```

```
5584      1A6F      51              LD      D,C              ;Remember last value
5585      1A70      CD 1B34          CALL    CNTFUL          ;Count full cycle
5586      1A73      D8              RET      C              ;Aborted
5587      1A74      79              LD      A,C              ;Get count
5588      1A75      FE DE            CP      0DEH            ;0DE = Max count
5589      1A77      30 F3            JR      NC,SYN05          ;Too long, reset number of pulses
5590      1A79      FE 05            CP      5              ;5 = Min count
5591      1A7B      38 EF            JR      C,SYN05           ;Too short, reset number of pulses
5592                                     ;
5593                                     ; Now compare with last pulse width and approve this as a good pulse
5594                                     ; if this is similar to last one.
5595                                     ;
5596      1A7D      92              SUB      D              ;current - last
5597      1A7E      30 02            JR      NC,SYN11
5598      1A80      2F              CPL              ;result was negative, negate it
5599      1A81      3C              INC      A
5600      1A82                                     SYN11:
5601      1A82      FE 04            CP      4              ;within a wow allowance?
5602      1A84      30 E6            JR      NC,SYN05          ;no, reset number of pulse ever seen
5603      1A86      2B              DEC      HL
5604      1A87      7C              LD      A,H
5605      1A88      B5              OR      L
5606      1A89      20 E4            JR      NZ,SYN10          ;Loop till seen enough good pulses
5607                                     ;
5608      1A8B                                     SYN20:
5609                                     ;
5610                                     ; Next, calculate the mean width of pulse.
5611                                     ;
5612      1A8B      21 0000          LD      HL,0              ;Initialize sum
5613      1A8E      45              LD      B,L              ;Initialize high byte of [BC] pair
5614      1A8F      55              LD      D,L              ;Loop 256 times
```

```
5615      1A90                      SYN30:
5616      1A90      CD 1B34          CALL    CNTFUL
5617      1A93      D8                RET     C
5618      1A94      09                ADD     HL,BC
5619      1A95      15                DEC     D
5620      1A96      C2 1A90           JP      NZ,SYN30
5621      1A99      01 06AE          LD       BC,06AEH          ;compensate over head
5622      1A9C      09                ADD     HL,BC
5623
5624      ; Set various values for read routine. Those are,
5625      ;
5626      ; LOWLIM - lower limit of the width of start bit. [H]*1.5
5627      ; WINWID - width of window to count the transition.
5628      ;
5629      1A9D      7C                LD       A,H              ;[H] has mean pulse width
5630      1A9E      1F                RRA
5631      1A9F      E6 7F            AND      7FH
5632      1AA1      57                LD       D,A              ;[D]=[mean]/2
5633      1AA2      29                ADD     HL,HL
5634      1AA3      7C                LD       A,H              ;[A]=[mean]x2
5635      1AA4      92                SUB     D                  ;[A]=[mean]x1.5
5636      1AA5      57                LD       D,A              ;save
5637      1AA6      D6 06            SUB     6                  ;compensate overhead at DATAR
5638      1AA8      32 FCA4          LD       (LOWLIM),A
5639
5640      ; Set width of window 'WINWID'
5641      ; CNTFUL takes 40T for a loop, RDBIT takes 60T for loop
5642      ; set WINWID as 3 times wider than single short pulse ([mean]/2)
5643      ; [WINWID]=[mean] x 1.5 x 40/60
5644      ;      =[D] x 2/3
5645      ;
```

5646	1AAB	7A	LD	A,D	;get [mean width]x1.75
5647	1AAC	87	ADD	A,A	;x2
5648	1AAD	06 00	LD	B,0	;clear quotient
5649	1AAF		SULOP:		
5650	1AAF	D6 03	SUB	3	
5651	1AB1	04	INC	B	
5652	1AB2	30 FB	JR	NC,SULOP	;loop till get carry
5653	1AB4	78	LD	A,B	;[A]=[mean]x1.75x2/3
5654	1AB5	D6 03	SUB	3	;compensate overhead in RDBIT routine
5655	1AB7	32 FCA5	LD	(WINWID),A	
5656	1ABA	B7	OR	A	
5657	1ABB	C9	RET		

```
5658
5659      1ABC      TAPIN:
5660      ;
5661      ; Read a byte from cassette
5662      ;
5663      1ABC      3A FCA4      LD      A,(LOWLIM)
5664      1ABF      57          LD      D,A          ;[D] has lower limit for start bit
5665      1AC0      DATAR:
5666      1AC0      CD 046F      CALL    BREAKX
5667      1AC3      D8          RET      C          ;Aborted
5668      1AC4      DB A2      IN      A,(PSG.DR)    ;Get cassette
5669      1AC6      07          RLCA          ;High state?
5670      1AC7      30 F7      JR      NC,DATAR      ;No
5671      1AC9      DATAR0:
5672      1AC9      CD 046F      CALL    BREAKX
5673      1ACC      D8          RET      C          ;Aborted
5674      1ACD      DB A2      IN      A,(PSG.DR)    ;Get cassette
5675      1ACF      07          RLCA          ;falling egde?
5676      1AD0      38 F7      JR      C,DATAR0      ;No
5677      1AD2      1E 00      LD      E,0          ;Initialize edge mask
5678      1AD4      CD 1B1F      CALL    CNTHLF      ;Get width in [C]
5679      1AD7      DATAR1:
5680      1AD7      41          LD      B,C          ;Save old width
5681      1AD8      CD 1B1F      CALL    CNTHLF      ;Get new width in [C]
5682      1ADB      D8          RET      C          ;aborted
5683      1ADC      78          LD      A,B          ;Add width of 2 pulses
5684      1ADD      81          ADD     A,C
5685      1ADE      DA 1AD7      JP      C,DATAR1    ;Pulse too long
5686      1AE1      BA          CP      D          ;Longer than lower limit?
5687      1AE2      38 F3      JR      C,DATAR1    ;No
5688      ;
```

```
5689                ; Now, a valid start bit has been found.
5690                ; [E] = 0      if NORMAL polarity,
5691                ;      =255     if REVERSE polarity.
5692                ;
5693    1AE4    2E 08                LD      L,8                ;Initialize counter
5694    1AE6                                DATARL:
5695    1AE6    CD 1B03            CALL    RDBIT
5696    1AE9    FE 04                CP      3+1                ;Legal transitions?
5697    1AEB    3F                CCF
5698    1AEC    D8                RET      C                ;Too many transitions
5699    1AED    FE 02                CP      2
5700    1AEF    3F                CCF                ;Set carry if 2 or 3 transitions
5701    1AF0    CB 1A            RR      D
5702                ;
5703                ; We've just assembled a bit. A check must be done to make sure
5704                ; that we're at the start of next bit field.
5705                ;
5706    1AF2    79                LD      A,C                ;Reget number of transitions
5707    1AF3    0F                RRCA
5708    1AF4    D4 1B23            CALL    NC,CNTHL0            ;Wait for next transition if 0 or 2
5709    1AF7    CD 1B1F            CALL    CNTHLF
5710    1AFA    2D                DEC     L
5711    1AFB    C2 1AE6            JP      NZ,DATARL            ;Loop till done
5712    1AFE    CD 046F            CALL    BREAKX            ;return with carry set if breaked
5713    1B01    7A                LD      A,D
5714    1B02    C9                RET
5715    1B03                                RDBIT:
5716                ;
5717                ; Count number of transitions within a period specified by 'WINWID'
5718                ;
5719                ; length of window = 17uSec x [WINWID] + 12.3 uSec
```



```
5720      ;
5721      ; [D],[H] and [L] are preserved.
5722      ; [E] is updated to prepare for next edge
5723      ;
5724      1B03      3A FCA5      LD      A,(WINWID)      ;Get width of window
5725      1B06      47          LD      B,A
5726      1B07      0E 00      LD      C,0              ;Clear # of transitions seen
5727      1B09      RDBITL:
5728      1B09      DB A2      IN      A,(PSG.DR)      ;Get a bit
5729      1B0B      AB          XOR      E              ;Any changes?
5730      1B0C      F2 1B17    JP      P,NOTRAN        ;No
5731      1B0F      7B          LD      A,E              ;Transition seen
5732      1B10      2F          CPL          ;Prepare for next transition
5733      1B11      5F          LD      E,A
5734      1B12      0C          INC      C              ;Increment # of transitions
5735      1B13      10 F4      DJNZ     RDBITL
5736      1B15      79          LD      A,C              ;Get transition count
5737      1B16      C9          RET
5738      1B17      NOTRAN:
5739      ;
5740      1B17      00          NOP                    ;Compensate time
5741      1B18      00          NOP
5742      1B19      00          NOP
5743      1B1A      00          NOP
5744      1B1B      10 EC      DJNZ     RDBITL
5745      ;
5746      1B1D      79          LD      A,C              ;Get transition count
5747      1B1E      C9          RET
```

```

5748
5749      1B1F      CNTHLF:
5750                ;
5751                ; Count half cycle
5752                ;      1T      =279.4nS
5753                ;      period=[C] x 11.18 + 35.48uS
5754                ;
5755      1B1F      CD 046F      CALL      BREAKX      ;Break?      (87 T)
5756      1B22      D8          RET      C      ;Yes, aborted      ( 6 T)
5757      1B23      CNTHL0:
5758      1B23      0E 00      LD      C,0      ;Initialize counter      ( 8 T)
5759      1B25      CNTHL1:
5760      1B25      0C          INC      C      ;# of state for this loop
5761                ;40T=11.18usec      ( 5 T)
5762      1B26      28 0A      JR      Z,TIMOUT      ;Pulse too long      ( 8 T)
5763      1B28      DB A2      IN      A,(PSG.DR)      ;Read cassette      (11 T)
5764      1B2A      AB          XOR      E      ;Desired transition?      ( 5 T)
5765      1B2B      F2 1B25      JP      P,CNTHL1      ;No      (11 T)
5766      1B2E      7B          LD      A,E      ;Complement edge mask      ( 5 T)
5767      1B2F      2F          CPL      ;      ( 5 T)
5768      1B30      5F          LD      E,A      ;      ( 5 T)
5769      1B31      C9          RET      ;      (11 T)
5770      1B32      TIMOUT:
5771                ;
5772      1B32      0D          DEC      C      ;Load 255
5773      1B33      C9          RET
5774      1B34      CNTFUL:
5775                ;
5776                ; Count full cycle
5777                ;
5778      1B34      CD 046F      CALL      BREAKX

```

(MSX ROM BASIC BIOS) Macro-80
-CASET- Cassette drivers stuff

3.44

01-Jan-85

PAGE 61-1

211

5779	1B37	D8	RET	C	;Aborted
5780	1B38	DB A2	IN	A,(PSG.DR)	;Get cassette
5781	1B3A	07	RLCA		;Low state?
5782	1B3B	38 F7	JR	C,CNTFUL	;No
5783	1B3D	1E 00	LD	E,0	;Initialize edge mask
5784	1B3F	CD 1B23	CALL	CNTHL0	
5785	1B42	C3 1B25	JP	CNTHL1	
5786			SUBTTL	- BIO -	OUTDO routine

- BIO - OUTDO routine

```

5787      1B45      OUTDO:
5788      ;
5789      ; OUTDO ( RST 18H )
5790      ; Prints char in [A], to either terminal or disk
5791      ; or printer depending on the flags:
5792      ; PRTFLG if non-zero print to printer
5793      ; PTRFIL if non-zero print to disk file pointed
5794      ; to by PTRFIL
5795      ;
5796      ;
5797      1B45      F5      PUSH      AF      ;Save character
5798      1B46      CD FEE4      CALL      H.OUTD
5799      1B49      CD 145F      CALL      ISFLIO      ;Doing I/O to file?
5800      1B4C      28 08      JR      Z,LPTCOD      ;Nope, check for output to printer
5801      1B4E      F1      POP      AF      ;Restore char.
5802      1B4F      DD 21 6C48      LD      IX,FILOUL      ;Jump with pointer to FILE OUT routine
5803      1B53      C3 01FF      JP      CALBAS
5804      ;
5805      1B56      LPTCOD:
5806      1B56      3A F416      LD      A,(PRTFLG)      ;Output to printer?
5807      1B59      B7      OR      A
5808      1B5A      28 5F      JR      Z,TTYCHR      ;Nope, output to console
5809      1B5C      3A F418      LD      A,(RAWPRT)      ;Print in "RAW" mode?
5810      1B5F      A7      AND      A
5811      1B60      20 49      JR      NZ,LPTCH1      ;Yes, send char to printer
5812      1B62      F1      POP      AF      ;restore char
5813      ;
5814      1B63      OUTDLP:
5815      1B63      F5      PUSH      AF
5816      ;
5817      1B64      NTBKS2:

```

```

5818 1B64 FE 09 CP 9 ;TAB?
5819 1B66 20 0E JR NZ,NOTABL ;No
5820 ;
5821 1B68 MORSP1:
5822 1B68 3E 20 LD A, ' ' ;Print a space
5823 1B6A CD 1B63 CALL OUTDLP
5824 1B6D 3A F415 LD A,(LPTPOS) ;Get current LPOS
5825 1B70 E6 07 AND 7 ;At TAB stop?
5826 1B72 20 F4 JR NZ,MORSP1 ;No, back for more space
5827 1B74 F1 POP AF ;Discard character
5828 1B75 C9 RET
5829 ;
5830 1B76 NOTABL:
5831 1B76 D6 0D SUB 0DH ;Check if CR. If so load a zero
5832 1B78 28 0A JR Z,ZERLP1 ;It is, clear LPTPOS and send CR
5833 1B7A 38 0B JR C,LPTCH0 ;Code is 0..0CH, just send.
5834 ;without modify LPTPOS
5835 1B7C FE 13 CP "-13 ;See if control character
5836 1B7E 38 07 JR C,LPTCH0 ;Code is 0EH..1FH, ditto
5837 1B80 3A F415 LD A,(LPTPOS) ;Get LPOS
5838 1B83 3C INC A
5839 ;
5840 1B84 ZERLP1:
5841 1B84 32 F415 LD (LPTPOS),A ;Update LPOS
5842 ;
5843 1B87 LPTCH0:
5844 1B87 3A F417 LD A,(NTMSXP) ;Output to MSX standard printer
5845 1B8A A7 AND A
5846 1B8B 28 1E JR Z,LPTCH1 ;No mapping for KATAKANA to HIRAGANA
5847 1B8D F1 POP AF ;restore char to print
5848 1B8E CD 089D CALL CNVCHR ;See if graphic header

```

```

5849  1B91  D0                RET    NC                ;Yep
5850  1B92  20 23            JR      NZ,MAPSPC        ;Graphic symbol, map to space
5851  1B94  A7                AND     A                ;
5852  1B95  F2 1BAC          JP      P,LPTCHR          ;
5853  1B98  FE 86            CP      86H              ;Graphic symbol?
5854  1B9A  38 1B            JR      C,MAPSPC        ;Yes, map this to space too!
5855  1B9C  FE A0            CP      0A0H             ;A HIRAGANA(part 1)?
5856  1B9E  30 04            JR      NC,NTHIRA        ;
5857  1BA0  C6 20            ADD     A,' '            ;Map to KATAKANA
5858  1BA2  18 08            JR      LPTCHR           ;
5859  1BA4                NTHIRA:
5860  1BA4  FE E0            CP      0E0H              ;HIRAGANA(part 2)?
5861  1BA6  38 04            JR      C,LPTCHR         ;No
5862  1BA8  D6 20            SUB     ' '              ;Map to KATAKANA
5863  1BAA  38                DB      38H              ;'JRC' instruction (Skip next byte)
5864  1BAB                LPTCH1:
5865  1BAB  F1                POP     AF                ;Restore char
5866                ;
5867  1BAC                LPTCHR:
5868  1BAC  CD 085D          CALL    LPTOUT          ;Send character out
5869  1BAF  D0                RET      NC              ;Sent successful
5870  1BB0  DD 21 73B2      LD      IX,DIOERR        ;Direct I/O error
5871  1BB4  C3 01FF          JP      CALBAS          ;
5872  1BB7                MAPSPC:
5873  1BB7  3E 20            LD      A,' '            ;
5874  1BB9  18 F1            JR      LPTCHR           ;
5875  1BBB                TTYCHR:
5876                ;
5877                ; Output to console
5878                ;
5879  1BBB  F1                POP     AF                ;Get the character

```

(MSX ROM BASIC BIOS) Macro-80
- BIO - OUTDO routine

3.44

01-Jan-85

PAGE 62-3

215

5880 1BBC C3 08BC
5881

JP CHPUT

SUBTTL -MSXCHR- MSX character set

```

5882
5883      1BBF      CGTABL:
5884      1BBF      00 00 00 00      DB      00H,00H,00H,00H,00H,00H,00H
5885      1BC3      00 00 00
5886      1BC6      00 7E 42 7E      DB      00H,7EH,42H,7EH,42H,7EH,42H
5887      1BCA      42 7E 42
5888      1BCD      82 00 10 92      DB      82H,00H,10H,92H,54H,10H,28H
5889      1BD1      54 10 28
5890      1BD4      44 82 00 12      DB      44H,82H,00H,12H,14H,0F8H,14H
5891      1BD8      14 F8 14
5892      1BDB      34 52 92 00      DB      34H,52H,92H,00H,10H,10H,0FEH
5893      1BDF      10 10 FE
5894      1BE2      10 38 54 92      DB      10H,38H,54H,92H,00H,10H,28H
5895      1BE6      00 10 28
5896      1BE9      7C 92 38 54      DB      7CH,92H,38H,54H,0FEH,00H,10H
5897      1BED      FE 00 10
5898      1BF0      10 10 7C 10      DB      10H,10H,7CH,10H,10H,0FEH,00H
5899      1BF4      10 FE 00
5900      1BF7      7E 42 42 7E      DB      7EH,42H,42H,7EH,42H,42H,7EH
5901      1BFB      42 42 7E
5902      1BFE      00 40 7E 48      DB      00H,40H,7EH,48H,3CH,28H,7EH
5903      1C02      3C 28 7E
5904      1C05      08 00 FE 92      DB      08H,00H,0FEH,92H,92H,0FEH,82H
5905      1C09      92 FE 82
5906      1C0C      82 86 00 04      DB      82H,86H,00H,04H,0EEH,0A4H,0EFH
5907      1C10      EE A4 EF
5908      1C13      A2 EA 06 00      DB      0A2H,0EAH,06H,00H,28H,44H,82H
5909      1C17      28 44 82
5910      1C1A      3C 14 24 4C      DB      3CH,14H,24H,4CH,00H,28H,0C8H
5911      1C1E      00 28 C8
5912      1C21      5C EA 6C C8      DB      5CH,0EAH,6CH,0C8H,50H,00H,7CH
  
```


5913	1C25	50 00 7C		
5914	1C28	20 7C 44 7C	DB	20H,7CH,44H,7CH,44H,7CH,00H
5915	1C2C	44 7C 00		
5916	1C2F	0C 70 10 FE	DB	0CH,70H,10H,0FEH,10H,10H,10H
5917	1C33	10 10 10		
5918	1C36	00 7E 10 1E	DB	00H,7EH,10H,1EH,12H,22H,44H
5919	1C3A	12 22 44		
5920	1C3D	08 00 00 7C	DB	08H,00H,00H,7CH,28H,28H,28H
5921	1C41	28 28 28		
5922	1C44	4E 00 00 10	DB	4EH,00H,00H,10H,10H,10H,0FFH
5923	1C48	10 10 FF		
5924	1C4B	00 00 00 00	DB	00H,00H,00H,00H,00H,00H,00H
5925	1C4F	00 00 00		
5926	1C52	FF 10 10 10	DB	0FFH,10H,10H,10H,10H,10H,10H
5927	1C56	10 10 10		
5928	1C59	10 F0 10 10	DB	10H,0F0H,10H,10H,10H,10H,10H
5929	1C5D	10 10 10		
5930	1C60	10 10 1F 10	DB	10H,10H,1FH,10H,10H,10H,10H
5931	1C64	10 10 10		
5932	1C67	10 10 10 FF	DB	10H,10H,10H,0FFH,10H,10H,10H
5933	1C6B	10 10 10		
5934	1C6E	10 10 10 10	DB	10H,10H,10H,10H,10H,10H,10H
5935	1C72	10 10 10		
5936	1C75	10 10 00 00	DB	10H,10H,00H,00H,00H,0FFH,00H
5937	1C79	00 FF 00		
5938	1C7C	00 00 00 00	DB	00H,00H,00H,00H,00H,00H,1FH
5939	1C80	00 00 1F		
5940	1C83	10 10 10 10	DB	10H,10H,10H,10H,00H,00H,00H
5941	1C87	00 00 00		
5942	1C8A	F0 10 10 10	DB	0F0H,10H,10H,10H,10H,10H,10H
5943	1C8E	10 10 10		

5944	1C91	10 1F 00 00	DB	10H,1FH,00H,00H,00H,00H,10H
5945	1C95	00 00 10		
5946	1C98	10 10 F0 00	DB	10H,10H,0F0H,00H,00H,00H,00H
5947	1C9C	00 00 00		
5948	1C9F	81 42 24 18	DB	81H,42H,24H,18H,18H,24H,42H
5949	1CA3	18 24 42		
5950	1CA6	81 10 7C 10	DB	81H,10H,7CH,10H,10H,28H,44H
5951	1CAA	10 28 44		
5952	1CAD	82 00 10 10	DB	82H,00H,10H,10H,0FEH,92H,0FEH
5953	1CB1	FE 92 FE		
5954	1CB4	10 10 00 10	DB	10H,10H,00H,10H,10H,54H,54H
5955	1CB8	10 54 54		
5956	1CBB	92 10 30 00	DB	92H,10H,30H,00H,00H,00H,00H
5957	1CBF	00 00 00		
5958	1CC2	00 00 00 00	DB	00H,00H,00H,00H,00H,20H,20H
5959	1CC6	00 20 20		
5960	1CC9	20 20 00 00	DB	20H,20H,00H,00H,20H,00H,50H
5961	1CCD	20 00 50		
5962	1CD0	50 50 00 00	DB	50H,50H,00H,00H,00H,00H,00H
5963	1CD4	00 00 00		
5964	1CD7	50 50 F8 50	DB	50H,50H,0F8H,50H,0F8H,50H,50H
5965	1CDB	F8 50 50		
5966	1CDE	00 20 78 A0	DB	00H,20H,78H,0A0H,70H,28H,0F0H
5967	1CE2	70 28 F0		
5968	1CE5	20 00 C0 C8	DB	20H,00H,0C0H,0C8H,10H,20H,40H
5969	1CE9	10 20 40		
5970	1CEC	98 18 00 40	DB	98H,18H,00H,40H,0A0H,40H,0A8H
5971	1CF0	A0 40 A8		
5972	1CF3	90 98 60 00	DB	90H,98H,60H,00H,10H,20H,40H
5973	1CF7	10 20 40		
5974	1CFA	00 00 00 00	DB	00H,00H,00H,00H,00H,10H,20H

5975	1CFE	00 10 20		
5976	1D01	40 40 40 20	DB	40H,40H,40H,20H,10H,00H,40H
5977	1D05	10 00 40		
5978	1D08	20 10 10 10	DB	20H,10H,10H,10H,20H,40H,00H
5979	1D0C	20 40 00		
5980	1D0F	20 A8 70 20	DB	20H,0A8H,70H,20H,70H,0A8H,20H
5981	1D13	70 A8 20		
5982	1D16	00 00 20 20	DB	00H,00H,20H,20H,0F8H,20H,20H
5983	1D1A	F8 20 20		
5984	1D1D	00 00 00 00	DB	00H,00H,00H,00H,00H,00H,00H
5985	1D21	00 00 00		
5986	1D24	20 20 40 00	DB	20H,20H,40H,00H,00H,00H,78H
5987	1D28	00 00 78		
5988	1D2B	00 00 00 00	DB	00H,00H,00H,00H,00H,00H,00H
5989	1D2F	00 00 00		
5990	1D32	00 00 60 60	DB	00H,00H,60H,60H,00H,00H,00H
5991	1D36	00 00 00		
5992	1D39	08 10 20 40	DB	08H,10H,20H,40H,80H,00H,70H
5993	1D3D	80 00 70		
5994	1D40	88 98 A8 C8	DB	88H,98H,0A8H,0C8H,88H,70H,00H
5995	1D44	88 70 00		
5996	1D47	20 60 A0 20	DB	20H,60H,0A0H,20H,20H,20H,0F8H
5997	1D4B	20 20 F8		
5998	1D4E	00 70 88 08	DB	00H,70H,88H,08H,10H,60H,80H
5999	1D52	10 60 80		
6000	1D55	F8 00 70 88	DB	0F8H,00H,70H,88H,08H,30H,08H
6001	1D59	08 30 08		
6002	1D5C	88 70 00 10	DB	88H,70H,00H,10H,30H,50H,90H
6003	1D60	30 50 90		
6004	1D63	F8 10 10 00	DB	0F8H,10H,10H,00H,0F8H,80H,0E0H
6005	1D67	F8 80 E0		

6006	1D6A	10 08 10 E0	DB	10H,08H,10H,0E0H,00H,30H,40H
6007	1D6E	00 30 40		
6008	1D71	80 F0 88 88	DB	80H,0F0H,88H,88H,70H,00H,0F8H
6009	1D75	70 00 F8		
6010	1D78	88 10 20 20	DB	88H,10H,20H,20H,20H,20H,00H
6011	1D7C	20 20 00		
6012	1D7F	70 88 88 70	DB	70H,88H,88H,70H,88H,88H,70H
6013	1D83	88 88 70		
6014	1D86	00 70 88 88	DB	00H,70H,88H,88H,78H,08H,10H
6015	1D8A	78 08 10		
6016	1D8D	60 00 00 00	DB	60H,00H,00H,00H,20H,00H,00H
6017	1D91	20 00 00		
6018	1D94	20 00 00 00	DB	20H,00H,00H,00H,00H,20H,00H
6019	1D98	00 20 00		
6020	1D9B	00 20 20 40	DB	00H,20H,20H,40H,18H,30H,60H
6021	1D9F	18 30 60		
6022	1DA2	C0 60 30 18	DB	0C0H,60H,30H,18H,00H,00H,00H
6023	1DA6	00 00 00		
6024	1DA9	F8 00 F8 00	DB	0F8H,00H,0F8H,00H,00H,00H,0C0H
6025	1DAD	00 00 C0		
6026	1DB0	60 30 18 30	DB	60H,30H,18H,30H,60H,0C0H,00H
6027	1DB4	60 C0 00		
6028	1DB7	70 88 08 10	DB	70H,88H,08H,10H,20H,00H,20H
6029	1DBB	20 00 20		
6030	1DBE	00 70 88 08	DB	00H,70H,88H,08H,68H,0A8H,0A8H
6031	1DC2	68 A8 A8		
6032	1DC5	70 00 20 50	DB	70H,00H,20H,50H,88H,88H,0F8H
6033	1DC9	88 88 F8		
6034	1DCC	88 88 00 F0	DB	88H,88H,00H,0F0H,48H,48H,70H
6035	1DD0	48 48 70		
6036	1DD3	48 48 F0 00	DB	48H,48H,0F0H,00H,30H,48H,80H

6037	1DD7	30 48 80		
6038	1DDA	80 80 48 30	DB	80H,80H,48H,30H,00H,0E0H,50H
6039	1DDE	00 E0 50		
6040	1DE1	48 48 48 50	DB	48H,48H,48H,50H,0E0H,00H,0F8H
6041	1DE5	E0 00 F8		
6042	1DE8	80 80 F0 80	DB	80H,80H,0F0H,80H,80H,0F8H,00H
6043	1DEC	80 F8 00		
6044	1DEF	F8 80 80 F0	DB	0F8H,80H,80H,0F0H,80H,80H,80H
6045	1DF3	80 80 80		
6046	1DF6	00 70 88 80	DB	00H,70H,88H,80H,0B8H,88H,88H
6047	1DFA	B8 88 88		
6048	1DFD	70 00 88 88	DB	70H,00H,88H,88H,88H,0F8H,88H
6049	1E01	88 F8 88		
6050	1E04	88 88 00 70	DB	88H,88H,00H,70H,20H,20H,20H
6051	1E08	20 20 20		
6052	1E0B	20 20 70 00	DB	20H,20H,70H,00H,38H,10H,10H
6053	1E0F	38 10 10		
6054	1E12	10 90 90 60	DB	10H,90H,90H,60H,00H,88H,90H
6055	1E16	00 88 90		
6056	1E19	A0 C0 A0 90	DB	0A0H,0C0H,0A0H,90H,88H,00H,80H
6057	1E1D	88 00 80		
6058	1E20	80 80 80 80	DB	80H,80H,80H,80H,80H,0F8H,00H
6059	1E24	80 F8 00		
6060	1E27	88 D8 A8 A8	DB	88H,0D8H,0A8H,0A8H,88H,88H,88H
6061	1E2B	88 88 88		
6062	1E2E	00 88 C8 C8	DB	00H,88H,0C8H,0C8H,0A8H,98H,98H
6063	1E32	A8 98 98		
6064	1E35	88 00 70 88	DB	88H,00H,70H,88H,88H,88H,88H
6065	1E39	88 88 88		
6066	1E3C	88 70 00 F0	DB	88H,70H,00H,0F0H,88H,88H,0F0H
5067	1E40	88 88 F0		

6068	1E43	80 80 80 00	DB	80H,80H,80H,00H,70H,88H,88H
6069	1E47	70 88 88		
6070	1E4A	88 A8 90 68	DB	88H,0A8H,90H,68H,00H,0F0H,88H
6071	1E4E	00 F0 88		
6072	1E51	88 F0 A0 90	DB	88H,0F0H,0A0H,90H,88H,00H,70H
6073	1E55	88 00 70		
6074	1E58	88 80 70 08	DB	88H,80H,70H,08H,88H,70H,00H
6075	1E5C	88 70 00		
6076	1E5F	F8 20 20 20	DB	0F8H,20H,20H,20H,20H,20H,20H
6077	1E63	20 20 20		
6078	1E66	00 88 88 88	DB	00H,88H,88H,88H,88H,88H,88H
6079	1E6A	88 88 88		
6080	1E6D	70 00 88 88	DB	70H,00H,88H,88H,88H,88H,50H
6081	1E71	88 88 50		
6082	1E74	50 20 00 88	DB	50H,20H,00H,88H,88H,88H,0A8H
6083	1E78	88 88 A8		
6084	1E7B	A8 D8 88 00	DB	0A8H,0D8H,88H,00H,88H,88H,50H
6085	1E7F	88 88 50		
6086	1E82	20 50 88 88	DB	20H,50H,88H,88H,00H,88H,88H
6087	1E86	00 88 88		
6088	1E89	88 70 20 20	DB	88H,70H,20H,20H,20H,00H,0F8H
6089	1E8D	20 00 F8		
6090	1E90	08 10 20 40	DB	08H,10H,20H,40H,80H,0F8H,00H
6091	1E94	80 F8 00		
6092	1E97	70 40 40 40	DB	70H,40H,40H,40H,40H,40H,70H
6093	1E9B	40 40 70		
6094	1E9E	00 88 50 20	DB	00H,88H,50H,20H,70H,20H,70H
6095	1EA2	70 20 70		
6096	1EA5	20 00 70 10	DB	20H,00H,70H,10H,10H,10H,10H
6097	1EA9	10 10 10		
6098	1EAC	10 70 00 20	DB	10H,70H,00H,20H,50H,88H,00H

6099	1EB0	50 88 00		
6100	1EB3	00 00 00 00	DB	00H,00H,00H,00H,00H,00H,00H
6101	1EB7	00 00 00		
6102	1EBA	00 00 00 F8	DB	00H,00H,00H,0F8H,00H,40H,20H
6103	1EBE	00 40 20		
6104	1EC1	10 00 00 00	DB	10H,00H,00H,00H,00H,00H,00H
6105	1EC5	00 00 00		
6106	1EC8	00 70 08 78	DB	00H,70H,08H,78H,88H,78H,00H
6107	1ECC	88 78 00		
6108	1ECF	80 80 B0 C8	DB	80H,80H,0B0H,0C8H,88H,0C8H,0B0H
6109	1ED3	88 C8 B0		
6110	1ED6	00 00 00 70	DB	00H,00H,00H,70H,88H,80H,88H
6111	1EDA	88 80 88		
6112	1EDD	70 00 08 08	DB	70H,00H,08H,08H,68H,98H,88H
6113	1EE1	68 98 88		
6114	1EE4	98 68 00 00	DB	98H,68H,00H,00H,00H,70H,88H
6115	1EE8	00 70 88		
6116	1EEB	F8 80 70 00	DB	0F8H,80H,70H,00H,10H,28H,20H
6117	1EEF	10 28 20		
6118	1EF2	F8 20 20 20	DB	0F8H,20H,20H,20H,00H,00H,00H
6119	1EF6	00 00 00		
6120	1EF9	68 98 98 68	DB	68H,98H,98H,68H,08H,70H,80H
6121	1EFD	08 70 80		
6122	1F00	80 F0 88 88	DB	80H,0F0H,88H,88H,88H,88H,00H
6123	1F04	88 88 00		
6124	1F07	20 00 60 20	DB	20H,00H,60H,20H,20H,20H,70H
6125	1F0B	20 20 70		
6126	1F0E	00 10 00 30	DB	00H,10H,00H,30H,10H,10H,10H
6127	1F12	10 10 10		
6128	1F15	90 60 40 40	DB	90H,60H,40H,40H,48H,50H,60H
6129	1F19	48 50 60		

6130	1F1C	50 48 00 60	DB	50H,48H,00H,60H,20H,20H,20H
6131	1F20	20 20 20		
6132	1F23	20 20 70 00	DB	20H,20H,70H,00H,00H,00H,0D0H
6133	1F27	00 00 D0		
6134	1F2A	A8 A8 A8 A8	DB	0A8H,0A8H,0A8H,0A8H,00H,00H,00H
6135	1F2E	00 00 00		
6136	1F31	B0 C8 88 88	DB	0B0H,0C8H,88H,88H,88H,00H,00H
6137	1F35	88 00 00		
6138	1F38	00 70 88 88	DB	00H,70H,88H,88H,88H,70H,00H
6139	1F3C	88 70 00		
6140	1F3F	00 00 B0 C8	DB	00H,00H,0B0H,0C8H,0C8H,0B0H,80H
6141	1F43	C8 B0 80		
6142	1F46	80 00 00 68	DB	80H,00H,00H,68H,98H,98H,68H
6143	1F4A	98 98 68		
6144	1F4D	08 08 00 00	DB	08H,08H,00H,00H,0B0H,0C8H,80H
6145	1F51	B0 C8 80		
6146	1F54	80 80 00 00	DB	80H,80H,00H,00H,00H,78H,80H
6147	1F58	00 78 80		
6148	1F5B	F0 08 F0 00	DB	0F0H,08H,0F0H,00H,40H,40H,0F0H
6149	1F5F	40 40 F0		
6150	1F62	40 40 48 30	DB	40H,40H,48H,30H,00H,00H,00H
6151	1F66	00 00 00		
6152	1F69	90 90 90 90	DB	90H,90H,90H,90H,68H,00H,00H
6153	1F6D	68 00 00		
6154	1F70	00 88 88 88	DB	00H,88H,88H,88H,50H,20H,00H
6155	1F74	50 20 00		
6156	1F77	00 00 88 A8	DB	00H,00H,88H,0A8H,0A8H,0A8H,50H
6157	1F7B	A8 A8 50		
6158	1F7E	00 00 00 88	DB	00H,00H,00H,88H,50H,20H,50H
6159	1F82	50 20 50		
6160	1F85	88 00 00 00	DB	88H,00H,00H,00H,88H,88H,98H

6161	1F89	88 88 98		
6162	1F8C	68 08 70 00	DB	68H,08H,70H,00H,00H,0F8H,10H
6163	1F90	00 F8 10		
6164	1F93	20 40 F8 00	DB	20H,40H,0F8H,00H,18H,20H,20H
6165	1F97	18 20 20		
6166	1F9A	40 20 20 18	DB	40H,20H,20H,18H,00H,20H,20H
6167	1F9E	00 20 20		
6168	1FA1	20 00 20 20	DB	20H,00H,20H,20H,20H,00H,0C0H
6169	1FA5	20 00 C0		
6170	1FA8	20 20 10 20	DB	20H,20H,10H,20H,20H,0C0H,00H
6171	1FAC	20 C0 00		
6172	1FAF	40 A8 10 00	DB	40H,0A8H,10H,00H,00H,00H,00H
6173	1FB3	00 00 00		
6174	1FB6	00 00 00 00	DB	00H,00H,00H,00H,00H,00H,00H
6175	1FBA	00 00 00		
6176	1FBD	00 00 10 38	DB	00H,00H,10H,38H,7CH,0FEH,0FEH
6177	1FC1	7C FE FE		
6178	1FC4	38 7C 00 6C	DB	38H,7CH,00H,6CH,0FEH,0FEH,0FEH
6179	1FC8	FE FE FE		
6180	1FCB	7C 38 10 00	DB	7CH,38H,10H,00H,38H,38H,0FEH
6181	1FCF	38 38 FE		
6182	1FD2	FE D6 10 7C	DB	0FEH,0D6H,10H,7CH,00H,10H,38H
6183	1FD6	00 10 38		
6184	1FD9	7C FE 7C 38	DB	7CH,0FEH,7CH,38H,10H,00H,00H
6185	1FDD	10 00 00		
6186	1FE0	78 84 84 84	DB	78H,84H,84H,84H,84H,78H,00H
6187	1FE4	84 78 00		
6188	1FE7	00 78 FC FC	DB	00H,78H,0FCH,0FCH,0FCH,0FCH,78H
6189	1FEB	FC FC 78		
6190	1FEE	00 40 FE 48	DB	00H,40H,0FEH,48H,70H,48H,82H
6191	1FF2	70 48 82		

6192	1FF5	7C 00 00 00	DB	7CH,00H,00H,00H,10H,7EH,3CH
6193	1FF9	10 7E 3C		
6194	1FFC	5A 34 00 00	DB	5AH,34H,00H,00H,00H,40H,42H
6195	2000	00 40 42		
6196	2003	42 52 20 00	DB	42H,52H,20H,00H,00H,00H,1CH
6197	2007	00 00 1C		
6198	200A	1C 22 02 0C	DB	1CH,22H,02H,0CH,00H,00H,00H
6199	200E	00 00 00		
6200	2011	18 7E 18 30	DB	18H,7EH,18H,30H,6EH,00H,00H
6201	2015	6E 00 00		
6202	2018	00 12 7E 3C	DB	00H,12H,7EH,3CH,52H,34H,00H
6203	201C	52 34 00		
6204	201F	00 00 28 7C	DB	00H,00H,28H,7CH,2AH,22H,24H
6205	2023	2A 22 24		
6206	2026	00 00 00 08	DB	00H,00H,00H,08H,5CH,6AH,0CH
6207	202A	5C 6A 0C		
6208	202D	30 00 00 00	DB	30H,00H,00H,00H,08H,0EH,38H
6209	2031	08 0E 38		
6210	2034	4C 3A 00 00	DB	4CH,3AH,00H,00H,00H,00H,3CH
6211	2038	00 00 3C		
6212	203B	02 02 1C 00	DB	02H,02H,1CH,00H,00H,00H,00H
6213	203F	00 00 00		
6214	2042	00 00 00 00	DB	00H,00H,00H,00H,00H,20H,0FEH
6215	2046	00 20 FE		
6216	2049	20 7C AA B2	DB	20H,7CH,0AAH,0B2H,64H,00H,00H
6217	204D	64 00 00		
6218	2050	80 82 82 82	DB	80H,82H,82H,82H,90H,60H,00H
6219	2054	90 60 00		
6220	2057	1C 00 7C 02	DB	1CH,00H,7CH,02H,02H,04H,18H
6221	205B	02 04 18		
6222	205E	00 38 00 FE	DB	00H,38H,00H,0FEH,08H,30H,50H

6223	2062	08 30 50		
6224	2065	9E 00 20 FA	DB	9EH,00H,20H,0FAH,22H,7CH,0A2H
6225	2069	22 7C A2		
6226	206C	A2 4C 00 40	DB	0A2H,4CH,00H,40H,44H,0F2H,4AH
6227	2070	44 F2 4A		
6228	2073	48 88 30 00	DB	48H,88H,30H,00H,10H,0FCH,08H
6229	2077	10 FC 08		
6230	207A	3E 04 80 7C	DB	3EH,04H,80H,7CH,00H,18H,18H
6231	207E	00 18 18		
6232	2081	30 60 60 30	DB	30H,60H,60H,30H,18H,00H,04H
6233	2085	18 00 04		
6234	2088	84 BE 84 84	DB	84H,0BEH,84H,84H,84H,48H,00H
6235	208C	84 48 00		
6236	208F	00 FC 02 00	DB	00H,0FCH,02H,00H,40H,80H,7EH
6237	2093	40 80 7E		
6238	2096	00 10 16 F8	DB	00H,10H,16H,0F8H,08H,7CH,80H
6239	209A	08 7C 80		
6240	209D	78 00 80 80	DB	78H,00H,80H,80H,80H,80H,84H
6241	20A1	80 80 84		
6242	20A4	88 70 00 08	DB	88H,70H,00H,08H,0FEH,08H,38H
6243	20A8	FE 08 38		
6244	20AB	48 38 08 00	DB	48H,38H,08H,00H,04H,44H,0FEH
6245	20AF	04 44 FE		
6246	20B2	44 44 40 3E	DB	44H,44H,40H,3EH,00H,64H,28H
6247	20B6	00 64 28		
6248	20B9	30 FE 20 40	DB	30H,0FEH,20H,40H,3CH,00H,00H
6249	20BD	3C 00 00		
6250	20C0	00 00 00 00	DB	00H,00H,00H,00H,00H,00H,00H
6251	20C4	00 00 00		
6252	20C7	00 00 00 00	DB	00H,00H,00H,00H,60H,90H,60H
6253	20CB	60 90 60		

6254	20CE	00 38 20 20	DB	00H,38H,20H,20H,20H,00H,00H
6255	20D2	20 00 00		
6256	20D5	00 00 00 00	DB	00H,00H,00H,00H,00H,20H,20H
6257	20D9	00 20 20		
6258	20DC	20 E0 00 00	DB	20H,0E0H,00H,00H,00H,00H,00H
6259	20E0	00 00 00		
6260	20E3	80 40 20 00	DB	80H,40H,20H,00H,00H,00H,00H
6261	20E7	00 00 00		
6262	20EA	30 30 00 00	DB	30H,30H,00H,00H,00H,0F8H,08H
6263	20EE	00 F8 08		
6264	20F1	F8 08 10 20	DB	0F8H,08H,10H,20H,40H,00H,00H
6265	20F5	40 00 00		
6266	20F8	00 F0 10 60	DB	00H,0F0H,10H,60H,40H,80H,00H
6267	20FC	40 80 00		
6268	20FF	00 10 20 60	DB	00H,10H,20H,60H,0A0H,20H,20H
6269	2103	A0 20 20		
6270	2106	00 00 20 F0	DB	00H,00H,20H,0F0H,90H,10H,20H
6271	210A	90 10 20		
6272	210D	40 00 00 00	DB	40H,00H,00H,00H,0F0H,20H,20H
6273	2111	F0 20 20		
6274	2114	20 F0 00 00	DB	20H,0F0H,00H,00H,20H,0F0H,60H
6275	2118	20 F0 60		
6276	211B	A0 A0 20 00	DB	0A0H,0A0H,20H,00H,00H,40H,0F8H
6277	211F	00 40 F8		
6278	2122	48 50 40 40	DB	48H,50H,40H,40H,00H,00H,00H
6279	2126	00 00 00		
6280	2129	70 10 10 10	DB	70H,10H,10H,10H,0F8H,00H,00H
6281	212D	F8 00 00		
6282	2130	00 F0 10 F0	DB	00H,0F0H,10H,0F0H,10H,0F0H,00H
6283	2134	10 F0 00		
6284	2137	00 00 A8 A8	DB	00H,00H,0A8H,0A8H,08H,10H,20H

6285	213B	08 10 20		
6286	213E	00 00 00 00	DB	00H,00H,00H,00H,0F8H,00H,00H
6287	2142	F8 00 00		
6288	2145	00 00 F8 08	DB	00H,00H,0F8H,08H,28H,30H,20H
6289	2149	28 30 20		
6290	214C	20 40 00 08	DB	20H,40H,00H,08H,10H,20H,60H
6291	2150	10 20 60		
6292	2153	A0 20 20 00	DB	0A0H,20H,20H,00H,20H,0F8H,88H
6293	2157	20 F8 88		
6294	215A	88 08 10 20	DB	88H,08H,10H,20H,00H,00H,0F8H
6295	215E	00 00 F8		
6296	2161	20 20 20 20	DB	20H,20H,20H,20H,0F8H,00H,10H
6297	2165	F8 00 10		
6298	2168	F8 10 30 50	DB	0F8H,10H,30H,50H,90H,10H,00H
6299	216C	90 10 00		
6300	216F	20 F8 28 28	DB	20H,0F8H,28H,28H,28H,48H,88H
6301	2173	28 48 88		
6302	2176	00 20 F8 20	DB	00H,20H,0F8H,20H,0F8H,20H,20H
6303	217A	F8 20 20		
6304	217D	20 00 78 48	DB	20H,00H,78H,48H,88H,08H,08H
6305	2181	88 08 08		
6306	2184	10 20 00 40	DB	10H,20H,00H,40H,78H,50H,90H
6307	2188	78 50 90		
6308	218B	10 10 20 00	DB	10H,10H,20H,00H,00H,0F8H,08H
6309	218F	00 F8 08		
6310	2192	08 08 08 F8	DB	08H,08H,08H,0F8H,00H,50H,0F8H
6311	2196	00 50 F8		
6312	2199	50 50 10 10	DB	50H,50H,10H,10H,20H,00H,00H
6313	219D	20 00 00		
6314	21A0	C0 08 C8 08	DB	0C0H,08H,0C8H,08H,10H,0E0H,00H
6315	21A4	10 E0 00		

6316	21A7	00 F8 08 10	DB	00H,0F8H,08H,10H,20H,50H,88H
6317	21AB	20 50 88		
6318	21AE	00 40 F8 48	DB	00H,40H,0F8H,48H,50H,40H,40H
6319	21B2	50 40 40		
6320	21B5	38 00 88 88	DB	38H,00H,88H,88H,48H,08H,10H
6321	21B9	48 08 10		
6322	21BC	20 40 00 78	DB	20H,40H,00H,78H,48H,78H,88H
6323	21C0	48 78 88		
6324	21C3	08 10 20 00	DB	08H,10H,20H,00H,10H,0E0H,20H
6325	21C7	10 E0 20		
6326	21CA	F8 20 20 40	DB	0F8H,20H,20H,40H,00H,0A8H,0A8H
6327	21CE	00 A8 A8		
6328	21D1	A8 08 08 10	DB	0A8H,08H,08H,10H,20H,00H,70H
6329	21D5	20 00 70		
6330	21D8	00 F8 20 20	DB	00H,0F8H,20H,20H,20H,40H,00H
6331	21DC	20 40 00		
6332	21DF	40 40 60 50	DB	40H,40H,60H,50H,48H,40H,40H
6333	21E3	48 40 40		
6334	21E6	00 20 F8 20	DB	00H,20H,0F8H,20H,20H,20H,20H
6335	21EA	20 20 20		
6336	21ED	40 00 00 70	DB	40H,00H,00H,70H,00H,00H,00H
6337	21F1	00 00 00		
6338	21F4	00 F8 00 00	DB	00H,0F8H,00H,00H,0F8H,08H,0D0H
6339	21F8	F8 08 D0		
6340	21FB	20 50 88 00	DB	20H,50H,88H,00H,20H,0F8H,08H
6341	21FF	20 F8 08		
6342	2202	30 E8 20 20	DB	30H,0E8H,20H,20H,00H,08H,08H
6343	2206	00 08 08		
6344	2209	08 10 20 40	DB	08H,10H,20H,40H,80H,00H,20H
6345	220D	80 00 20		
6346	2210	10 48 48 48	DB	10H,48H,48H,48H,48H,88H,00H

6347	2214	48 88 00		
6348	2217	80 80 F8 80	DB	80H,80H,0F8H,80H,80H,80H,78H
6349	221B	80 80 78		
6350	221E	00 F8 08 08	DB	00H,0F8H,08H,08H,08H,10H,20H
6351	2222	08 10 20		
6352	2225	40 00 00 40	DB	40H,00H,00H,40H,0A0H,10H,08H
6353	2229	A0 10 08		
6354	222C	08 00 00 20	DB	08H,00H,00H,20H,0F8H,20H,20H
6355	2230	F8 20 20		
6356	2233	A8 A8 20 00	DB	0A8H,0A8H,20H,00H,00H,0F8H,08H
6357	2237	00 F8 08		
6358	223A	08 50 20 10	DB	08H,50H,20H,10H,00H,0F0H,00H
6359	223E	00 F0 00		
6360	2241	60 00 00 F0	DB	60H,00H,00H,0F0H,08H,00H,10H
6361	2245	08 00 10		
6362	2248	20 40 80 90	DB	20H,40H,80H,90H,88H,0F8H,00H
6363	224C	88 F8 00		
6364	224F	08 08 08 50	DB	08H,08H,08H,50H,20H,50H,80H
6365	2253	20 50 80		
6366	2256	00 78 20 F8	DB	00H,78H,20H,0F8H,20H,20H,20H
6367	225A	20 20 20		
6368	225D	18 00 40 F8	DB	18H,00H,40H,0F8H,48H,48H,50H
6369	2261	48 48 50		
6370	2264	40 40 00 00	DB	40H,40H,00H,00H,70H,10H,10H
6371	2268	70 10 10		
6372	226B	10 10 F8 00	DB	10H,10H,0F8H,00H,00H,0F8H,08H
6373	226F	00 F8 08		
6374	2272	F8 08 08 F8	DB	0F8H,08H,08H,0F8H,00H,70H,00H
6375	2276	00 70 00		
6376	2279	F8 08 08 10	DB	0F8H,08H,08H,10H,20H,00H,48H
6377	227D	20 00 48		

6378	2280	48 48 48 48	DB	48H,48H,48H,48H,10H,20H,00H
6379	2284	10 20 00		
6380	2287	10 50 50 50	DB	10H,50H,50H,50H,50H,58H,90H
6381	228B	50 58 90		
6382	228E	00 40 40 40	DB	00H,40H,40H,40H,48H,48H,50H
6383	2292	48 48 50		
6384	2295	60 00 00 F8	DB	60H,00H,00H,0F8H,88H,88H,88H
6385	2299	88 88 88		
6386	229C	88 F8 00 F8	DB	88H,0F8H,00H,0F8H,88H,88H,08H
6387	22A0	88 88 08		
6388	22A3	08 10 20 00	DB	08H,10H,20H,00H,00H,0C0H,00H
6389	22A7	00 C0 00		
6390	22AA	08 08 10 E0	DB	08H,08H,10H,0E0H,00H,90H,48H
6391	22AE	00 90 48		
6392	22B1	00 00 00 00	DB	00H,00H,00H,00H,00H,00H,60H
6393	22B5	00 00 60		
6394	22B8	90 60 00 00	DB	90H,60H,00H,00H,00H,00H,00H
6395	22BC	00 00 00		
6396	22BF	40 FE 40 5E	DB	40H,0FEH,40H,5EH,80H,0A0H,9EH
6397	22C3	80 A0 9E		
6398	22C6	00 20 FE 40	DB	00H,20H,0FEH,40H,0F8H,04H,04H
6399	22CA	F8 04 04		
6400	22CD	78 00 00 00	DB	78H,00H,00H,00H,0FCH,02H,02H
6401	22D1	FC 02 02		
6402	22D4	04 38 00 00	DB	04H,38H,00H,00H,0FEH,0CH,30H
6403	22D8	FE 0C 30		
6404	22DB	40 40 38 00	DB	40H,40H,38H,00H,10H,12H,1CH
6405	22DF	10 12 1C		
6406	22E2	30 40 40 3E	DB	30H,40H,40H,3EH,00H,24H,0F2H
6407	22E6	00 24 F2		
6408	22E9	48 48 9C AA	DB	48H,48H,9CH,0AAH,10H,00H,80H

6409	22ED	10 00 80		
6410	22F0	9E 80 80 A0	DB	9EH,80H,80H,0A0H,0BEH,0C0H,00H
6411	22F4	BE C0 00		
6412	22F7	44 4C 7A AA	DB	44H,4CH,7AH,0AAH,0A6H,0AAH,6CH
6413	22FB	A6 AA 6C		
6414	22FE	00 40 EC 52	DB	00H,40H,0ECH,52H,62H,0CEH,4AH
6415	2302	62 CE 4A		
6416	2305	4C 00 00 38	DB	4CH,00H,00H,38H,54H,92H,0A2H
6417	2309	54 92 A2		
6418	230C	A2 4C 00 04	DB	0A2H,4CH,00H,04H,0BEH,84H,84H
6419	2310	BE 84 84		
6420	2313	9E A4 5C 00	DB	9EH,0A4H,5CH,00H,08H,4CH,0C6H
6421	2317	08 4C C6		
6422	231A	46 44 44 38	DB	46H,44H,44H,38H,00H,20H,18H
6423	231E	00 20 18		
6424	2321	20 16 8A CA	DB	20H,16H,8AH,0CAH,18H,00H,00H
6425	2325	18 00 00		
6426	2328	20 70 D8 8C	DB	20H,70H,0D8H,8CH,06H,02H,00H
6427	232C	06 02 00		
6428	232F	3E 84 BE 84	DB	3EH,84H,0BEH,84H,9CH,0A6H,18H
6429	2333	9C A6 18		
6430	2336	00 08 7E 08	DB	00H,08H,7EH,08H,7EH,38H,4CH
6431	233A	7E 38 4C		
6432	233D	3A 00 E0 24	DB	3AH,00H,0E0H,24H,24H,7EH,0A4H
6433	2341	24 7E A4		
6434	2344	A4 68 00 20	DB	0A4H,68H,00H,20H,0FCH,24H,62H
6435	2348	FC 24 62		
6436	234B	A0 62 3C 00	DB	0A0H,62H,3CH,00H,04H,44H,7CH
6437	234F	04 44 7C		
6438	2352	C6 AA 92 64	DB	0C6H,0AAH,92H,64H,00H,20H,20H
6439	2356	00 20 20		

(MSX ROM BASIC BIOS) Macro-80
 -MSXCHR- MSX character set

3.44 01-Jan-85 PAGE 63-18

234

6440	2359	78 20 78 22	DB	78H,20H,78H,22H,1CH,00H,00H
6441	235D	1C 00 00		
6442	2360	48 FC 4A 42	DB	48H,0FCH,4AH,42H,4CH,40H,00H
6443	2364	4C 40 00		
6444	2367	08 BC CA 8A	DB	08H,0BCH,0CAH,8AH,0BCH,08H,30H
6445	236B	BC 08 30		
6446	236E	00 08 08 0E	DB	00H,08H,08H,0EH,08H,78H,8CH
6447	2372	08 78 8C		
6448	2375	72 00 38 84	DB	72H,00H,38H,84H,80H,0FCH,0C2H
6449	2379	80 FC C2		
6450	237C	02 38 00 00	DB	02H,38H,00H,00H,42H,42H,42H
6451	2380	42 42 42		
6452	2383	62 04 18 00	DB	62H,04H,18H,00H,7CH,08H,30H
6453	2387	7C 08 30		
6454	238A	DC 62 92 7C	DB	0DCH,62H,92H,7CH,00H,20H,2CH
6455	238E	00 20 2C		
6456	2391	F4 24 64 E4	DB	0F4H,24H,64H,0E4H,26H,00H,7CH
6457	2395	26 00 7C		
6458	2398	18 20 5C 82	DB	18H,20H,5CH,82H,02H,7CH,00H
6459	239C	02 7C 00		
6460	239F	40 60 DC 62	DB	40H,60H,0DCH,62H,42H,0C2H,5CH
6461	23A3	42 C2 5C		
6462	23A6	00 10 30 20	DB	00H,10H,30H,20H,70H,48H,0CEH
6463	23AA	70 48 CE		
6464	23AD	84 00 00 00	DB	84H,00H,00H,00H,00H,00H,00H
6465	23B1	00 00 00		
6466	23B4	00 00 00 00	DB	00H,00H,00H,00H,00H,00H,00H
6467	23B8	00 00 00		
6468	23BB	00 00 00 00	DB	00H,00H,00H,00H
6469				

SUBTTL - MSXINL, Screen editor - Line input and function character

```

6470
6471      23BF          PINLIN:
6472                      ;
6473                      ; Main entry point
6474                      ;
6475      23BF      CD FDD8          CALL      H.PINL
6476      23C2      3A F6AA          LD        A,(AUTFLG)      ;During AUTO mode?
6477      23C5      A7              AND        A
6478      23C6      20 0D          JR         NZ,INLIN          ;Yes, then fake INLIN to prevent 0 from
6479                      ;deleting line number
6480      23C8      2E 00          LD        L,0
6481      23CA      18 14          JR         INLIN1
6482      23CC          QINLIN:
6483                      ;
6484                      ; Output question mark then get input
6485                      ;
6486      23CC      CD FDE0          CALL      H.QINL
6487      23CF      3E 3F          LD        A,'?'
6488      23D1      DF              RST        18H
6489      23D2      3E 20          LD        A,' '
6490      23D4      DF              RST        18H
6491      23D5          INLIN:
6492      23D5      CD FDE5          CALL      H.INLI
6493      23D8      2A F3DC          LD        HL,(CSRY)
6494      23DB      2D              DEC        L
6495      23DC      C4 0C29          CALL      NZ,TERMIN          ;Terminate previous line
6496      23DF      2C              INC        L
6497      23E0          INLIN1:
6498      23E0      22 FBFA          LD        (FSTPOS),HL      ;Mark first position
6499      23E3      AF              XOR        A
6500      23E4      32 FC9B          LD        (INTFLG),A
  
```

```
6501      23E7      INLIN2:
6502      23E7      CD 10CB      CALL    CHGET
6503      23EA      21 2437      LD      HL,SCITBL-2
6504      23ED      0E 0B      LD      C,0BH      ;SCI Max
6505      23EF      CD 0919      CALL    INDJMP      ;Do functions
6506      23F2      F5      PUSH    AF
6507      23F3      C4 23FF      CALL    NZ,INLOUT      ;Output a character
6508      23F6      F1      POP     AF
6509      23F7      30 EE      JR      NC,INLIN2      ;Not a terminator
6510      ;
6511      ; return to BASIC (break or CR)
6512      ;
6513      23F9      21 F55D      LD      HL,BUFMIN
6514      23FC      C8      RET      Z      ;Cnt-C, return with carry set
6515      23FD      3F      CCF      ;No, return carry clear
6516      23FE      RETURN:
6517      23FE      C9      RET
```

```

6518
6519      23FF          INLOUT:
6520          ;
6521      23FF      F5          PUSH      AF          ;Save character to output
6522      2400      FE 09          CP          9          ;TAB?
6523      2402      20 0F          JR          NZ,OUTNTB      ;Nope
6524      2404      F1          POP          AF          ;Discard stack
6525      2405          OUTTAB:
6526      2405      3E 20          LD          A,' '          ;Map to space
6527      2407      CD 23FF          CALL      INLOUT
6528      240A      3A F3DD          LD          A,(CSRX)
6529      240D      3D          DEC          A          ;Make it zero based.
6530      240E      E6 07          AND          7          ;Reached TAB stop?
6531      2410      20 F3          JR          NZ,OUTTAB      ;Not yet, continue...
6532      2412      C9          RET
6533      2413          OUTNTB:
6534          ;
6535      2413      F1          POP          AF          ;Restore character
6536      2414      21 FCA8          LD          HL,INSFLG      ;points insert mode flag
6537      2417      FE 01          CP          1          ;Graphic header byte?
6538      2419      28 0B          JR          Z,INLOT0      ;Yes, send as is
6539      241B      FE 20          CP          ' '          ;control char?
6540      241D      38 09          JR          C,INLOT1      ;branch if so. - Reset insert mode
6541      241F      F5          PUSH      AF          ;save char to output
6542      2420      7E          LD          A,(HL)          ;get insert mode flag
6543      2421      A7          AND          A          ;test
6544      2422      C4 24F2          CALL      NZ,INSERT      ;if insert mode, make room to insert
6545      2425      F1          POP          AF          ;restore char to output
6546      2426          INLOT0:
6547      2426      DF          RST          18H          ;output char
6548      2427      C9          RET

```

```
6549      2428          INLOT1:
6550          ;
6551      2428      36 00          LD      (HL),0          ;reset insert mode
6552      242A      DF          RST      18H          ;send this control char
6553      242B      3E          DB      3EH
6554      242C          SETINS:
6555      242C      3E          DB      3EH          ;Set insert mode and exit
6556      242D          SETOVW:
6557      242D      AF          XOR      A          ;Set overwrite mode
6558      242E      F5          PUSH     AF
6559      242F      CD 0A2E      CALL     CKERCS
6560      2432      F1          POP      AF
6561      2433      32 FCAA      LD      (CSTYLE),A
6562      2436      C3 09E1      JP      CKDPCS
```

```

6563
6564      2439      SCITBL:
6565      ;
6566      ; Table of function characters
6567      ;
6568      2439      08      DB      08H      ;Delete previous char
6569      243A      2561      DW      DELETE
6570      243C      12      DB      12H      ;Toggle insert flag
6571      243D      24E5      DW      TGLINS
6572      243F      1B      DB      1BH      ;Escape
6573      2440      23FE      DW      RETURN
6574      2442      02      DB      02H      ;Back word
6575      2443      260E      DW      LBCKWD
6576      2445      06      DB      06H      ;Next word
6577      2446      25F8      DW      LNXTWD
6578      2448      0E      DB      0EH
6579      2449      25D7      DW      LAPPND
6580      244B      05      DB      05H      ;Erase to end of line
6581      244C      25B9      DW      TRUNC
6582      244E      03      DB      03H      ;Abort
6583      244F      24C5      DW      LBREAK
6584      2451      0D      DB      0DH      ;Carriage return
6585      2452      245A      DW      LCRRET
6586      2454      15      DB      15H      ;Delete whole line
6587      2455      25AE      DW      LERASE
6588      2457      7F      DB      7FH      ;Delete character at cursor
6589      2458      2550      DW      LDELNX
6590      SUBTTL - MSXINL, Screen editor - Process special characters
    
```

- MSXINL, Screen editor - Process special characters

```

6591
6592      245A      LCRRET:
6593                  ;;;;;;;;;;;;;;;;;;
6594                  ; ;
6595                  ; Carriage return ;
6596                  ; ;
6597                  ;;;;;;;;;;;;;;;;;;
6598      245A      CD 266C      CALL      GTFRST      ;L=line number of first visual
6599      245D      3A F6AA      LD        A,(AUTFLG)    ;During AUTO mode?
6600      2460      A7          AND        A
6601      2461      28 02      JR         Z,NOTAUT      ;No
6602      2463      26 01      LD         H,1          ;Always get from top of line during AUTO mode
6603      2465      NOTAUT:
6604      2465      E5          PUSH       HL
6605                  ;
6606                  ; Put logical starting at L into BUF
6607                  ;
6608      2466      CD 0A2E      CALL      CKERCS
6609      2469      E1          POP        HL
6610      246A      11 F55E      LD         DE,BUF      ;Line buffer pointer
6611      246D      06 FE      LD         B,0FEH      ;Max count
6612      246F      2D          DEC        L
6613      2470      LCR1:
6614      2470      2C          INC        L
6615      2471      LCR2:
6616      2471      D5          PUSH       DE          ;Save buffer pointer
6617      2472      C5          PUSH       BC          ;Save buffer count
6618      2473      CD 0BD8      CALL      GETVRM      ;Get current character in Acc
6619      2476      C1          POP        BC          ;Restore buffer count
6620      2477      D1          POP        DE          ;Restore buffer pointer
6621      2478      A7          AND        A          ;Null?

```


6622	2479	28 14	JR	Z,LCRNUL	;Yes, ignore this
6623	247B	FE 20	CP	' '	;Special graphic character?
6624	247D	30 0B	JR	NC,LCRNRM	;No, proceed normally
6625	247F	05	DEC	B	;Decrement BUF size counter before storing
6626	2480	28 1D	JR	Z,LBLKSP	;At end of BUF, so ignore this
6627	2482	4F	LD	C,A	
6628	2483	3E 01	LD	A,1	;Store header byte for graphic symbol
6629	2485	12	LD	(DE),A	
6630	2486	13	INC	DE	
6631	2487	79	LD	A,C	
6632	2488	C6 40	ADD	A,'@'	
6633	248A			LCRNRM:	
6634	248A	12	LD	(DE),A	;Store byte in buffer
6635	248B	13	INC	DE	;Bump buffer pointer
6636	248C	05	DEC	B	;Decrement BUF size counter
6637	248D	28 10	JR	Z,LBLKSP	;At end of BUF
6638	248F			LCRNUL:	
6639	248F	24	INC	H	;Next column
6640	2490	3A F3B0	LD	A,(LINLEN)	;Max column reached?
6641	2493	BC	CP	H	;
6642	2494	30 DB	JR	NC,LCR2	;Not yet
6643	2496	D5	PUSH	DE	;Save buffer pointer
6644	2497	CD 0C1D	CALL	GETTRM	;Is this line terminated?
6645	249A	D1	POP	DE	;Restore buffer pointer
6646	249B	26 01	LD	H,1	;Assume not, start from top of next line
6647	249D	28 D1	JR	Z,LCR1	;No
6648	249F			LBLKSP:	
6649				;	
6650				; Suppress trailing blanks, [DE]=last+1	
6651				;	
6652	249F	1B	DEC	DE	;Back up buffer pointer

```

6653      24A0      1A              LD      A,(DE)          ;Get stored character
6654      24A1      FE 20          CP      ' '            ;Is it space?
6655      24A3      28 FA          JR      Z,LBLKSP        ;Yes, ignore this
6656      24A5      E5              PUSH     HL
6657      24A6      D5              PUSH     DE
6658      24A7      CD 09E1        CALL     CKDPCS
6659      24AA      D1              POP      DE
6660      24AB      E1              POP      HL
6661
6662          ;
6663          ; Terminate
6664      24AC      13              INC      DE              ;Point past last valid character
6665      24AD      AF              XOR      A              ;Load terminator
6666      24AE      12              LD      (DE),A          ;Put it in BUF
6667      24AF          FAKECR:
6668      24AF      3E 0D          LD      A,0DH            ;Load character to echo to console
6669      24B1      A7              AND      A              ;Reset Z-flag, (say not break)
6670      24B2          LNXTLN:
6671      24B2      F5              PUSH     AF              ;Save this flag
6672      24B3      CD 0C29        CALL     TERMIN
6673      24B6      CD 088E        CALL     POSIT            ;Save current cursor position
6674      24B9      3E 0A          LD      A,0AH
6675      24BB      DF              RST      18H            ;Move cursor to start of next line
6676      24BC      AF              XOR      A              ;Clear possible INSFLG
6677      24BD      32 FCA8        LD      (INSFLG),A
6678      24C0      F1              POP      AF              ;Restore flags
6679      24C1      37              SCF                      ;Set carry indicating end of input
6680      24C2      E1              POP      HL              ;Discard return address (XRA A;RET)
6681      24C3      C9              RET                    ;If break, Z flag is set
6682      24C4          LBREK0:
6683          ;

```

6684				; Control-C input
6685				;
6686	24C4	2C	INC L	;Bump line counter
6687	24C5		LBREAK:	
6688	24C5	CD 0C1D	CALL GETTRM	;Line terminated?
6689	24C8	28 FA	JR Z, LBREK0	;No, check next line
6690	24CA	CD 242D	CALL SETOVW	;Set to overwrite mode
6691	24CD	AF	XOR A	;Load 0 in Acc, and set Z flag
6692	24CE	32 F55E	LD (BUF), A	;Say no character in BUF
6693	24D1	26 01	LD H, 1	;Set to first column
6694	24D3	E5	PUSH HL	;Save cursor position
6695	24D4	CD 04BD	CALL GICINI	;Initialize sound chip and queue
6696	24D7	CD 0454	CALL CKSTTP	;Check if STOP trap is active or not
6697	24DA	E1	POP HL	
6698	24DB	38 D2	JR C, FAKECR	;Yes, fake CR
6699	24DD	3A FBB1	LD A, (BASROM)	;Executing BASIC program in ROM?
6700	24E0	A7	AND A	
6701	24E1	20 CC	JR NZ, FAKECR	;Yes, fake CR
6702	24E3	18 CD	JR LNXTLN	

```

6703
6704      24E5      TGLINS:
6705              ;
6706              ; Toggle insert mode flag
6707              ;
6708      24E5      21 FCA8      LD      HL,INSFLG      ;Get current insert flag
6709      24E8      7E          LD      A,(HL)
6710      24E9      EE FF      XOR      0FFH          ;Toggle insert status and affect Z flag
6711      24EB      77          LD      (HL),A
6712      24EC      CA 242D      JP      Z,SETOVW      ;Set to overwrite mode
6713      24EF      C3 242C      JP      SETINS       ;Set to insert mode
6714      24F2      INSERT:
6715              ;
6716              ; Insert a blank
6717              ;
6718      24F2      CD 0A2E      CALL    CKERCS        ;Erase cursor before operation
6719      24F5      2A F3DC      LD      HL,(CSRY)
6720      24F8      0E 20      LD      C,' '          ;Load raw code for space
6721      24FA      INS1:
6722      24FA      E5          PUSH    HL              ;Save current cursor position
6723      24FB      INS2:
6724      24FB      C5          PUSH    BC              ;Save previous character
6725      24FC      CD 0BD8      CALL    GETVRM        ;Get current character in C
6726      24FF      D1          POP     DE              ;Restore previous character in [E]
6727      2500      C5          PUSH    BC              ;Save current character
6728      2501      4B          LD      C,E            ;C=previous character
6729      2502      CD 0BE6      CALL    PUTVRM        ;Put it on screen
6730      2505      C1          POP     BC              ;Restore current character in C
6731      2506      3A F3B0      LD      A,(LINLEN)     ;Check if end of line
6732      2509      24          INC     H              ;Bump column counter
6733      250A      BC          CP      H              ;End of line?

```

```

6734      250B      7A                      LD      A,D          ;Get current attribute in Acc
6735      250C      30 ED                    JR      NC,INS2      ;If not, continue till end of line
6736
6737      ;
6738      ; Now we just finished a line, code of character wrapped to next
6739      ; line is held in [C].
6740      250E      E1                      POP      HL          ;Restore current cursor position
6741      250F      CD 0C1D                  CALL     GETTRM       ;Is this line terminated?
6742      2512      28 37                    JR      Z,INS6       ;Line not terminated on this visual
6743
6744      ;
6745      ; The current line is terminated. A check must be made to
6746      ; determine if a wrapped character is a space, or we're inserting
6747      ; at the end-of-line. If so, we have to open a next line to
6748      ; insert.
6749      2514      79                      LD      A,C          ;Move last character to A for comparison
6750      2515      FE 20                    CP      ' '
6751      2517      F5                      PUSH     AF          ;Save the condition
6752      2518      20 0A                    JR      NZ,INS3      ;No, open next line
6753      251A      3A F3B0                  LD      A,(LINLEN)    ;Are we trying to insert at the EOL?
6754      251D      BC                      CP      H
6755      251E      28 04                    JR      Z,INS3      ;Yes, open next line
6756      2520      F1                      POP      AF          ;Discard stack
6757      2521      C3 09E1                  JP      CKDPCS       ;Display cursor again
6758      2524
6759      INS3:
6760      2524      CD 0C2A                  CALL     UNTERM      ;Unterminate this line
6761      2527      2C                      INC      L          ;Go to next row
6762      2528      C5                      PUSH     BC          ;Save character code
6763      2529      E5                      PUSH     HL          ;Save position of character in operation
6764      252A      CD 0C32                  CALL     GETLEN      ;Bottom of screen?

```

```

6765      252D      BD                      CP      L                      ;
6766      252E      38 05                  JR      C,INS4                ;Yes
6767                                          ;
6768                                          ; Scroll down starting at line L
6769                                          ;
6770      2530      CD 0AB7                  CALL     INSLN0                ;Insert a blank line there
6771      2533      18 0F                  JR      INS5
6772      2535                      INS4:
6773                                          ;
6774                                          ; Scroll up
6775                                          ;
6776      2535      21 F3DC                  LD      HL,CSRY
6777      2538      35                      DEC      (HL)
6778      2539      20 01                  JR      NZ,INS45
6779      253B      34                      INC      (HL)
6780      253C                      INS45:
6781      253C      2E 01                  LD      L,1
6782      253E      CD 0A88                  CALL     DELLN0
6783      2541      E1                      POP      HL
6784      2542      2D                      DEC      L
6785      2543      E5                      PUSH     HL
6786      2544                      INS5:
6787      2544      E1                      POP      HL
6788      2545      C1                      POP      BC
6789      2546      F1                      POP      AF                    ;Restore flags
6790      2547      CA 09E1                  JP      Z,CKDPCS                ;If we were trying to insert at the
6791                                          ;end-of-line, nothing else to do
6792      254A      2D                      DEC      L                    ;Cancel next 'INR L'
6793      254B                      INS6:
6794                                          ;
6795                                          ; Not end of logical line, pass character to next line

```

6796						
6797	254B	2C		INC	L	;Bump row counter
6798	254C	26 01		LD	H,l	;Start from first column
6799	254E	18 AA		JR	INS1	;Pass character to next line

```

6800
6801      2550      LDELNX:
6802                ;
6803                ; Delete current character
6804                ;
6805      2550      3A F3B0      LD      A,(LINLEN)
6806      2553      BC          CP      H          ;At rightmost position?
6807      2554      20 05      JR      NZ,LDELX1    ;Nope
6808      2556      CD 0C1D      CALL     GETTRM      ;Is this a terminated line?
6809      2559      20 3A      JR      NZ,DELET5      ;Yes, place a space there.
6810      255B      LDELX1:
6811      255B      3E 1C      LD      A,1CH          ;Move cursor right
6812      255D      DF          RST      18H
6813      255E      2A F3DC      LD      HL,(CSRY)      ;Fall into 'delete prev. character'
6814      2561      DELETE:
6815                ;
6816                ; Delete previous character
6817                ;
6818      2561      E5          PUSH     HL
6819      2562      CD 0A2E      CALL     CKERCS
6820      2565      E1          POP      HL
6821      2566      25          DEC      H          ;Are we at top of line?
6822      2567      C2 257A      JP      NZ,DELET2      ;No
6823      256A      24          INC      H          ;Yes
6824      256B      E5          PUSH     HL          ;Save current cursor position
6825      256C      2D          DEC      L          ;Look a line above
6826      256D      28 0A      JR      Z,DELET1        ;At top of screen
6827      256F      3A F3B0      LD      A,(LINLEN)
6828      2572      67          LD      H,A
6829      2573      CD 0C1D      CALL     GETTRM      ;Is previous line terminated?
6830      2576      20 01      JR      NZ,DELET1        ;Yes

```


6831	2578	E3	EX	(SP),HL	;No, substitue by current HL
6832	2579		DELET1:		
6833	2579	E1	POP	HL	;Get saved cursor position
6834	257A		DELET2:		
6835	257A	22 F3DC	LD	(CSRY),HL	;Set new cursor position
6836	257D		DELET3:		
6837	257D	3A F3B0	LD	A,(LINLEN)	
6838	2580	BC	CP	H	
6839	2581	28 12	JR	Z,DELET5	;Just over strike with blank
6840	2583	24	INC	H	
6841	2584		DELET4:		
6842	2584	CD 0BD8	CALL	GETVRM	;Get current character and attribute
6843	2587	25	DEC	H	
6844	2588	CD 0BE6	CALL	PUTVRM	;Output it to left of current position
6845	258B	24	INC	H	
6846	258C	24	INC	H	
6847	258D	3A F3B0	LD	A,(LINLEN)	
6848	2590	3C	INC	A	
6849	2591	BC	CP	H	
6850	2592	20 F0	JR	NZ,DELET4	;Do next till end of visual
6851	2594	25	DEC	H	
6852	2595		DELET5:		
6853	2595	0E 20	LD	C,' '	;Load raw code for space
6854	2597	CD 0BE6	CALL	PUTVRM	
6855	259A	CD 0C1D	CALL	GETTRM	
6856	259D	C2 09E1	JP	NZ,CKDPCS	;End of line, all done
6857	25A0	E5	PUSH	HL	
6858	25A1	2C	INC	L	
6859	25A2	26 01	LD	H,1	
6860	25A4	CD 0BD8	CALL	GETVRM	;Get first character next visual
6861	25A7	E3	EX	(SP),HL	

MSX ROM BASIC BIOS) Macro-80 3.44 01-Jan-85
· MSXINL, Screen editor - Process special characters

PAGE 69-2

250

6862	25A8	CD 0BE6	CALL	PUTVRM	;Put at last position last line
6863	25AB	E1	POP	HL	
6864	25AC	18 CF	JR	DELET3	

```

6865
6866      25AE          LERASE:
6867                      ;
6868                      ; Erase logical line
6869                      ;
6870      25AE      CD 0A2E          CALL      CKERCS
6871      25B1      CD 266C          CALL      GTFIRST
6872      25B4      22 F3DC          LD          (CSRY),HL
6873      25B7      18 05          JR          TRUNC1
6874      25B9          TRUNC:
6875                      ;
6876      6876          ; Truncate logical line
6877                      ;
6878      25B9      E5          PUSH      HL
6879      25BA      CD 0A2E          CALL      CKERCS
6880      25BD      E1          POP       HL
6881      25BE          TRUNC1:
6882      25BE      CD 0C1D          CALL      GETTRM
6883      25C1      F5          PUSH      AF
6884      25C2      CD 0AEE          CALL      EOL
6885      25C5      F1          POP       AF
6886      25C6      20 05          JR          NZ,DPCSOW
6887      25C8      26 01          LD          H,1
6888      25CA      2C          INC        L
6889      25CB      18 F1          JR          TRUNC1
6890      25CD          DPCSOW:
6891                      ;
6892      25CD      CD 09E1          CALL      CKDPCS
6893      25D0      AF          XOR        A
6894      25D1      32 FCA8          LD          (INSFLG),A
6895      25D4      C3 242D          JP          SETOVW

```

;Set L=first visual this logical line

;Is this line terminated?

;Save the condition

;Erase to end-of-line

;Restore condition

;Yes

;Go to next line

;Bump row counter

;And continue

```

6896      25D7      LAPPND:
6897      ;
6898      ; Append to current line
6899      ;
6900      25D7      CD 0A2E      CALL      CKERCS      ;Erase cursor
6901      25DA      2A F3DC      LD        HL,(CSRY)    ;Get current cursor position
6902      25DD      2D          DEC        L
6903      25DE      LAP1:
6904      25DE      2C          INC        L
6905      25DF      CD 0C1D      CALL      GETTRM      ;Line terminated?
6906      25E2      28 FA      JR         Z,LAP1      ;No, look at next line
6907      25E4      3A F3B0      LD        A,(LINLEN)
6908      25E7      67          LD        H,A
6909      25E8      24          INC        H
6910      25E9      LAP2:
6911      25E9      25          DEC        H      ;Reached start of line?
6912      25EA      28 07      JR         Z,LAP3      ;Yes
6913      25EC      CD 0BD8      CALL      GETVRM      ;Get a character at the cursor
6914      25EF      FE 20      CP         ' '      ;Space?
6915      25F1      28 F6      JR         Z,LAP2      ;Yes, skip this
6916      25F3      LAP3:
6917      25F3      CD 0A5B      CALL      ADVCUR      ;Advance cursor to point to end of line
6918      25F6      18 D5      JR         DPCSOW      ;Re-display cursor
6919      25F8      LNXTWD:
6920      ;
6921      ; Move to next word
6922      ;
6923      25F8      CD 0A2E      CALL      CKERCS
6924      25FB      CD 2634      CALL      PRVCHK
6925      25FE      LNWL:
6926      25FE      CD 2624      CALL      NXTCHK      ;Still in word?

```

6927	2601	28 CA	JR	Z,DPCSOW	;Reached screen bottom, abort
6928	2603	38 F9	JR	C,LNW1	;Yes
6929	2605		LNW2:		
6930	2605	CD 2624	CALL	NXTCHK	;Reached word?
6931	2608	28 C3	JR	Z,DPCSOW	;Reached screen bottom, abort
6932	260A	30 F9	JR	NC,LNW2	;Not yet
6933	260C	18 BF	JR	DPCSOW	
6934	260E		LBCKWD:		
6935			;		
6936			;	Move to previous word	
6937			;		
6938	260E	CD 0A2E	CALL	CKERCS	
6939	2611		LBW1:		
6940	2611	CD 2634	CALL	PRVCHK	;Still in separator?
6941	2614	28 B7	JR	Z,DPCSOW	;Reached screen top, abort
6942	2616	30 F9	JR	NC,LBW1	;Yes
6943	2618		LBW2:		
6944	2618	CD 2634	CALL	PRVCHK	;Reached separator?
6945	261B	28 B0	JR	Z,DPCSOW	;Reached screen top, abort
6946	261D	38 F9	JR	C,LBW2	;Not yet
6947	261F	CD 0A5B	CALL	ADVCUR	
6948	2622	18 A9	JR	DPCSOW	
6949	2624		NXTCHK:		
6950			;		
6951			;	Move right and check	
6952			;		
6953	2624	2A F3DC	LD	HL,(CSRY)	;Get current cursor position
6954	2627	CD 0A5B	CALL	ADVCUR	;Advance cursor
6955	262A	CD 0C32	CALL	GETLEN	;Get an actual height of screen
6956	262D	5F	LD	E,A	;[D],[E] hold the dead end position
6957	262E	3A F3B0	LD	A,(LINLEN)	

```

6958      2631      57                      LD      D,A
6959      2632      18 09                    JR      PRVCK1
6960      2634                      PRVCHK:
6961                      ;
6962                      ; Move left and check
6963                      ;
6964      2634      2A F3DC                    LD      HL,(CSRY)      ;Get current cursor position
6965      2637      CD 0A4C                    CALL    BS           ;Regress cursor
6966      263A      11 0101                    LD      DE,0101H      ;[D],[E] hold the dead end position
6967      263D                      PRVCK1:
6968                      ;
6969                      ; Check current character
6970                      ; Carry set if the character is regarded as separator
6971                      ;
6972      263D      2A F3DC                    LD      HL,(CSRY)      ;Get updated cursor position
6973      2640      E7                          RST      20H          ;Reached dead end?
6974      2641      C8                          RET      Z           ;Yes, return with Z flag
6975      2642      11 2668                    LD      DE,RESZRO      ;Jump to RESZRO when done
6976      2645      D5                          PUSH    DE
6977      2646      CD 0BD8                    CALL    GETVRM          ;Get ASCII code of character at [H],[L]
6978      2649      FE 30                      CP      '0'          ;Set carry if "0".."9"
6979      264B      3F                          CCF
6980      264C      D0                          RET      NC
6981      264D      FE 3A                      CP      ':'
6982      264F      D8                          RET      C
6983      2650      FE 41                      CP      'A'          ;Set carry if "A".."Z"
6984      2652      3F                          CCF
6985      2653      D0                          RET      NC
6986      2654      FE 5B                      CP      'Z'+1
6987      2656      D8                          RET      C
6988      2657      FE 61                      CP      'a'          ;Set carry if "a".."z"

```

```

6989      2659      3F              CCF
6990      265A      D0              RET      NC
6991      265B      FE 7B           CP        'z'+1
6992      265D      D8              RET      C
6993      265E      FE 86           CP        86H          ;Check for Hiragana (86H)
6994      2660      3F              CCF
6995      2661      D0              RET      NC
6996      2662      FE A0           CP        0A0H
6997      2664      D8              RET      C
6998      2665      FE A6           CP        0A6H
6999      2667      3F              CCF
7000      2668                      RESZRO:
7001      2668      3E 00           LD        A,0          ;Reset Z flag without affecting C flag
7002      266A      3C              INC      A
7003      266B      C9              RET
7004
7005                      ;
7006                      ; Set H,L to first visual line in logical line
7007                      ;
7007      266C                      GTFRST:
7008      266C      2D              DEC      L              ;Look a line just above
7009      266D      28 05           JR        Z,GTfst1      ;If we're at top of screen, all done
7010      266F      CD 0C1D        CALL     GETTRM        ;Get terminator
7011      2672      28 F8           JR        Z,GTFRST      ;More to get above in this logical
7012      2674                      GTfst1:
7013      2674      2C              INC      L              ;L=line number of first visual
7014      2675      3A FBCA         LD        A,(FSTPOS)      ;Get first line
7015      2678      BD              CP        L              ;Same?
7016      2679      26 01         LD        H,1              ;Assume not
7017      267B      C0              RET      NZ              ;Good assumption
7018      267C      2A FBCA         LD        HL,(FSTPOS)      ;Get first line and column
7019      267F      C9              RET

```

(MSX ROM BASIC BIOS) Macro-80 3.44 01-Jan-85
- MSXINL, Screen editor - Process special characters

PAGE 70-5

256

7020

END

MSX BIOS CROSS REFERENCE

ACTION	1#	2664	3518#							
ADVCUR	1#	1930	2166#	6917	6947	6954				
ALPJMP	1#	2892#	3041							
ASCPCT1	1#	5236								
ASCPCT2	1#	5238								
ATRBAS	1#	1163	1255	1296	1387	1437				
ATRBYT	1#	4407	4725	4756	5113	5168	5215	5256	5404	5431
AUTFLG	1#	6476	6599							
BAKCLR	1#	1574	1584	1660	1684					
BASROM	1#	923	2571	6699						
BDRCLR	1#	1690								
BEEP	1#	170	1914	3485#						
BEGIN	30#									
BIT0	1#	5517	5523#							
BIT1	1#	5516	5519	5520	5533#					
BIT1OT	1#	5494	5535	5542	5544#					
BITOUT	1#	5511	5530	5552#						
BRDATR	1#	5259	5282	5377	5427					
BREAKX	1#	167	1008#	1733	5500	5521	5666	5672	5712	5755 5778
BS	1#	1916	1932	2144#	2297	6965				
BUF	1#	6610	6692							
BUFEND	1#	2087	2391	2478	2497					
BUFMIN	1#	6513								
CALATR	1#	136	1430#							
CALBAS	1#	252	363#	2768	5803	5871				
CALESL	1#	412	419#							
CALLF	1#	90	366#							
CALPAT	1#	135	1413#							
CALSLT	1#	57	365	404#	437					
CAPST	1#	3055	3193	3275						
CGCAP1	1#	3201	3203#							
CGPBAS	1#	1140	1159	1471	2083					

[illegible]

[illegible]

DATAWL	1#	5514#	5518							
DCOMPR	1#	59	4146#							
DELET1	1#	6826	6830	6832#						
DELET2	1#	6822	6834#							
DELET3	1#	6836#	6864							
DELET4	1#	6841#	6850							
DELET5	1#	6809	6839	6852#						
DELETE	1#	6569	6814#							
DELLN0	1#	1868	2222#	6782						
DELLN1	1#	2240#	2249							
DIOERR	1#	5870								
DISSC1	1#	1175	1182#							
DISSCR	1#	108	1131	1150	1176#	1249	1290			
DLN	1953	2215#								
DOWN	1#	1864	1936	1959	2173#					
DOWN1	1#	2180	2185#							
DOWNC	1#	216	4876#							
DPCSOW	1#	6886	6890#	6918	6927	6931	6933	6941	6945	6948
DSFKCL	1#	2395#	2398							
DSPCS1	1#	2091	2093#	2098						
DSPCSR	1#	2058	2066#							
DSPFK1	1#	2386	2389#							
DSPFK2	1#	2413#	2437							
DSPFK4	1#	2405#	2408							
DSPFK5	1#	2417#	2422	2430						
DSPFK6	1#	2425	2428#							
DSPFK8	1#	2423	2426#							
DSPFKE	1#	2403	2411	2438#						
DSPFNK	1#	175	2366#	2821						
DWNC10	1#	4869	4873	4885#						
EASYTB	1#	2937#	3163							
ELN	1#	1949	2226	2250	2263	2289	2301#	2356		

[illegible]

[illegible]

GPRT60	1#	4447	4452#			
GPRT70	1#	4462	4465#			
GPRT80	1#	4468	4470#			
GRPACX	1#	4410	4443	4453	4459	
GRPACY	1#	4408	4461	4471		
GRPATR	1#	1254				
GRPCGP	1#	1576	4612	4862	4901	
GRPCOL	1#	1573				
GRPCR	1#	4401	4446	4451	4456#	
GRPDIF	1#	4688	5111	5115	5153	5202
GRPHED	1#	1787				
GRPNAM	1#	1256	1283			
GRPPAT	1#	1252				
GRPPRT	1#	138	4389#			
GRPTAB	1#	3365	3377#			
SPAD1	1#	1422	1425#			
SPSIZ	1#	137	1420	1440#		
TASPC	1#	228	5232#			
TFRST	1#	6598	6871	7007#	7011	
TFST1	1#	7009	7012#			
TPAD	1#	186	3867#			
TPAD0	1#	3893	3899#			
TPAT1	1#	1526#	1540			
TPDL	1#	187	3807#			
TPDP1	1#	3888	3891#			
TROW8	1#	2689	3698	3726#	3804	
TSTCK	1#	184	3683#			
TTRIG	1#	185	3783#			
I.CHGE	1#	3410				
I.CHPU	1#	1819				
I.DSPC	1#	2070				
I.DSPF	1#	2370				

H.ERAC	1#	2125				
H.ERAF	1#	2350				
H.FORM	1#	4203				
H.INIP	1#	1470				
H.INLI	1#	6492				
H.ISFL	1#	4139				
H.KEYC	1#	2993				
H.KEYI	1#	2621				
H.KYEA	1#	3160				
H.LPTO	1#	1730				
H.LPTS	1#	1759				
H.NMI	1#	4061				
H.OUTD	1#	5798				
H.PHYD	1#	4199				
H.PINL	1#	6475				
H.QINL	1#	6486				
H.TIMI	1#	2625				
H.TOTE	1#	1703				
HEADER	1#	5485				
HIGH	1#	5551				
HRSSCL	1#	4521	4526#			
HRZMOV	1#	4811	4841#			
HRZMV1	1#	4791	4807	4822	4837	4845#
ILN	1#	1951	2251#			
INDJMP	1#	1889#	1897	1994	6505	
INESC	1#	1846	1987#			
INESC1	1#	1989	1995#			
INESC2	1#	2005	2009#			
INGI	1#	1051	3476#	3723	3915	3987 4012
INIFNK	1#	99	4065#			
INIGR1	1#	1260#	1263	1264		
INIGRP	1#	129	1245#	1722		

INIML1	1#	1300#	1313			
INIML2	1#	1302#	1310			
INIML3	1#	1305#	1308			
INIMLT	1#	130	1286#	1723		
INIPAT	1#	1143	1166	1466#		
INIPT1	1#	1477#	1490			
INIT	1#	919				
INIT32	1#	128	1146#	1720		
INITIO	1#	98	1038#			
INITQ	1#	1088	4328#			
INITXT	1#	127	1127#	1719		
INLIN	1#	164	6478	6491#		
INLIN1	1#	6481	6497#			
INLIN2	1#	6501#	6509			
INLOT0	1#	6538	6546#			
INLOT1	1#	6540	6549#			
INLOUT	1#	6507	6519#	6527		
INS1	1#	6721#	6799			
INS2	1#	6723#	6735			
INS3	1#	6752	6755	6758#		
INS4	6766	6772#				
INS45	1#	6778	6780#			
INS5	1#	6771	6786#			
INS6	1#	6742	6793#			
INSERT	1#	6544	6714#			
INSFLG	1#	6536	6677	6708	6894	
INSLN0	1#	2258#	6770			
INSLN1	1#	2279#	2288			
INTCNT	1#	2638	2647			
INTFLG	1#	927	944	3217	3419	6500
INTRET	1#	2624	2672	2720	2723	2731#
INTVAL	1#	2645				

ISCNTC	1#	168	922#	3437		
ISFLIO	1#	247	4135#	5799		
JFLVRM	1#	1579#	1593			
JIFFY	1#	2651	2653			
JMPBC	1#	1887	1902	1905#		
JMPWRT	1#	5191	5204	5218#		
JPPPAL	1#	4398	4403	4412	4454#	4472
JPUTCH	1#	3025	3032#	3060		
KAIUEO	1#	3260	3265#			
KANAMD	1#	1053	3256			
KANANO	1#	3267	3290#			
KANASF	1#	3269	3307#			
KANAST	1#	3002	3173			
KANJNO	1#	3261	3324#			
KANJSF	1#	3263	3341#			
KEEPH	1#	5560#	5562			
KEEPL	1#	5555#	5557			
KEYANY	1#	2795	2828#			
KEYBUF	1#	3401				
KEYCHK	1#	2719	2746#			
KEYCK1	1#	2753#	2760			
KEYCK2	1#	2773#	2779			
KEYCK3	1#	2778	2781#			
KEYCK4	1#	2730	2780	2785#		
KEYCK5	1#	2789#	2798			
KEYCOD	1#	2848	2983#			
KEYINT	1#	97	2603#			
KEYNOM	1#	2896	3053#			
KEYSFT	1#	2895	3050#			
KEYTRG	1#	3786	3802#			
KILBUF	1#	251	962	1002#		
KSTKTB	1#	3703	3765#			

KY1CNT	2900	2914#			
KY1NOM	2902	2903#			
KY1SFC	2899	2924#			
KY1SFT	2901	2908#			
KYALP	1#	2865	3034#		
KYANY1	1#	2844#	2852		
KYClTB	1#	2898#	3063		
KYCLAC	1#	2999	3005#		
KYCLAS	1#	2995	3007#	3017	
KYCLS	2881	3150#			
KYCOD1	1#	2863	3061#		
KYEASY	1#	2867	2875	2879	2883 3156#
KYFNC1	1#	3086	3090#		
KYFNC2	1#	3098#	3120		
KYFNC3	1#	3107#	3113		
KYFUNC	1#	2873	3080#		
KYGRAP	1#	3001	3360#		
KYJTAB	1#	2859#	2992		
KYKAN1	1#	3262	3264	3268	3270#
KYKANA	1#	3004	3252#		
KYKLOK	1#	2871	3169#		
KYLOCK	1#	2869	3189#		
KYNUM	1#	2861	3018#		
KYSTCK	1#	3686	3696#		
KYSTOP	1#	2877	3206#		
KYSTP1	1#	3214	3216#		
LAP1	1#	6903#	6906		
LAP2	1#	6910#	6915		
LAP3	1#	6912	6916#		
LAPPND	1#	6579	6896#		
LBCKWD	1#	6575	6934#		
LBLKSP	1#	6626	6637	6648#	6655

[illegible]

LOWLIM	1#	5638	5663			
LPT.DW	1#	623#	1740			
LPT.SB	1#	624#	1055	1742	1744	
LPT.ST	1#	625#				
LPTABO	1#	1734	1748#			
LPTCH0	1#	5833	5836	5843#		
LPTCH1	1#	5811	5846	5864#		
LPTCHR	1#	5852	5858	5861	5867#	5874
LPTCOD	1#	5800	5805#			
LPTOUT	1#	159	1726#	5868		
LPTPOS	1#	1751	5824	5837	5841	
LPTSTT	1#	160	1735	1757#		
MAPSPC	1#	5850	5854	5872#		
MAPXYC	1#	219	4413	4540#		
MDNC	1#	4884	5006	5010	5016#	
MHCMOV	1#	4966	4987#			
MHZMV1	1#	4949	4960	4972	4982	4991#
MLFTC	1#	4834	4977#			
MLFTC1	1#	4975	4983#			
MLTATR	1#	1295				
MLTCGP	1#	1591	4650	5001	5025	
MLTNAM	1#	1297	1333			
MLTPAT	1#	1293				
MMPXY1	1#	4631	4633#			
MMPXYC	1#	4590	4624#			
MNSTCX	1#	5071	5221#	5230		
MORACT	1#	3577#	3596	3619	3631	
MORSPL	1#	5821#	5826			
MOTRON	1#	4045	4048#			
MOTRWT	1#	5479#	5483			
MREADC	1#	4684	4706#			
MRGTC	1#	4804	4955#			

IRGTC1	1#	4953	4961#			
ISCANL	1#	5371	5411#	5420		
ISCANR	1#	5269	5336#	5347		
ISCNR1	1#	5341	5350#			
ISCNR2	1#	5356#	5361			
ISETC	1#	4738	4745#			
ISETC1	1#	4759	4764#			
ITDNC	1#	4860	4996#			
ITLFT	1#	4819	4967#			
ITRGT	1#	4788	4941#			
ITSBRD	1#	5340	5360	5417	5421#	
ITUPC	1#	4899	5022#			
IUPC	1#	4926	5030	5033	5038#	
IUSCLL	1#	1073#	1076			
IUSICF	1#	1070	2657	3642	3657	3670
IUSINT	1#	2660#	2669			
IUSITB	1#	1098	1114#			
IVTMOV	1#	5021	5043#			
IVTMV1	1#	5045	5047#			
JAMBAS	1#	1138	1157	1553	2557	
JEWKEY	1#	2752	2771	2788		
JMI	1#	124	4057#			
JMSFTB	1#	2885#	3027			
JOKEY	1#	3187#				
JONEG1	1#	3935	3938#			
JONEG2	1#	3943	3946#			
JOSTOP	1#	2763	2766	2769#		
JOTABL	1#	5819	5830#			
JOTAUT	1#	6601	6603#			
JOTRAN	1#	5730	5738#			
JSETCX	1#	227	5055#	5328	5395	
JSTC10	1#	5080#	5084			

ISTC20	1#	5076	5094#			
ISTC30	1#	5108#	5117			
ISTC40	1#	5107	5118#			
ISTC50	1#	5125	5139#			
ISTCSP	1#	5082	5130#			
ITBKS2	1#	5817#				
ITBOTM	1#	2379	2381#			
ITHIRA	1#	5856	5859#			
ITINTT	1#	2642	2646#			
ITMSXP	1#	5844				
IXCHK	1#	6926	6930	6949#		
LDKEY	1#	1029	1031	2725	2726	2787
LDSCR	1#	1134	1153	1702		
NBRD1	1#	4797	4827	4914#	4954	4976
NBRDR	1#	4874	4912#			
NGSBF	1#	3145	3147			
UTDLP	1#	248	5814#	5823		
UTDO	1#	55	5788#			
UTGI	1#	3986	3995	3999	4010	4017#
UTNTB	1#	6523	6533#			
UTTAB	1#	6525#	6531			
ADX	1#	3895	3950			
ADX1	1#	3926	3928	3932	3953#	
ADY	1#	3897	3952			
ATBAS	1#	1161	1253	1294	1380	1427
ATWR1	1#	5181	5205#			
ATWRK	1#	1523	4414			
ATWRT	1#	4740	5089	5142#		
BDHRT	1#	1832#	2853	3430	3679	
DL1	1#	3833#	3835			
DL2	1#	3856#	3861			
DL3	1#	3859	3863#			

PUTVRM	1#	1854	2105	2131	2300	2512#	6729	6844	6854	6862
QINLIN	1#	166	6482#							
QSTART	1#	4333	4368	4377#						
QUEBAK	1#	4324								
QUEUEN	1#	3560	3674							
QUEUES	1#	4384								
RAMLOW	1#	296	871	872						
RAWPRT	1#	5809								
RDBIT	1#	5695	5715#							
RDBITL	1#	5727#	5735	5744						
RDESLT	1#	291	299#							
RDPSG	1#	148	3481#	3712	3846	3909				
RDSLTL	1#	49	289#	304	1482	1531				
RDVDP	1#	241	4112#							
RDVRM	1#	111	1606#	4685	4690	4708	4750	5151	5155	
READC	1#	225	4674#	5285	5310	5382	5425			
READC0	1#	4696#	4712							
READC1	1#	4694	4701#	4711						
READYR	1#	2767								
REDCOD	1#	3927	3930	3959#						
REDLOP	1#	3983#	3996							
REDPAD	1#	3924	3925	3964	3969	3975#				
REPCNT	1#	1033	2721	2784						
REQSTP	1#	965	988	991						
REQTRP	1#	967	2634	2644	2701	2704	2707	2710	2713	3127#
RESZRO	1#	6975	7000#							
RETRET	1#	5495	5531#							
RETURN	1#	6516#	6573							
RGOSAV	1#	1205	1214	1232	1273	1322				
RG1SAV	1#	1173	1180	1219	1237	1278	1327	1376	1400	1444
RGHTC1	1#	4796	4808#							
RGTEXT	1#	5122	5126#							

RIGHT	1#	1855	1961	2135#	2170					
RIGHTC	1#	212	4798#	5227	5390	5418				
RSET10	1#	2038	2043#							
RSLREG	1#	239	4116#							
RSTFL1	1#	3645#	3647							
RSTM0D	1#	1969	1976#							
RUBOUT	1#	1853	2293#							
RUNFLG	1#	3902	4023	5279						
SAMEBG	1#	5170	5192#							
SAMEFG	1#	5176	5185	5198#						
SAVSTK	1#	979								
SCALXY	1#	218	4411	4475#						
SCANL	1#	231	5364#							
SCANL1	1#	5379#	5387							
SCANL2	1#	5384	5388#							
SCANL3	1#	5381	5391#							
SCANL4	1#	5334	5397#							
SCANR	1#	230	5261#							
SCANR1	1#	5284#	5293							
SCANR2	1#	5287	5296#							
SCANR3	1#	5306#	5314							
SCANR4	1#	5309	5312	5315#						
SCITBL	1#	6503	6564#							
SCLXOK	1#	4513	4518#							
SCLYOK	1#	4497	4502#							
SCNCNT	1#	2670								
SCRMOD	1#	1133	1152	1251	1292	1551	1648	2455	2540	4537
SELEXP	1#	301	342	420	486	544#				
SELPRM	1#	290	331	411	477	500#				
SETATR	1#	224	4714#							
SETC	1#	226	4425	4727#	5226	5435				
SETCHK	1#	2352	2372	2446#						

[illegible]

STOREC	1#	222	4435	4665#	5323	5331
STRTMS	1#	149	3651#			
STSTYL	1#	2027	2040#			
SULOP	1#	5649#	5652			
SYN05	1#	5577#	5589	5591	5602	
SYN10	1#	5583#	5606			
SYN11	1#	5597	5600#			
SYN20	1#	5608#				
SYN30	1#	5615#	5620			
SYNCHR	1#	46				
SYNCW1	1#	5486	5489#			
SYNLP1	1#	5493#	5499			
F32ATR	1#	1162				
F32CGP	1#	1158				
F32COL	1#	1662				
F32NAM	1#	1156	1242			
F32PAT	1#	1160				
FAB	1#	1918	2190#	2199		
FAPIN	1#	194	5659#			
FAPIOF	1#	195	5462#			
FAPION	1#	193	5568#			
FAPOFF	1#	198	5450#			
FAPOON	1#	196	5469#			
FAPOUT	1#	197	5501#			
FDOWNC	1#	217	4436	4850#		
FERMIN	1#	2314	2579#	6495	6672	
FGLINS	1#	6571	6704#			
FIMOUT	1#	5762	5770#			
FLEFT	1#	4812#	5380	5415		
FOTEXT	1#	176	973	1696#		
FRGFLG	1#	2694				
FRIG1	1#	3794	3796#			

[illegible]

MSX BIOS SYMBOL TABLE

042C ABORT	10F9 CKCNTC	0A88 DELLNO
F847 ARG	FBD9 CLIKFL	FD99 DEVICE
F7B5 ARYTA2	F3DB CLIKSW	F662 DIMFLG
F6C4 ARYTAB	F935 CLINEF	0577 DISSCR
F40B ASCPCT1	F3B2 CLMLST	F665 DONUM
F40D ASCPCT2	F92A CLOC	F6B5 DOT
F931 ASPECT	F38C CLPRIM	0A61 DOWN
F928 ATRBAS	06A8 CLRSPR	172A DOWNC
F3F2 ATRBYT	0848 CLS	FCBD DRWANG
F6AA AUTFLG	F92C CMASK	FCBB DRWFLG
F6AD AUTINC	F936 CNPNTS	FCBC DRWSCL
F6AB AUTLIN	F3DE CNSDFG	F699 DSCPTR
F3EA BAKCLR	08B0 CNVCH1	F698 DSCTMP
FBB1 BASROM	08B2 CNVCH2	0B2B DSPFNK
F3EB BDRCLR	08B4 CNVCH3	1B63 DUTDLP
1113 BEEP	089D CNVCHR	0570 ENASCR
FC48 BOTTOM	FBCC CODSAV	025E ENASLT
FCB2 BRDATR	F66A CONLO	267F ENDBIOS
046F BREAKX	F668 CONSAV	F660 ENDBUF
3FDC BRKTXR	F666 CONXTX	F6A1 ENDFOR
F55E BUF	F669 CONTYP	F40F ENDPRG
FC18 BUFEND	F939 CPCNT	FFCA ENDWRK
F55D BUFMIN	F93B CPCNT8	026B ENESLT
06F9 CALATR	F938 CPLOTf	FBB0 ENSTOP
01FF CALBAS	F93D CRCSUM	0989 ENTESC
022E CALESL	F3B1 CRTCNT	0B15 ERAFNK
0205 CALLF	F3FC CS120	F414 ERRFLG
06E4 CALPAT	F942 CSAVEA	F6B3 ERRLIN
0217 CALSLT	F944 CSAVEM	F6B7 ERRTXT
FCAB CAPST	F941 CSCCLXY	FCC1 EXPTBL
FCB1 CASPRV	FCA9 CSRSW	F7F8 FACLO
F933 CENCNT	F3DD CSRX	F7C5 FBUFFR
F924 CGPBAS	F3DC CSRY	1639 FETCHC
F91F CGPNT	F93F CSTCNT	F871 FILNM2
1BBF CGTABL	FCAA CSTYLE	F860 FILTAB
0F3D CHGCAP	F41C CURLIN	0815 FILVRM
07F7 CHGCLR	F945 CXOFF	13A9 FKTABL
10CB CHGET	F947 CYOFF	FCAE FLBMEM
084F CHGMOD	F7F6 DAC	F6A6 FLGINP
0F7A CHGSND	F6A3 DATLIN	FBCE FNKFLG
0D62 CHKBUF	F6C8 DATPTR	0B26 FNKSB
02D7 CHKRAM	146A DCOMPR	F87F FNKSTR
0B9F CHKSCR	F7F4 DECCNT	FBCD FNKSWI
08BC CHPUT	268C DECSUB	F3E9 FORCLR
08DF CHPUT1	F7F2 DECTM2	148E FORMAT
2686 CHRGR	F7F0 DECTMP	F3F5 FRCNEW
0D6A CHSNS	F6CA DEFTBL	F69B FRETOP

FBCA FSTPOS	FEEE H.DSKC	FE67 H.MERG
F7BA FUNACT	FE12 H.DSKF	FE3A H.MKD
F3FA GETPNT	FE17 H.DSKI	FE30 H.MKI
1474 GETVC2	FDEF H.DSKO	FE35 H.MKS
1470 GETVCP	FDA9 H.DSPC	FDF9 H.NAME
2689 GETYPR	FDB3 H.DSPF	FF3E H.NEWS
04BD GICINI	FEA3 H.EOF	FDD6 H.NMI
FCB7 GRPACX	FDAE H.ERAC	FEB7 H.NODE
FCB9 GRPACY	FDB8 H.ERAF	FE58 H.NOFO
F3CD GRPATR	FF02 H.ERRF	FF34 H.NOTR
F3CB GRPCGP	FFB1 H.ERRO	FE62 H.NTFL
F3C9 GRPCOL	FEFD H.ERRP	FF2F H.NTFN
FCA6 GRPHED	FF70 H.EVAL	FF6B H.NTPL
F3C7 GRPNAM	FE2B H.FIEL	FE5D H.NULO
F3CF GRPPAT	FE7B H.FILE	FF75 H.OKNO
1510 GRPPRT	FE85 H.FILO	FDEA H.ONGO
0704 GSPSIZ	FF1B H.FINE	FEE4 H.OUTD
18C7 GTASPC	FF7A H.FING	FEB2 H.PARD
12AC GTPAD	FF16 H.FINI	FFA7 H.PHYD
1273 GTPDL	FF5C H.FINP	FDD8 H.PINL
11EE GTSTCK	FEA8 H.FOPS	FFC5 H.PLAY
1253 GTTRIG	FFAC H.FORM	FEB3 H.POSD
FCB3 GXPOS	FF9D H.FRET	FEF8 H.PRGE
FCB5 GYPOS	FF66 H.FRME	FF52 H.PRTF
F40A HEADER	FF93 H.FRQI	FFA2 H.PTRG
FE1C H.ATTR	FEC6 H.GEND	FDE0 H.QINL
FEAD H.BAKU	FE4E H.GETP	FF07 H.READ
FE76 H.BINL	FF43 H.GONE	FF4D H.RETU
FE71 H.BINS	FE8A H.INDS	FE26 H.RSET
FF8E H.BUFL	FDC7 H.INIP	FE8F H.RSLF
FDC2 H.CHGE	FDE5 H.INLI	FECB H.RUNC
FDA4 H.CHPU	FE03 H.IPL	FE94 H.SAVD
FF48 H.CHRG	FEDF H.ISFL	FE6C H.SAVE
FED0 H.CLEA	FF7F H.ISMI	FF98 H.SCNE
FE0D H.CMD	FF2A H.ISRE	FFC0 H.SCRE
FF57 H.COMP	FDCC H.KEYC	FE53 H.SETF
FE08 H.COPY	FD9A H.KEYI	FDF4 H.SETS
FEE9 H.CRDO	FDFF H.KILL	FF39 H.SNGF
FF20 H.CRUN	FDD1 H.KYEA	FEDA H.STKE
FF25 H.CRUS	FF89 H.LIST	FD9F H.TIMI
FE49 H.CVD	FE99 H.LOC	FDBD H.TOTE
FE3F H.CVI	FE9E H.LOF	FF61 H.TRMN
FE44 H.CVS	FED5 H.LOPD	FF84 H.WIDT
FEF3 H.DDGR	FFB6 H.LPTO	F408 HIGH
FEC1 H.DEVN	FFBB H.LPTS	FC4A HIMEM
FE80 H.DGET	FE21 H.LSET	F83E HOLD
FF11 H.DIRD	FF0C H.MAIN	F836 HOLD2

F806 HOLD8	15DF MAPXYC	18CF PNTINI
098F INESC	F92F MAXDEL	088E POSIT
139D INIFNK	F85F MAXFIL	F7B4 PRMFLG
05D2 INIGRP	F3EC MAXUPD	F6E6 PRMLEN
061F INIMLT	F958 MCLFLG	F74E PRMLN2
2680 INIT	FB3B MCLLEN	F74C PRMPRV
0538 INIT32	FB3C MCLPTR	F6E4 PRMSTK
049D INITIO	F956 MCLTAB	FD89 PROCNM
050E INITXT	F672 MEMSIZ	FB35 PRSCNT
23D5 INLIN	F92D MINDEL	F416 PRTFLG
FCA8 INSFLG	F3EF MINUPD	F864 PTRFIL
FCA2 INTCNT	F3D7 MLTATR	F6A9 PTRFLG
FC9B INTFLG	F3D5 MLTCGP	0F55 PUTCHR
FCA0 INTVAL	F3D3 MLTCOL	F3F8 PUTPNT
03FB ISCNTC	F3D1 MLTNAM	1492 PUTQ
145F ISFLIO	F3D9 MLTPAT	23CC QINLIN
FC9E JIFFY	F951 MOVCNT	F971 QUEBAK
FCAD KANAMD	FB3F MUSICF	F959 QUETAB
FCAC KANAST	F922 NAMBAS	FB3E QUEUEN
F41F KBUF	FBE5 NEWKEY	F3F3 QUEUES
0D89 KEYANY	4601 NEWSTT	F418 RAWPRT
FBF0 KEYBUF	F87C NLONLY	F380 RDPRIM
0E3B KEYCOD	1398 NMI	110E RDPSPG
0C3C KEYINT	F7B7 NOFUNS	01B6 RDSLTL
0468 KILBUF	1809 NSETCX	7E1A RDSLTLW
0F10 KYEASY	F417 NTMSXP	1449 RDVDP
107D KYGRAP	F862 NULBUF	07D7 RDVRM
0F36 KYLOCK	FBDA OLDKEY	1647 READC
0F46 KYSTOP	F6BE OLDLIN	F3F7 REPCNT
070F LDIRMV	FCB0 OLDSCR	FC6A REQSTP
0744 LDIRVM	F6C0 OLDTXT	F3DF RG0SAV
16EE LEFTC	F6BB ONEFLG	F3E0 RG1SAV
F954 LFPROG	F6B9 ONELIN	F3E1 RG2SAV
14EB LFTQ	FBD8 ONGSBF	F3E2 RG3SAV
F3AF LINL32	F664 OPRTYP	F3E3 RG4SAV
F3AE LINL40	1B45 OUTDO	F3E4 RG5SAV
F3B0 LINLEN	FC9D PADX	F3E5 RG6SAV
FBF2 LINTTB	FC9C PADY	F3E6 RG7SAV
F94B LOHADR	F6E8 PARM1	16C5 RIGHTC
F94D LOHCNT	F750 PARM2	F857 RNDX
F94A LOHDIR	F926 PATBAS	FAF5 RS2IQ
F949 LOHMSK	FC40 PATWRK	144C RSLREG
F406 LOW	08DB PBDHRT	F955 RTPROG
FCA4 LOWLIM	F953 PDIREC	FC9A RTYCNT
085D LPTOUT	148A PHYDIO	FCBE RUNBNF
F415 LPTPOS	23BF PINLIN	F866 RUNFLG
0884 LPTSTT	FB40 PLYCNT	F87D SAVEND

FCBF	SAVENT	1A63	TAPION
FB36	SAVSP	19DD	TAPOFF
F6B1	SAVSTK	19F1	TAPOON
F6AF	SAVTXT	1A19	TAPOUT
FB39	SAVVOL	170A	TDOWNC
1599	SCALXY	F6A7	TEMP
197A	SCANL	F6BC	TEMP2
18E4	SCANR	F69D	TEMP3
2439	SCITBL	F69F	TEMP8
F3F6	SCNCNT	F7B8	TEMP9
FCAF	SCRMOD	F678	TEMPPT
02A3	SELEXP	F67A	TEMPST
027E	SELPRM	083B	TOTEXT
1676	SETATR	F7C4	TRCFLG
167E	SETC	F3E8	TRGFLG
0602	SETGRP	FC4C	TRPTBL
0659	SETMLT	F661	TTYPOS
07EC	SETRD	173C	TUPC
05B4	SETT32	F3B9	TXATAT
0C2B	SETTRM	F3B7	TXTCGP
0594	SETTXT	F3B5	TXTCOL
07DF	SETWRT	F3B3	TXTNAM
FBEB	SFTKEY	F3BB	TXTPAT
F94F	SKPCNT	F676	TXTTAB
120C	SLSTCK	175D	UPC
FCC9	SLTATR	F39A	USRTAB
FCC5	SLTTBL	F663	VALTYP
FD09	SLTWRK	F6C2	VARTAB
1452	SNSMAT	FB41	VCBA
F3E7	STATFL	FB66	VCBB
F674	STKTOP	FB8B	VCBC
1384	STMOTR	F419	VLZADR
0A69	STOCSR	F41B	VLZDAT
1640	STOREC	F975	VOICAQ
F6C6	STREND	F9F5	VOICBQ
6678	STROUT	FA75	VOICCQ
11C4	STRTMS	FB38	VOICEN
F6A5	SUBFLG	FCA5	WINWID
F7BC	SWPTMP	F385	WRPRIM
2683	SYNCHR	01D1	WRSLT
F3C3	T32ATR	1102	WRTPSG
F3C1	T32CGP	057F	WRTVDP
F3BF	T32COL	07CD	WRTVRM
F3BD	T32NAM	144F	WSLREG
F3C5	T32PAT		
1ABC	TAPIN		
19E9	TAPIOF		

APPENDIX A

TITLE MSX USA version
SUBTTL Symbol definition
page 36

0000'

.Z80
ASEG

.COMMENT %

Differences between Japanese version and overseas versions

- 1) The default screen mode has been changed from 1 to 0.
- 2) The default border color has been changed from 7 to 4. The default function key string for F6 key has been also changed to reflect this change.
- 3) The character generator pattern has been changed.
- 4) The Hiragana to Katakana conversion in LPT output routine has been removed.
- 5) The ASCII load problem has been fixed.
- 6) The null device name problem has been fixed.
- 7) The format symbol in PRINT USING statement has been changed.
- 8) The reserved key matrix area now has a table for ten-key support

	United States	United Kingdom
Vsync:	60Hz	50Hz
Screen size:	39 (default)	37 (default)
Layout:	QWERTY	QWERTY
Deadkey:	4 deadkeys supported.	4 deadkeys supported.
Currency:	Dollar sign	British Pound sign
Special note:	None	None
Status:	Finalized	Finalized

%


```

009C      POND      EQU      9CH      ;character code for pound sign
0006      DEADNUM   EQU      6

PRINTV    MACRO      VALUE
          IF1
          .PRINTX * VALUE bytes left *
          ENDIF
          ENDM

;
;      MSX ROM references
;

006C      INITXT      EQU      6CH      ;initialize screen to 40 character text
0132      CHGCAP      EQU      132H
0F10      KYEASY      EQU      0F10H
0F55      PUTCHR      EQU      0F55H      ;put a character in queue
0F64      GENCLK      EQU      0F64H      ;generate click sound
10C2      UPDATE      EQU      10C2H      ;update put/get pointer
FBEB      SFTKEY      EQU      0FBEBH      ;current shift key status
FCAB      CAP_LOCK    EQU      0FCABH      ;capital lock status      (CAPST)
FCAC      DEAD_STATUS EQU      0FCACH      ;current dead-key status (KANAST)
                                          ; if 0 no preceding dead-key
                                          ; if 1      dead-key
                                          ; if 2      shifted-dead-key
                                          ; if 3      code-dead-key
                                          ; if 4      code-shift-dead-key

          IF1
          .PRINTX / USA version /
          ENDIF
          ;IF1

```

ORG 2BH

; The format of ID byte is as follows

```

; 2BH: b7 b6 b5 b4 b3 b2 b1 b0
;      | | | | | | | |
;      | | | | +---+---+---+ kind of character generator
;      | | | | 0:Japanese 1:International
;      | +---+---+---+ format of date
;      | 0:Y-M-D 1:M-D-Y 2:D-M-Y
;      +---+---+---+ frequency of interrupt
;      1:50Hz 0:60Hz

```

002B 11

DEFB 00010001B ;UK - DEFB 1010001B

```

; 2CH: b7 b6 b5 b4 b3 b2 b1 b0
;      | | | | | | | |
;      | | | | +---+---+---+ kind of keyboard
;      | | | | 0:Japan 1:International
;      | | | | 2:French 3:UK 4:DIN
;      +---+---+---+ version of BASIC (print using etc.)

```

002C 11

DEFB 11H ;UK - DEFB 13H

; 34H .. 37H

; Range of first byte for 2-byte characters such as KANJI

0D9B 1021

;

ORG 0D9BH

DEFW KEYCOD

SUBTTL Key code table (0DA5H..0EC4H)

ORG 0DA5H

```

;*****
;
;      Table of codes for various shift conditions.  Note that 0FFH
;      (255) is reserved for dead-key.
;
;*****

```

```

;;;;;;;;;;;;
;
;      Keyboard encode table for 'QWERTY' layout
;
;*****

```

```

;
;      Normal codes
;

```

NORMAL:

0DA5		DEFB	'01234567'	
0DA5	30 31 32 33			
0DA9	34 35 36 37			
0DAD	38 39 2D 3D	DEFB	'89-=[];'	'89-= \ [];
0DB1	5C 5B 5D 3B			
0DB5	27 60 2C 2E	DEFB	''',./',0FFH,'ab'	;' ' ` ,./',0ffh,'ab'
0DB9	2F FF 61 62			
0DBD	63 64 65 66	DEFB	'cdefghij'	
0DC1	67 68 69 6A			
0DC5	6B 6C 6D 6E	DEFB	'klmnopqr'	
0DC9	6F 70 71 72			
0DCD	73 74 75 76	DEFB	'stuvwxyz'	
0DD1	77 78 79 7A			

```

;
; Codes when shift key pressed
;
SHIFT:
0DD5
0DD5 29 21 40 23      DEFB  ')!@#$$%&'          ')!@#$$ ^ &'
0DD9 24 25 5E 26
0DD  2A 28 5F 2B      DEFB  '*(_+|_:'          '*(_+|{ }:'
0DE1 7C 7B 7D 3A
0DE5 22 7E 3C 3E      DEFB  '"°$¶?' ,0FFH, 'AB'  '"~< >?' ,0 ffH, 'AB'
0DE9 3F FF 41 42
0DED 43 44 45 46      DEFB  'CDEFGHIJ'
0DF1 47 48 49 4A
0DF5 4B 4C 4D 4E      DEFB  'KLMNOPQR'
0DF9 4F 50 51 52
0DFD 53 54 55 56      DEFB  'STUVWXYZ'
0E01 57 58 59 5A

```

```

;
; Codes when graph key pressed
;
;
GRAPH:
0E05
0E05 09 AC AB BA      DEFB  009H,0ACH,0ABH,0BAH,0EFH,0BDH,0F4H,0FBH ;0
0E09 EF BD F4 FB
0E0D EC 07 17 F1      DEFB  0ECH,007H,017H,0F1H,01EH,001H,00DH,006H ;1
0E11 1E 01 0D 06
0E15 05 BB F3 F2      DEFB  005H,0BBH,0F3H,0F2H,01DH,0FFH,0C4H,011H ;2
0E19 1D FF C4 11
0E1D BC C7 CD 14      DEFB  0BCH,0C7H,0CDH,014H,015H,013H,0DCH,0C6H ;3
0E21 15 13 DC C6
0E25 DD C8 0B 1B      DEFB  0DDH,0C8H,00BH,01BH,0C2H,0DBH,0CCH,018H ;4
0E29 C2 DB CC 18

```

0E2D	D2 12 C0 1A	DEFB	0D2H,012H,0C0H,01AH,0CFH,01CH,019H,00FH ;5
0E31	CF 1C 19 0F		

;
;
Codes when graph and shift keys pressed

			0	1	2	3	4	5	6	7
0E35		GRAPH_SHIFT:								
0E35	0A 00 FD FC	DEFB	00AH,000H,0FDH,0FCH,000H,000H,0F5H,000H ;0							
0E39	00 00 F5 00									
0E3D	00 08 1F F0	DEFB	000H,008H,01FH,0F0H,016H,002H,00EH,004H ;1							
0E41	16 02 0E 04									
0E45	03 F7 AE AF	DEFB	003H,0F7H,0AEH,0AFH,0F6H,0FFH,0FEH,000H ;2							
0E49	F6 FF FE 00									
0E4D	FA C1 CE D4	DEFB	0FAH,0C1H,0CEH,0D4H,010H,0D6H,0DFH,0CAH ;3							
0E51	10 D6 DF CA									
0E55	DE C9 0C D3	DEFB	0DEH,0C9H,00CH,0D3H,0C3H,0D7H,0CBH,0A9H ;4							
0E59	C3 D7 CB A9									
0E5D	D1 00 C5 D5	DEFB	0D1H,000H,0C5H,0D5H,0D0H,0F9H,0AAH,0F8H ;5							
0E61	D0 F9 AA F8									

;
;
Codes when code key pressed

			0	1	2	3	4	5	6	7
0E65		CODE:								
0E65	EB 9F D9 BF	DEFB	0EBH,09FH,0D9H,0BFH,09BH,098H,0E0H,0E1H ;0							
0E69	9B 98 E0 E1									
0E6D	E7 87 EE E9	DEFB	0E7H,087H,0EEH,0E9H,000H,0EDH,0DAH,0B7H ;1							
0E71	00 ED DA B7									
0E75	B9 E5 86 A6	DEFB	0B9H,0E5H,086H,0A6H,0A7H,0FFH,084H,097H ;2							
0E79	A7 FF 84 97									
0E7D	8D 8B 8C 94	DEFB	08DH,08BH,08CH,094H,081H,0B1H,0A1H,091H ;3							

0E81 81 B1 A1 91
0E85 B3 B5 E6 A4
0E89 A2 A3 83 93
0E8D 89 96 82 95
0E91 88 8A A0 85

DEFB 0B3H,0B5H,0E6H,0A4H,0A2H,0A3H,083H,093H ;4

DEFB 089H,096H,082H,095H,088H,08AH,0A0H,085H ;5

;

;

Codes when code and shift keys pressed

;

;

0 1 2 3 4 5 6 7

0E95

CODE__SHIFT:

0E95 D8 AD 9E BE

DEFB 0D8H,0ADH,09EH,0BEH,09CH,09DH,000H,000H ;0

0E99 9C 9D 00 00

DEFB 0E2H,080H,000H,000H,000H,0E8H,0EAH,0B6H ;1

0E9D E2 80 00 00

0EA1 00 E8 EA B6

DEFB 0B8H,0E4H,08FH,000H,0A8H,0FFH,08EH,000H ;2

0EA5 B8 E4 8F 00

0EA9 A8 FF 8E 00

DEFB 000H,000H,000H,099H,09AH,0B0H,000H,092H ;3

0EAD 00 00 00 99

0EB1 9A B0 00 92

DEFB 0B2H,0B4H,000H,0A5H,000H,0E3H,000H,000H ;4

0EB5 B2 B4 00 A5

0EB9 00 E3 00 00

DEFB 000H,000H,090H,000H,000H,000H,000H,000H ;5

0EBD 00 00 90 00

0EC1 00 00 00 00

IF1

IF (\$-NORMAL) NE (48*6)

.PRINTX * Table length not correct *

ENDIF

ENDIF

MSX USA version Macro-80
Key code table (0DA5H..0EC4H)

3.44

01-Jan-85

PAGE

5

296

0F17 1003

;

ORG 0F17H

DEFW EASYTB-48

SUBTTL Dead key handler (0F1FH..0F34H)


```
                                ORG      0F1FH
                                ;
                                DEAD_KEY:
0F1F      3A FBEB                LD      A,(SFTKEY)
0F22      5F                    LD      E,A
0F23      F6 FE                OR      1111110B      ;extract shift key status only
0F25      CB 63                BIT     4,E          ;code key pressed?
0F27      20 02                JR      NZ,DEAD_KEY1  ;no
0F29      E6 FD                AND     11111101B
0F2B                                DEAD_KEY1:
0F2B      2F                    CPL
0F2C      3C                    INC     A            ;make 1..4
0F2D      32 FCAC              LD      (DEAD_STATUS),A
0F30      18 32                JR      GENCLK       ;generate click sound

                                PRINTV  %(0F35H-$)
```

```
                                ORG      0F5AH
                                ;
0F5A      105B                DEFW     NEW_UPDATE

                                SUBTTL   Keyboard encoder (0F83H..10C1H)
```

```

                                ORG      0F83H
                                ;
                                ;       Beginning of the table-driven key encoder
                                ;
                                ;       [C] = raw code for pressed key
                                ;
0F83      INTKEY:
0F83      3A FBEB      LD      A,(SFTKEY)      ;get current shift key status
0F86      5F          LD      E,A            ;save shift key status in [E]
0F87      1F          RRA                    ;move control key status to carry
0F88      1F          RRA
0F89      F5          PUSH     AF            ;remember control key status (carry
                                ;reset if pressed)
0F8A      7B          LD      A,E            ;restore shift key status
0F8B      2F          CPL
0F8C      30 10       JR      NC,IS_CONTROL ;control key being pressed
                                ;
                                ;       Get an offset into SFTTAB using current shift key status and
                                ;       code lock status.
                                ;
0F8E      1F          RRA
0F8F      1F          RRA
0F90      07          RLCA
0F91      E6 03       AND      11B
0F93      CB 4F       BIT      1,A          ;is graph shift on?
0F95      20 09       JR      NZ,INTKEY_1    ;yes, ignore code key
0F97      CB 63       BIT      4,E          ;is code pressed?
0F99      20 05       JR      NZ,INTKEY_1    ;no
0F9B      F6 04       OR      100B          ;set code bit
0F9D      11          DEFB     11H          ;'LD DE,XXXX' instruction
                                ;

```

```

;      Control key is being pressed. Ignore the graph and code lock
;      status.
;
0F9E      IS_CONTROL:
0F9E      E6 01      AND      1      ;valid is only shift key status
;
;      Now we have in [Acc] '00000CGS'
;
;      |||
;      ||+-- shift \
;      |+--- graph >-- 1 when pressed
;      +---- code /
;
0FA0      INTKEY_1:
0FA0      5F      LD      E,A
0FA1      87      ADD     A,A
0FA2      83      ADD     A,E
0FA3      87      ADD     A,A
0FA4      87      ADD     A,A
0FA5      87      ADD     A,A
0FA6      87      ADD     A,A
0FA7      5F      LD      E,A
0FA8      16 00   LD      D,0
0FAA      21 0DA5 LD      HL,NORMAL
0FAD      19      ADD     HL,DE      ;[HL] = the address of table
0FAE      42      LD      B,D      ;[BC] = offset into code table
0FAF      09      ADD     HL,BC
0FB0      F1      POP     AF      ;restore control key status into carry
0FB1      7E      LD      A,(HL)   ;get real code
0FB2      3C      INC     A      ;dead-key?
0FB3      CA 0F1F JP      Z,DEAD_KEY ;yes
0FB6      3D      DEC     A      ;should code be generated?
0FB7      C8      RET     Z      ;no code should be generated

```

```

0FB8    38 16                JR      C,WASNT_CONTROL ;control was not pressed
0FBA    E6 DF                AND     11011111B      ;force to upper case
0FBC    D6 40                SUB     40H           ;make control character
0FBE    FE 20                CP      ' '
0FC0    D0                   RET     NC           ;cannot make control code
0FC1                                JPUTCHR:
0FC1    18 92                JR      PUTCHR        ;skip 2 byte code check and case
                                                ;translation

;
0FC3                                KYFUNC:
0FC3    3A FBEB              LD      A,(SFTKEY)
0FC6    0F                  RRCA
0FC7    38 04                JR      C,KYFNC1
0FC9    79                  LD      A,C
0FCA    C6 05                ADD     A,5
0FCC    4F                  LD      C,A
0FCD                                KYFNC1:
0FCD    C3 0EC5              JP      0EC5H

;
0FD0                                WASNT_CONTROL:
0FD0    FE 20                CP      ' '           ;2 byte code?
0FD2    30 0B                JR      NC,NOT_2BYTE ;no
0FD4    F5                  PUSH    AF
0FD5    3E 01                LD      A,1           ;put graphic header byte
0FD7    CD 0F55              CALL    PUTCHR
0FDA    F1                  POP     AF
0FDB    C6 40                ADD     A,40H        ;add offset
0FDD    18 E2                JR      JPUTCHR        ;skip case translation

;
;      Check if case translation is necessary
;
0FDF                                NOT_2BYTE:

```

0FDF	21 FCAB	LD	HL,CAP_LOCK	;capital lock active?
0FE2	34	INC	(HL)	
0FE3	35	DEC	(HL)	
0FE4	28 0A	JR	Z,CHECK_DEAD	;no
0FE6	FE 61	CP	'a'	;normal alphabet?
0FE8	38 27	JR	C,CHECK_SPECIAL	;no, check if special alphabet
0FEA	FE 7B	CP	'z'+1	
0FEC	30 23	JR	NC,CHECK_SPECIAL	
0FEE	E6 DF	AND	11011111B	;force to upper case
0FF0		CHECK_DEAD:		
0FF0	ED 5B FCAC	LD	DE,(DEAD_STATUS)	
0FF4	1C	INC	E	;dead-key active?
0FF5	1D	DEC	E	
0FF6	28 C9	JR	Z,JPUTCHR	;no
0FF8	57	LD	D,A	;save encoded code
0FF9	F6 20	OR	00100000B	;force to lower case
0FFB	21 1066	LD	HL,VOWELS+DEADNUM-1	
0FFE	0E 06	LD	C,DEADNUM	
1000	ED B9	CPDR		;is input character vowel?
1002	7A	LD	A,D	;restore code
1003	20 BC	JR	NZ,JPUTCHR	;no
1005	23	INC	HL	
1006	0E 06	LD	C,DEADNUM	
1008		DEAD1:		
1008	09	ADD	HL,BC	
1009	1D	DEC	E	
100A	20 FC	JR	NZ,DEAD1	
100C	7E	LD	A,(HL)	;get from table
100D	CB 6A	BIT	5,D	;is input code lower or upper?
100F	20 B0	JR	NZ,JPUTCHR	;lower, no case translation necessary
1011		CHECK_SPECIAL:		
1011	0E 1F	LD	C,TABLE_LENGTH	;number of special alphabets

```

1013    21 109D          LD      HL,SPECIAL_UPPER-1
1016    ED B9           CPDR           ;found in lower case table?
1018    20 A7           JR      NZ,JPUTCHR ;no
101A    0E 1F          LD      C,TABLE_LENGTH ;number of special alphabets
101C    23             INC      HL          ;compensate [HL] so it points to the
                                           ;data that matched
101D    09             ADD      HL,BC       ;add table length to get address of
                                           ;the character
101E    7E             LD      A,(HL)      ;get code from table
101F    18 A0          JR      JPUTCHR

;
;      Here with raw code in [C]
;
1021    KEYCOD:
1021    79             LD      A,C          ;get raw code
1022    21 1B96        LD      HL,KYJTAB
1025    CD FDCC        CALL    0FDCCH
1028    16 0F          LD      D,0FH
102A    KYCLAS:
102A    BE            CP      (HL)
102B    23            INC     HL
102C    5E            LD      E,(HL)
102D    23            INC     HL
102E    D5            PUSH    DE
102F    D8            RET     C
1030    D1            POP     DE
1031    18 F7          JR      KYCLAS

;
1033    EASYTB:
1033    00            DEFB     0            ;Shift          (48)
1034    00            DEFB     0            ;Control         (49)
1035    00            DEFB     0            ;Graph           (50)

```

Keyboard encoder (0F83H..10C1H)

1036	00	DEFB	0	;Cap lock	(51)
1037	00	DEFB	0	;Kana lock	(52)
1038	00	DEFB	0	;F1	(53)
1039	00	DEFB	0	;F2	(54)
103A	00	DEFB	0	;F3	(55)
103B	00	DEFB	0	;F4	(56)
103C	00	DEFB	0	;F5	(57)
103D	1B	DEFB	27	;Es cape	(58)
103E	09	DEFB	9	;Tab	(59)
103F	00	DEFB	0	;Stop	(60)
1040	08	DEFB	8	;Back space	(61)
1041	18	DEFB	'X'-'@'	;Select	(62)
1042	0D	DEFB	13	;Enter	(63)
1043	20	DEFB	32	;Space	(64)
1044	0C	DEFB	12	;Clear	(65)
1045	12	DEFB	'R'-'@'	;Insert	(66)
1046	7F	DEFB	127	;Rubout	(67)
1047	1D	DEFB	29	;Left	(68)
1048	1E	DEFB	30	;Up	(69)
1049	1F	DEFB	31	;Down	(70)
104A	1C	DEFB	28	;Right	(71)

;

;

;

For additional key matrix

104B	00	DEFB	0	;	(72)
104C	00	DEFB	0	;	(73)
104D	00	DEFB	0	;	(74)
104E	30	DEFB	'0'	;	(75)
104F	31	DEFB	'1'	;	(76)
1050	32	DEFB	'2'	;	(77)
1051	33	DEFB	'3'	;	(78)
1052	34	DEFB	'4'	;	(79)

```
1053 35      DEFB '5'      ;      (80)
1054 36      DEFB '6'      ;      (81)
1055 37      DEFB '7'      ;      (82)
1056 38      DEFB '8'      ;      (83)
1057 39      DEFB '9'      ;      (84)
1058 2D      DEFB '-'      ;      (85)
1059 2C      DEFB ','      ;      (86)
105A 2E      DEFB '.'      ;      (87)
;
105B      NEW_UPDATE:
105B AF      XOR      A      ;clear DEAD_STATUS since code generated
105C 32 FCAC  LD      (DEAD_STATUS),A
105F 18 61    JR      UPDATE
;
1061      VOWELS:
1061 61 65 69 6F DEFB 'aeiouy'
1065 75 79
;
;      Table of codes when vowels are used with a dead key.
;
;
;      For 'dead-key' (non-shifted)
;
1067 85      DEFB 85H      ;a accent grave
1068 8A      DEFB 8AH      ;e accent grave
1069 8D      DEFB 8DH      ;i accent grave
106A 95      DEFB 95H      ;o accent grave
106B 97      DEFB 97H      ;u accent grave
106C 79      DEFB 'y'
;
;      For shifted dead-key
;
```



```
106D    A0                DEFB    0A0H                ;a accent egu
106E    82                DEFB    82H                ;e accent egu
106F    A1                DEFB    0A1H                ;i accent egu
1070    A2                DEFB    0A2H                ;o accent egu
1071    A3                DEFB    0A3H                ;u accent egu
1072    79                DEFB    'y'
;
;      For code dead-key
;
1073    83                DEFB    83H                ;a accent circonflex
1074    88                DEFB    88H                ;e accent circonflex
1075    8C                DEFB    8CH                ;i accent circonflex
1076    93                DEFB    93H                ;o accent circonflex
1077    96                DEFB    96H                ;u accent circonflex
1078    79                DEFB    'y'
;
;      For shifted-code dead key
;
1079    84                DEFB    84H                ;a umlaut
107A    89                DEFB    89H                ;e umlaut
107B    8B                DEFB    8BH                ;i umlaut
107C    94                DEFB    94H                ;o umlaut
107D    81                DEFB    81H                ;u umlaut
107E    98                DEFB    98H                ;y umlaut
;
;      Table of special alphabets
;
;      Used to determine if a key should be affected by capital lock
;
107F    SPECIAL_ALPHABET:
107F    83                DEFB    83H                ;a accent circonflex
```

1080	88	DEFB	88H	;e accent circonflex
1081	8C	DEFB	8CH	;i accent circonflex
1082	93	DEFB	93H	;o accent circonflex
1083	96	DEFB	96H	;u accent circonflex
1084	84	DEFB	84H	;a umlaut
1085	89	DEFB	89H	;e umlaut
1086	8B	DEFB	8BH	;i umlaut
1087	94	DEFB	94H	;o umlaut
1088	81	DEFB	81H	;u umlaut
1089	98	DEFB	98H	;y umlaut
108A	A0	DEFB	0A0H	;a accent ugu
108B	82	DEFB	82H	;e accent ugu
108C	A1	DEFB	0A1H	;i accent ugu
108D	A2	DEFB	0A2H	;o accent ugu
108E	A3	DEFB	0A3H	;u accent ugu
108F	85	DEFB	85H	;a accent grave
1090	8A	DEFB	8AH	;e accent grave
1091	8D	DEFB	8DH	;i accent grave
1092	95	DEFB	95H	;o accent grave
1093	97	DEFB	97H	;u accent grave
1094	B1	DEFB	0B1H	;a tilde
1095	B3	DEFB	0B3H	;i tilde
1096	B5	DEFB	0B5H	;o tilde
1097	B7	DEFB	0B7H	;u tilde
1098	A4	DEFB	0A4H	;n tilde
1099	86	DEFB	86H	;a circle
109A	87	DEFB	87H	;c cedille

109B	91	DEFB	91H	;ae
109C	B9	DEFB	0B9H	;ij
109D	79	DEFB	'y'	
001F		TABLE_LENGTH	EQU	\$-SPECIAL_ALPHABET
		;		
109E		SPECIAL_UPPER:		
109E	41	DEFB	'A'	;A accent circonflex
109F	45	DEFB	'E'	;E accent circonflex
10A0	49	DEFB	'I'	;I accent circonflex
10A1	4F	DEFB	'O'	;O accent circonflex
10A2	55	DEFB	'U'	;U accent circonflex
10A3	8E	DEFB	8EH	;A umlaut
10A4	45	DEFB	'E'	;E umlaut
10A5	49	DEFB	'I'	;I umlaut
10A6	99	DEFB	99H	;O umlaut
10A7	9A	DEFB	9AH	;U umlaut
10A8	59	DEFB	'Y'	;Y umlaut
10A9	41	DEFB	'A'	;A accent egu
10AA	90	DEFB	90H	;E accent egu
10AB	49	DEFB	'I'	;I accent egu
10AC	4F	DEFB	'O'	;O accent egu
10AD	55	DEFB	'U'	;U accent egu
10AE	41	DEFB	'A'	;A accent grave
10AF	45	DEFB	'E'	;E accent grave
10B0	49	DEFB	'I'	;I accent grave
10B1	4F	DEFB	'O'	;O accent grave
10B2	55	DEFB	'U'	;U accent grave
10B3	B0	DEFB	0B0H	;A tilde

10B4	B2	DEFB	0B2H	;I tilde
10B5	B4	DEFB	0B4H	;O tilde
10B6	B6	DEFB	0B6H	;U tilde
10B7	A5	DEFB	0A5H	;N tilde
10B8	8F	DEFB	8FH	;A circle
10B9	80	DEFB	80H	;C cedille
10BA	92	DEFB	92H	;AE
10BB	B8	DEFB	0B8H	;IJ
10BC	59	DEFB	'Y'	

```
IF      TABLE_LENGTH NE ($-SPECIAL_UPPER)
.PRINTX * Upper case table inconsistent *
ENDIF
```

```
PRINTV %(10C2H-$)
```

```
SUBTTL  Function key content
```

```
1404 34      ORG      1404H
;
;      Patch to change the default border color to 4
;
      DEFB      '4'          ;change default border color to 4

      SUBTTL  Dispatch table (1B94H..1BAAH)
```

```

                                ORG      1B94H
                                ;
                                ;      Patch to ignore the katakana to hiragana mapping
                                ;
1B94      18 16                JR      1BACH
                                ;
                                KYJTAB:
1B96                                DEFB      48
1B96      30                                DEFB      LOW INTKEY
1B97      83                                DEFB      51
1B98      33                                DEFB      LOW KYEASY
1B99      10                                DEFB      52
1B9A      34                                DEFB      LOW 0F36H          ;capital lock
1B9B      36                                DEFB      53
1B9C      35                                DEFB      LOW KYEASY          ;code
1B9D      10                                DEFB      58
1B9E      3A                                DEFB      LOW KYFUNC          ;function key
1B9F      C3                                DEFB      60
1BA0      3C                                DEFB      LOW KYEASY
1BA1      10                                DEFB      61
1BA2      3D                                DEFB      LOW 0F46H          ;stop key
1BA3      46                                DEFB      65
1BA4      41                                DEFB      LOW KYEASY
1BA5      10                                DEFB      66
1BA6      42                                DEFB      LOW 0F06H          ;CLS/HOME key
1BA7      06                                DEFB      255
1BA8      FF                                DEFB      LOW KYEASY
1BA9      10

```

```

IF2
IF      (HIGH INTKEY) NE 0FH
.PRINTX * INTKEY not on 0FxxH *

```

SX USA version Macro-80
ispatch table (1B94H..1BAAH)

3.44

01-Jan-85

PAGE 10-1

311

```
ENDIF
IF      (HIGH KYFUNC) NE 0FH
.PRINTX * KYFUNC not on 0FxxH *
ENDIF
ENDIF
```

```
PRINTV  %(1BABH-$)
SUBTTL  Character font
```

5X USA version Macro-80
character font

3.44

01-Jan-85

PAGE 11

312

ORG 1BBFH
.list

(Font Image of each version)

1BBFH to 23BEH

currency symbol and print formatter symbols

Address	Size	Disassembly	Comment
3499	24	ORG 3499H DEFB '\$'	;UK - 9CH, Pound Sign
3549	24	ORG 3549H DEFB '\$'	;UK - 9CH, Pound sign
5600	CD 7FB7	ORG 5600H CALL PATCH1	
60E3	5C	ORG 60E3H DEFB 'r'	
60F1	5C	ORG 60F1H DEFB '\'	
6109	26	ORG 6109H DEFB '&'	
611F	5C	ORG 611FH DEFB '\'	
6126	24	ORG 6126H DEFB '\$'	;UK - 9CH, Pound sign
6135	24	ORG 6135H DEFB '\$'	;UK - 9CH, Pound Sign
		SUBTTL	Miscellaneous patches

```

                                ORG      738AH
                                ;
                                ;      Patch to allow graphic characters in ASCII load
                                ;
738A      FE 0A                  CP        0AH                      ;line feed?
738C      28 EE                  JR        Z,737CH                  ;yes, ignore this

                                ORG      7754H
                                ;
                                ;      TCONST
                                ;      Store original value - do not change
                                ;      ; 60*120*4/2 = 14400 ;
                                ;      ; 50*120*4/2 = 12000 ;
7754      40                    DEFB      40H                      ;UK - 0 (2nd byte of mantissa)
7755      00                    DEFB      00H                      ;UK - 0 (3rd byte of mantissa)
7756      45                    DEFB      45H                      ;UK - 45H (exponent)
7757      14                    DEFB      14H                      ;UK - 12H (1st byte of mantissa)

                                ORG      7D2EH
                                ;
                                ;      Patch to change to 40 character mode
                                ;
7D2E      CD 006C                CALL      INITXT

                                ORG      7F55H
                                ;
                                ;      Patch to change to 37 character mode
                                ;
7F55      27                    DEFB      39                      ;39 character mode for NTSC

                                ORG      7F92H                      ;UK - 37 character mode for PAL
```

```

;      Patch to change the default border color to 4
;
7F92    04      DEFB    4

;
;      Patch code to fix ":xxx" file names
;
;      ORG      7FB7H
PATCH1:
7FB7      LD      DE,0FD89H      ;load PROCNM
7FB7    11 FD89      AND      A      ;is device name null?
7FBA    A7          RET      NZ      ;no
7FBB    C0          INC      B      ;yes, fake 1
7FBC    04          RET
7FBD    C9

7FBE      LASTWR EQU      $

      END
```

Macros:
PRINTV

Symbols:

FCAB	CAP_LOCK	0FF0	CHECK_DEAD	1011	CHECK_SPECIAL
0132	CHGCAP	0E65	CODE	0E95	CODE_SHIFT
1008	DEAD1	0006	DEADNUM	0F1F	DEAD_KEY
0F2B	DEAD_KEY1	FCAC	DEAD_STATUS	1033	EASYTB
0F64	GENCLK	0E05	GRAPH	0E35	GRAPH_SHIFT
006C	INITXT	0F83	INTKEY	0FA0	INTKEY_1
0F9E	IS_CONTROL	0FC1	JPUTCHR	1021	KEYCOD
102A	KYCLAS	0F10	KYEASY	0FCD	KYFNCL
0FC3	KYFUNC	1B96	KYJTAB	7FBE	LASTWR
105B	NEW_UPDATE	0DA5	NORMAL	0FDF	NOT_2BYTE
7FB7	PATCH1	009C	POND	0F55	PUTCHR
FBEB	SFTKEY	0DD5	SHIFT	107F	SPECIAL_ALPHABET
109E	SPECIAL_UPPER	001F	TABLE_LENGTH	10C2	UPDATE
1061	VOWELS	0FD0	WASNT_CONTROL		

No Fatal error(s)

List of some ROM BIOS calls used by BASIC:

Name: SYNCHR, 0008H
Function: Checks if the current character pointed by HL is the one we want. If not, generates 'Syntax error', otherwise falls into CHRGT. Entry: HL, character to be checked be placed at the next location to this RST.
Returns: HL points to next character, A has the character.
Carry flag set if number, Z flag set if end of statement.
Modifies: AF, HL

Name: CHRGT, 0010H
Function: Gets next character (or token) from BASIC text.
Entry: HL
Returns: HL points to next character, A has the character. Carry flag set if number, Z flag set if end of statement encountered.
Modifies: AF, HL

Name: OUTDO, 0018H
Function: Outputs to current device
Entry: A, PTRFIL, PRTFLG
Returns: None
Modifies: None

Name: DCOMPR, 0020H
Function: Compares HL with DE
Entry: HL, DE
Returns: Flags
Modifies: AF

Name: GETYPR, 0028H
Function: Returns the type of FAC
Entry: FAC
Returns: Flags
Modifies: AF

Name: CALLF, 0030H
Function: Performs far_call (i.e., inter-slot call)
Entry: None
Returns: Who knows?
Modifies: ditto
Note: Calling sequence is as follows.

RST 6
DB destination slot
DW destination address

For precise description about parameters, see
CALSLT.

Name: CHSNS, 009CH
Function: Checks the status of keyboard buffer.
Entry: None
Returns: Z flag reset if there's any character in buffer
Modifies: AF

Name: CHGET, 009FH
Function: Waits until any characters are typed, and return
 with the character code.
Entry: None
Returns: Character code in [Acc]
Modifies: AF

Name: CHPUT, 00A2H
Function: Outputs a character to console.
Entry: Character code to be output in [Acc]
Returns: None
Modifies: None

Name: LPTOUT, 00A5H
Function: Outputs a character to LPT
Entry: Character code to be output in [Acc]
Returns: Carry flag set if aborted
Modifies: F

Name: LPTSTT, 00A8H
Function: Checks line printer status
Entry: None
Returns: 255 in [Acc] and Z flag reset if printer ready,
 0 and Z flag set if not.
Modifies: AF

Name: CNVCHR, 00ABH
Function: Checks graphic header byte and convert code
Entry: Character code in [Acc]
Returns: Carry flag reset - graphic header byte
 Carry flag set, Z flag set - converted graphic co
 Carry flag set, Z flag reset - non converted code
Modifies: AF

Name: PINLIN, 00AEH
 Function: Accepts a line from console until a CR or STOP is typed, and stores the line in buffer
 Entry: None
 Returns: Address of buffer top-1 in [HL], carry flag set if STOP is typed.
 Modifies: All

Name: INLIN, 00BlH
 Function: Same as PINLIN, except in case AUTFLG is set.
 Entry: None
 Returns: Address of buffer top-1 in [HL], carry flag set if STOP is pressed.
 Modifies: All

Name: QINLIN, 00B4H
 Function: Outputs a '?' mark and a space then fall into INLIN.
 Entry: None
 Returns: Address of buffer top-1 in [HL], carry flag set if STOP is pressed.
 Modifies: All

Name: BREAKX, 00B7H
 Function: Checks the status of Control-STOP key
 Entry: None
 Returns: Carry flag set if being pressed
 Modifies: AF
 Note: This routine is used to check Control-STOP when interrupts are disabled.

Name: ISCNTC, 00BAH
 Function: Checks the status of SHIFT-STOP key
 Entry: None
 Returns: None
 Modifies: None

Name: CKCNTC, 00BDH
 Function: Same as ISCNTC, used by BASIC
 Entry: None
 Returns: None
 Modifies: None

Name: BEEP, 00C0H
 Function: Beeps buzzer, reset sound chip.
 Entry: None
 Returns: None
 Modifies: All

Name: CLS, 00C3H
 Function: Clears screen
 Entry: None
 Returns: None
 Modifies: AF, BC, DE

Name: POSIT, 00C6H
 Function: Locates cursor at specified position.
 Entry: Column in [H], row in [L]
 Returns: None
 Modifies: AF

Name: FNKSB, 00C9H
 Function: Checks if function key display is active. If
 so, displays it, otherwise do nothing.
 Entry: FNKFLG
 Returns: None
 Modifies: All

Name: ERAFNK, 00CCH
 Function: Erases function key display
 Entry: None
 Returns: None
 Modifies: All

Name: DSPFNK, 00CFH
 Function: Displays function key display
 Entry: None
 Returns: None
 Modifies: All

Name: TOTEXT, 00D2H
 Function: Forces screen to text mode
 Entry: None
 Returns: None
 Modifies: All

Following are used to access game I/O

Name: GTSTCK, 00D5H
 Function: Returns the current status of joy stick
 Entry: Joy stick ID in [Acc]
 Returns: Direction in [Acc]
 Modifies: All

Name: GTTRIG, 00D8H
 Function: Returns the current status of trigger button
 Entry: Trigger button ID in [Acc]
 Returns: Returns 0 in [Acc] if not pressed, 255
 otherwise.
 Modifies: AF

Name: GTPAD, 00DBH
Function: Checks current status of touch PAD
Entry: ID in [Acc]
Returns: Value in [Acc]
Modifies: All

Name: GTPDL, 00DEH
Function: Returns the value of paddle
Entry: Paddle ID in [Acc]
Returns: Value in [Acc]
Modifies: All

Following are used to access cassette tape

Name: TAPION, 00ElH
Function: Sets motor on and reads header from tape
Entry: None
Returns: Carry flag set if aborted
Modifies: All

Name: TAPIN, 00E4H
Function: Inputs from tape
Entry: None
Returns: Data in [Acc], carry flag set if aborted.
Modifies: All

Name: TAPIOF, 00E7H
Function: Stops reading from tape
Entry: None
Returns: None
Modifies: None

Name: TAPOON, 00EAH
Function: Sets motor on and writes header block to cassette.
Entry: [Acc] holds non-0 value if a long header desired, 0 if a short header desired.
Returns: Carry flag set if aborted
Modifies: All

Name: TAPOUT, 00EDH
Function: Outputs to tape
Entry: Data to be output in [Acc]
Returns: Carry flag set if aborted
Modifies: All

Name: TAPOOF, 00F0H
Function: Stops writing to tape
Entry: None
Returns: None
Modifies: None

Name: STMOTR, 00F3H
Function: Sets cassette motor
Entry: 0 in [Acc] to stop, 1 to start, 255 to flip.
Returns: None
Modifies: AF

Following are used to handle queues

Name: LFTQ, 00F6H
Function: Returns how many bytes are left in queue
Entry:
Returns:
Modifies:

Name: PUTQ, 00F9H
Function: Puts a byte in queue
Entry:
Returns:
Modifies:

Following are used by GENGRP and ADVGRP modules

Name: FETCHC, 0114H
Function: Fetches current physical address and mask pattern.
Entry: None
Returns: Address in [HL], mask pattern in [Acc]
Modifies: A, HL

Name: STOREC, 0117H
Function: Stores to physical address and mask pattern
Entry: Address in [HL], mask pattern in [Acc]
Returns: None
Modifies: None

Name: GTASPC, 0126H
Function: Returns aspect ratio
Entry: None
Returns: DE, HL
Modifies: DE, HL

Name: PNTINI, 0129H
Function: Initializes for PAINT
Entry:
Returns:
Modifies:

Name: SCANR, 012CH
Function: Scans pixels to right
Entry:
Returns:
Modifies:

Name: SCANL, 012FH
Function: Scans pixels to left
Entry:
Returns:
Modifies:

Following are the additional entries

Name: CHGCAP, 0132H
Function: Changes the status of CAP lamp
Entry: 0 in [Acc] to turn off the lamp, non 0 otherwise.
Returns: None
Modifies: AF

Name: CHGSND, 0135H
Function: Changes the status of 1 bit sound port.
Entry: 0 in [Acc] to turn off, non 0 otherwise.
Returns: None
Modifies: AF

Name: RSLREG, 0138H
Function: Reads what is currently output to primary slot register.
Entry: None
Returns: Result in [Acc]
Modifies: A

Name: WSLREG, 013BH
Function: Writes to primary slot register.
Entry: Value in [Acc]
Returns: None
Modifies: None

Name: RDVDP, 013EH
Function: Reads VDP's status register.
Entry: None
Returns: Data in [Acc]
Modifies: A

Name: SNSMAT, 0141H
Function: Returns the status of specified row of a keyboard matrix.
Entry: Row # in [Acc]
Returns: Status in [Acc], corresponding bit is reset to 0 if being pressed.
Modifies: AF

Name: ISFLIO, 014AH
Function: Checks if we're doing device I/O
Entry: None
Returns: Non zero if so, zero otherwise
Modifies: AF

Name: OUTDLP, 014DH
Function: Outputs to LPT
Entry: Code in [Acc]
Returns: None
Modifies: F
Note: This entry differs from LPTOUT in that:
1) TABs are expanded to spaces,
2) HIRAGANA and graphics symbol are converted when non-MSX printer is in use,
3) a jump to 'device I/O error' is made when aborted.

Name: KILBUF, 0156H
Function: Clears keyboard buffer
Entry: None
Returns: None
Modifies: HL

Name: CALBAS, 0159H
Function: Performs far_call (i.e., inter-slot call) into BASIC interpreter.
Entry: Address in [IX]
Returns: Who knows?
Modifies: ditto

APPENDIX B

INTERNATIONAL MSX VERSIONS

o Character Set (Common to DIN, French, INT, UK, and USA)

Character Code Table (International)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4																
5																
6																
7																
8																
9																
A																
B																
C																
D																
E																
F																

Note: The font of the character '0' (Zero) is different for DIN version. See figure.

```

***
*   *
*   *
* * *
*   *
*   *
***

```

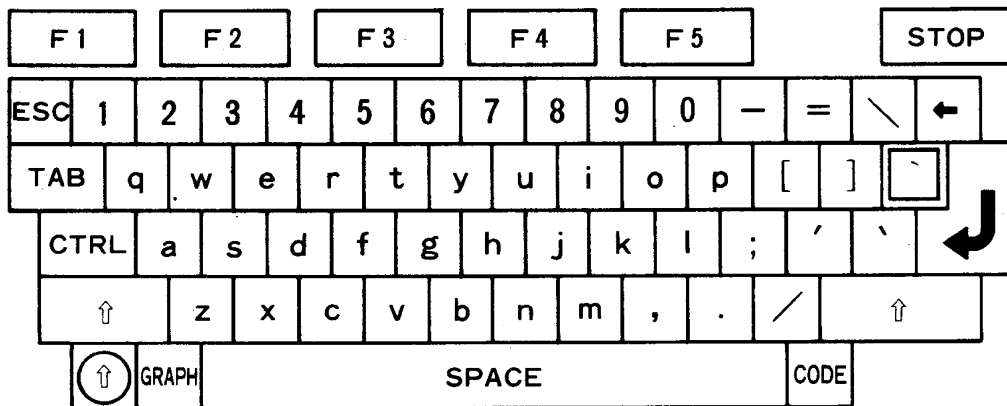
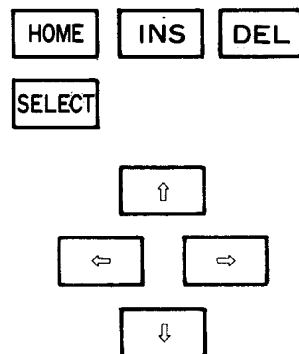
INTERNATIONAL MSX VERSIONS

o Decode International (USA)

I N T			0	1	2	3	4	5	6	7
0	Normal		0 30	1 31	2 32	3 33	4 34	5 35	6 36	7 37
		Shift) 29	' 21	@ 40	# 23	\$ 24	% 25	^ 5E	& 26
	Graph		○ 09	¼ AC	½ AB	¾ BA	η EF	‰ BD	ƒ F4	√ FB
		Shift	◉ 0A		² FD	" FC			J F5	
	Code		δ EB	f 9F	‡ D9	§ BF	¢ 9B	ÿ 98	α E0	β E1
		Shift	Δ D8	ι AD	Pt 9E	¶ BE	£ 9C	Υ 9D		
1	Normal		8 38	9 39	— 2D	= 3D	\\ 5C	[5B] 5D	; 3B
		Shift	* 2A	(28	_ 5F	+ 2B	! 7C	! 7B	! 7D	: 3A
	Graph		∞ EC	• 07	— 17	± F1	\\ 1E	☺ 01	♪ 0D	♠ 06
		Shift		■ 08	+ 1F	≡ F0	16	☉ 02	♫ 0E	♦ 04
	Code		γ E7	ç 87	ε EE	θ E9		φ ED	ω DA	ü B7
		Shift	l' E2	Ç 80				Φ E8	Ω EA	Û B6
2	Normal		ˆ 27	ˆ 60	, 2C	. 2E	/ 2F		a 61	b 62
		Shift	ˆ 22	ˆ 7E	< 3C	> 3E	? 3F		A 41	B 42
	Graph		♣ 05	BB	≤ F3	≥ F2	/ 1D		■ C4	⊥ 11
		Shift	♥ 03	≈ F7	⋈ AE	⋈ AF	÷ F6		■ FE	
	Code		ij B9	σ E5	å 86	a A6	o A7		ä 84	ü 97
		Shift	IJ B8	Σ E4	Å 8F		ı A8		Ä 8E	
3	Normal		c 63	d 64	e 65	f 66	g 67	h 68	i 69	j 6A
		Shift	C 43	D 44	E 45	F 46	G 47	H 48	I 49	J 4A
	Graph		◇ BC	◼ C7	▼ CD	† 14	+ 15	† 13	■ DC	■ C6
		Shift	FA	◼ C1	▲ CE	■ D4	† 10	■ D6	■ DF	■ CA
	Code		ı 8D	ı 8B	ı 8C	ö 94	ü 81	ä B1	ı A1	æ 91
		Shift				Ö 99	Ü 9A	Ä B0		Æ 92
4	Normal		k 6B	l 6C	m 6D	n 6E	o 6F	p 70	q 71	r 72
		Shift	K 4B	L 4C	M 4D	N 4E	O 4F	P 50	Q 51	R 52
	Graph		■ DD	■ C8	♂ 0B	┘ 1B	■ C2	■ DB	/// CC	┘ 18
		Shift	■ DE	■ C9	♀ 0C	■ D3	■ C3	■ D7	/// CB	┘ A9
	Code		ı B3	o B5	u E6	n A4	ó A2	ü A3	ä 83	ö 93
		Shift	I B2	Ö B4		Ñ A5		ll E3		
5	Normal		s 73	t 74	u 75	v 76	w 77	x 78	y 79	z 7A
		Shift	S 53	T 54	U 55	V 56	W 57	X 58	Y 59	Z 5A
	Graph		♠ D2	┘ 12	■ C0	┘ 1A	▶ CF	× 1C	┘ 19	⊗ 0F
		Shift	⊗ D1		■ C5	■ D5	◀ D0	● F9	┘ AA	○ F8
	Code		ë 89	û 96	é 82	ò 95	ē 88	è 8A	á A0	ä 85
		Shift			É 90					

INTERNATIONAL MSX VERSIONS

o Layout International (USA)



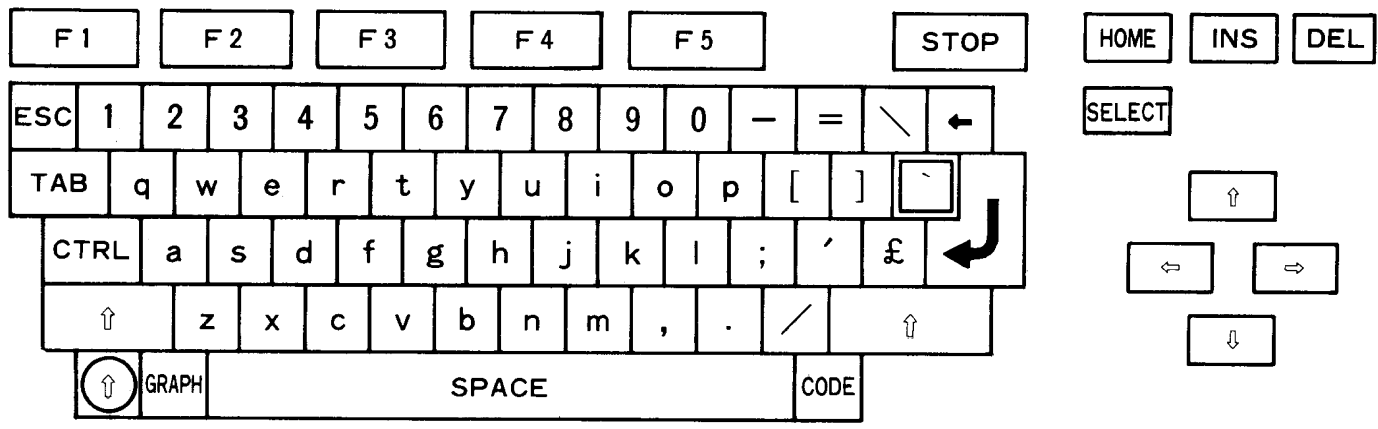
INTERNATIONAL MSX VERSIONS

o Decode UK

U K			0	1	2	3	4	5	6	7
0	Normal		0 30	1 31	2 32	3 33	4 34	5 35	6 36	7 37
		Shift) 29	! 21	@ 40	# 23	\$ 24	% 25	^ 5E	& 26
	Graph		○ 09	¼ AC	½ AB	¾ BA	¾ EF	‰ BD	ƒ F4	√ FB
		Shift	◉ 0A		² FD	³ FC			Ƶ F5	
	Code		δ EB	ƒ 9F	‡ D9	§ BF	€ 9B	¥ 98	α E0	β E1
Shift		Δ D8	ι AD	Π 9E	π BE	£ 9C	¥ 9D			
1	Normal		8 38	9 39	— 2D	= 3D	\ 5C	[5B] 5D	; 3B
		Shift	* 2A	(28	_ 5F	+ 2B	7C	7B	7D	: 3A
	Graph		∞ EC	• 07	— 17	± F1	\ 1E	☺ 01	♪ 0D	♠ 06
		Shift		■ 08	+ 1F	≡ F0	16	☉ 02	♫ 0E	♦ 04
	Code		γ E7	ç 87	ε EE	θ E9	60	φ ED	⊙ DA	û B7
Shift		Γ E2	Ç 80				Φ E8	Ω EA	Û B6	
2	Normal		˙ 27	£ 9C	˙ 2C	˙ 2E	/ 2F	dead key	a 61	b 62
		Shift	˙ 22	˙ 7E	< 3C	> 3E	? 3F		A 41	B 42
	Graph		♣ 05	˙ BB	≤ F3	≥ F2	/ 1D		■ C4	⊥ 11
		Shift	♥ 03	≈ F7	« AE	» AF	÷ F6		■ FE	
	Code		ij B9	σ E5	à 86	á A6	ó A7		ä 84	ü 97
Shift		IJ B8	Σ E4	À 8F		¿ A8	À 8E			
3	Normal		c 63	d 64	e 65	f 66	g 67	h 68	i 69	j 6A
		Shift	C 43	D 44	E 45	F 46	G 47	H 48	I 49	J 4A
	Graph		◇ BC	■ C7	▼ CD	† 14	+ 15	† 13	■ DC	■ C6
		Shift	˙ FA	■ C1	▲ CE	■ D4	+ 10	■ D6	■ DF	■ CA
	Code		i 8D	ï 8B	ï 8C	ö 94	ü 81	ä B1	í A1	æ 91
Shift					Ö 99	Ü 9A	Ä B0		Æ 92	
4	Normal		k 6B	l 6C	m 6D	n 6E	o 6F	p 70	q 71	r 72
		Shift	K 4B	L 4C	M 4D	N 4E	O 4F	P 50	Q 51	R 52
	Graph		■ DD	■ C8	♂ 0B	┘ 1B	■ C2	■ DB	≡ CC	┘ 18
		Shift	■ DE	■ C9	♀ 0C	■ D3	■ C3	■ D7	≡ CB	┘ A9
	Code		i B3	ö B5	μ E6	ñ A4	ó A2	ú A3	â 83	ô 93
Shift		I B2	Ö B4		Ñ A5		Π E3			
5	Normal		s 73	t 74	u 75	v 76	w 77	x 78	y 79	z 7A
		Shift	S 53	T 54	U 55	V 56	W 57	X 58	Y 59	Z 5A
	Graph		⌘ D2	┘ 12	■ C0	┘ 1A	▶ CF	× 1C	┘ 19	⊛ 0F
		Shift	⌘ D1		■ C5	■ D5	◀ D0	● F9	┘ AA	○ F8
	Code		ë 89	ù 96	é 82	ò 95	ê 88	ë 8A	â A0	à 85
Shift				É 90						

INTERNATIONAL MSX VERSIONS

o Layout UK



o Character Code Table (Japanese)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
B	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
C	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
D	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
F	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

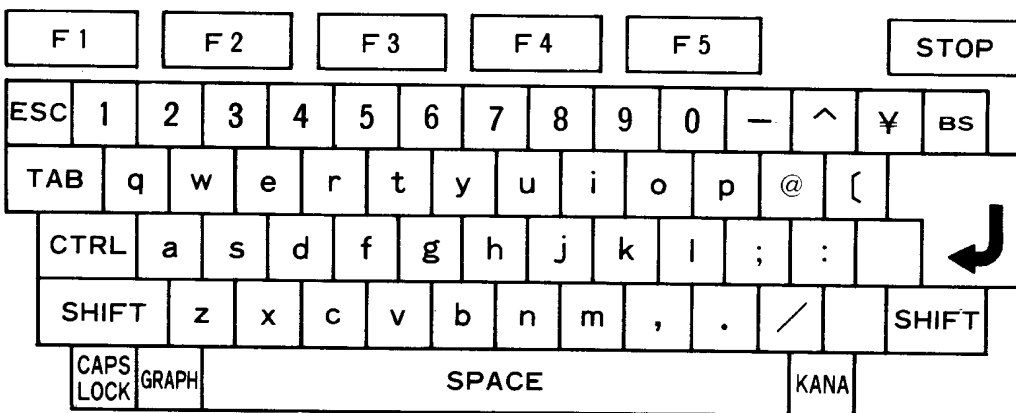
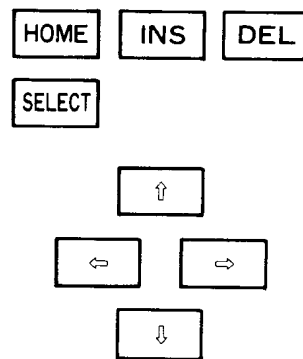
INTERNATIONAL MSX VERSIONS

o Decode Japanese 1

J I S			0	1	2	3	4	5	6	7
0	Normal		0 30	1 31	2 32	3 33	4 34	5 35	6 36	7 37
		Shift		! 21	" 22	# 23	\$ 24	% 25	& 26	' 27
	Graph		万 0F	日 07	月 01	火 02	水 03	木 04	金 05	土 06
	Kana		わ FC	ぬ E7	ふ EC	あ 91	う 93	え 94	お 95	や F4
		Caps	ワ DC	ヌ C7	フ CC	ア B1	ウ B3	エ B4	オ B5	ヤ D4
1	Normal		8 38	9 39	-- 2D	^ 5E	¥ 5C	@ 40	[5B	; 3B
		Shift	(28) 29	= 3D	~ 7E	! 7C	' 60	7B	+ 2B
	Graph		百 0D	千 E0	-- 17		円 09		○ 84	♣ 82
	Kana		ゆ F5	よ F6	ほ EE	へ ED	ー B0	* DE	° DF	れ FA
		Caps	ユ D5	ヨ D6	ホ CE	ヘ CD	ー B0	* DE	° DF	レ DA
2	Normal		: 3A) 5D	, 2C	. 2E	/ 2F		a 61	b 62
		Shift	* 2A	7D	< 3C	> 3E	? 3F	_ 5F	A 41	B 42
	Graph		♥ 81	● 85	小 1F	大 1D	♠ 80	◆ 83		┘ 1B
	Kana		け 99	む F1	ね E8	る F9	め F2	ろ FB	ち E1	こ 9A
		Caps	ケ B9	ム D1	ネ C8	ル D9	メ D2	ロ DB	チ C1	コ BA
3	Normal		c 63	d 64	e 65	f 66	g 67	h 68	i 69	j 6A
		Shift	C 43	D 44	E 45	F 46	G 47	H 48	I 49	J 4A
	Graph		ㄥ 1A	ㄣ 14	ㄱ 18	ㅍ 15	ㅊ 13	時 0A	16	
	Kana		そ 9F	し 9C	い 92	は EA	き 97	く 98	に E6	ま EF
		Caps	ソ BF	シ BC	イ B2	ハ CA	キ B7	ク B8	ニ C6	マ CF
4	Normal		k 6B	l 6C	m 6D	n 6E	o 6F	p 70	q 71	r 72
		Shift	K 4B	L 4C	M 4D	N 4E	O 4F	P 50	Q 51	R 52
	Graph			中 1E	分 0B			π 10		〒 12
	Kana		の E9	り F8	も F3	み F0	ら F7	せ 9E	た E0	す 9D
		Caps	ノ C9	リ D8	モ D3	ミ D0	ラ D7	セ BE	タ C0	ス BD
5	Normal		s 73	t 74	u 75	v 76	w 77	x 78	y 79	z 7A
		Shift	S 53	T 54	U 55	V 56	W 57	X 58	Y 59	Z 5A
	Graph		秒 0C	ㄣ 19		ㄣ 11		× 1C	年 08	
	Kana		と E4	か 96	な E5	ひ EB	て E3	さ 9B	ん FD	つ E2
		Caps	ト C4	カ B6	ナ C5	ヒ CB	テ C3	サ BB	ン DD	ツ C2

INTERNATIONAL MSX VERSIONS

o Layout Japanese



INTERNATIONAL MSX VERSIONS

o Decode Japanese 2

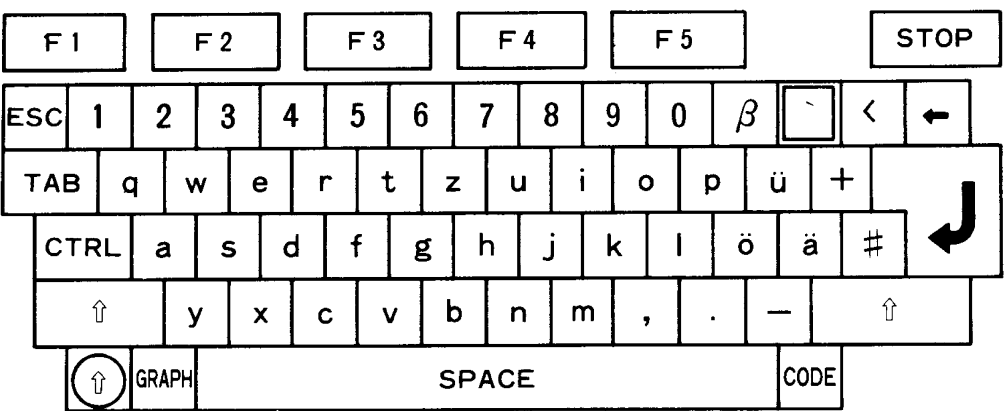
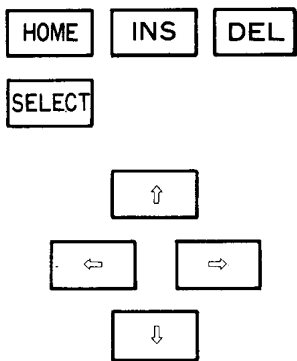
KANJI+SHIFT		0	1	2	3	4	5	6	7
0		を 86			あ 87	う 89	え 8A	お 8B	や 8C
	Caps	ヲ A6			ア A7	ウ A9	エ AA	オ AB	ヤ AC
1		ゆ 8D	よ 8E					「 A2	
	Caps	ユ AD	ヨ AE					「 A2	
2			」 A3	A4	。 A1	・ A5			
	Caps		」 A3	、 A4	。 A1	・ A5			
3				い 88					
	Caps			イ A8					
5									っ 8F
	Caps								ッ AF

INTERNATIONAL MSX VERSIONS

o Decode DIN

D I N		0	1	2	3	4	5	6	7	
0	Normal		0 30	1 31	2 32	3 33	4 34	5 35	6 36	7 37
		Shift	= 3D	! 21	" 22	§ BF	\$ 24	% 25	& 26	/ 2F
	Graph		○ 09	¼ AC	½ AB	¾ BA	η EF	‰ BD	∫ F4	/ 1D
		Shift	◻ 0A		² FD	³ FC		÷ F6	∫ F5	\ 1E
	Code		δ EB	∫ 7C	@ 40	€ EE	¢ 87	¢ 9B	γ E7	\ 5C
		Shift	Δ D8	∫ AD	Pt 9E	π BE	Ç 80	£ 9C	Γ E2	
1	Normal		8 38	9 39	β E1		< 3C	ü 81	+ 2B	ö 94
		Shift	(28) 29	? 3F	dead key	> 3E	Ü 9A	* 2A	Ö 99
	Graph		∞ EC	• 07	♪ 0D	60	◁ AE	☺ 01	± F1	♠ 06
		Shift		■ 08	♫ 0E	27	▷ AF	☹ 02	+ 1F	♦ 04
	Code		[5B] 5D	θ E9	dead key	≤ F3	φ ED	ω DA	ū B7
		Shift	7B	7D	∞ A8	dead key	≥ F2	Φ E8	Ω EA	Û B6
2	Normal		ä 84	# 23	˙ 2C	˙ 2E	- 2D		a 61	b 62
		Shift	Ä 8E	^ 5E	; 3B	: 3A	_ 5F		A 41	B 42
	Graph		♣ 05	˘ 7E	√ FB	16	- 17		■ C4	⊥ 11
		Shift	♥ 03	˘ BB	≈ F7		= F0		■ FE	
	Code		ij B9	σ E5	â 86	ä A6	ü A7		α E0	û 97
		Shift	IJ B8	Σ E4	Å 8F					
3	Normal		c 63	d 64	e 65	f 66	g 67	h 68	i 69	j 6A
		Shift	C 43	D 44	E 45	F 46	G 47	H 48	I 49	J 4A
	Graph		◇ BC	◼ C7	▼ CD	† 14	+ 15	† 13	■ DC	■ C6
		Shift	- FA	◼ C1	▲ CE	■ D4	+ 10	■ D6	■ DF	■ CA
	Code		î 8D	ï 8B	î 8C	f 9F	y 98	ä B1	í A1	æ 91
		Shift						Ä B0		Æ 92
4	Normal		k 6B	l 6C	m 6D	n 6E	o 6F	p 70	q 71	r 72
		Shift	K 4B	L 4C	M 4D	N 4E	O 4F	P 50	Q 51	R 52
	Graph		■ DD	■ C8	♂ 0B	┘ 1B	■ C2	■ DB	≡ CC	┘ 18
		Shift	■ DE	■ C9	♀ 0C	■ D3	■ C3	■ D7	≡ CB	┘ A9
	Code		ï B3	ö B5	μ E6	ñ A4	ó A2	ú A3	ä 83	ô 93
		Shift	Ï B2	Ö B4		Ñ A5		Π E3		
5	Normal		s 73	t 74	u 75	v 76	w 77	x 78	z 7A	y 79
		Shift	S 53	T 54	U 55	V 56	W 57	X 58	Z 5A	Y 59
	Graph		♠ D2	† 12	■ C0	┘ 1A	▶ CF	× 1C	┘ 19	✱ 0F
		Shift	✱ D1	† D9	■ C5	■ D5	◀ D0	● F9	┘ AA	○ F8
	Code		ë 89	û 96	é 82	ó 95	è 88	è 8A	à A0	á 85
		Shift			É 90					¥ 9D

o Layout DIN



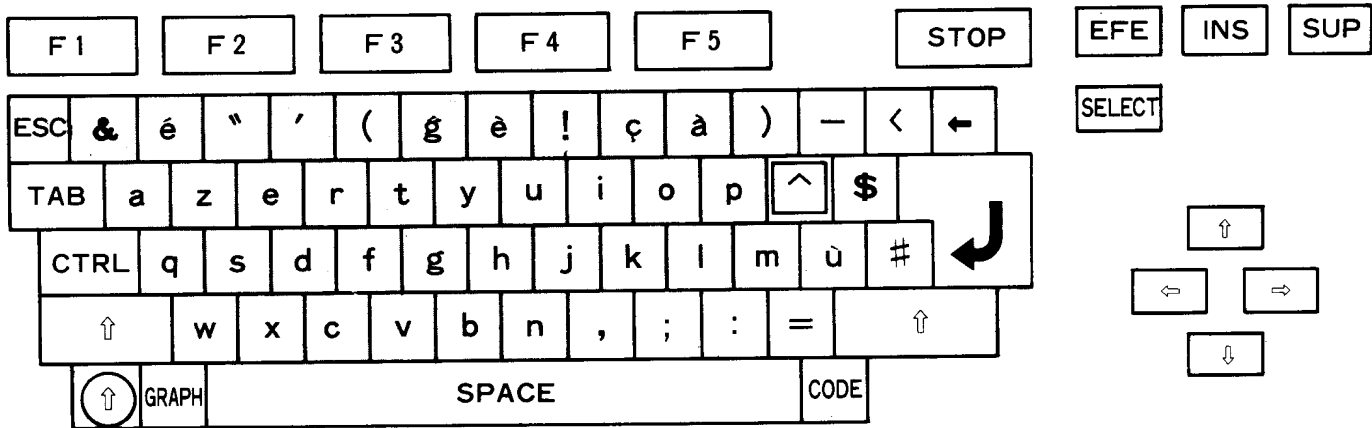
INTERNATIONAL MSX VERSIONS

o Decode French

F R			0	1	2	3	4	5	6	7
0	Normal		à 85	& 26	é 82	" 22	' 27	(28	§ BF	è 8A
		Shift	0 30	1 31	2 32	3 33	4 34	5 35	6 36	7 37
	Graph		○ 09	£ AC	½ AB	¼ BA	~ BB	η EF	/ F4	√ FB
		Shift	☐ 0A	16	² FD	" FC	≈ F7		J F5	
	Code		δ EB	7C	α 40	α E0	` 60	7B	^ 5E	€ EE
Shift		Δ D8	AD	Ê 90	Pt 9E		[5B	π BE	~ 7E	
1	Normal		' 21	ç 87) 29	- 2D	< 3C	^	\$ 24	m 6D
		Shift	8 38	9 39	○ F8	- 5F	> 3E	..	* 2A	M 4D
	Graph		∞ EC	• 07	☺ 01	- 17	⌵ AE	^	♪ 0D	♠ 06
		Shift		■ 08	☹ 02	+ 1F	⌶ AF	..	♫ 0E	♦ 04
	Code		γ E7	θ E9	7D	φ ED	≤ F3	^	€ 9B	ü B7
Shift		Γ E2	C 80] 5D	Φ E8	≥ F2	..		Û B6	
2	Normal		ù 97	# 23	; 3B	: 3A	= 3D		q 71	b 62
		Shift	% 25	£ 9C	. 2E	/ 2F	+ 2B		Q 51	B 42
	Graph		♣ 05	‰ BD	÷ F6	\ 1E	± F1		■ C4	⊥ 11
		Shift	♥ 03			/ 1D	≡ F0		■ FE	
	Code		ij B9	σ E5	à 86	ä A6	o A7		ä 84	β E1
Shift		IJ B8	Σ E4	Å 8F	\ 5C			Ä 8E		
3	Normal		c 63	d 64	e 65	f 66	g 67	h 68	i 69	j 6A
		Shift	C 43	D 44	E 45	F 46	G 47	H 48	I 49	J 4A
	Graph		◇ BC	■ C7	▼ CD	14	+ 15	+ 13	■ DC	■ C6
		Shift	- FA	■ C1	▲ CE	■ D4	+ 10	■ D6	■ DF	■ CA
	Code		ï 8D	ï 8B	î 8C	ö 94	ü 81	ä B1	í A1	æ 91
Shift					Ö 99	Ü 9A	Ä B0		Æ 92	
4	Normal		k 6B	l 6C	, 2C	n 6E	o 6F	p 70	a 61	r 72
		Shift	K 4B	L 4C	? 3F	N 4E	O 4F	P 50	A 41	R 52
	Graph		■ DD	■ C8	♂ 0B	1B	■ C2	■ DB	/// CC	┘ 18
		Shift	■ DE	■ C9	♀ 0C	■ D3	■ C3	■ D7	/// CB	┘ A9
	Code		ï B3	ö B5	μ E6	ñ A4	ó A2	ú A3	á 83	ô 93
Shift		Ï B2	Õ B4	ù A8	Ñ A5		ü E3			
5	Normal		s 73	t 74	u 75	v 76	z 7A	x 78	y 79	w 77
		Shift	S 53	T 54	U 55	V 56	Z 5A	X 58	Y 59	W 57
	Graph		✠ D2	┘ 12	■ C0	1A	► CF	× 1C	┘ 19	⚙ 0F
		Shift	✠ D1	‡ D9	■ C5	■ D5	◀ D0	● F9	┘ AA	
	Code		ë 89	û 96	ÿ 98	ö 95	é 88	f 9F	ä A0	ω DA
Shift								¥ 9D	Ω EA	

INTERNATIONAL MSX VERSIONS

o Layout French



Following are definition of hooks and their functions

name - name of hook
 where - where in what module it is used
 purpose - what purpose it is used for

```

FD9A  (HOKJMP,0)
;      name:      H.KEYI
;      where:     MSXIO, at the beginning of interrupt handler
;      purpose:   to do additional interrupt handling such as
;                RS232C
;
FD9A  (H.KEYI,5)
;      name:      H.TIMI
;      where:     MSXIO, in timer interrupt handler
;      purpose:   to allow other interrupt handling invoked by
;                timer
;
FD9F  (H.TIMI,5)
;      name:      H.CHPU
;      where:     MSXIO, at the beginning of CHPUT (CHAracter
;                outPUT) routine
;      purpose:   to allow other console output devices to be used
;
FDA4  (H.CHPU,5)
;      name:      H.DSPC
;      where:     MSXIO, at the beginning of DSPCSR (DiSPlay
;                CurSoR) routine
;      purpose:   to allow other console output devices to be used
;
FDA9  (H.DSPC,5)
;      name:      H.ERAC
;      where:     MSXIO, at the beginning of ERACSR (ERASe CurSoR)
;                routine
;      purpose:   to allow other console output devices to be used

```

```

FDAE  (H.ERAC,5)
;      name:          H.DSPF
;      where:         MSXIO, at the beginning of DSPFNK (DiSPlay
;                      FuNction Key) routine
;      purpose:       to allow other console output devices to be used
;
FDB3  (H.DSPF,5)
;      name:          H.ERAF
;      where:         MSXIO, at the beginning of ERAFNK (ERaSe
;                      FuNction Key) routine
;      purpose:       to allow other console output devices to be used
;
FDB8  (H.ERAF,5)
;      name:          H.TOTE
;      where:         MSXIO, at the beginning of TOTEXT (force screen
;                      TO TEXT mode) routine
;      purpose:       to allow other console output devices to be used
;
FDBD  (H.TOTE,5)
;      name:          H.CHGE
;      where:         MSXIO, at the beginning of CHGET (CHAracter
;                      GET) routine
;      purpose:       to allow other console input devices to be used
;
FDC2  (H.CHGE,5)
;      name:          H.INIP
;      where:         MSXIO, at the beginning of INIPAT (INITialize
;                      PATtern) routine
;      purpose:       to allow other character sets to be used
;
FDC7  (H.INIP,5)
;      name:          H.KEYC
;      where:         MSXIO, at the beginning of KEYCOD (KEY CODer)
;                      routine
;      purpose:       to allow other key assignments to be used
;
FDCC  (H.KEYC,5)

```

```

;      name:      H.KYEA
;      where:     MSXIO, at the beginning of KYEASY (KeY EASY)
;                routine
;      purpose:   to allow other key assignments to be used
;
FDD1  (H.KYEA,5)
;      name:      H.NMI
;      where:     MSXIO, at the beginning of NMI (Non Maskable
;                Interrupt) routine
;      purpose:   to allow NMI handling
;
FDD6  (H.NMI, 5)
;      name:      H.PINL
;      where:     MSXINL, at the beginning of PINLIN (Program
;                INput LINE) routine
;      purpose:   to allow other console input devices or other
;                input design to be used
;
FDDB  (H.PINL,5)
;      name:      H.QINL
;      where:     MSXINL, at the beginning of QINLIN (Question
;                mark and INput LINE) routine
;      purpose:   to allow other console input devices or other
;                input design to be used
;
FDE0  (H.QINL,5)
;      name:      H.INLI
;      where:     MSXINL, at the beginning of INLIN (INput LINE)
;                routine
;      purpose:   to allow other console input devices or other
;                input design to be used
;
FDE5  (H.INLI,5)
;      name:      H.ONGO
;      where:     MSXSTS, at the beginning of ONGOTP (ON GOTo
;                Procedure) routine
;      purpose:   to allow other interrupting devices to be used

```

```

FDEA  (H.ONGO,5)
;      name:      H.DSKO
;      where:     MSXSTS, at the beginning of DSKO$ (DiSK Output)
;               routine
;      purpose:    to install disk driver
;
FDEF  (H.DSKO,5)
;      name:      H.SETS
;      where:     MSXSTS, at the beginning of SETS (SET
;               attributeS) routine
;      purpose:    to install disk driver
;
FDF4  (H.SETS,5)
;      name:      H.NAME
;      where:     MSXSTS, at the beginning of NAME (reNAME) routine
;      purpose:    to install disk driver
;
FDF9  (H.NAME,5)
;      name:      H.KILL
;      where:     MSXSTS, at the beginning of KILL (KILL file)
;               routine
;      purpose:    to install disk driver
;
FDFF  (H.KILL,5)
;      name:      H.IPL
;      where:     MSXSTS, at the beginning of IPL (Initial Program
;               Load) routine
;      purpose:    to install disk driver
;
FE03  (H.IPL, 5)
;      name:      H.COPY
;      where:     MSXSTS, at the beginning of COPY (COPY files)
;               routine
;      purpose:    to install disk driver
;
FE08  (H.COPY,5)

```

```

;      name:      H.CMD
;      where:     MSXSTS, at the beginning of CMD (CoMmAnD)
;               routine
;      purpose:   to install disk driver
;
FE0D (H.CMD, 5)
;      name:      H.DSKF
;      where:     MSXSTS, at the beginning of DSKF (DiSK Free)
;               routine
;      purpose:   to install disk driver
;
FE12 (H.DSKF,5)
;      name:      H.DSKI
;      where:     MSXSTS, at the beginning of DSKI (DiSK Input)
;               routine
;      purpose:   to install disk driver
;
FE17 (H.DSKI,5)
;      name:      H.ATTR
;      where:     MSXSTS, at the beginning of ATTR$ (ATTRibute)
;               routine
;      purpose:   to install disk driver
;
FE1C (H.ATTR,5)
;      name:      H.LSET
;      where:     MSXSTS, at the beginning of LSET (Left SET)
;               routine
;      purpose:   to install disk driver
;
FE21 (H.LSET,5)
;      name:      H.RSET
;      where:     MSXSTS, at the beginning of RSET (Right SET)
;               routine
;      purpose:   to install disk driver
;
FE26 (H.RSET,5)
;      name:      H.FIEL

```



```

;      where:      MSXSTS, at the beginning of FIELD (FIELD)
;
;      purpose:    to install disk driver
;
FE2B (H.FIEL,5)
;      name:      H.MKI$
;      where:     MSXSTS, at the beginning of MKI$ (MaKe Int)
;      routine
;      purpose:    to install disk driver
;
FE30 (H.MKI$,5)
;      name:      H.MKS$
;      where:     MSXSTS, at the beginning of MKS$ (Make Single)
;      routine
;      purpose:    to install disk driver
;
FE35 (H.MKS$,5)
;      name:      H.MKD$
;      where:     MSXSTS, at the beginning of MKD$ (Make Double)
;      routine
;      purpose:    to install disk driver
;
FE3A (H.MKD$,5)
;      name:      H.CVI
;      where:     MSXSTS, at the beginning of CVI (Convert Int)
;      routine
;      purpose:    to install disk driver
;
FE3F (H.CVI,5)
;      name:      H.CVS
;      where:     MSXSTS, at the beginning of CVS (Convert Sng)
;      routine
;      purpose:    to install disk driver
;
FE44 (H.CVS,5)
;      name:      H.CVD
;      where:     MSXSTS, at the beginning of CVD (Convert Dbl)

```

```

; routine
; purpose: to install disk driver
;
FE49 (H.CVD,5)
; name: H.GETP
; where: SPCDSK, at the GETPTR (GET file PoinTeR) routine
; purpose: to install disk driver
;
FE4E (H.GETP,5)
; name: H.SETF
; where: SPCDSK, at the SETFIL (SET FILE pointer) routine
; purpose: to install disk driver
;
FE53 (H.SETF,5)
; name: H.NOFO
; where: SPCDSK, at the NOFOR (NO FOR clause) routine
; purpose: to install disk driver
;
FE58 (H.NOFO,5)
; name: H.NULO
; where: SPCDSK, at the NULOPN (NULL file OPeN) routine
; purpose: to install disk driver
;
FE5D (H.NULO,5)
; name: H.NTFL
; where: SPCDSK, at the NTFL0 (NoT File number 0) routine
; purpose: to install disk driver
;
FE62 (H.NTFL,5)
; name: H.MERG
; where: SPCDSK, at the MERGE (MERGE program files) routine
; purpose: to install disk driver
;
FE67 (H.MERG,5)
; name: H.SAVE
; where: SPCDSK, at the SAVE routine

```

```

;      purpose:      to install disk driver
;
FE6C (H.SAVE,5)
;      name:         H.BINS
;      where:        SPCDSK, at the BINSAB (BINary SAVe) routine
;      purpose:      to install disk driver
;
FE71 (H.BINS,5)
;      name:         H.BINL
;      where:        SPCDSK, at the BINLOD (BINary LOaD) routine
;      purpose:      to install disk driver
;
FE76 (H.BINL,5)
;      name:         H.FILE
;      where:        SPCDSK, at the FILES command
;      purpose:      to install disk driver
;
FE7B (H.FILE,5)
;      name:         H.DGET
;      where:        SPCDSK, at the DGET (Disk GET) routine
;      purpose:      to install disk driver
;
FE80 (H.DGET,5)
;      name:         H.FILO
;      where:        SPCDSK, at the FILOUL (FILE Out 1) routine
;      purpose:      to install disk driver
;
FE85 (H.FILO,5)
;      name:         H.INDS
;      where:        SPCDSK, at the INDSKC (INput DiSK Character)
;                  routine
;      purpose:      to install disk driver
;
FE8A (H.INDS,5)
;      name:         H.RSLF
;      where:        SPCDSK, to re-select old drive
;      purpose:      to install disk driver

```

```

;
FE8F (H.RSLF,5)
;      name:      H.SAVD
;      where:      SPCDSK, to save current drive
;      purpose:    to install disk driver
;
FE94 (H.SAVD,5)
;      name:      H.LOC
;      where:      SPCDSK, at the LOC (LOCation) function
;      purpose:    to install disk driver
;
FE99 (H.LOC, 5)
;      name:      H.LOF
;      where:      SPCDSK, at the LOF (Length Of File) function
;      purpose:    to install disk driver
;
FE9E (H.LOF, 5)
;      name:      H.EOF
;      where:      SPCDSK, at the EOF (End Of File) function
;      purpose:    to install disk driver
;
FEA3 (H.EOF, 5)
;      name:      H.FPOS
;      where:      SPCDSK, at the FPOS (File POSition) function
;      purpose:    to install disk driver
;
FEA8 (H.FPOS,5)
;      name:      H.BAKU
;      where:      SPCDSK, at the BAKUPT (BACK UP) routine
;      purpose:    to install disk driver
;
FEAD (H.BAKU,5)
;      name:      H.PARD
;      where:      SPCDEV, at the PARDEV (PARse DEVice name)
;                  routine
;      purpose:    to expand logical device names
;

```

```

FEB2  (H.PARD,5)
;      name:          H.NODE
;      where:         SPCDEV, at the NODEVN (NO DEvice Name) routine
;      purpose:       to set other default device
;
FEB7  (H.NODE,5)
;      name:          H.POSD
;      where:         SPCDEV, at the POSDSK (POSSibly DiSK) routine
;      purpose:       to install disk driver
;
FEBC  (H.POSD,5)
;      name:          H.DEVN
;      where:         SPCDEV, at the DEVNAM (DEvice NAME) routine
;      purpose:       to expand logical device names
;
FEC1  (H.DEVN,5)
;      name:          H.GEND
;      where:         SPCDEV, at the GENDSP (GENeral device
;                      DiSPatcher)
;      purpose:       to expand logical device names
;
FEC6  (H.GEND,5)
;      name:          H.RUNC
;      where:         BIMISC, at the RUNC (RUN Clear) routine
;      purpose:
;
FECB  (H.RUNC,5)
;      name:          H.CLEA
;      where:         BIMISC, at the CLEARC (CLEAR Clear) routine
;      purpose:
;
FED0  (H.CLEA,5)
;      name:          H.LOPD
;      where:         BIMISC, at the LOPDFT (LOOp and set DeFault)
;                      routine
;      purpose:       to use other defaults for variables
;

```

```

FED5  (H.LOPD,5)
;      name:      H.STKE
;      where:     BIMISC, at the STKERR (STacK ERRor) routine
;      purpose:
;
FEDA  (H.STKE,5)
;      name:      H.ISFL
;      where:     BIMISC, at the ISFLIO (IS FiLe I/O) routine
;      purpose:
;
FEDF  (H.ISFL,5)
;      name:      H.OUTD
;      where:     BIO, at the OUTDO (OUT DO) routine
;      purpose:
;
FEE4  (H.OUTD,5)
;      name:      H.CRDO
;      where:     BIO, at the CRDO (CRlf DO) routine
;      purpose:
;
FEE9  (H.CRDO,5)
;      name:      H.DSKC
;      where:     BIO, at the DSKCHI (DiSK CHaracter Input)
;                  routine
;      purpose:
;
FEEE  (H.DSKC,5)
;      name:      H.DOGR
;      where:     GENGRP, at the DOGRPH (DO GRaPH) routine
;      purpose:
;
FEF3  (H.DOGR,5)
;      name:      H.PRGE
;      where:     BINTRP, at the PRGEN (PRoGram END) routine
;      purpose:
;
FEF8  (H.PRGE,5)

```

```

;      name:      H.ERRP
;      where:     BINTRP, at the ERRPRT (ERRor PRinT) routine
;      purpose:
;
FEFD  (H.ERRP,5)
;      name:
;      where:     BINTRP
;      purpose:
;
FF02  (H.ERRF,5)
;      name:      H.READ.
;      where:     BINTRP, at the READY entry
;      purpose:
;
FF07  (H.READ,5)
;      name:      H.MAIN
;      where:     BINTRP, at the MAIN entry
;      purpose:
;
FF0C  (H.MAIN,5)
;      name:      H.DIRD
;      where:     BINTRP, at the DIRDO (DIRect statement DO).
;      purpose:
;
FF11  (H.DIRD,5)
;      name:
;      where:     BINTRP
;      purpose:
;
FF16  (H.FINI,5)
;      name:
;      where:     BINTRP
;      purpose:
;
FF1B  (H.FINE,5)
;      name:
;      where:     BINTRP

```

```
      ;      purpose:
      ;
FF20  (H.CRUN,5)
      ;      name:
      ;      where:      BINTRP
      ;      purpose:
      ;
FF25  (H.CRUS,5)
      ;      name:
      ;      where:      BINTRP
      ;      purpose:
      ;
FF2A  (H.ISRE,5)
      ;      name:
      ;      where:      BINTRP
      ;      purpose:
      ;
FF2F  (H.NTFN,5)
      ;      name:
      ;      where:      BINTRP
      ;      purpose:
      ;
FF34  (H.NOTR,5)
      ;      name:
      ;      where:      BINTRP
      ;      purpose:
      ;
FF39  (H.SNGF,5)
      ;      name:
      ;      where:      BINTRP
      ;      purpose:
      ;
FF3E  (H.NEWS,5)
      ;      name:
      ;      where:      BINTRP
      ;      purpose:
      ;
```



```
FF43 (H.GONE,5)
;      name:
;      where:      BINTRP
;      purpose:
;
FF48 (H.CHRG,5)
;      name:
;      where:      BINTRP
;      purpose:
;
FF4D (H.RETU,5)
;      name:
;      where:      BINTRP
;      purpose:
;
FF52 (H.PRTE,5)
;      name:
;      where:      BINTRP
;      purpose:
;
FF57 (H.COMP,5)
;      name:
;      where:      BINTRP
;      purpose:
;
FF5C (H.FINP,5)
;      name:
;      where:      BINTRP
;      purpose:
;
FF61 (H.TRMN,5)
;      name:
;      where:      BINTRP
;      purpose:
;
FF66 (H.FRME,5)
;      name:
```

```

;      where:      BINTRP
;      purpose:
;
FF6B (H.NTPL,5)
;      name:
;      where:      BINTRP
;      purpose:
;
FF70 (H.EVAL,5)
;      name:
;      where:      BINTRP
;      purpose:
;
FF75 (H.OKNO,5)
;      name:
;      where:      BINTRP
;      purpose:
;
FF7A (H.FING,5)
;      name:      H.ISMI
;      where:      BINTRP, at the ISMID$ (IS MID$) routine
;      purpose:
;
FF7F (H.ISMI,5)
;      name:      H.WIDT
;      where:      BINTRP, at the WIDTHS (WIDTH) routine
;      purpose:
;
FF84 (H.WIDT,5)
;      name:      H.LIST
;      where:      BINTRP, at the LIST routine
;      purpose:
;
FF89 (H.LIST,5)
;      name:      H.BUFL
;      where:      BINTRP, at the BUFLIN (BUffer LIne) routine
;      purpose:

```

```

;
FF8E (H.BUFL,5)
;   name:      H.FRQI
;   where:     BINTRP, at the FRQINT routine
;   purpose:
;
FF93 (H.FRQI,5)
;   name:
;   where:     BINTRP
;   purpose:
;
FF98 (H.SCNE,5)
;   name:      H.FRET
;   where:     BISTRP, at the FRETMP (FREe up TeMPoraries)
;              routine
;   purpose:
;
FF9D (H.FRET,5)
;   name:      H.PTRG
;   where:     BIPTRG, at the PTRGET (PoinTeR GET) routine
;   purpose:   to use other variable names than default
;
FFA2 (H.PTRG,5)
;   name:      H.PHYD
;   where:     MSXIO, at the PHYDIO (PHYSical Disk I/O) routine
;   purpose:   to install disk driver
;
FFA7 (H.PHYD,5)
;   name:      H.FORM
;   where:     MSXIO, at the FORMAT (disk FORMATter) routine
;   purpose:   to install disk driver
;
FFAC (H.FORM,5)
;   name:      H.ERRO
;   where:     BINTRP, at the ERROR routine
;   purpose:   to trap errors from application programs
;

```

```

FFB1  (H.ERRO,5)
      ;      name:      H.LPTO
      ;      where:     MSXIO, at the LPTOUT (Line PrinTer OUTput)
      ;                  routine
      ;      purpose:    to use other printer than default
      ;
FFB6  (H.LPTO,5)
      ;      name:      H.LPTS
      ;      where:     MSXIO, at the LPTSTT (Line PrinTer STaTus)
      ;                  routine
      ;      purpose:    to use other printer than default
      ;
FFBB  (H.LPTS,5)
      ;      name:      H.SCRE
      ;      where:     MSXSTS, at the entry to SCREEN statement.
      ;      purpose:    To expand SCREEN statement.
      ;
FFC0  (H.SCRE,5)
      ;      name:      H.PLAY
      ;      where:     MSXSTS, at the entry to PLAY statement.
      ;      purpose:    To expand PLAY statement.
      ;
FFC5  (H.PLAY,5)
      ;
FFCA  (ENDWRK,0)          ;end of work area

```

ISBN 0-933063-00-8