APPENDICES

Appendix A Error Codes and Error Messages

Error Code	Message

1

NEXT without FOR (English)
NEXT sans FOR (French)
NEXT ohne FOR (German)

A NEXT statement was encountered without a corresponding FOR statement.

Possible causes:

(1) Improperly nested FOR/NEXT loops (or variables specified in the wrong order in a common NEXT statement for loops that end at the same point).
(2) Variable in NEXT statement does not correspond to any previously executed FOR statement variable.
(3) More than one NEXT statement specified for one FOR statement.
(4) Execution branched to a point within a FOR/NEXT loop from elsewhere in the program.

2

Syntax error (English)
Erreur de syntaxe (French)
Syntaxfehler (German)

A statement does not conform to the syntax rules of MFBASIC.

Possible causes:

(1) Incorrectly typed keywords.
(2) Unmatched parentheses.
(3) Wrong delimiting punctuation (commas, periods, colons, or semicolons) used between statements, expressions, or arguments.
(4) Variable name beginning with other than a letter.
(5) Keyword used as the first letters of a variable name.
(6) Wrong number or type of arguments specified in a function or statement.
(7) Type of value included in a DATA statement did not match the corresponding variable in a READ statement's list of variables.

3

RETURN without GOSUB (English)
RETURN sans GOSUB (French)
RETURN ohne GOSUB (German)

A RETURN statement was encountered which did not correspond to a previously executed GOSUB statement.

Possible causes:

(1) Execution branched to a subroutine with a GOTO statement.
(2) Line number specified in a RUN command was a line in a subroutine.
(3) END statement was not included at the end of the main routine to keep execution from moving into a subroutine.

4 **Out of data (English)**
Plus de données (French)
Außerhald des datenberichs (German)

A READ statement was executed when there was no unread data remaining in the program's DATA statements.

Possible causes:

(1) Insufficient number of data items in DATA statement(s).
(2) Incorrect specification of a RESTORE statement.
(3) Incorrect delimiting punctuation used in a DATA statement.

5 **Illegal function call (English)**
Appel de fonction illégal (French)
Nicht erlaubter funktionsaufruf (German)

A statement or function was incorrectly specified.

Possible causes:

(1) Specification of a negative number or a number which is too large as an array variable subscript.
(2) Specification of zero or a negative number as the argument in the LOG function.
(3) Specification of a negative number as the argument of the SQR function.
(4) Specification of a non-integer exponent with a negative mantissa.
(5) A call to a USR function for which the starting address has not yet been defined.
(6) An incorrectly specified argument in any of the following functions or statements: MID$, LEFT$, RIGHT$, INP, OUT, WAIT, PEEK, POKE, TAB, SPC, STRING$, SPACE$, INSTR, ON...GOSUB, ON...GOTO, ASC, SCREEN, STYLE$, VARPTR.
(7) Specification of a non-existent line number in a DELETE statement.
(8) Attempting to erase a non-existent variable array with an ERASE statement.

(9) Attempting to edit a program loaded after it was SAVEd with the P option specified.
(10) Specifying coordinates in a PAINT statement which are outside the effective range of coordinates for the graphic screen.
(11) Execution of a RENUM command with parameters which do not conform to the rules for specifying such commands.
(12) Specification of an undefined array variable or a variable whose value has not yet been defined in a SWAP statement.
(13) Output of more than 1.75K bytes of data to the CMOS RAM file.

6 **Overflow (English)**
Dépassement de capacité (French)
Bereichsüberschreitung (German)

A numeric value was encountered whose magnitude exceeds the limits prescribed by MFBASIC (if underflow occurs, zero is assumed and execution continues without error).

Possible causes:

(1) The result of an integer calculation was outside the range from -32768 to 32767.
(2) The result of a single or double precision number calculation was outside the range from 1.70141E38 to -1.70141E38.
(3) One of the operands of a logical operation was not in the range from -32768 to 32767.
(4) The argument specified for the CINT function was outside the range from -32768 to 32767.
(5) The argument specified for the HEX$ or OCT$ function was outside the range from -32768 to 65535.

7 **Out of memory (English)**
Mémoire insuffisante (French)
Außerhab des speicherbereichs (German)

Memory available is insufficient for processing required.

Possible causes:

(1) Program is too long.
(2) The program uses too many variables.
(3) The subscript range specified in a DIM statement is too large.
(4) An expression has too many levels of parentheses.
(5) FOR...NEXT loops or GOSUB...RETURN sequences are nested to too many levels.

(6) The stack area size or machine language area specified in a CLEAR statement is too large.

8 **Undefined line number (English)**
Numéro de ligne inconnu (French)
Undefinierbare zeilennummer (German)

A non-existent line number was specified in one of the following commands or statements: GOTO, GOSUB, RESTORE, RUN, RENUM.

9 **Subscript out of range (English)**
Plus de place pour phrase (French)
Index ausserhalb des hereichs (German)

The subscript specified in a statement referencing an array element is outside the range permitted for that array.

Possible causes:

(1) Subscript specified was greater than the maximum specified in the DIM statement defining that array.
(2) Wrong number of subscripts specified in a statement referencing an array variable.
(3) A subscript greater than 10 was used without executing a DIM statement to define that array.
(4) Zero was used as a subscript after executing OPTION BASE 1.

10 **Duplicate definition (English)**
Double définition (French)
Doppelte definition (German)

A variable array was defined more than once.

Possible causes:

(1) A second DIM statement was executed for an array without erasing that array with an ERASE statement.
(2) An undeclared array was used, then an attempt was made to redimension that array with a DIM statement.
(3) The OPTION BASE statement was executed more than once, or was executed after an array had already been dimensioned (either by a DIM statement or implicitly by assignment of a value to a variable with a subscripted name).

11 **Division by zero (English)**
Division par zéro (French)
Division durch null (German)

An operation was encountered which included division by zero.

Possible causes:

(1) Zero was used as a divisor.
(2) Division was attempted using an undefined variable as a divisor.

12 **Illegal direct (English)**
Commande directe illégale (French)
Unerlaubter direkter modus (German)

A statement that is illegal in the direct mode (such as DEF FN) was entered as a direct mode command.

13 **Type mismatch (English)**
Non-concordance de type (French)
Schreibfehler (German)

A string expression was used where a numeric expression is required, or vice versa.

Possible causes:

(1) An attempt was made to assign a numeric value to a string variable.
(2) An attempt was made to assign a string to a numeric variable.
(3) The wrong type of value was specified as the argument of a function.

14 **Out of string space (English)**
Plus d'espace de phrases (French)
Außerhalb des zeichen bereichs (German)

Insufficient memory space is available for storage of characters in string variables.

15 **String too long (English)**
Phrase trop longue (French)
Zeichenkette zu lang (German)

An attempt was made to create a string whose length exceeds 255 characters.

16 **String formula too complex (English)**
Traitement de phrase trop complexe (French)
Zeichenkette zu komplex (German)

The complexity of a string operation is too great.

17 **Can't continue (English)**
Ne peut continuer (French)
Fortsetzung nicht möglich (German)

An attempt was made to resume execution of a program when continuation was not possible.

Possible causes:

(1) Program execution was terminated due to an error.
(2) The program was modified while execution was suspended.
(3) The BREAK key was pressed during execution of an INPUT statement.
(4) The program had not yet been executed.

18 **Undefined user function (English)**
Fonction utilisateur non définie (French)
Undefinierte usr funktion (German)

A call was made to an undefined user function.

Possible causes:

(1) The letters FN were used at the beginning of a variable name.
(2) The function name was specified incorrectly in the DEF FN statement or when the function was called.
(3) The user function was called before the corresponding DEF FN statement was executed.

19 **No RESUME (English)**
Pas de RESUME (French)
Kein RESUME (German)

No RESUME statement was included in an error processing routine. (All error processing routines must conclude with an END or RESUME statement.)

20 **RESUME without error (English)**
RESUME sans erreur (French)
RESUME ohne fehler (German)

A RESUME statement was encountered outside of an error processing routine.

Possible causes:

(1) Execution was branched to within an error processing routine by a GOTO or GOSUB statement.

(2) No END statement was included at the end of the main routine to keep execution from moving on into an error processing routine.

21 **Unprintable error (English)**
Erreur non définie (French)
Undruckbarer fehler (German)

No error message has been assigned to the error condition which exists; this message is also issued for error codes 31-49, 56, 59, 60, 65, and 72-255 (usually due to execution of an ERROR statement specifying one of these codes).

22 **Missing operand (English)**
Opérande manquant (French)
Fehlender operand (German)

An expression contains an operator without a following operand.

23 **Line buffer overflow (English)**
Dépassement de la ligne tampon (French)
Zeilensspeicher überschreitung (German)

An attempt was made to input a line that contains too many characters.

24 **Device time out (English)**
Désynchronisation sur E/S (French)
Gerätezeit aus (German)

An input or output device was not ready when an input or output operation was attempted.

Possible causes:

(1) Transmission via the RS-232C interface was not enabled within a certain period of time after an OPEN"O"statement was executed with auto enable or the DSR send check set to ON by option "c" of the communications format specification.

(2) The BREAK key was pressed while output to the RS-232C interface was being deferred for some reason.

(3) The DSR or CD line did not become high within a certain period of time after an OPEN"I" statement was executed with the DSR receive check or CD check set to ON by option "c" of the communications format specification.

(4) The printer was off-line when output to the printer was attempted.

25 **Device fault (English)**
Erreur sur E/S (French)
Falsches gerät (German)
The level of the signal on the DSR or CD line became low during input from the RS-232C interface after the DSR receive check or CD check had been set to ON (by option "c" of the communications format specification in the OPEN"I" statement executed to open the interface).

26 **FOR without NEXT (English)**
FOR sans NEXT (French)
FOR ohne NEXT (German)
A FOR statement was encountered without a corresponding NEXT.

27 **Out of paper (English)**
Plus de papier (French)
Kein Papier (German)
There is no paper in the printer or the printer is not connected.

28 **Communication buffer overflow (English)**
Dépassement de communication tampon (French)
Kommunikationsspeicher Überschreitung (German)
The receive buffer overflowed during receipt of data via an RS-232C interface. This error is likely to occur when the speed with which receive processing is performed is lower than that at which data is being received, but is unlikely if the communication rate is set to 1200 bps or less.

29 **WHILE without WEND (English)**
WHILE sans WEND (French)
WHILE ohne WEND (German)
A WHILE statement was encountered without a corresponding WEND.

30 **WEND without WHILE (English)**
WEND sans WHILE (French)
WEND ohne WHILE (German)
A WEND statement was encountered without a corresponding WHILE.

50 **FIELD overflow (English)**
Dépassement de champ de données (French)
Feldüberschreitung (German)
A FIELD statement attempted to allocate more bytes in a random file buffer than were specified for that buffer when the file was opened.

51 **Internal error (English)**
Erreur interne (French)
Interner fehler (German)
An internal malfunction occurred in MFBASIC.

52 **Bad file number (English)**
Numéro de fichier erroné (French)
Falsche dateinummer (German)
A statement or command references a file that has not been opened, or the file number specified in an OPEN statement is outside of the range of file numbers that was specified when MFBASIC was started.

53 **File not found (English)**
Fichier non trouvé (French)
Datei nicht auffindbar (German)
The file name specified in a LOAD, KILL, NAME, or OPEN statement does not exist on the disk in the accessed drive.

54 **Bad file mode (English)**
Mode de fichier incorrect (French)
Falscher dateimodus (German)
A statement or function was used with a file of the wrong type.

Possible causes:
(1) An attempt was made to use PUT, GET, or LOF with a sequential file.
(2) A random file was specified in a LOAD command.
(3) A file mode other than I, O, or R was specified in an OPEN statement.
(4) An attempt was made to MERGE a file that was not saved in ASCII format.

55 **File already open (English)**
Fichier déjà ouvert (French)
Datei bereits eröffnet (German)
An OPEN"O"statement was executed for a file which was already

open, or a KILL command was executed for a file that was open.

57 **Device I/O error (English)**
Erreur I/O de E/S (French)
Geräte I/O fehler (German)
An error occurred involving input or output to a peripheral device.

Possible causes:
(1) An I/O error occurred during a disk I/O operation. (This is a fatal error; i.e., one from which the operating system cannot recover.)
(2) A parity error, overrun error, or framing error occurred during input from an RS-232C interface. (In this case, the error condition will be reset if input is continued, but there is no assurance that data received will be normal.)
(3) The printer power was off or printer trouble occurred when data was output to the printer.

58 **File already exists (English)**
Fichier déjà existant (French)
Datei bereits vorhanden (German)
The new file name specified in a NAME statement is already being used with another file on the disk.

61 **Disk full (English)**
Disque plein (French)
Diskette voll (German)
All storage on the disk is in use.

62 **Input past end (English)**
Entrée après fin de fichier (French)
Eingabe nach ende (German)
An INPUT statement was executed for an empty file or after all data in the file had been read (to avoid this error, use the EOF function to detect the end of the file.) Or, the BREAK key was pressed while input from an RS-232C interface was pending with INPUT#, INPUT$, or the like.

63 **Bad record number (English)**
Numéro d'enregistrement erroné (French)
Falsche satznummer (German)
The record number specified in a PUT or GET statement was either zero or greater than the maximum allowed.

64 **Bad file descriptor (English)**
Fichier descripteur erroné (French)
Falscher dateiname (German)

An illegal file name was specified in a LOAD, SAVE, or KILL command or an OPEN statement (for example, a file name with too many characters).

66 **Direct statement in file (English)**
Commande directe dans fichier (French)
Direkte anweisung in der datei (German)

A program line without a line number was encountered during execution of a LOAD or MERGE command; or, an attempt was made to LOAD a data file or machine language program.

67 **Too many files (English)**
Trop de fichiers (French)
Zu viele dateien (German)

An attempt was made to create a new disk file after all 255 directory entries were full.

68 **Device unavailable (English)**
Périphérique non utilisable (French)
Geräte nicht verfügbar (German)

An attempt was made to open COMl: to COM4: when the corresponding option card was not installed. Or, an attempt was made to access a drive which did not contain a flexible disk.

69 **Disk write protect (English)**
Disque protégé en écriture (French)
Diskette schreibgeschützt (German)

An attempt was made to write data to a file for which the write protect attribute was set with the SET statement or STAT command of CP/M. Or, an attempt was made to write data to a disk which is write protected with a write protect tab.

70 **Disk read error (English)**
Erreur de lecture du disque (French)
Plattenlesungsfehler (German)

A read error occurred while data was being read from a flexible disk.

71 **Disk write error (English)**
Erreur d'écriture du disque (French)
Plattenschreibfehler (German)

A write error occurred while data was being written to a flexible disk.

Appendix B Table of Reserved Words

ABS
AND
ASC
ATN
ATTR$
AUTO
BASE
BEEP
BIT
CALL
CDBL
CHAIN
CHR$
CINT
CIRCLE
CLEAR
CLOSE
CLS
COLOR
COMMON
CONNECT
CONT
COPY
COS
COUNTRY
CSNG
CSRLIN
CURRENCY
CVD
CVI
CVS
DATA
DATE
DATE$
DAY
DEF
DEFDBL
DEFINT
DEFSNG
DEFSTR
DELETE
DIM
DSKF
EDIT
ELSE
END
EOF
EQV
ERASE
ERL
ERR
ERROR
EXP
FIELD
FILES
FIX
FN
FONT
FOR
FRE
GCURSOR
GET
GET@
GOTO, GOSUB
HEX$
IF
IMP
INKEY$
INP
INPUT
INPUT#
INPUT$
INSTR
INT
KEY
KILL
LEFT$
LEN
LET
LINE
LIST
LLIST
LOAD
LOC
LOCATE
LOF
LOG
LPOS
LPRINT
LSET
MERGE
MID$
MKD$
MKI$
MKS$
MOD
NAME
NEW
NEXT
NOT
OCT$
OFF
ON
OPEN
OPTION
OR
OUT
PAINT
PEEK
PEN
POINT
POKE
POS
PRESET
PRINT
PRINT#
PSET
PUT
PUT@
RANDOMIZE
READ
REM
RENUM
RESET
RESTORE
RESUME
RETURN
RIGHT$
RND
RSET
RUN
SAVE
SCREEN
SET
SGN
SIN
SOUND
SPACE$
SPC
SQR
STEP
STOP
STR$
STRING$
STYLE
STYLE$
SWAP
SYSTEM
TAB
TAN
THEN
TIME
TIME$
TO
TROFF
TRON
USING
USR
VAL
VARPTR
WAIT
WEND
WHILE
WIDTH
WRITE
WRITE#
XOR

Appendix C MFBASIC Console Escape Sequences

Control code	Function
ESC "("	No operation
ESC ")"	No operation
ESC "*"	Clear screen
ESC "0"	Reverse on
ESC "1"	Reverse off
ESC "2"	Cursor off
ESC "3"	Cursor on
ESC "4"	Underline
ESC "5"	Underline off
ESC "<"	Push cursor position
ESC "="	Set cursor position
ESC ">"	Pop cursor position
ESC "C"	Change character set & keyboard
ESC "F"	Change character style
ESC "L"	Change CRT color
ESC "P"	Screen dump
ESC "T"	Erase end of line
ESC "Y"	Erase end of screen
ESC 123	Secret
ESC 125	Non secret
ESC 160	INS LED on
ESC 161	INS LED off
ESC 176	Function key check mode on
ESC 177	Function key check mode off

ESC "("
No operation.

ESC ")"
No operation.

ESC "*"
This escape sequence clears the CRT screen to the background color and positions the cursor in the upper left hand corner of the screen.

ESC "0"
This escape sequence reverses the display.

ESC "1"
This escape sequence terminates reverse display.

ESC "2"
This escape sequence turns off the cursor.

ESC "3"
This escape sequence turns on the cursor.

ESC "4"
This escape sequence causes characters input to be underlined.

ESC "5"
This escape sequence terminates character underlining.

ESC "<"
This escape sequence saves the cursor position, display attributes (reverse, underline, highlight, etc.) and color and, when the first byte of a 2-byte character has already been sent to the console, saves that byte. Saving is possible up to 8 levels; further entries of the sequence are invalid.

ESC "="
This escape sequence makes it possible to specify the cursor position; this is done as follows.

```
PRINT CHR$ (27) ; "=" ; CHR$ (m+32) ; CHR$ (n+32)
```

m specifies the vertical cursor position and n specifies the horizontal position. Possible values of m and n are as shown below.

m: 0 to 19
n: 0 to 79 for the MFBASIC WIDTH80 mode
0 to 39 for the MFBASIC WIDTH40 mode

ESC ">"

This escape sequence restores the cursor position, attribute, color, and first byte of a 2-byte character most recently saved with ESC "<.

ESC "C"

This escape sequence is used to select one of the eight international character sets and keyboard arrangements as follows.

US ASCII	PRINT CHR$ (27); "CU" or "Cu"
French	PRINT CHR$ (27); "CF" or "Cf"
German	PRINT CHR$ (27); "CG" or "Cg"
English	PRINT CHR$ (27); "CE" or "Ce"
Danish	PRINT CHR$ (27); "CD" or "Cd"
Swedish	PRINT CHR$ (27); "CW" or "Cw"
Italian	PRINT CHR$ (27); "CI" or "Ci"
Spanish	PRINT CHR$ (27); "CS" or "Cs"

ESC "F"

This escape sequence changes the font of 1-byte characters in the MFBASIC WIDTH40 mode. This is specified as follows.

```
PRINT CHR$ (27) ; "Fn"
```

Here, n is a hexadecimal number from 0 to F. (This escape sequence has no effect on 2-byte characters.)

Example

```
10 PRINT CHR$(27);"F8"
20 PRINT "abcEPSONdef"
RUN

abcEPSONdef
Ok
```

ESC "L"

This escape sequence changes the foreground or background color of the CRT display. The sequence for changing the foreground color is as follows.

```
PRINT CHR$ (27) ; "L0n"
```

Here, n is an integer with the following meanings.

n	0	1	2	3	4	5	6	7
	Black	Blue	Red	Violet	Green	Light blue	Yellow	White

NOTE:
In the case of the monochrome display, specification of any number other than 0 (black) for n is regarded as equivalent to n = 7 (white).

For the background color, the sequence is as follows.

```
PRINT CHR$ (27) ; "L1n"
```

Here, n is an integer from 0 to 7 with meanings which are the same as indicated above.

NOTE:
This sequence is not effective in the case of the monochrome display (the background color is always black).

ESC "P"

This escape sequence outputs a hard copy of the display screen contents to the printer. This escape sequence is not effective with some combinations of mode, CRT and printer. When a color display is used, colors other than background color are printed.

ESC "T"

This escape sequence clears the current line from the position of the cursor to the line's end.

ESC "Y"

This escape sequence clears the current screen from the position of the cursor to the screen's end.

ESC CHR$ (123)

This escape sequence causes all characters to be displayed on the screen as blanks (secret mode).

ESC CHR$ (125)

This escape sequence terminates the secret mode.

ESC CHR$ (160)
This escape sequence lights the LED built into the INS key.

ESC CHR$ (161)
This escape sequence turns off the LED built into the INS key.

ESC CHR$ (176)
This escape sequence sets the console in the function key check mode (the mode in which a special check is made of the programmable function keys and numeric pad keys) and turns on FUNCFLG ((&HFED1) = &HFF).
In this mode, the internal code corresponding to each key is entered as it is pressed as follows.

	Key	Internal code
Programmable function keys	PFK1	&HE0
	PFK2	&HE1
	PFK3	&HE2
	PFK4	&HE3
	PFK5	&HE4
	PFK6	&HE5
	PFK7	&HE6
	PFK8	&HE7
	PFK9	&HE8
	PFK10	&HE9
	00	&HF4
	(German keyboard only) 000	&HF5
Numeric pad keys	↵	Same as ASCII code
	*	Same as ASCII code
	+	Same as ASCII code
	, or △	Same as ASCII code
	—	Same as ASCII code
	.	Same as ASCII code
	/	Same as ASCII code
	0 to 9	Same as ASCII code
		Other keys are same as ASCII code 0

ESC CHR$ (177)
This escape sequence resets the function key check mode ((&HFED1) = 0).
The programmable function keys, the "00" key and the "000" key generate more than one code in sequence. For example, the "000" key generates three zeros in sequence.

Appendix D Proportional Printing

Proportional printing is possible when QX-10 MFBASIC is used with most EPSON printers. During proportional printing, the TAB function manages the print head position on a dot basis. The head position is determined as follows when TAB(n) is specified.

DP = 12 * (n-1)

Here, DP is the print position for double density printing, starting with 0 at the left side.

When using the TAB function during output of 2-byte characters, the position of the head is managed on a dot basis regardless of whether proportional printing is being performed.

Example

```
10 LPRINT CHR$(27);"a";:GOSUB 50
20 LPRINT
30 LPRINT CHR$(27);"b";:GOSUB 80
40 END
50 LPRINT "The quick brown fox jumped"
60 LPRINT "over the lazy yellow dog."
70 RETURN
80 LPRINT "The quick brown fox jumped over the"
90 LPRINT "lazy yellow dog."
100 RETURN
```

```
RUN
The quick brown fox jumped
over the lazy yellow dog.

The quick brown fox jumped over the
lazy yellow dog.
```

Appendix E Machine Language Subroutines

The CALL and USR statements of MFBASIC make it possible to execute machine language subroutines from programs written in MFBASIC. Such subroutines must be written into memory in machine language with the POKE statement before they can be called. (It is also possible to use the MACRO-80 assembler and LINK-80 linker/loader to assemble and load routines written in assembly language; however these programs are not provided on the MF CP/M diskette.) See any of the various handbooks available on the Z80 microcomputer or the Z80 assembly language for the Z80 instruction code set.

When preparing machine language subroutines, remember that the presence of even a single error in the machine code is likely to result in destruction of all data included in the QX-10's memory (including MFBASIC itself). Therefore, be sure to back up all data and programs in memory on a disk before attempting to test or debug such routines.

1. Memory Allocation

Memory space must be reserved for storage of the instruction codes of machine language subroutines before they can be written into memory with the POKE statement. This is done using the CLEAR statement of MFBASIC or the /M: option of the MFBASIC command. When using the /M: option, the starting address of the machine language area is the address specified, and the ending address is equal to &HE400 (the starting address of MFBASIC working area (2) minus the size of the stack area specified in the /T: option. (See the figure below.) Remember that the size of the memory area reserved must be no larger than &HE400 minus the stack area size specified in the /T: option.

When a machine language subroutine is called, the stack pointer is set up for 8 levels (16 bytes) of stack storage. If more stack space is required, MFBASIC's stack can be saved and a new stack set up for use by the machine language subroutine. MFBASIC's stack must be restored, however, before returning from the subroutine.

2. USR Function Calls

With MFBASIC, the format used for calling USR functions is as follows.

USR[<digit>](argument)

Here, <digit> is a number from 0 to 9 and the argument specified is any numeric or string expression. <digit> specifies which of the USR routines is being called, and corresponds to the digit specified in the USF USR statement for that routine. If <digit> is omitted, USR0 is assumed. The address specified in the DEF USR statement determines the starting address of the subroutine.

When a USR function call is made, a value is placed in CPU register A which specifies the type of argument specified. The value placed in A may be any of the following.

Value	Type of argument
2	Two-byte integer (two's complement)
3	String
4	Single precision floating point number
8	Double precision floating point number

If the argument is a number, the HL register pair points to the Floating Point Accumulator (FAC) where the argument is stored.

If the argument is an integer, FAC-3 contains the lower 8 bits of the argument and FAC-2 contains the upper 8 bits. If the argument is a single precision floating point number, FAC-3 contains the lowest 8 bits of the mantissa, FAC-2 contains the middle 8 bits of the mantissa, and FAC-1 contains the highest 7 bits of the mantissa (with leading 1 suppressed). Bit 7 is the sign of the number (0 for positive and 1 for negative), and FAC is the exponent minus 128. The binary point is the bit to the left of the most significant bit of the mantissa.

If the argument is a double precision floating point number, FAC-7 to FAC-4 contain four more bytes of the mantissa (with the lowest 8 bits in FAC-7).

If the argument is a string, the DE register pair points to three bytes called the "string descriptor." Byte 0 of the string descriptor contains the length of the string (0 to 255); and bytes 1 and 2, respectively, are the lower and upper 8 bits of the starting address of the string in string space.

CAUTION: If the argument is a string literal in the program, the string descriptor will point to program text. Be careful not to alter or destroy your program in this way. To avoid unpredictable results, add +"" to the string literal in the program.

Example A$ = "MFBASIC" + ""

This will copy the string literal into string space and prevent alteration of program text during a subroutine call.

3. CALL Statement

MFBASIC user function calls may also be made with the CALL statement. A CALL statement with no arguments generates a simple "CALL" instruction. The corresponding subroutine should return to the MFBASIC program via a simple "RET" instruction. ("CALL" and "RET" are Z80 opcodes; see a Z80 reference manual for details.)

A subroutine CALL with arguments results in a somewhat more complex calling sequence. For each argument in the CALL argument list, a parameter is passed to the subroutine. That parameter is the address of the low byte of the argument. Therefore, parameters always occupy two bytes each, regardless of type.

The method of passing parameters depends on the number of parameters to be passed as follows.

(1) If the number of parameters is less than or equal to 3, they are passed in the registers. Parameter 1 will be in HL, 2 (if present) in DE, and 3 (if present) in BC.
(2) If the number of parameters is greater than 3, they are passed as follows.
    (a) Parameter 1 in HL.
    (b) Parameter 2 in DE.
    (c) Parameters 3 through n in a contiguous data block. Register pair BC will point to the low byte of this data block (i.e., to the low byte of parameter 3).

Note that, with this scheme, the subroutine must know how many parameters to expect in order to find them. Conversely, the calling program is responsible for passing the correct number of parameters. There are no checks for correct number or type of parameters.

When accessing parameters in a subroutine, don't forget that they are pointers to the actual arguments passed.

NOTE:
It is entirely up to the programmer to ensure that arguments in the calling program match those expected by the subroutine in number, type, and length. This applies to MFBASIC subroutines, as well as those written in machine language.

4. Interrupts

Machine language subroutines can be written to handle interrupts. All interrupt handling routines should save the stack, registers A to L, and the PSW. Since an interrupt received automatically disables all further interrupts, interrupts should always be reenabled before returning from the subroutine. (The user should note that, with MFBASIC, all interrupt vectors are free.)

Appendix F Derived Functions

Functions that are not intrinsic to MFBASIC may be calculated as follows.

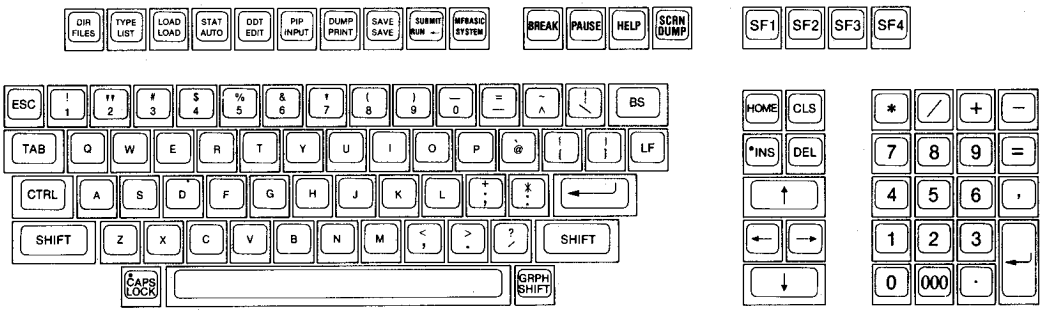
Function	MFBASIC Equivalent
SECANT	SEC(X) = 1/COS(X)
COSECANT	CSC(X) = 1/SIN(X)
COTANGENT	COT(X) = 1/TAN(X)
INVERSE SINE	ARCSIN(X) = ATN(X/SQR(1 − X * X + 1))
INVERSE COSINE	ARCCOS(X) = − ATN(X/SQR(1 − X * X)) + 1.570796326794897
INVERSE SECANT	ARCSEC(X) = ATN(SQR(X * X − 1)) + (SGN(X) − 1) * 1.570796326794897
INVERSE COSECANT	ARCCSC(X) = ATN(1/SQR(X * X − 1)) + (SGN(X) − 1) * 1.570796326794897
INVERSE COTANGENT	ARCCOT(X) = − ATN(X) + 1.570796326794897
HYPERBOLIC SINE	SINH(X) = (EXP(X) − EXP(− X))/2
HYPERBOLIC COSINE	COSH(X) = (EXP(X) + EXP(− X))/2
HYPERBOLIC TANGENT	TANH(X) = (EXP(X) − EXP(− X))/(EXP(X) + EXP(− X))
HYPERBOLIC SECANT	SECH(X) = 2/(EXP(X) + EXP(− X))
HYPERBOLIC COSECANT	CSCH(X) = 2/(EXP(X) − EXP(− X))
HYPERBOLIC COTANGENT	COTH(X) = (EXP(X) + EXP(− X))/(EXP(X) − EXP(− X))
INVERSE HYPERBOLIC SINE	ARCSINH(X) = LOG(X + SQR(X * X + 1))
INVERSE HYPERBOLIC COSINE	ARCCOSH(X) = LOG(X + SQR(X * X − 1)
INVERSE HYPERBOLIC TANGENT	ARCTANH(X) = LOG((1 + X)/(1 − X))/2
INVERSE HYPERBOLIC SECANT	ARCSECH(X) = LOG((SQR(1 − X * X) + 1)/X)
INVERSE HYPERBOLIC CASECANT	ARCCSCH(X) = LOG((1 + SGN(X) * SQR(1 + X * X))/X
INVERSE HYPERBOLIC COTANGENT	ARCCOTH(X) = LOG((X + 1)/(X − 1))/2

Any of these functions can easily be used in a program by defining it with a DEF FN statement. This is illustrated in the example below.

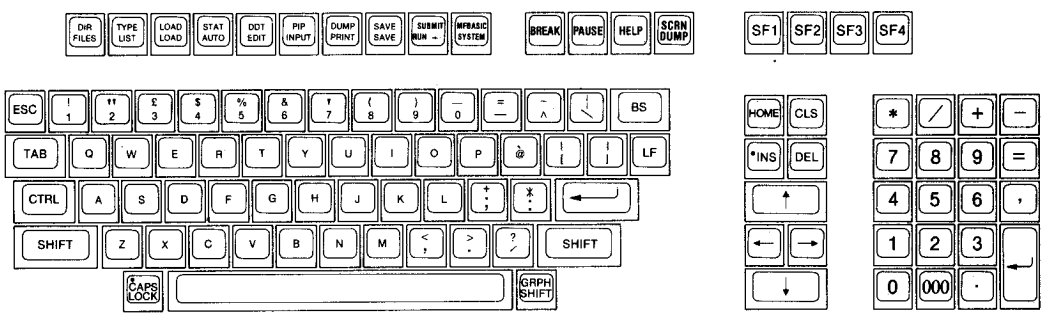
Example

Function definition: DEF FN SINH(X) = (EXP(X)-(EXP(X))/2
Function call: A = FNSINH(Y)

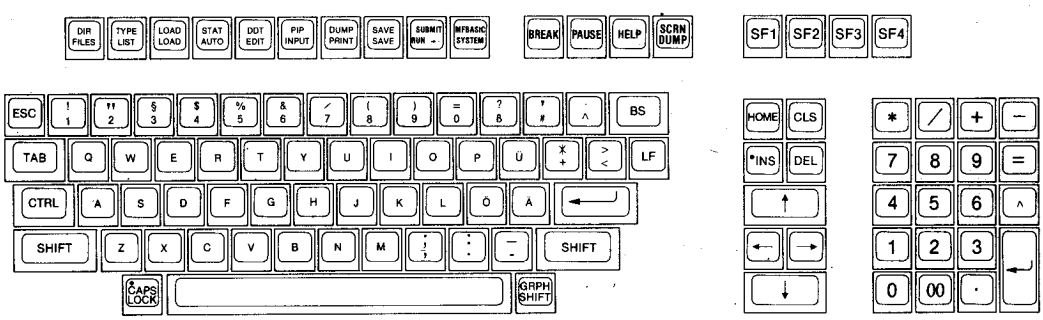
Appendix G Keyboard Arrangements



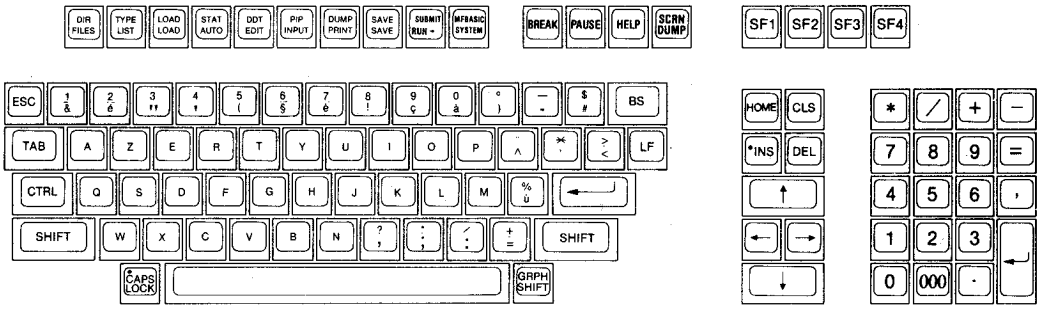
USASCII Keyboard



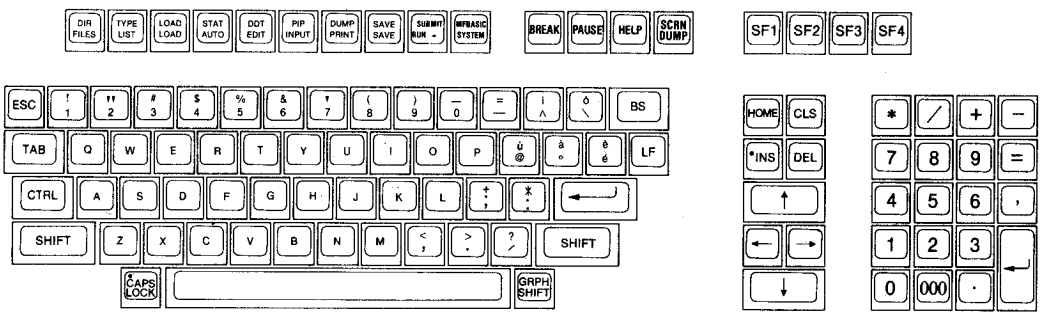
English Keyboard



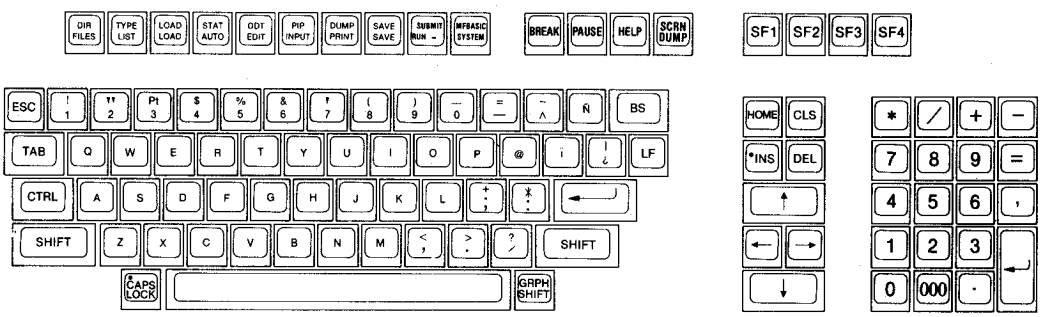
German Keyboard



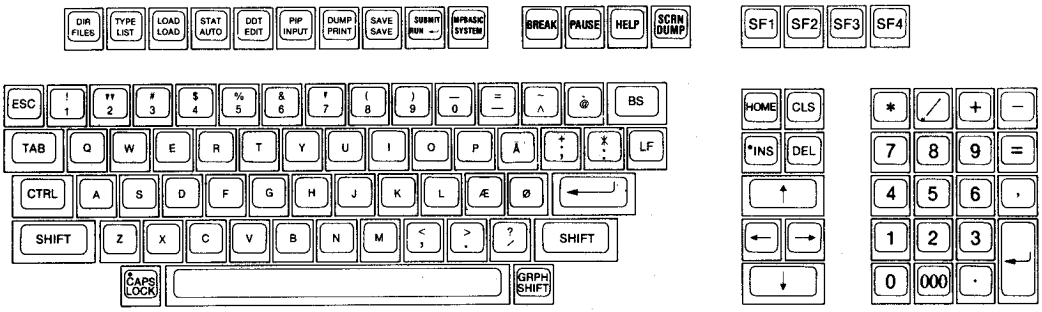
French Keyboard



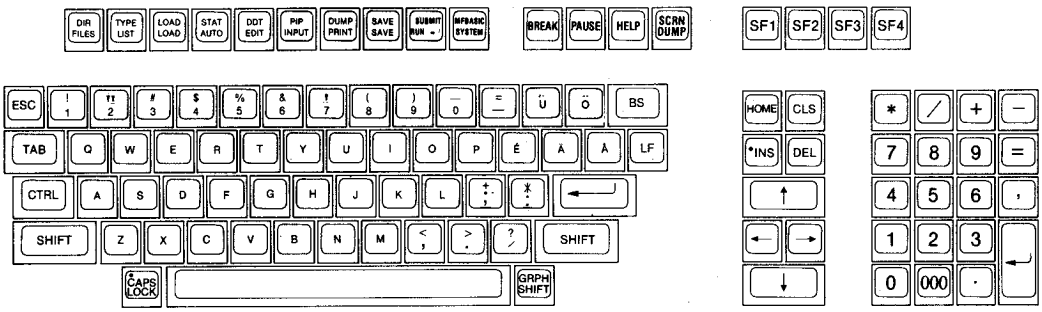
Italian Keyboard



Spanish Keyboard



Danish Keyboard



Swedish Keyboard

Appendix H ASCII Character Codes

Hex. No.	Binary No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0000	0	16	SPACE 32	0 48	@ 64	P 80	` 96	p 112	+ 128	○ 144						
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	⊥ 129	♠ 145						
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	⊤ 130	♥ 146						
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	◆ 147						
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148						
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149						
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	┼ 134	◘ 150						
7	0111	BEL 7	23	' 39	7 55	G 71	W 87	g 103	w 119	├ 135	◘ 151						
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	◘ 152						
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	◘ 153						
A	1010	LF 10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	◘ 154						
B	1011	11	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123	┌ 139	◘ 155						
C	1110	CLS 12	→ 28	, 44	< 60	L 76	\ 92	l 108	¦ 124	■ 140	◘ 156						
D	1101	← 13	← 29	− 45	= 61	M 77	] 93	m 109	} 125	▬ 141	◘ 157						
E	1110	14	↑ 30	. 46	> 62	N 78	^ 94	n 110	~ 126	▮ 142	÷ 158						
F	1111	15	↓ 31	/ 47	? 63	O 79	_ 95	o 111	△ 127	● 143	± 159						

Graphic symbols
Multiple fonts
User-defined characters

USASCII in MFBASIC Mode

NOTES:

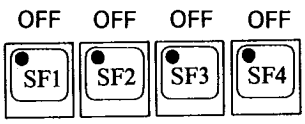
1. *$(0)_D$ through $(31)_D$ are control characters.*
2. *$(32)_D$ through $(127)_D$ are ASCII characters.*
3. *$(128)_D$ through $(159)_D$ are graphic symbols.*
4. *$(160)_D$ through $(255)_D$ are multiple font characters.*
5. *Reverse display is used for some of the graphic symbols shown in the table above, depending on whether the symbols are displayed in the WIDTH 80 mode or the WIDTH 40 mode.*

Appendix I Style Selection

Characters printed or displayed are as shown at right when the style selection keys are set as shown at left.

Style Selection

0:

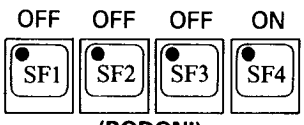


(OCR B-FONT)

```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | }
```

(Although OCR B-FONT characters can be entered from the keyboard in the non-MFBASIC mode, they cannot be entered using the style selection keys in the MFBASIC mode.)

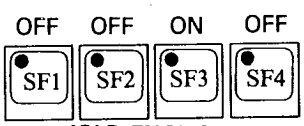
1:



(BODONI)

```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ Δ
```

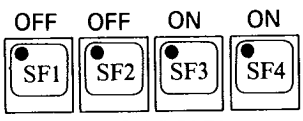
2:



(OLD ENGLISH)

```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ Δ
```

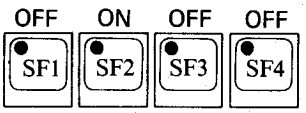
3:



(FLASH BOLD)

```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ Δ
```

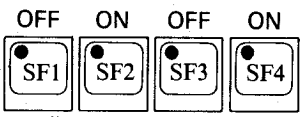
4:



(COMMERCIAL SCRIPT)

```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ Δ
```

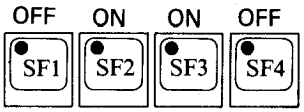
5:



(HELVETICA LIGHT)

```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ Δ
```

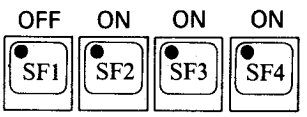
6:



(HELVETICA LIGHT ITALIC)

```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ Δ
```

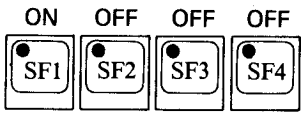
7:



(HELVETICA MEDIUM ITALIC)

```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ Δ
```

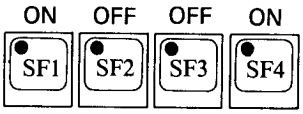
8:



(BROADWAY)

```
 !"#$%&' ()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~Δ
```

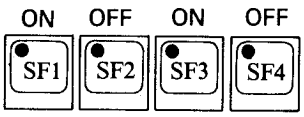
9:



(AMERICAN TYPEWRITER MEDIUM)

```
 !"#$%&' ()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~Δ
```

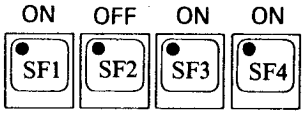
A:



(LIGHT ITALIC)

```
 !"#$%&' ()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~Δ
```

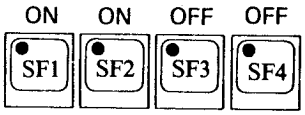
B:



(HELVETICA MEDIUM)

```
 !"#$%&' ()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~Δ
```

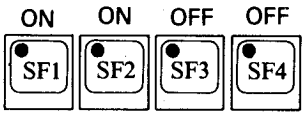
C:



(BODONI ITALIC)

```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ Δ
```

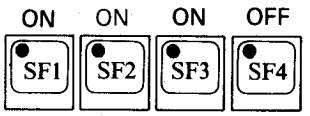
D:



(SANS SERIF SHADED)

```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ Δ
```

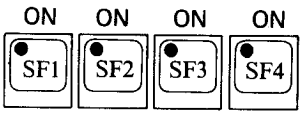
E:



(MICROGRAMA EXTENDED)

```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ Δ
```

F:



(OLD GERMANY)

```
 ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [ \ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ Δ
```

Appendix J MultiFonts/International Character Sets

International Character Sets

The QX-10 provides international character sets for the following eight countries:

United States	England
Germany	France
Italy	Spain
Denmark	Sweden

The characters displayed or printed for the above countries are as follows.

Country / Dec. code	U.S.A.	France	Germany	England	Denmark	Sweden	Italy	Spain
35 (23)	#	#	#	£	#	#	#	Pt
36 (24)	$	$	$	$	$	¤	$	$
64 (40)	@	à	§	@	@	É	@	@
91 (5B)	[	°	Ä	[	Æ	Ä	°	¡
92 (5C)	\	ç	Ö	\	Ø	Ö	\	Ñ
93 (5D)	]	§	Ü	]	Å	Å	é	¿
94 (5E)	^	^	^	^	^	Ü	^	^
96 (60)	`	`	`	`	`	é	ù	`
123 (7B)	{	é	ä	{	æ	ä	à	¨
124 (7C)	¦	ù	ö		ø	ö	ò	ñ
125 (7D)	}	è	ü	}	å	å	è	{
126 (7E)	~	¨	ß	~	~	ü	ì	~

NOTE: Numbers in parentheses are hexadecimal codes.

NOTES:

1) *Numbers in parentheses are apply to the non-MFBASIC mode only.*
2) *In the WIDTH 80 or WIDTH 40 mode of MFBASIC, certain characters are blank when option style 16 (OCR B font) is selected.*
3) *Depending on the international character set selected, certain characters are blank when option style 15 (Old Germany) is selected.*

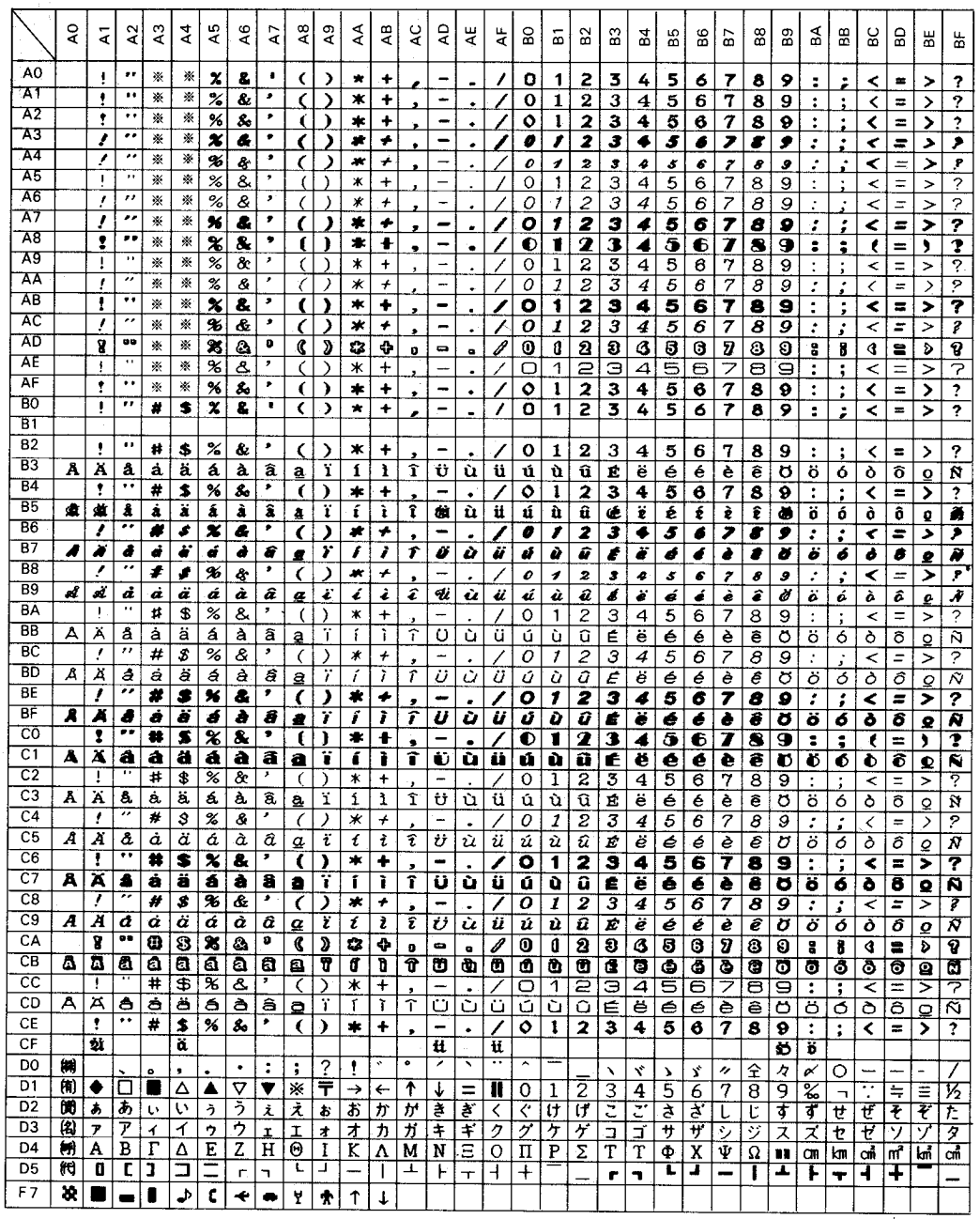
Country Hex. code (Dec code)	U.S.A.	France	Germany	England	Denmark	Sweden	Italy	Spain
A0A3 (160-163)	#	#	#	£	#	#	#	
A0A4 (160-164)	$	$	$	$	$	¤	$	$
A0C0 (160-192)	@		§	@	@		@	@
A0DB (160-219)	[			[				
A0DC (160-220)	\			\			\	
A0DD (160-221)	]	§		]				
A0DE (160-222)	^	^	^	^	^		^	^
A0E0 (160-224)	`	`	`	`	`			`
A0FB (160-251)	{			{				
A0FC (160-252)								
A0FD (160-253)	}			}				}
A0FE (160-254)								
A1A3 (161-163)	#	#	#	£	#	#	#	Pt
A1A4 (161-164)	$	$	$	$	$	¤	$	$
A1C0 (161-192)	@	à	§	@	@	É	@	@
A1DB (161-219)	[	°	Ä	[	Æ	Ä	°	¡
A1DC (161-220)	\	Ç	Ö	\	Ø	Ö	\	Ñ
A1DD (161-221)	]	§	Ü	]	Å	Å	é	¿
A1DE (161-222)	^	^	^	^	^	Ü	^	^
A1E0 (161-224)	`	`	`	`	`	é	ù	`
A1FB (161-251)	{	é	ä	{	æ	ä	à	¨
A1FC (161-252)	¦	ù	ö	¦	ø	ö	ò	ñ
A1FD (161-253)	}	è	ü	}	å	å	è	}
A1FE (161-254)	~	¨	ß	~	~	ü	ì	~
A2A3 (162-163)	#	#	#	£	#	#	#	
A2A4 (162-164)	$	$	$	$	$	¤	$	$
A2C0 (162-192)	@	à	§	@	@	É	@	@
A2DB (162-219)	[	°		[			°	¡
A2DC (162-220)	\	Ç		\	Ø		\	
A2DD (162-221)	]	§		]			é	¿
A2DE (162-222)	^	^	^	^	^		^	^
A2E0 (162-224)	`	`	`	`	`	é	ù	`
A2FB (162-251)	{	é	ä	{	æ	ä	à	¨
A2FC (162-252)	¦	ù	ö	¦	ø	ö	ò	ñ
A2FD (162-253)	}	è	ü	}	å	å	è	}
A2FE (162-254)	~	¨	ß	~	~	ü	ì	~
A3A3 (163-163)	#	#	#	£	#	#	#	
A3A4 (163-164)	$	$	$	$	$	¤	$	$
A3C0 (163-192)	@	à	§	@	@	É	@	@
A3DB (163-219)	[	°	Ä	[	Æ	Ä	°	¡
A3DC (163-220)	\	Ç	Ö	\	Ø	Ö	\	Ñ
A3DD (163-221)	]	§	Ü	]	Å	Å	é	¿
A3DE (163-222)	^	^	^	^	^	Ü	^	^
A3E0 (163-224)	`	`	`	`	`	é	ù	`
A3FB (163-251)	{	é	ä	{	æ	ä	à	¨
A3FC (163-252)	¦	ù	ö	¦	ø	ö	ò	ñ
A3FD (163-253)	}	è	ü	}	å	å	è	}
A3FE (163-254)	~	¨	ß	~	~	ü	ì	~

Country Hex. code (Dec code)	U.S.A.	France	Germany	England	Denmark	Sweden	Italy	Spain
A4A3 (164-163)								
A4A4 (164-164)								
A4C0 (164-192)								
A4DB (164-219)								
A4DC (164-220)								
A4DD (164-221)								
A4DE (164-222)								
A4E0 (164-224)								
A4FB (164-251)								
A4FC (164-252)								
A4FD (164-253)								
A4FE (164-254)								
A5A3 (165-163)	#	#	#	£	#	#	#	Pt
A5A4 (165-164)	$	$	$	$	$	¤	$	$
A5C0 (165-192)	@	à	§	@	@	É	@	@
A5C0 (165-219)	[	°	Ä	[	Æ	Ä	°	¡
A5DC (165-220)	\	Ç	Ö	\	Ø	Ö	\	Ñ
A5DD (165-221)	]	§	Ü	]	Å	Å	é	¿
A5DE (165-222)	^	^	^	^	^	Ü	^	^
A5E0 (165-224)	`	`	`	`	`	é	ù	`
A5FB (165-251)	{	é	ä	{	æ	ä	à	¨
A5FC (165-252)	¦	ù	ö	¦	ø	ö	ò	ñ
A5FD (165-253)	}	è	ü	}	å	å	è	}
A5FE (165-254)	~	¨	ß	~	~	ü	ì	~
A6A3 (166-163)	#	#	#	£	#	#	#	Pt
A6A4 (166-164)	$	$	$	$	$	¤	$	$
A6C0 (166-192)	@	à	§	@	@	É	@	@
A6DB (166-219)	[	°	Ä	[	Æ	Ä	°	¡
A6DC (166-220)	\	Ç	Ö	\	Ø	Ö	\	Ñ
A6DD (166-221)	]	§	Ü	]	Å	Å	é	¿
A6DE (166-222)	^	^	^	^	^	Ü	^	^
A6E0 (166-224)	`	`	`	`	`	é	ù	`
A6FB (166-251)	{	é	ä	{	æ	ä	à	¨
A6FC (166-252)	¦	ù	ö	¦	ø	ö	ò	ñ
A6FD (166-253)	}	è	ü	}	å	å	è	}
A6FE (166-254)	~	¨	ß	~	~	ü	ì	~
A7A3 (167-163)	#	#	#	£	#	#	#	Pt
A7A4 (167-164)	$	$	$	$	$	¤	$	$
A7C0 (167-192)	@	à	§	@	@	É	@	@
A7DB (167-219)	[	°	Ä	[	Æ	Ä	°	¡
A7DC (167-220)	\	Ç	Ö	\	Ø	Ö	\	Ñ
A7DD (167-221)	]	§	Ü	]	Å	Å	é	¿
A7DE (167-222)	^	^	^	^	^	Ü	^	^
A7E0 (167-224)	`	`	`	`	`	é	ù	`
A7FB (167-251)	{	é	ä	{	æ	ä	à	¨
A7FC (167-252)	¦	ù	ö	¦	ø	ö	ò	ñ
A7FD (167-253)	}	è	ü	}	å	å	è	}
A7FE (167-254)	~	¨	ß	~	~	ü	ì	~

Country Hex. code (Dec code)	U.S.A.	France	Germany	England	Denmark	Sweden	Italy	Spain
A8A3 (168-163)								
A8A4 (168-164)								
A8C0 (168-192)								
A8DB (168-219)								
A8DC (168-220)								
A8DD (168-221)								
A8DE (168-222)								
A8E0 (168-224)								
A8FB (168-251)								
A8FC (168-252)								
A8FD (168-253)								
A8FE (168-254)								
A9A3 (169-163)								
A9A4 (169-164)								
A9C0 (169-192)								
A9DB (169-219)								
A9DC (169-220)								
A9DD (169-221)								
A9DE (169-222)								
A9E0 (169-224)								
A9FB (169-251)								
A9FC (169-252)								
A9FD (169-253)								
A9FE (169-254)								
AAA3 (170-163)								
AAA4 (170-164)								
AAC0 (170-192)								
AADB (170-219)								
AADC (170-220)								
AADD (170-221)								
AADE (170-222)								
AAE0 (170-224)								
AAFB (170-251)								
AAFC (170-252)								
AAFD (170-253)								
AAFE (170-254)								
ABA3 (171-163)								
ABA4 (171-164)								
ABC0 (171-192)								
ABDB (171-219)								
ABDC (171-220)								
ABDD (171-221)								
ABDE (171-222)								
ABE0 (171-224)								
ABFB (171-251)								
ABFC (171-252)								
ABFD (171-253)								
ABFE (171-254)								

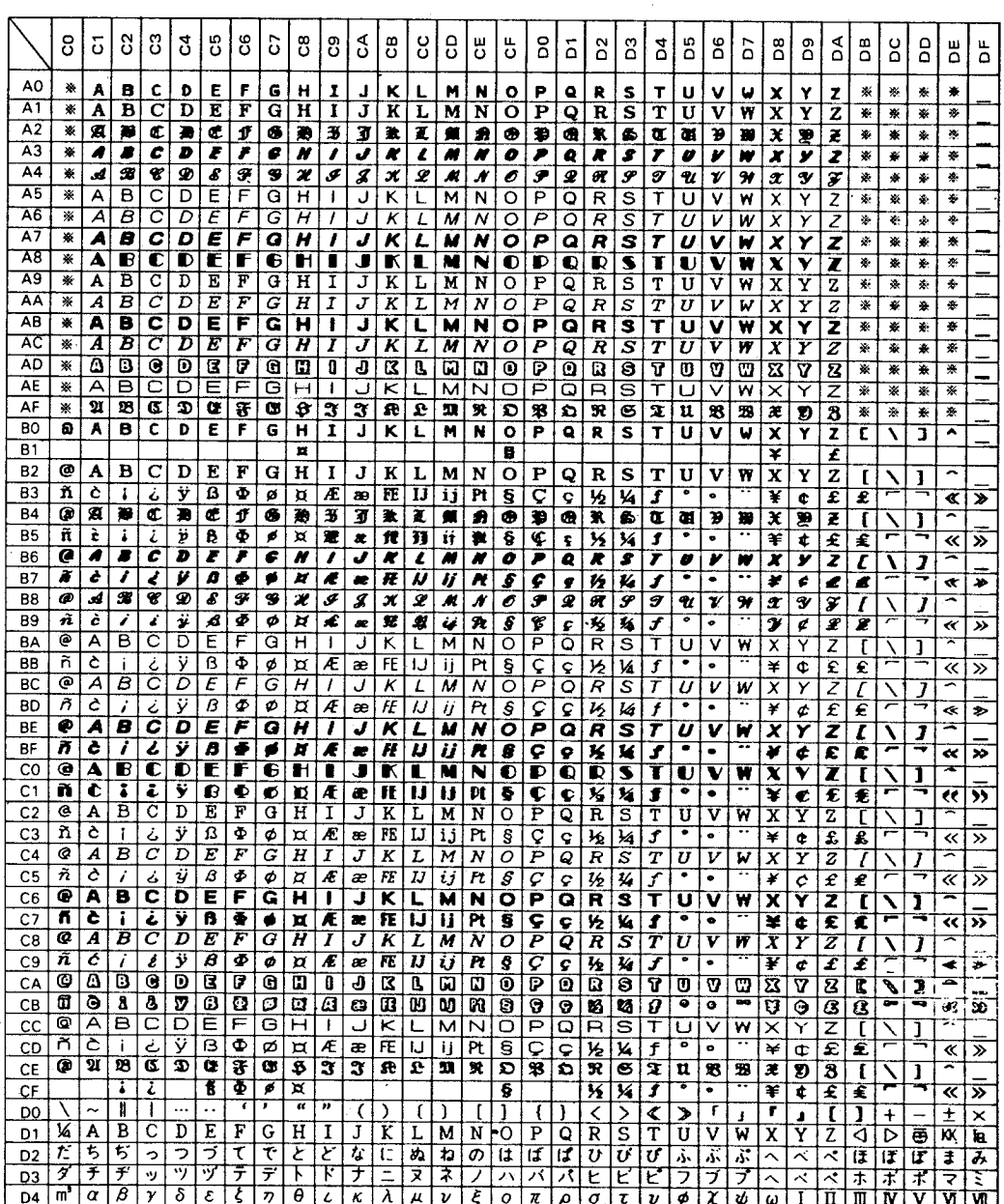
Country Hex. code (Dec code)	U.S.A.	France	Germany	England	Denmark	Sweden	Italy	Spain
ACA3 (172-163)	#	#	#	£	#	#	#	Pt
ACA4 (172-164)	$	$	$	$	$	¤	$	$
ACC0 (172-192)	@	à	§	@	@	É	@	@
ACDB (172-219)	[	°	Ä	[	Æ	Ä	°	¡
ACDC (172-220)	\	Ç	Ö	\	Ø	Ö	\	Ñ
ACDD (172-221)	]	§	Ü	]	Å	Å	é	¿
ACDE (172-222)	^	^	^	^	^	Ü	^	^
ACE0 (172-224)	`	`	`	`	`	é	ù	`
ACFB (172-251)	{	é	ä	{	æ	ä	à	{
ACFC (172-252)	|	ù	ö	|	ø	ö	ò	ñ
ACFD (172-253)	}	è	ü	}	å	å	è	}
ACFE (172-254)	~	¨	ß	~	~	ü	ì	~
ADA3 (173-163)	■	■	■	■	■	■	■	■
ADA4 (173-164)	■	■	■	■	■	■	■	■
ADC0 (173-192)	■	■	■	■	■	■	■	■
ADDB (173-219)	■	■	■	■	■	■	■	■
ADDC (173-220)	■	■	■	■	■	■	■	■
ADDD (173-221)	■	■	■	■	■	■	■	■
ADDE (173-222)	■	■	■	■	■	■	■	■
ADE0 (173-224)	■	■	■	■	■	■	■	■
ADFB (173-251)	■	■	■	■	■	■	■	■
ADFC (173-252)	■	■	■	■	■	■	■	■
ADFD (173-253)	■	■	■	■	■	■	■	■
ADFE (173-254)	■	■	■	■	■	■	■	■
AEA3 (174-163)	#	#	#	£	#	#	#	Pt
AEA4 (174-164)	$	$	$	$	$	¤	$	$
AEC0 (174-192)	@	à	§	@	@	É	@	@
AEDB (174-219)	[	°	Ä	[	Æ	Ä	°	¡
AEDC (174-220)	\	Ç	Ö	\	Ø	Ö	\	Ñ
AEDD (174-221)	]	§	Ü	]	Å	Å	é	¿
AEDE (174-222)	^	^	^	^	^	Ü	^	^
AEE0 (174-224)	`	`	`	`	`	é	ù	`
AEFB (174-251)	{	é	ä	{	æ	ä	à	{
AEFC (174-252)	|	ù	ö	|	ø	ö	ò	ñ
AEFD (174-253)	}	è	ü	}	å	å	è	}
AEFE (174-254)	~	¨	ß	~	~	ü	ì	~
AFA3 (175-163)	#	#	#	£	#	#	#	$
AFA4 (175-164)	$	$	$	$	$	¤	$	●
AFC0 (175-192)	●	•	§	●	●	■	●	¡
AFDB (175-219)	[	■	■	[	■	■	°	■
AFDC (175-220)	\	■	■	\	■	■	\	■
AFDD (175-221)	]	§	■	]	■	■	■	¿
AFDE (175-222)	^	^	^	^	^	■	^	^
AFE0 (175-224)	`	`	■	`	`	■	`	`
AFFB (175-251)	{	■	■	{	■	■	■	■
AFFC (175-252)	|	■	■	|	■	■	■	■
AFFD (175-253)	}	■	■	}	■	■	■	}
AFFE (175-254)	~	¨	ß	~	~	■	■	~

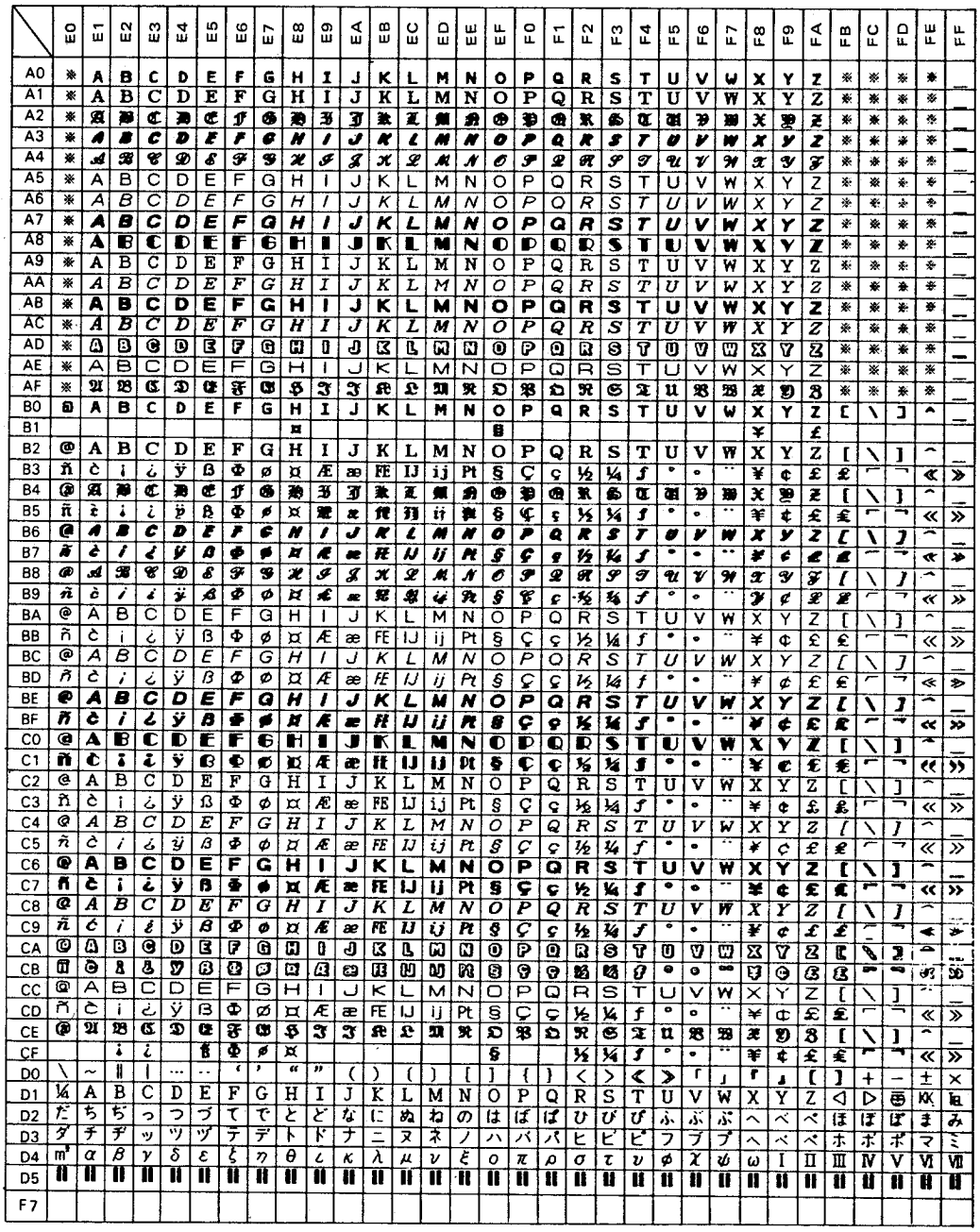
Multiple Fonts Character Set



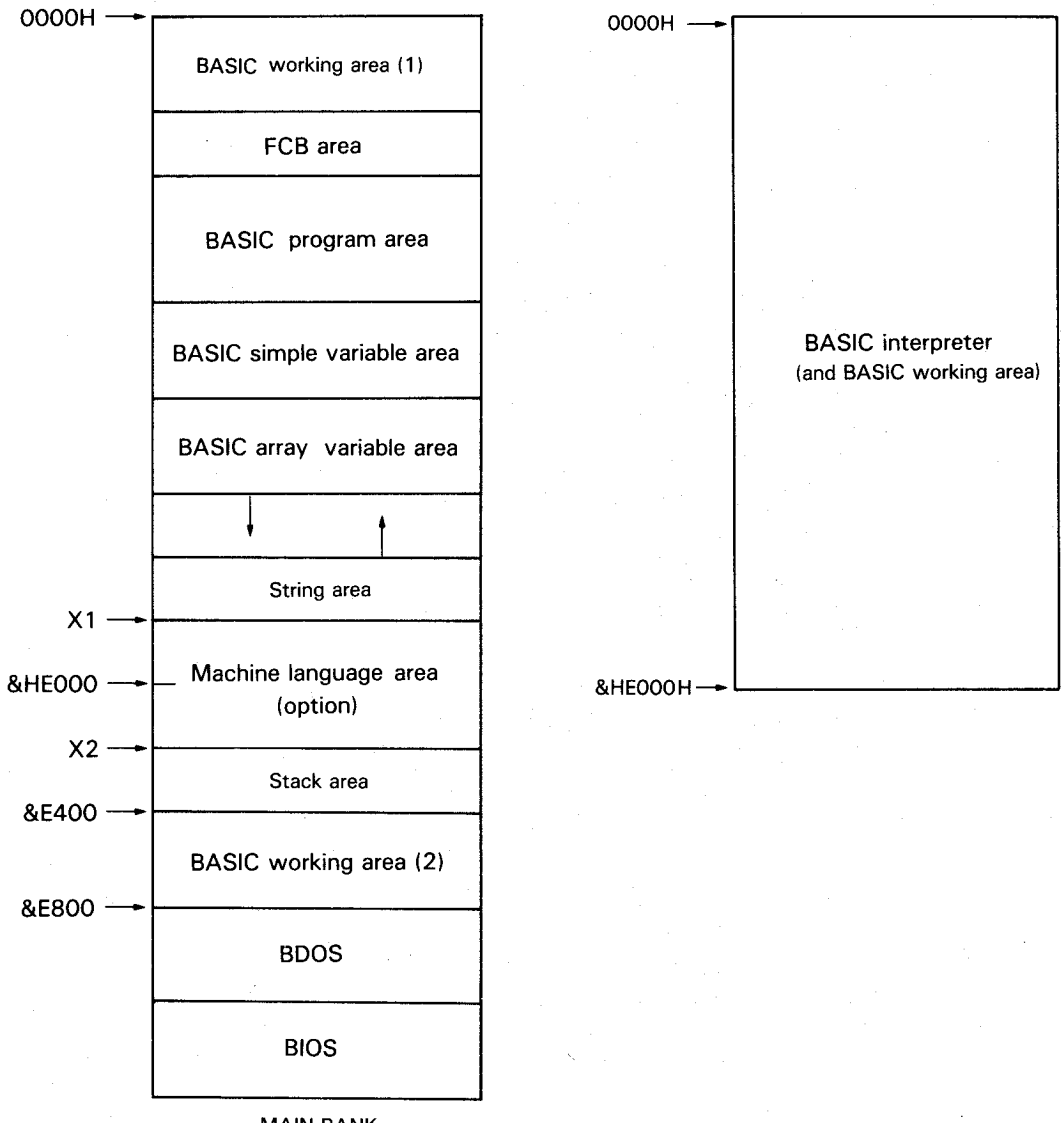
Characters marked with * are those characters that can be changed by "OPTION COUNTRY" command to correspond to the specific country's language requirements.

"▮▮"is generated for codes other than those listed in these tables.





Appendix K Memory Map



With QX-10 MFBASIC, the BASIC interpreter is located in user bank 1 and the program text, variables, and machine language areas are located in the main bank. Therefore, the PEEK, POKE, USR, and CALL statements/functions operate on memory addresses in the main bank. Also, the floating accumulator is located in BASIC working area (2), making it easy to access from machine language programs.