APPENDIXES

APPENDIX A Error Messages

/0 11

Division by zero
A division by zero is encountered in an expression.
- The divisor in an expression is zero.
- Division by an undefined variable is encountered in an expression.
- The argument of TAN function is $\pi/2$.

AO 52

File already open
A file of the specified number is already open.
- A file opened has not been closed in direct mode.

BD 58

Bad data in file
Data format in a file is incorrect.
- An attempt is made to read a programme file in binary format as a data file in ASCII format.
- An attempt is made to read a machine language programme file as a data file.

BF 51

Bad file mode
A file mode is incorrect.
- An attempt is made to open a file with a file mode not allowed for that device (e.g., open the printer for the input mode, etc.).
- An attempt is made to execute an I/O command that is inconsistent with the mode in which the file was opened (e.g., output data to a file opened for the input mode, etc.).

BN 50

Bad file number
A file number is incorrect.
- A file number not in the range 1 to 16 is used in an OPEN statement.
- A file number not in the range 1 to 16 is used in an I/O statement.

BO 61

Buffer overflow
An overflow occurred in the input buffer.
- Data input from the RS-232C port ("COM0:") is overflowed. (In data transfer without handshaking, the bit rate is too fast. The bit rate should be reduced.)

BS 9

Bad subscript
A subscript that is outside the dimensions of the array, or the wrong number of subscripts is used.
- The size of array variable elements in a DIM statement is too large. (Normally, this will result in an OM or OV error.)
- The value of a subscript other than that declared in a DIM statement is used.
- The number of dimensions for an array is incorrect.

- A subscript with a value greater than 11 is used without declaration by a DIM statement.
- A subscript specified as 0 is referenced after execution of an OPTION BASE 1.
- A record number in a PUT% or GET% statement is too large.

CN 17	Can't continue Programme execution cannot be resumed. ● A programme has halted due to an error. ● The programme has been modified after it was BREAKed. ● A programme is not executed. ● An abort has occurred, as BREAK key was pressed during I/O operation.
DD 10	Duplicate definition An array or user function is defined in duplication. ● An array of the same name is declared without executing an ERASE statement. ● Undeclared array variables are used and then declared by a DIM statement. ● Attempts to execute an OPTION BASE statement was made twice. ● A DIM or DEF FN statement exists in a loop.
DS 56	Direct statement in file During a LOAD or MERGE operation, an unnumbered programme line is read. ● A data file is read. ● A machine language programme is read.
DU 60	Device unavailable A device is not available. ● A device which is not connected to the HX-20 is specified.
FC 5	Illegal function call A statement or function is called incorrectly. ● A parameter for a statement or function is out of range. (Many functions cannot be used with a negative or zero parameter.) ● The value of a subscript in an array is negative. ● An undefined USR function is used. ● The number of characters specified in a PRINT USING statement exceeds 25. ● An undeclared array or a variable to which no value has been assigned is used in a SWAP statement. ● A line number greater than 64000 is encountered during execution of a RENUM command. ● In a PCOPY command, the specified area is the programme area currently LOGged IN, or a programme already exists in the specified area, or no programme exists in the programme area currently LOGged IN. ● Offset of a DEFFIL statement is too large. ● PEEK or POKE is executed against the EPSON BASIC programme area or the I/O area up to address &H4D.

FD 55	Bad file descriptor A file descripter is incorrect. ● An element of the file descriptor is misspelled.
FN 23	FOR without NEXT NEXT statements are insufficient. ● One NEXT statement is shared by two or more FOR statements. ● FOR and NEXT do not correspond one to one. (The control variable name is written incorrectly.)
ID 12	Illegal direct A statement that is illegal in direct mode is entered. ● DEF FN, INPUT and RANDOMIZE statements, etc., cannot be executed in direct mode.
IE 54	Input past end All the data in the file has been read. ● The number of data and the number of variables to be read do not match. ● An attempt is made to read data continuously without using the EOF function.
IO 53	Device I/O error An error has occurred during communication with a peripheral device. ● A cassette tape is defective. ● The level adjustments of the audio cassette are mismatched. ● The interface conditions of the RS-232C (bit rate, handshaking lines, etc.,) are mismatched.
IU 59	Device in use The specified device is busy. ● Wrong device name. ● The same OPEN statement is executed twice. ● Execution of CLOSE statement is neglected.
LS 15	String too long A string is too long. ● An attempt is made to assign a string variable longer than 256 characters.
MO 22	Missing operand A required parameter is missing in an expression. ● A full stop is used instead of a comma between numbers. ● An essential parameter is omitted.

NE 63 File not exist
A file does not exist under the specified name.
- A file name is written incorrectly.
- A ROM cartridge that does not contain the specified file is used.

NF 1 NEXT without FOR
FOR statements are insufficient.
- Incorrect looping is executed.
- Two or more NEXT statements exist for one FOR statement.
- Accidental jump to FOR–NEXT loop from other programme line.

NO 57 File not OPEN
A file number is used for a file that has not been OPENed.
- A file number is written incorrectly.
- An OPEN statement is not programmed.

NR 19 No resume
No RESUME statement is contained in an error trapping routine.
- At the end of an error trapping routine, there must be one of the following statements: END, RESUME, and ON ERROR GOTO.

OD 4 Out of data
A READ statement is encountered when there is no data to read.
- The number of data is insufficient.
- A RESTORE statement is incorrect.
- Use of delimiters in a DATA statement is incorrect.

OM 7 Out of memory
Memory capacity is insufficient.
- A programme is too long.
- A programme has too many variables.
- An array variable is too large.
- Expressions are too complicated.
- A programme has too many FOR...NEXT or GOSUB...RETURN loops.
- The string space or RAM file size specified by a CLEAR command is too large.
- The address number specified by a MEMSET statement is too large.

OS 14 Out of string space
A string space is insufficient.
- The string space specified by a CLEAR command is too small.

OV 6 Overflow
The result of a calculation is too large.
- The result of an operation on integer constants is not in the range −32768 to 32767.
- The result of an operation on real numbers is not in the range −1.70141E38 to 1.70141E38.
- A value in a command using the address as a parameter exceeds the specified range.

PP 62 Protected programme
The programme is protected.
- A NEW or LOAD command cannot be executed for the programme in the area which has been named by a TITLE statement.

RG 3 RETURN without GOSUB
A RETURN statement is encountered before the execution of a GOSUB statement.
- Execution is branched to a subroutine by a GOTO statement.
- A subroutine is executed by a RUN command.
- In the absence of an END statement at the last line of the main programme the following subroutine is executed.

RW 20 RESUME without error
A RESUME statement is executed when no error exists.
- An error trapping routine is entered by a GOTO or GOSUB statement.
- In the absence of an END statement at the last line of the main programme, the following error trapping routine is executed.

SN 2 Syntax error
A programme is not written according to the syntax of the language used.
- A reserved word other than those recognised is used.
- Unmatched parentheses.
- A delimiter is mistyped (comma for full stop, colon for semicolon, etc.).
- A variable name does not start with an alphabetic character.
- A variable name starts with a reserved word.
- The number of parameters for a function or statement is incorrect.
- Unrelated characters are written in the latter part of a line not visible on the physical screen.
- A string variable is used before the variable name list in a PUT% or GET% statement.

ST 16 String formula too complex
A string expression is too complex.
- A string expression written in one line is too long or complex. Too many nested parentheses are used in a string expression.

TM 13 Type mismatch
A mismatch in the type of variable.
- A numeric value name is assigned to a string variable.
- A string value name is assigned to a numeric variable.
- A type mismatch exists in the argument of a function.

UF 18 Undefined user function
A USR function is not defined.
- A variable name starting with "FN" is used.
- The function name in a DEF FN statement is incorrect.
- The DEF FN statement is not executed. (Execution of a programme is started from the middle of the programme by a GOTO or similar statement.)

UL 8 Undefined line number
An error in the line number.
- Line number is not specified.
- A line number specified in a GOTO, GOSUB, RESTORE or RUN statement does not exist.
- The line to be referenced when a RENUM statement is executed does not exist.

UP 21 Unprintable error
Indicates an error with an undefined error code.
- An ERROR statement is executed in the absence of any error trapping routine.
- Error codes 26 to 49 and 64 to 255 will cause this message to be displayed.

WE 24 WHILE without WEND
- This message is used in Disk BASIC.

WH 25 WEND without WHILE
- This message is used in Disk BASIC.

How to Read Error Message Table

SN 2 Syntax error
The programme is not written according to the syntax.
- A reserved word other than those recognised is used.

Meaning and explanation of the error message
Error code
Error message

APPENDIX B Device Names

Device name	Equipment name	Input	Output	Remarks
KYBD:	Keyboard	○	×	
SCRN:	Screen	×	○	
LPT0:	Built-in microprinter	×	○	
COM0:	RS-232C port	○	○	
CAS0:	Microcassette drive	○	○	Option
CAS1:	Audio cassette	○	○	
PAC0:	ROM cartridge	○	×	Option
A:	Flexible disk drive A	○	○	Device
B:	Flexible disk drive B	○	○	names for
C:	Flexible disk drive C	○	○	DISK BASIC
D:	Flexible disk drive D	○	○	

○: Applicable ×: Not applicable

APPENDIX C
Correspondence Table between Device Names and BASIC Commands

Device / Command	KYBD:	SCRN:	LPT0:	COM0:	CAS0:	CAS1:	PAC0:
LOAD	×	×	×	○	○	○	○
LOADM	×	×	×	×	○	○	○
LOAD?	×	×	×	×	○	○	×
RUN "<file descriptor>"	×	×	×	○	○	○	○
MERGE	×	×	×	○	○	○	○
FILES	×	×	×	×	○	○	○
INPUT#	○	×	×	○	○	○	○
INPUT$	○	×	×	○	○	○	○
FOF	–	×	×	○	○	○	○
LOF	–	×	×	○	–	–	○
SAVE	×	○	○	○	○	○	×
SAVEM	×	×	×	×	○	○	×
LIST	×	○	○	○	○	○	×
PRINT# (USING)	×	○	○	○	○	○	×
POS	×	○	○	○	–	–	×
OPEN mode	I	O	O	I/O	I/O	I/O	I

NOTE:
○ or × used in this table indicates that when a device is specified for a command or statement, the device

○: Can be used.
×: Cannot be used. An FC error occurs.
–: Causes no error but the command is invalid.

APPENDIX D Formatting Characters

Format string	Function
!	Specifies to output only the first character in a given string.
\…\	Specifies to the number of characters to be output from the beginning of a given string.
&	Specifies the output positions of characters in a given string.
#	Specifies each digit position.
.	Specifies the position of the decimal point.
+	Outputs the sign of a number (plus or minus) before or after the number.
–	Outputs negative numbers with a trailing minus sign.
**	Causes leading spaces in the numeric field to be filled with asterisks.
$$	Causes a dollar sign to be output to the immediate left of the formatted number.
**$	Causes leading spaces to be filled with asterisks and a dollar sign to be output before the number.
,	Causes a comma to be output to the every 3rd digit to the left of the decimal point.
^^^^	Outputs a numeric value in exponential format.
_	Outputs any of the above formatting characters as a literal character.

NOTE: The formatting characters shown above apply to the ASCII character set. If your selected character set is other than ASCII, some of the formatting characters will be output differently as shown below.

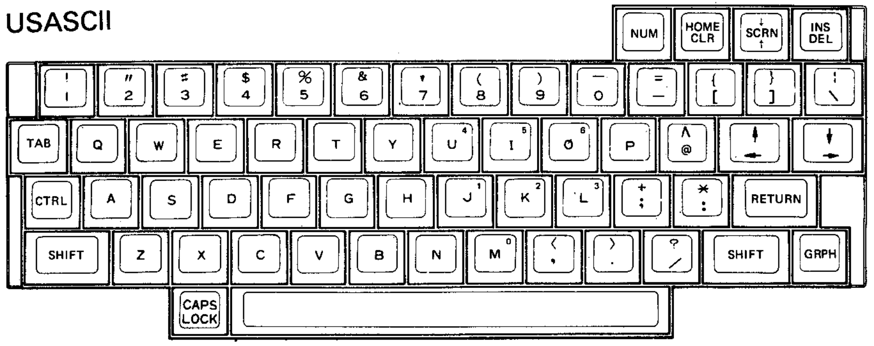
U.S.A.	France	Germany	England	Denmark	Sweden	Italy	Spain
#	#	#	£	#	#	#	Pt
$	$	$	$	$	¤	$	$
\	ç	Ö	\	ϕ	Ö	\	Ñ
^	^	^	^	^	Ü	^	^

(See Chapter 3 POKE).

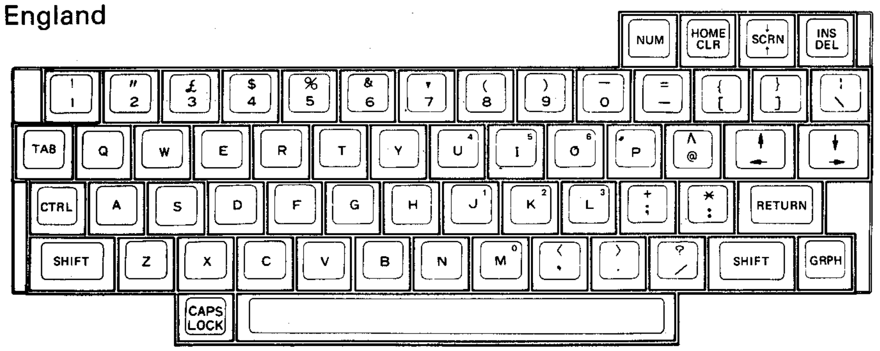
APPENDIX E
Keyboard Layouts and Key Assignments

1. Keyboard Layouts

USASCII



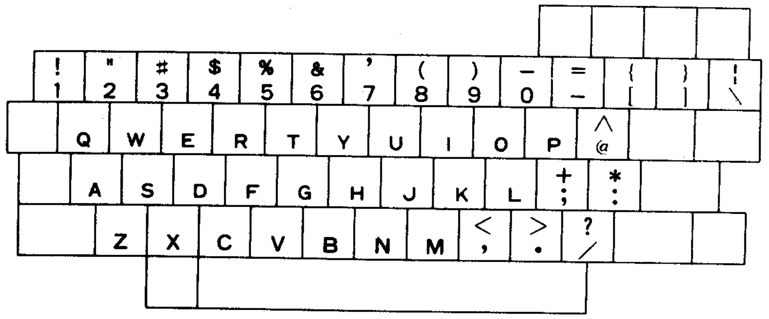
England



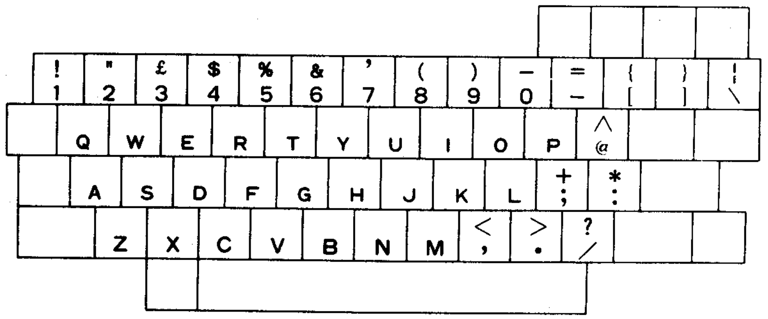
2. Key Assignments for Each Keyboard Mode

(1) Uppercase Mode and Lowercase Mode

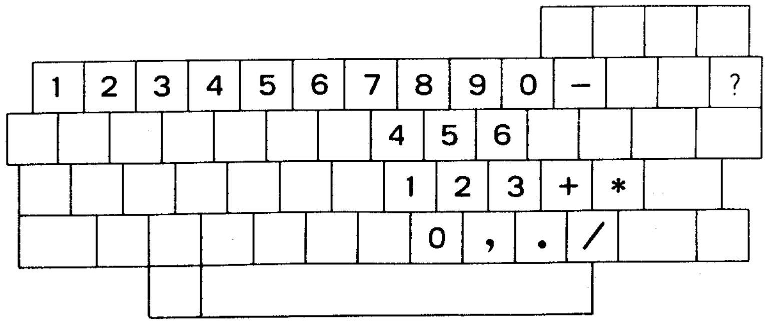
USASCII



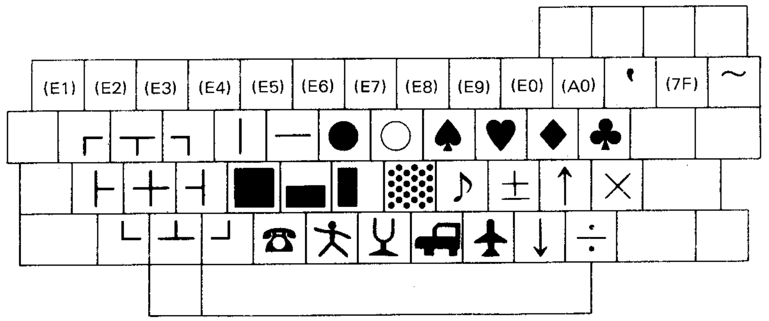
England



(2) Numeric Key Mode



(3) Graphic Key Locations


(E1) (E2) (E3) (E4) (E5) (E6) (E7) (E8) (E9) (E0) (A0) ‘ (7F) ~

NOTE: E0 through E9, A0 and 7F shown in parentheses in the above figure are character codes in hexadecimal numbers and can be input by pressing the corresponding keys while holding down the GRPH key. The character codes for ‘ and ~ are 60 and 7E, respectively. (See Chapter 8, "Definition of Graphic Pattern" in the HX-20 Operation Manual.)

APPENDIX F Character Code Tables

1. USASCII

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	32 SP	48 0	64 @	80 P	96 `	112 p	128 ┼	144 ○	160	176	192	208	224	240
1	0001	1	17	33 !	49 1	65 A	81 Q	97 a	113 q	129 ┴	145 ♠	161	177	193	209	225	241
2	0010	2	18	34 "	50 2	66 B	82 R	98 b	114 r	130 ┬	146 ♥	162	178	194	210	226	242
3	0011	3	19	35 #	51 3	67 C	83 S	99 c	115 s	131 ┤	147 ♦	163	179	195	211	227	243
4	0100	4	20	36 $	52 4	68 D	84 T	100 d	116 t	132 ├	148 ♣	164	180	196	212	228	244
5	0101	5	21	37 %	53 5	69 E	85 U	101 e	117 u	133 ─	149 🚶	165	181	197	213	229	245
6	0110	6	22	38 &	54 6	70 F	86 V	102 f	118 v	134 │	150 ☎	166	182	198	214	230	246
7	0111	7	23	39 '	55 7	71 G	87 W	103 g	119 w	135 ┌	151 ♪	167	183	199	215	231	247
8	1000	8	24	40 (	56 8	72 H	88 X	104 h	120 x	136 ┐	152 🚗	168	184	200	216	232	248
9	1001	9	25	41)	57 9	73 I	89 Y	105 i	121 y	137 └	153 Ψ	169	185	201	217	233	249
A	1010	10	26	42 *	58 :	74 J	90 Z	106 j	122 z	138 ┘	154 ✈	170	186	202	218	234	250
B	1011	11	27	43 +	59 ;	75 K	91 [	107 k	123 {	139 ▒	155 ↑	171	187	203	219	235	251
C	1100	12	28	44 ,	60 <	76 L	92 \	108 l	124 |	140 █	156 ↓	172	188	204	220	236	252
D	1101	13	29	45 -	61 =	77 M	93]	109 m	125 }	141 ▬	157 ×	173	189	205	221	237	253
E	1110	14	30	46 .	62 >	78 N	94 ^	110 n	126 ~	142 ▌	158 ÷	174	190	206	222	238	254
F	1111	15	31	47 /	63 ?	79 O	95 _	111 o	127	143 ●	159 ±	175	191	207	223	239	255

2. ENGLAND

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	**Binary No.**	**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
0	0000	0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	£ 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	☎ 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	✈ 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	🚗 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	🏆 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	🚶 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	[91	k 107	{ 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	\ 92	l 108	¦ 124	█ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	] 93	m 109	} 125	▬ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	▮ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

3. FRANCE

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	**Binary No.**	**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
0	0000	0	16	SP 32	0 48	à 64	P 80	` 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	☎ 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	✈ 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	🚗 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	🏆 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	🚶 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	° 91	k 107	é 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	ç 92	l 108	ù 124	█ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	§ 93	m 109	è 125	▬ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	^ 94	n 110	¨ 126	▮ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

4. GERMANY

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	§ 64	P 80	` 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	☎ 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	✈ 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	🚗 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	🍸 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	🏃 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	Ä 91	k 107	ä 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	Ö 92	l 108	ö 124	█ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	Ü 93	m 109	ü 125	▬ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	^ 94	n 110	ß 126	▌ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

5. DENMARK

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	@ 64	P 80	` 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	☎ 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	✈ 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	🚗 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	🍸 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	🏃 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	Æ 91	k 107	æ 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	Ø 92	l 108	ø 124	█ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	Å 93	m 109	å 125	▬ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	^ 94	n 110	~ 126	▌ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

6. SWEDEN

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	É 64	P 80	é 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	¤ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	☎ 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	✈ 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	🚗 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	⛵ 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	🏃 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	Ä 91	k 107	ä 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	Ö 92	l 108	ö 124	█ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	Å 93	m 109	å 125	▬ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	Ü 94	n 110	ü 126	▌ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

7. ITALY

	Hex. No.	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Hex. No.	Binary No.	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	0000	0	16	SP 32	0 48	@ 64	P 80	ù 96	p 112	┼ 128	○ 144	160	176	192	208	224	240
1	0001	1	17	! 33	1 49	A 65	Q 81	a 97	q 113	┴ 129	♠ 145	161	177	193	209	225	241
2	0010	2	18	" 34	2 50	B 66	R 82	b 98	r 114	┬ 130	♥ 146	162	178	194	210	226	242
3	0011	3	19	# 35	3 51	C 67	S 83	c 99	s 115	┤ 131	♦ 147	163	179	195	211	227	243
4	0100	4	20	$ 36	4 52	D 68	T 84	d 100	t 116	├ 132	♣ 148	164	180	196	212	228	244
5	0101	5	21	% 37	5 53	E 69	U 85	e 101	u 117	─ 133	♪ 149	165	181	197	213	229	245
6	0110	6	22	& 38	6 54	F 70	V 86	f 102	v 118	│ 134	☎ 150	166	182	198	214	230	246
7	0111	7	23	' 39	7 55	G 71	W 87	g 103	w 119	┌ 135	✈ 151	167	183	199	215	231	247
8	1000	8	24	(40	8 56	H 72	X 88	h 104	x 120	┐ 136	🚗 152	168	184	200	216	232	248
9	1001	9	25	) 41	9 57	I 73	Y 89	i 105	y 121	└ 137	⛵ 153	169	185	201	217	233	249
A	1010	10	26	* 42	: 58	J 74	Z 90	j 106	z 122	┘ 138	🏃 154	170	186	202	218	234	250
B	1011	11	27	+ 43	; 59	K 75	° 91	k 107	à 123	▒ 139	↑ 155	171	187	203	219	235	251
C	1100	12	28	, 44	< 60	L 76	\ 92	l 108	ò 124	█ 140	↓ 156	172	188	204	220	236	252
D	1101	13	29	- 45	= 61	M 77	é 93	m 109	è 125	▬ 141	× 157	173	189	205	221	237	253
E	1110	14	30	. 46	> 62	N 78	^ 94	n 110	ì 126	▌ 142	÷ 158	174	190	206	222	238	254
F	1111	15	31	/ 47	? 63	O 79	_ 95	o 111	127	● 143	± 159	175	191	207	223	239	255

8. SPAIN

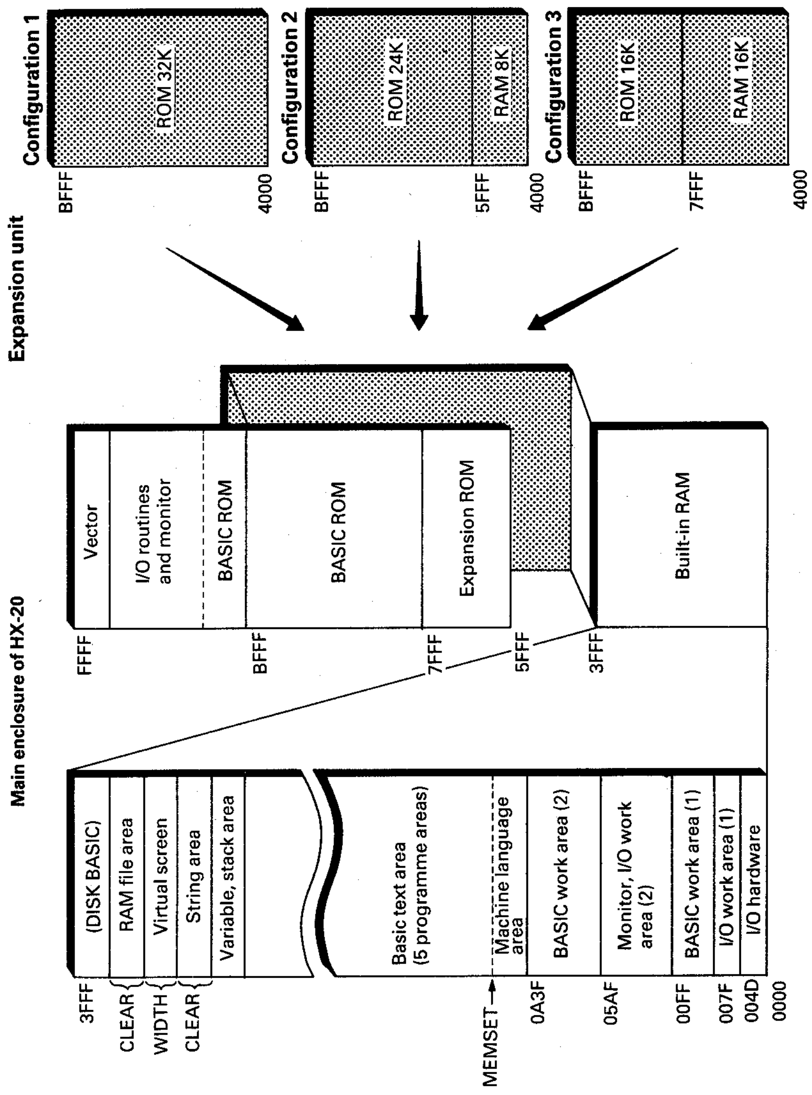
Hex. No.	Binary No.	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	9 1001	A 1010	B 1011	C 1100	D 1101	E 1110	F 1111
0	0000	0	16	32 SP	48 0	64 @	80 P	96 `	112 p	128 ┼	144 ○	160	176	192	208	224	240
1	0001	1	17	33 !	49 1	65 A	81 Q	97 a	113 q	129 ┴	145 ♠	161	177	193	209	225	241
2	0010	2	18	34 "	50 2	66 B	82 R	98 b	114 r	130 ┬	146 ♥	162	178	194	210	226	242
3	0011	3	19	35 ₧	51 3	67 C	83 S	99 c	115 s	131 ┤	147 ♦	163	179	195	211	227	243
4	0100	4	20	36 $	52 4	68 D	84 T	100 d	116 t	132 ├	148 ♣	164	180	196	212	228	244
5	0101	5	21	37 %	53 5	69 E	85 U	101 e	117 u	133 ─	149 ♪	165	181	197	213	229	245
6	0110	6	22	38 &	54 6	70 F	86 V	102 f	118 v	134 │	150 [illegible]	166	182	198	214	230	246
7	0111	7	23	39 '	55 7	71 G	87 W	103 g	119 w	135 ┌	151 [illegible]	167	183	199	215	231	247
8	1000	8	24	40 (	56 8	72 H	88 X	104 h	120 x	136 ┐	152 [illegible]	168	184	200	216	232	248
9	1001	9	25	41)	57 9	73 I	89 Y	105 i	121 y	137 └	153 [illegible]	169	185	201	217	233	249
A	1010	10	26	42 *	58 :	74 J	90 Z	106 j	122 z	138 ┘	154 [illegible]	170	186	202	218	234	250
B	1011	11	27	43 +	59 ;	75 K	91 ¡	107 k	123 ¨	139 ▒	155 ↑	171	187	203	219	235	251
C	1100	12	28	44 ,	60 <	76 L	92 Ñ	108 l	124 ñ	140 █	156 ↓	172	188	204	220	236	252
D	1101	13	29	45 -	61 =	77 M	93 ¿	109 m	125 }	141 ▄	157 ×	173	189	205	221	237	253
E	1110	14	30	46 .	62 >	78 N	94 ^	110 n	126 ~	142 ▌	158 ÷	174	190	206	222	238	254
F	1111	15	31	47 /	63 ?	79 O	95 _	111 o	127	143 ●	159 ±	175	191	207	223	239	255

APPENDIX G Control Codes

Decimal	Hexadecimal	Function	Keys
1	01	Moves the physical screen to the left corner of the virtual screen.	CTRL + A
3	03	Escape AUTO mode.	CTRL + C
4	04	Moves the physical screen to the right.	CTRL + D, CTRL + →
5	05	Deletes all characters to the end of the line.	CTRL + E
6	06	Moves the physical screen to the right corner of the virtual screen.	CTRL + F
8	08	Backspace	CTRL + H, INS DEL
9	09	TAB (spaces every 8 columns)	CTRL + I, TAB
10	0A	Line feed	CTRL + J
11	0B	Moves the cursor to its home position.	CTRL + K, SHIFT + HOME CLR
12	0C	Clears the virtual screen.	CTRL + L, HOME CLR
13	0D	Carriage return	CTRL + M, RETURN
16	10	Moves the physical screen up.	CTRL + P SC RN
17	11	Moves the physical screen down.	CTRL + Q, SHIFT + SC RN
18	12	Insert mode	CTRL + R, SHIFT + INS DEL
19	13	Moves the physical screen to the left.	CTRL + S, CTRL + ←
22	16	Turns the cursor ON.	CTRL + V
23	17	Turns the cursor OFF.	CTRL + W
26	1A	Deletes all characters from the cursor position to the bottom line of the virtual screen.	CTRL + Z
28	1C	Moves the cursor to the right.	→
29	1D	Moves the cursor to the left.	←
30	1E	Moves the cursor up.	SHIFT + ↑
31	1F	Moves the cursor down.	SHIFT + ↓

Character codes 0 to 31 are not displayed as characters even if they are in an output statement. However, if they are included in a string, each character code will be counted as a one-character length.

APPENDIX H Memory Map



APPENDIX I Table of Reserved Words

ABS	ERL	LPRINT	SAVE
ALL	ERR	MEMSET	SAVEM
AND	ERROR	MERGE	SCREEN
ASC	EXEC	MID$	SCROLL
ATN	EXP	MOD	SGN
AUTO	FILES	MON	SIN
BASE	FIX	MOTOR	SOUND
CDBL	FN	NEW	SPACE$
CHR$	FOR	NEXT	SPC
CINT	FRE	NOT	SQR
CLEAR	GCLS	OCT$	STAT
CLOSE	GET	OFF	STEP
CLS	GO	ON	STOP
COLOR	HEX$	OPEN	STR$
CONT	IF	OPTION	STRING$
COPY	IMP	OR	SUB
COS	INKEY$	PCOPY	SWAP
CSNG	INPUT	PEEK	TAB
CSRLIN	INSTR	POINT	TAN
DATA	INT	POKE	TAPCNT
DATE	KEY	POS	THEN
DAY	LEFT$	PRESET	TIME
DEF	LEN	PRINT	TITLE
DEFDBL	LET	PSET	TO
DEFFIL	LINE	PUT	TROFF
DEFINT	LIST	RANDOMIZE	TRON
DEFSNG	LLIST	READ	USING
DEFSTR	LOAD	REM	USR
DELETE	LOAD?	RENUM	VAL
DIM	LOADM	RESTORE	VARPTR
ELSE	LOCATE	RESUME	WEND
END	LOCATES	RETURN	WHILE
EOF	LOF	RIGHT$	WIDTH
EQV	LOG	RND	WIND
ERASE	LOGIN	RUN	XOR

APPENDIX J
List of Commands and Statements

AUTO

FORMAT	AUTO [<line number>][,[<increment>]]
PURPOSE	To generate a line number automatically.
EXAMPLE	AUTO 100, 10 AUTO 200, AUTO 300 AUTO

CLEAR

FORMAT	CLEAR [<character area size>[,<RAM file size>]]
PURPOSE	To initialize variables and to set the size of the character area and the RAM file.
EXAMPLE	CLEAR 200, 256

CLOSE

FORMAT	CLOSE[[#]<file number>[,[#]<file number>...]]
PURPOSE	To close file(s).
EXAMPLE	CLOSE #3

*CLS

FORMAT	CLS
PURPOSE	To clear a text screen.
EXAMPLE	CLS

COLOR

FORMAT	COLOR [<foreground color>][,[<background color>] [,<color set>]]
PURPOSE	To specify the screen colors of the external display.
EXAMPLE	COLOR 0, 3, 0

CONT

FORMAT	CONT
PURPOSE	To resume the execution of a programme that has been stopped.
EXAMPLE	CONT

COPY

FORMAT	COPY
PURPOSE	To output the characters and graphics displayed on the LCD, on the built-in microprinter.
EXAMPLE	COPY

DATA

FORMAT	DATA <constant>[,<constant>...]
PURPOSE	To store the numeric and string constants that are accessed by the READ statement(s).
EXAMPLE	DATA HX, 20, EPSON

DEFFIL

FORMAT	DEFFIL <record length>, <relative address>
PURPOSE	To define the relative address of record 0 in a RAM file and the length of a single record.
EXAMPLE	DEFFIL 20, 200

DEF FN

FORMAT	DEF FN<name>[(<parameter>[,<parameter>....])] =<function definition>
PURPOSE	To define a function created by the user.
EXAMPLE	DEF FNZ(X, Y)=X∗2+Y∗3+A

DEFINT/SNG/DBL/STR

FORMAT	DEF | INT | <range(s) of letters> SNG DBL STR
PURPOSE	To declare variable types as integer, single precision, double precision, and string.
EXAMPLE	DEFSTR A,X–Z

DEF USR

FORMAT	DEF USR[<digit>]=<starting address>
PURPOSE	To specify the starting address of a machine language subroutine.
EXAMPLE	DEF USR6=&H0C00

DELETE

FORMAT	DELETE [<starting line number>][– [<ending line number>]]
PURPOSE	To delete specified programme lines.
EXAMPLE	DELETE 100–200 DELETE 100– DELETE –200 DELETE 100

DIM

FORMAT	DIM<variable>(<maximum subscript value>[,<maximum subscript value>...])[,...]
PURPOSE	To declare the size of array variable elements.
EXAMPLE	DIM A(40, 10), B$(50)

END

FORMAT	END
PURPOSE	To close all files and terminate programme execution.
EXAMPLE	END

ERASE

FORMAT	ERASE <array variable>[,<array variable>...]
PURPOSE	To eliminate arrays from a programme.
EXAMPLE	ERASE A, B

ERROR

FORMAT	ERROR<integer expression>
PURPOSE	To simulate the occurrence of an error; or to allow error codes to be defined by the user.
EXAMPLE	ERROR 225

EXEC

FORMAT	EXEC [<starting address>]
PURPOSE	To start execution of a machine language programme.
EXAMPLE	EXEC &H0C00

FILES

FORMAT	FILES["<device name>"]
PURPOSE	To display the names of all files residing on a specified memory device.
EXAMPLE	FILES "CAS1:"

FOR...TO...STEP-NEXT

FORMAT	FOR <variable>=<initial value> TO <final value> [STEP <increment>] . . . NEXT [<variable>[,<variable>]]
PURPOSE	To allow a series of instructions between FOR and NEXT statements to be performed in a loop a given number of times.
EXAMPLE	FOR I=0 TO 100 STEP 5 . . . NEXT I

GCLS

FORMAT	GCLS
PURPOSE	To clear a graphic screen.
EXAMPLE	GCLS

GET%

FORMAT	GET% <record number>,<variable name>[,<variable name>...]
PURPOSE	To read data from a RAM file into variables.
EXAMPLE	GET% O, A!, B#, C$

GOSUB...RETURN

FORMAT	GOSUB <line number> . . . RETURN
PURPOSE	To branch to and return from a subroutine.
EXAMPLE	GOSUB 500

GO TO/GOTO

FORMAT	(1) GO TO <line number>, or (2) GOTO <line number>
PURPOSE	To branch programme execution to a specified line number.
EXAMPLE	GOTO 300

IF...THEN...ELSE/IF...GOTO...ELSE

FORMAT	IF <expression> | THEN | <statement> || [ELSE | <statement> |] | | <line No.> | | | <line No.> | | GOTO <line No.> |
PURPOSE	To choose a particular route for programme execution based on conditions established in an expression.
EXAMPLE	IF A >10 THEN A=0 ELSE 200

INPUT

FORMAT	INPUT ["<prompt string>" | ; | <variable> | , | [,<variable>...]
PURPOSE	To allow input from the keyboard into a specified variable during programme execution.
EXAMPLE	INPUT"NAME"; A$

INPUT#

FORMAT	INPUT# <file number>, <variable>[,<variable ...]
PURPOSE	To read data items from a sequential file and assign them to programme variables.
EXAMPLE	INPUT#1, A, B, C$

KEY

FORMAT	KEY <key number>, <string>
PURPOSE	To define the programmable function keys.
EXAMPLE	KEY 1, "LIST"

KEY LIST/KEY LLIST

FORMAT	(1) KEY LIST (2) KEY LLIST
PURPOSE	To output the strings assigned to the programmable function keys on the screen and the microprinter, respectively.
EXAMPLE	KEY LLIST

LET

FORMAT	[LET]<variable>=<expression>
PURPOSE	To assign the value of an expression to a variable.
EXAMPLE	LET A=3.141592

LINE

FORMAT	LINE[(<horizontal coordinate 1>, <vertical coordinate 1>)]–(<horizontal coordinate 2>, <vertical coordinate 2>), | PSET | [,<colour>] | PRESET |
PURPOSE	To draw a straight line between two specified points.
EXAMPLE	LINE(0,0)–(119,31),PSET

LINE INPUT

FORMAT	LINE INPUT ["<prompt string>";]<string variable>
PURPOSE	To input an entire line to a string variable.
EXAMPLE	LINE INPUT "WHAT?";A$

LINE INPUT#

FORMAT	LINE INPUT#<file number>,<string variable>
PURPOSE	To read an entire line from a sequential data file to a string variable.
EXAMPLE	LINE INPUT #1, A$

LIST/LLIST

FORMAT	(1) LIST[<starting line number>][–[<ending line number>]] (2) LLIST[<starting line number>][–[<ending line number>]]
PURPOSE	To output a programme list (1) on the LCD or external display or (2) on the microprinter.
EXAMPLE	LIST 100–200 LIST –200 LIST 100– LIST 200 LIST LIST

LIST <file descriptor>

FORMAT	LIST<file descriptor>[,[<line number>] [–[<line number>]]]
PURPOSE	To output a programme list into a specified file.
EXAMPLE	LIST "COM0:"

*LIST "COM0:"

FORMAT	LIST"COM0:[(<BLPSC>)]"[, [<line number>][–[<line number>]]]
PURPOSE	To specify the interface conditions of the RS-232C port and execute LIST.
EXAMPLE	LIST"COM0:(2701B)"

LOAD
FORMAT LOAD[<file descriptor>[,R]]
PURPOSE To load a programme file into the memory.
EXAMPLE LOAD"CAS1:PROG1.ASC"

*LOAD "COM0:"
FORMAT LOAD"COM0:[(<BLPSC>)]"
PURPOSE To specify the interface conditions of the RS-232C port and execute LOAD.
EXAMPLE LOAD"COM0:(68N2B)"[,R]

LOADM
FORMAT LOADM<file descriptor>][,[offset value][,R]]
PURPOSE To load machine language programme file into the memory.
EXAMPLE LOADM"CAS1:ABC"

*LOAD?
FORMAT LOAD?[<file descriptor>]
PURPOSE To check files.
EXAMPLE LOAD?"CAS1:PROG1.ASC"

*LOCATE
FORMAT LOCATE<horizontal coordinate>,<vertical coordinate> [,<cursor switch>]
PURPOSE To specify the cursor position on the screen.
EXAMPLE LOCATE 10, 10, 0

*LOCATES
FORMAT LOCATES <horizontal coordinate>, <vertical coordinate>
PURPOSE To specify the position of the physical screen.
EXAMPLE LOCATES 0, 0

*LOGIN
FORMAT LOGIN <expression>[,R]
PURPOSE To switch the programme areas.
EXAMPLE LOGIN 3

*MEMSET
FORMAT MEMSET [<bottom address of memory>]
PURPOSE To specify the lower limit of the memory.
EXAMPLE MEMSET &HOD00

MERGE
FORMAT MERGE [<file descriptor>[,R]]
PURPOSE To merge a specified programme file into the programme currently in memory.
EXAMPLE "CAS1:PROG3.ASC"

*MERGE "COM0:"
FORMAT MERGE "COM0:[(<BLPSC>)"[,R]]
PURPOSE To specify the interface conditions of the RS-232C port and to execute MERGE.
EXAMPLE MERGE"COM0:(68N2B)",R

MID$
FORMAT MID$ (<string exp 1>,<n>[,<m>])=<string exp 2> where n and m are integer expressions and <string exp 1> and <string exp 2> are string expressions.
PURPOSE To replace a portion of one string with another string.
EXAMPLE MID$(A$,2)="BASIC"

MON
FORMAT MON
PURPOSE To transfer programme control to the machine language monitor.
EXAMPLE MON

MOTOR
FORMAT MOTOR [<switch>]
PURPOSE To turn ON/OFF the motor of the external audio cassette.
EXAMPLE MOTOR ON

*NEW
FORMAT NEW
PURPOSE To delete the programme in the memory and clear all variables.
EXAMPLE NEW

ON ERROR GOTO
FORMAT ON ERROR GOTO <line number>
PURPOSE To enable error trapping and specify the first line of the error handling subroutine.
EXAMPLE ON ERROR GOTO 1000

ON...GOSUB/ON...GOTO
FORMAT ON <expression> |GOSUB| <line number>
|GOTO|
[,<line number>...]
PURPOSE To branch to one of several specified line numbers.
EXAMPLE ON A GOSUB 100, 200, 300, 400
ON B GOTO 100, 200, 300, 400

OPEN
FORMAT OPEN"<mode>", [#] <file number>, <file descriptor>
PURPOSE To open a specified file for I/O.
EXAMPLE OPEN"O", #1 "CAS0:TEST.BAS"

*OPEN"COM0:"
FORMAT OPEN "<mode>", [#]<file number>, "COMO:[(<BLPSC>)]"
PURPOSE To specify the interface conditions for the RS-232C port and execute OPEN.
EXAMPLE OPEN "O", #1,"COM0:(68N2B)"

OPTION BASE
FORMAT OPTION BASE |0|
|1|
PURPOSE To declare the minimum value for array variable subscripts.
EXAMPLE OPTION BASE 1

*PCOPY
FORMAT PCOPY <expression>
PURPOSE To copy a BASIC programme into another programme area.
EXAMPLE PCOPY 3

*POKE
FORMAT POKE <address>, <numeric expression>
PURPOSE To write a byte into a specified memory location.
EXAMPLE POKE &HCOO0, &H39

PRESET
FORMAT PRESET (<horizontal coordinate>, <vertical coordinate>)
PURPOSE To erase a dot on a graphic screen.
EXAMPLE PRESET (40,25)

PRINT/LPRINT
FORMAT |PRINT |[<expression>[|,|<expression>...]]
|LPRINT| |;|
PURPOSE To output data on the screen or the built-in microprinter.
EXAMPLE PRINT "EPSON"

PRINT USING/LPRINT USING
FORMAT |PRINT | USING <format string>;
|LPRINT|
[<expression>[|;|[<expression> ...]]
|,|
PURPOSE To output strings or numerics using a specified format.
EXAMPLE PRINT USING "####"; A,B

*PRINT#
FORMAT PRINT# <file number>, [<expression>...]
PURPOSE To write data into a sequential file.
EXAMPLE PRINT#1,A,B

PRINT# USING
FORMAT PRINT #<file number>, USING<format string>;
[<expression>[|;|<expression>...]]
|,|
PURPOSE To write strings and numerics into a sequential file using a specified format.
EXAMPLE PRINT#1, USING"###"; A

PSET
FORMAT PSET (<horizontal coordinate>, <vertical coordinate>)[,<colour>]
PURPOSE To draw dots on a specified graphic screen.
EXAMPLE PSET (30,20)

*PUT%
FORMAT PUT% <record number>, <variable>[,<variable>...]
PURPOSE To write the values of variables into a RAM file.
EXAMPLE PUT%0, A!, B#, C$

RANDOMIZE
FORMAT RANDOMIZE [<expression>]
PURPOSE To reseed the random number generator.
EXAMPLE RANDOMIZE

READ
FORMAT READ<variable>[,<variable>...]
PURPOSE To read values from a DATA statement and assigning them to variables.
EXAMPLE READ A, I, C$

REM
FORMAT REM[<remark>]
PURPOSE To allow explanatory remarks to be inserted in a programme.
EXAMPLE REM COMMENT MESSAGE

RENUM
FORMAT RENUM [<new number>][,<old number>][,<increment>]]
PURPOSE To renumber programme lines.
EXAMPLE RENUM

RESTORE
FORMAT RESTORE [<line number>]
PURPOSE To allow DATA statements to be reread from a specified point.
EXAMPLE RESTORE 1000

RESUME
FORMATS RESUME [|NEXT |]
|<line number>|
PURPOSE To continue programme execution after an error recovery procedure has been performed.
EXAMPLE RESUME 100

RUN
FORMAT (1) RUN [<line number>], or
(2) RUN <file descriptor>[,R]
PURPOSE To start programme execution.
EXAMPLE (1) RUN 300
(2) RUN"CAS0:PROG4.ASC"

RUN"COM0:"
FORMAT RUN"COM0:[(<BLPSC>)]"[,R]
PURPOSE To specify the interface condition of the RS-232C port and execute RUN.
EXAMPLE RUN"COM0:(68N2B)"

SAVE
FORMAT SAVE <file descriptor>[,A][,V]
PURPOSE To save an EPSON BASIC programme on a specified file.
EXAMPLE SAVE"CAS0:ABC"

*SAVE"COM0:"
FORMAT SAVE"COM0:[(<BLPSC>)]", A
PURPOSE To specify the interface conditions of the RS-232C port and execute SAVE.
EXAMPLE SAVE"COM0: (68E13)",A

*SAVEM
FORMAT SAVEM <file descriptor>,<top address>,<bottom address> <execution starting address>[,V]
PURPOSE To save the memory contents on a specified file.
EXAMPLE SAVEM"CAS1:ABC", &H0B00, &H0C00, &H0B00

*SCREEN
FORMAT SCREEN <text>, <graphic mode>
PURPOSE To specify the text or graphic screen modes.
EXAMPLE SCREEN 0,2

*SCROLL
FORMAT SCROLL[<speed>][,<mode>][,<scroll step X>,<scroll step Y>]]
PURPOSE To specify the SCROLL function of the physical screen.
EXAMPLE SCROLL 9,0,10,4

*SOUND
FORMAT SOUND <tone>,<duration>
PURPOSE To sound a specified tone.
EXAMPLE SOUND 10, 10

*STAT
FORMAT STAT [|ALL |]
|<expression>|
PURPOSE To display the status of each programme area.
EXAMPLE STAT 3

STOP
FORMAT STOP
PURPOSE To terminate programme execution and return to command level.
EXAMPLE STOP

SWAP

FORMAT	SWAP <variable 1>,<variable 2>
PURPOSE	To exchange the values of two variables.
EXAMPLE	SWAP A$, B$

*TITLE

FORMAT	TITLE <programme name>
PURPOSE	To name programmes.
EXAMPLE	TITLE"TEST 1"

TRON/TROFF

FORMAT	TRON TROFF
PURPOSE	To trace the execution of programme statements.
EXAMPLE	TRON TROFF

*WIDTH

FORMAT	WIDTH <characters per line>, <number of lines> [,<scroll margin>]
PURPOSE	To set the size of the virtual screen.
EXAMPLE	WIDTH 20, 25, 5

WIDTH <device name>

FORMAT	WIDTH "LPT0:" , <number of digits> "COM0:"
PURPOSE	To set the print width of the printer.
EXAMPLE	WIDTH"LPT0:", 20

WIND

FORMAT	WIND[<counter value>]
PURPOSE	To control the microcassette drive for fast forward and rewind.
EXAMPLE	WIND 0

Functions

ABS

FORMAT	ABS(<numeric expression>)
PURPOSE	To return the absolute value of a numeric expression
EXAMPLE	A=ABS(−1.6)

ASC

FORMAT	ASC(<string>)
PURPOSE	To return the character code of a character.
EXAMPLE	A=ASC("A")

ATN

FORMAT	ATN(<numeric expression>)
PURPOSE	To return the arc tangent of a numeric expression.
EXAMPLE	A=ATN(0.5)

CDBL

FORMAT	CDBL(<numeric expression>)
PURPOSE	To convert integers and single precision numbers into double precision numbers.
EXAMPLE	A#=CDBL(B!/2)

CHR$

FORMAT	CHR$(<numeric expression>)
PURPOSE	To return the character corresponding to a specified character code.
EXAMPLE	A$=CHR$(&H41)

CINT

FORMAT	CINT(<numeric expression>)
PURPOSE	To convert single and double precision numbers into integers.
EXAMPLE	A%=CINT(B#/2)

COS

FORMAT	COS (<numeric expression>)
PURPOSE	To return the cosine of a numeric expression.
EXAMPLE	A=COS(3.1415926/2)

CSNG

FORMAT	CSNG(<numeric expression>)
PURPOSE	To convert integers and double precision numbers into single precision numbers.
EXAMPLE	A!=CSNG(B#)

CSRLIN

FORMAT	CSRLIN
PURPOSE	To return the vertical position of the cursor on the virtual screen.
EXAMPLE	Y=CSRLIN

DATE$

FORMAT	DATE$ [=MM/DD/YY]
PURPOSE	To set the current date in, and return the date kept by, the internal calendar clock.
EXAMPLE	PRINT DATE$

DAY

FORMAT	DAY
PURPOSE	To set the current day of the week in, and display the day of the week kept by, the internal calendar clock.
EXAMPLE	PRINT DAY

EOF

FORMAT	EOF(<file number>)
PURPOSE	To return the end-of-file code.
EXAMPLE	IF EOF(3) THEN CLOSE #1 ELSE GOTO 100

ERL/ERR

FORMAT	ERL ERR
PURPOSE	To return the error code of an occurred error and the line number where the error occurred.

EXAMPLE	A=ERL
	B=ERR

EXP

FORMAT	EXP (<numeric expression>)
PURPOSE	To return the value of an exponential with e as its base.
EXAMPLE	A=EXP(1)

FIX

FORMAT	FIX (<numeric expression>)
PURPOSE	To return the truncated integer part of a numeric expression.
EXAMPLE	A=FIX(−B/3)

FRE

FORMAT	FRE(<expression>)
PURPOSE	To return the size of an unused memory area.
EXAMPLE	PRINT FRE(0) PRINT FRE("A$")

HEX$

FORMAT	HEX$(<numeric expression>)
PURPOSE	To return a string which represents the hexadecimal value of the decimal argument.
EXAMPLE	A$=HEX$(65535)

INKEY$

FORMAT	INKEY$
PURPOSE	To return a one-character string of the pressed character key or a null string if no character key is pressed.
EXAMPLE	A$=INKEY$

INPUT$

FORMAT	INPUT$(<number of characters>[,[#]<file number>])
PURPOSE	To return a string of characters read from a specified file.
EXAMPLE	A$=INPUT$(5,#3)

INSTR

FORMAT	INSTR([<numeric expression>,]<string 1>,<string 2>)
PURPOSE	To search for the first occurrence of one string in another string and returns the position of the searched string.
EXAMPLE	B=INSTR(A$,"XYZ")

INT

FORMAT	INT(<numeric expression>)
PURPOSE	To return the largest integer value (truncated).
EXAMPLE	PRINT INT (−B/3)

LEFT$

FORMAT	LEFT$(<string>,<numeric expression>)
PURPOSE	To return an arbitrary length of string from the leftmost characters of a string.
EXAMPLE	B$=LEFT$(A$,4)

LEN

FORMAT	LEN(<string>)
PURPOSE	To return the total number of characters in a string.
EXAMPLE	A=LEN(A$)

LOF

FORMAT	LOF (<file number>)
PURPOSE	To return the size of a specified file.
EXAMPLE	A=LOF(3)

LOG

FORMAT	LOG(<numeric expression>)
PURPOSE	To return the natural logarithm of a numeric expression.
EXAMPLE	PRINT LOG(2.7812818)

MID$

FORMAT	MID$(<string>,<expression 1>[,<expression 2>])
PURPOSE	To return an arbitrary length of string from a string.
EXAMPLE	B$=MID$(A$,2,3)

OCT$

FORMAT	OCT$(<numeric expression>)
PURPOSE	To return a string which represents the octal value of the decimal argument.
EXAMPLE	PRINT OCT$(123+456)

*PEEK

FORMAT	PEEK(<address>)
PURPOSE	To return the byte read from a specified memory location.
EXAMPLE	A=PEEK(&H0C00)

POINT

FORMAT	POINT(<horizontal coordinate>,<vertical coordinate>)
PURPOSE	To return the status of a dot at a specified location on the graphic screen.
EXAMPLE	PRINT POINT(100,10)

POS

FORMAT	POS(<digit>)
PURPOSE	To return the horizontal position of the cursor on the virtual screen or the horizontal position of the printer head.
EXAMPLE	X=POS(0)

RIGHT$

FORMAT	RIGHT$(<string>,<numeric expression>)
PURPOSE	To return an arbitrary length of string from the rightmost characters of a string.
EXAMPLE	PRINT RIGHT$("ABCD",3)

RND

FORMAT	RND[(<numeric expression>)]
PURPOSE	To return a random number.
EXAMPLE	A=RND(1)

SGN

FORMAT	SGN(<numeric expression>)
PURPOSE	To return the sign of the value of a numeric expression.
EXAMPLE	B=SGN(A)

SIN

FORMAT	SIN(<numeric expression>)
PURPOSE	To return the sine of a numeric expression.
EXAMPLE	PRINT SIN(3.1415926/2)

SPACE$

FORMAT	SPACE$(<numeric expression>)
PURPOSE	To return a string of spaces of a specified length.
EXAMPLE	A$="A"+SPACE$(10)+"C"

SPC

FORMAT	SPC(<digit>)
PURPOSE	To output a specified number of blanks.
EXAMPLE	PRINT SPC(10);"A"

SQR

FORMAT	SQR(<numeric expression>)
PURPOSE	To return the square root of a numeric expression.
EXAMPLE	A=SQR(2)

STR$

FORMAT STR$(<numeric expression>)
PURPOSE To return a string representation of the value of a numeric expression.
EXAMPLE A$=STR$(123)

STRING$

FORMAT STRING$ (<integer expression>, | <string expression> |)
| <numeric expression> |
PURPOSE To return a string of specified characters.
EXAMPLE PRINT STRING$(10,65)

***TAB**

FORMAT TAB(<numeric expression>)
PURPOSE To space to a specified position on the line where the cursor is currently positioned.
EXAMPLE PRINT TAB(10); "ABC"

TAN

FORMAT TAN(<numeric expression>)
PURPOSE To return the tangent of a numeric expression.
EXAMPLE A=TAN(3.1416/4)

***TAPCNT**

FORMAT TAPCNT
PURPOSE To return the value of the microcassette drive counter.
EXAMPLE PRINT TAPCNT
A=TAPCNT

TIME$

FORMAT TIME$="HH:MM:SS"
PURPOSE To return the time kept by the internal calendar clock.
EXAMPLE PRINT TIME$

USR

FORMAT USR[<digit>](<argument>)
PURPOSE To call a machine language subroutine defined by DEFUSR statement.
EXAMPLE A=USR 1(B)

VAL

FORMAT VAL(<string expression>)
PURPOSE To return the numerical value of a string expression.
EXAMPLE A=VAL("−123")

VARPTR

FORMAT VARPTR(<variable name>)
PURPOSE To return the address of a variable or array.
EXAMPLE PRINT HEX$(VARPTR(A))

EPSON OVERSEAS MARKETING LOCATIONS

EPSON AMERICA, INC.
3415 Kashiwa Street
Torrance, CA 90505 U.S.A.
Phone: (213) 539-9140
Telex: 182412

EPSON UK LTD
Dorland House
388 High Road,
Wembley, Middlesex, HA9 6UH,UK
Phone: (01) 900-0466/7/8/9
Telex: 8814169

EPSON DEUTSCHLAND GmbH
Am Seestern 24
4000 Düsseldorf 11
F.R. Germany
Phone: (0211) 596-1001
Telex: 8584786