

National JR-800 — user manual

The [PockEmul](#) guide to the National JR-800

The National (Matsushita / Panasonic) JR-800 is a B5-size hand-held computer sold in Japan from 1983 (list price 128,000 yen). It runs JR-800 BASIC, a Hudson Soft BASIC ("JR-HuBASIC") of the JR family (JR-100, JR-200, JR-300), on an 8-line, 192 × 64 dot liquid crystal display.

This manual describes the machine as [PockEmul](#) emulates it. It was written from the 1983 catalogue specification, the memory map of the original BASIC manual, the command list published on pockemul.com, a complete disassembly of the 32 KB ROM and tests in [PockEmul](#): every keyword of the ROM tables was tried.

All screenshots were taken in [PockEmul](#).



The JR-800 in PockEmul

Contents

1. The machine
2. Getting started
3. The keyboard
4. The display
5. The screen editor
6. Program pages
7. The BASIC language
8. BASIC reference
9. Errors

- 10. Graphics
- 11. Sound
- 12. Clock and alarm
- 13. Function keys
- 14. Cassette tape
- 15. Printer JR-P20
- 16. Serial port
- 17. Machine language
- 18. The extended BASIC ROM and the external units
- 19. PockEmul specifics and limitations
- Appendices: A. Tokens, B. Character codes, C. Memory map, D. Sources

1. The machine

Specifications

From the 1983 catalogue.

Item	JR-800
CPU	Hitachi HD63A01V (CMOS, 6801 family with extra instructions), 4.9152 MHz crystal, 1.2288 MHz system clock
ROM	20 KB standard (16 KB at 8000h-BFFFh + the 4 KB inside the CPU at F000h-FFFFh); 32 KB with the 12 KB extended BASIC ROM at C000h-EFFFh (see section 18)
RAM	16 KB CMOS, battery backed (2000h-5FFFh), 24 KB with 8 KB more inside the case (6000h-7FFFh)
Display	LCD, 32 characters × 8 lines, 192 × 64 dots; 16 mode indicators; 192 characters + 32 user-defined; inverse and underline; contrast slider
Keyboard	79 keys, JIS layout with numeric pad; 10 PF keys giving 20 functions; caps lock, graphic, kana, function and control modes; one-key keywords and abbreviations; type-ahead, auto repeat, key click
Programs	8 independent program pages, battery backed; menu with auto-start
Cassette	FSK, about 1800 baud on average; recommended recorder RQ-8300
Serial	3-wire cable, 300 / 1200 / 9600 / 76800 baud: JR-P20 printer, link between two JR-800s, JH-600 voice input unit
Memory pack	28-pin connector: 8 KB RAM or 12 KB ROM packs
System bus	40-pin connector for the JR-U20 I/O unit
Sound	small dynamic speaker, one voice, 5 octaves
Clock	calendar clock (hour, minute, second, year, month, day, day of the week) with wake-up
Power	4 × AA (6 V, 400 mW max), AC adapter JL-045; auto power off after about 6 minutes
Battery life	about 70 h of keyboard input, 25 h of program execution

Item	JR-800
Size, weight	260 × 143 × 34 mm, about 710 g with batteries
Accessories	recording cable, serial cable, demonstration tape, BASIC manual, case, PF key card

The top side carries, from left to right: the LCD contrast slider, the serial jack (シリアル端子), the 40-pin external bus, the three tape recorder jacks (microphone, remote, monitor) and the 6 V power jack.

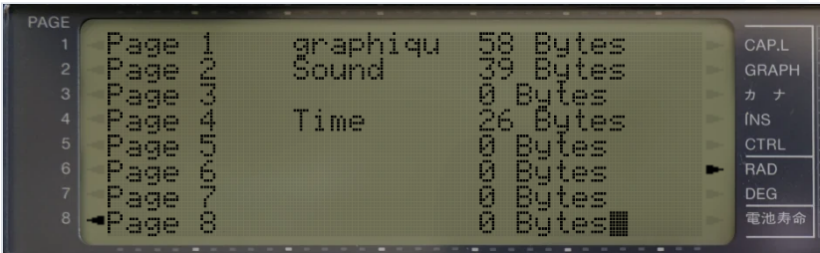
The emulated machine

PockEmul emulates a JR-800 with 24 KB of RAM and the full 32 KB ROM, so the extended BASIC ("JR-HuBASIC 2.0") is always present. The tape jacks, the serial jack (JR-P20 printer or **PockEmul**'s Serial Terminal), the clock with its alarm, the sound and the display with its markers are emulated. The external bus is empty: the commands of its units end with Error 29 (see [section 19](#)).

Note: the cassette tape ([section 14](#)), the serial port ([section 16](#)) and the alarm (WAKE , [section 12](#)) are still being tested in **PockEmul**: they will only be available in a future version of **PockEmul**.

2. Getting started

Create the pocket from the "New Pocket" list (**National JR-800**). **PockEmul** starts it with a demonstration session: the screen lists the 8 program pages, three of them hold small demonstration programs.



Page list at power on

Press **BREAK** to leave the list, then **HOME/CLS** to clear the screen. `Ok` and a blinking block cursor show that BASIC waits for a command. Type a command and press **RETURN**:

```
PRINT 6*7
```

A line that starts with a number is stored as a program line in the current page:

```
10 REM HELLO
20 FOR I=1 TO 5
30 PRINT "HELLO";I;SPC(I);""
40 NEXT I
LIST
RUN
```

```

40 NEXT I
LIST
10 REM HELLO
20 FOR I=1 TO 5
30 PRINT "HELLO";I;SPC(I);"*"
40 NEXT I
Ok

```

LIST

```

RUN
HELLO 1      *
HELLO 2      *
HELLO 3      *
HELLO 4      *
HELLO 5      *
Ok

```

RUN

Power

Key	Effect
OFF	switches off at once; memory, programs and the state of a running program are kept
ON	switches on; the JR-800 resumes exactly where it was switched off, even inside a running program or an <code>INPUT</code>
RESET (button under the display)	cold start: memory test, every page is erased , the PF keys and the clock are cleared, JR-HuBASIC 2.0 banner
MENU	page menu (see Program pages)

Holding **R**, **S** and **T** while switching on also forces a cold start. The JR-800 switches itself off after about 6 minutes without a key while it waits for a command or in the menu; a running program is never switched off. In [PockEmul](#) the PC key **F11** is the **ON** key.

3. The keyboard

The key layout is on the picture at the top of this manual.

Main keys

Key	Unshifted	With SHIFT
letters	lower case	upper case (keywords may be typed in lower case)
digit row	0-9	@ ! " # \$ % & ' () (shown on the keys)
: ; , ^ .	: ; , ^ .	_ ? < ¥ >
INS/RUB	RUB: delete the character left of the cursor	INS: insert a space at the cursor
↑/↓	cursor up	cursor down
←/→	cursor right	cursor left
HOME/CLS	CLS: clear the screen, cursor home	HOME: cursor to the top left corner
RETURN	enter the line under the cursor	
BREAK	stop a program, a listing, a tape or printer operation; abandon the line being typed	

Key	Unshifted	With SHIFT
MENU	page menu	
PF1-PF10	function keys PF1-PF10	PF11-PF20
numeric pad	0-9 . = + - * /	same

CTRL: keywords and modes

The words printed above the keys are typed with **CTRL** + the key:

Key	CTRL + key	Key	CTRL + key	Key	CTRL + key
A	AUTO	J	READ	S	STOP
B	RESTORE	K	KEY	T	THEN
C	CLEAR	L	LIST	U	TITLE
D	DIM	M	DATA	V	REM
E	END	N	NEXT	W	RETURN
F	FOR	O	PAGE	X	CONT
G	GOTO	P	PRINT	Y	LOCATE
H	POKE	Q	GOSUB	Z	SOUND
I	INPUT	R	RUN		

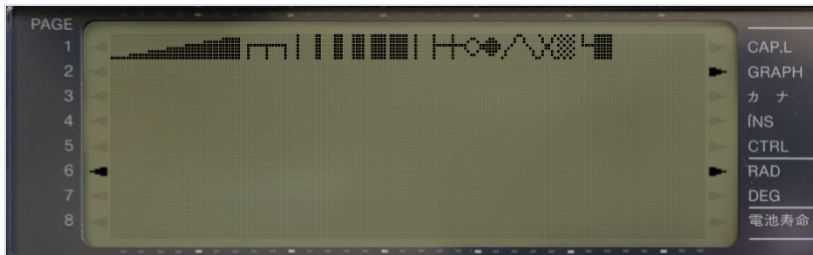
Key	CTRL + key
1	ERASE: erase from the cursor to the end of the logical line
2	HCOPY: print the screen on the JR-P20
3 to 7	type SAVE, LOAD, VERIFY, OPEN, CLOSE
8	CTRL ON/OFF: switch the CTRL mode (CTRL marker)
9	CAP.L: caps lock, letters upper case unshifted (CAP.L marker)
0	GRAPH: graphic characters (GRAPH marker)
^	カナ: katakana (カナ marker); GRAPH and kana cancel each other
; :	cursor left / right (arrows printed above the keys)
, .	cursor up / down

With the CTRL mode on (**CTRL+8**, CTRL marker lit), CTRL + a letter gives the control code 01h-1Ah instead of a keyword; CTRL+8 again returns to the keyword mode. In CTRL mode **CTRL+A** switches the insert mode (INS marker): the characters typed are then inserted; any cursor movement ends it.

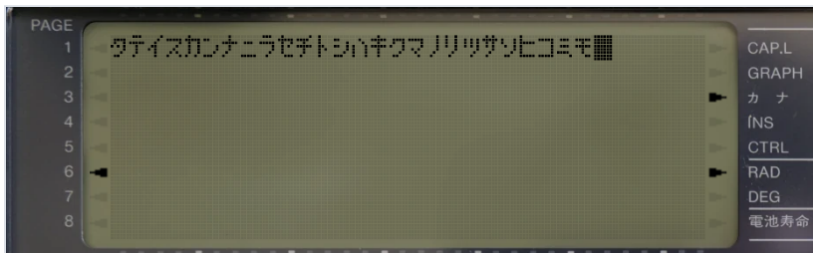
Keyword keys and modes act while a line is typed (command line, **INPUT**). **INKEY\$** returns the raw codes (see [Appendix B](#)).

GRAPH and kana modes

In GRAPH mode the letter keys and `,` `.` `:` `;` `^` give the graphic characters 80h-9Eh drawn on the key fronts; digits and the numeric pad keep their characters. In kana mode the keys give the katakana of the JIS layout printed on them, SHIFT the small kana.



GRAPH mode: q to m typed



Kana mode: q to m typed

Abbreviations

A keyword may be abbreviated by its first letters followed by a period: the JR-800 takes the first keyword of its table that starts with these letters (table order: GOTO, GOSUB, RUN, RETURN, RESTORE, RESUME, LIST, LLIST, DELETE, EDIT, AUTO, FOR, NEXT, PRINT, INPUT, IF, DATA, READ, DIM, REM, ...; see [Appendix A](#)). So `P.` is PRINT, `G.` GOTO, `GOS.` GOSUB, `R.` RUN, `L.` LIST. `?` is PRINT too.

Typing aids

- Type-ahead: 63 keys are kept while the JR-800 is busy.
- Auto repeat after about 0.7 s, then about 11 keys per second (not for the PF keys, the keyword keys, BREAK and MENU).
- Key click: a short beep for each key; `POKE &H89,1` silences it, `POKE &H89,0` brings it back.
- Holding **SHIFT** pauses a listing (`LIST` , `KEY LIST` ...), holding **SHIFT + CTRL** pauses a running program.

The PC keyboard in PockEmul

Letters, digits, space and RETURN are typed from the PC keyboard; **F1-F10** are PF1-PF10 (with Shift: PF11-PF20), **Insert** is INS/RUB, `↑` and `→` are the two cursor keys (use Shift for `↓` and `←`), **Shift** and **Ctrl** are SHIFT and CTRL, **F11** is ON. BREAK, HOME/CLS, MENU and OFF are clicked on the picture. A character that needs SHIFT on the JR-800 but not on the PC may not get through: to enter a program, paste it in [PockEmul](#)'s keyboard simulator (Keyboard tab), which types every character with the right JR-800 keys.

4. The display

Text screen

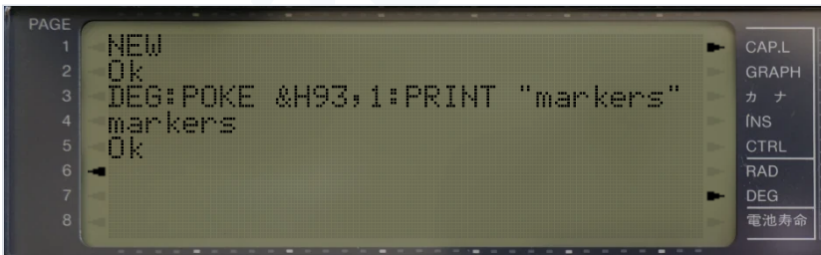
The LCD shows a window of 32 columns × 8 lines on a **virtual screen** of 40 columns × 16 lines (after a cold start). Characters are 6 × 8 dots. The window follows the cursor: always vertically, and horizontally while a line is typed (with a margin of 4 columns), so a long line can be typed and edited.

Statement / function	Use
<code>WIDTH c,l</code>	size of the virtual screen: 32 to 254 columns, at least 8 lines; clears the screen and stops a program
<code>LOCATE x,y</code>	cursor at column x, line y of the virtual screen (from 0)
<code>WLOCATE x,y</code>	top left corner of the LCD window on the virtual screen; the screen is redrawn
<code>POS(0) , CSRLIN</code>	cursor column, line
<code>SCRN\$(x,y,n)</code>	n characters read on the screen from column x, line y
<code>CLS</code>	clear the screen

```
WIDTH 64,20
LOCATE 60,19:PRINT "END";:WLOCATE 32,12
```

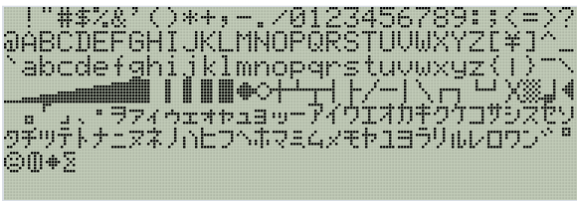
Markers

Left of the LCD, the **PAGE 1-8** column marks the current program page. Right of it, eight markers show the modes: **CAP.L**, **GRAPH**, **カナ** (kana), **INS** (insert), **CTRL** (CTRL mode), **RAD**, **DEG** (angle unit; none for GRAD) and **電池寿命** (batteries low).



Markers: page 6, CAP.L and DEG

Characters



Character set, codes 20h to FFh

Codes	Characters
00h-1Fh	control codes, see below (shown as spaces when read by <code>SCRN\$</code>)
20h-7Fh	ASCII (5Ch is ¥, 7Eh overline, 7Fh backslash)
80h-9Fh	graphics: bars, lines, corners, ●, ○, ♪ ... (GRAPH mode, 9Fh only by <code>CHR\$</code>)
A0h-DFh	katakana (JIS X 0201)
E0h-FFh	32 user-defined characters

The user-defined characters are kept at 2000h-20BFh, 6 bytes per character (one byte per column, left to right, bit 0 at the top, bit 7 = 8th row): character E0h + n uses addresses &H2000 + 6 × n to &H2005 + 6 × n. They are cleared by a cold start. The screenshot above defines E0h-E3h with:

```
10 FOR I=0 TO 23:READ A:POKE &H2000+I,A:NEXT
20 DATA 28,34,85,81,85,34,28,0,62,65,127,65,62,0,8,28,62,28,8,0,99,85,73,85
30 PRINT CHR$(&HE0);CHR$(&HE1);CHR$(&HE2);CHR$(&HE3)
```

Control codes

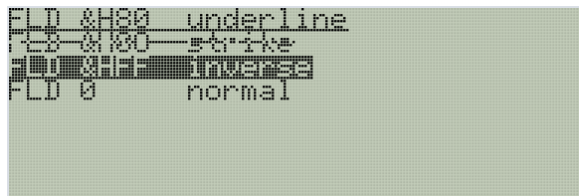
They act when printed (`PRINT CHR$(n)`) or typed:

Code	Key	Action
01h	CTRL+A (CTRL mode)	insert mode on / off (INS marker)
05h	CTRL+1 (ERASE)	erase from the cursor to the end of the logical line
08h	RUB	delete the character left of the cursor
0Bh	SHIFT+HOME	cursor home
0Ch	CLS	clear the screen
0Dh	RETURN	new line
10h	CTRL+2 (HCOPY)	screen copy on the printer
12h	SHIFT+INS	insert a space
1Ch, 1Dh, 1Eh, 1Fh	→ ← ↑ ↓	cursor moves

Other codes are ignored (no bell, no tab).

Underline, strike-through, inverse: FLD

`FLD n` (0-255) is XORed with each column of the characters drawn after it: `FLD &H80` underlines, `FLD &H08` strikes through, `FLD &HFF` inverts, `FLD 0` returns to normal. The mask applies when a character is drawn: when the screen scrolls or is redrawn, the whole window is drawn again with the current mask.



FLD

5. The screen editor

The JR-800 has a full-screen editor: any line on the screen can be changed with the cursor keys, INS and RUB, and entered again with RETURN. RETURN takes the whole **logical line** under the cursor (a line typed past the right edge continues on the next row), at most 128 characters, trailing spaces removed. An odd number of " gets a closing " .

Command	Use
LIST [range]	list the current page; range <code>n</code> , <code>n-</code> , <code>-m</code> , <code>n-m</code> (or <code>n,m</code>) , <code>.</code> = last line used
LLIST [range]	the same on the printer
EDIT [line]	show the line with the cursor on its text, ready to be changed
AUTO [start] [,step]	automatic line numbers (default 10,10); an existing line is shown for editing; BREAK ends
RENUM [new][,][old] [,step]]	renumber the lines from <i>old</i> (default: first) as <i>new</i> , <i>new+step</i> ... (default 10,first,10) and update GOTO, GOSUB, THEN, ELSE, RESTORE, RESUME, RETURN and ERL= references
DELETE first,last	delete lines <i>first</i> to <i>last</i> (both must exist; note the comma)
FIND "text"	list the lines of the current page that contain <i>text</i>
LFIND "text"	the same on the printer

A line with the number of an existing line replaces it; a number alone deletes the line. Line numbers go from 1 to 65534.

6. Program pages

The JR-800 keeps 8 independent programs, **pages 1 to 8**, each with a title of up to 8 characters. Program lines, RUN , LIST , GOTO ... always act on the current page; variables are common.

Command	Use
PAGE <code>n</code>	select page <code>n</code> (1-8) and show its line
PAGE <code>n,R</code>	select page <code>n</code> , clear the variables and run it (from a program too: chaining)
PAGE	show the line of the current page
PAGE LIST	show the 8 pages
TITLE "name"	title of the current page (8 characters; TITLE alone clears it)

Command	Use
NEW	erase the program of the current page only
NEW ON address	erase all the pages and start the BASIC area at <i>address</i> , keeping the memory below for machine code (from &H2E00)
MENU key	clear the screen, list the pages; press 1-8 to run that page, BREAK to leave

The page line shows the number, the title and the size in bytes:

```

Page 3      HELLO      28 Bytes
Page 4      Time      26 Bytes
Page 5              0 Bytes
Page 6              0 Bytes
Page 7              0 Bytes
Page 8              0 Bytes
Ok

```

PAGE LIST



MENU

With **MENU** and a digit, a page is started with two key presses: this is the "menu with auto-start" of the catalogue.

7. The BASIC language

Lines and statements

A program line is a number (1-65534) followed by statements separated by **:**. Keywords are recognised anywhere, spaces are optional and lower case is accepted; a variable name must not contain a keyword (**TOTAL** contains **TO**).

' is a short **REM** (listed as **REM**).

Numbers

Type	Suffix	Range	Precision
integer	%	-32768 to 32767	exact
single precision	! (exponent E)	about 1E-154 to 1E153	10 digits
double precision	# (exponent D)	same	20 digits

A variable without suffix is single precision unless a **DEFINT** , **DEFSNG** , **DEFDBL** or **DEFSTR** statement gave another type to its first letter. A constant with more than 10 digits, a **#** or a **D** exponent is double precision. **&H** gives a hexadecimal constant, **&O** an octal one (16 bits, **&HFFFF** = -1).

```
PRINT 1/3;1#/3
.3333333333 .33333333333333333333
```

Numbers print with a leading space (or -) and a trailing space, in fixed notation while it needs at most 10 digits in single precision (1E9 prints 1000000000, 1E-9 prints .000000001), otherwise as 1E+10 (D for double precision). An integer variable receives the value modulo 65536: A%=40000 stores -25536.

Strings

String variables end with \$, constants are between "; at most 255 characters (Error 15 beyond). + joins strings.

Variables and arrays

Names start with a letter, followed by letters and digits; **only the first 2 characters count** (ABC and ABD are the same variable). A, A%, A# and A\$ are different variables. Arrays must be declared by DIM before use (no automatic dimension); indexes start at 0, any number of dimensions.

Operators

In order of precedence:

1. ^ power
2. unary -, +
3. *, /
4. +, - (and + for strings)
5. comparisons =, <> or <>, <, >, <= or <=, >= or >= (numbers or strings), result -1 (true) or 0 (false)

There are **no logical operators** (no AND, OR, NOT, MOD). As the catalogue says, the logical operations are written with * (and) and + (or) on comparisons, true being -1:

```
IF (A>0)*(A<10) THEN PRINT "1 TO 9"
IF (A<0)+(A>9) THEN PRINT "OUT"
```

OR and XOR are words of the extended BASIC that only exist as drawing modes of the external graphics unit; in an expression they give Error 2.

Angles

RAD (radians, after a cold start), DEG (degrees) and GRAD (grades) select the unit of SIN, COS, TAN, ATN, ASN, ACS; the RAD and DEG markers show it.

8. BASIC reference

Syntax conventions: [] optional, | alternatives, *n* a numeric expression, *a\$* a string expression, *line* a line number written as a number. Unless stated otherwise a numeric argument outside its range gives Error 5. Keywords of the extended BASIC ROM are marked (ext); those that need an external unit are described in [section 18](#).

Statements

Statement	Description
AUTO [start][,step]	automatic line numbers, see section 5
CALL address	call a machine-code routine (see section 17)
CHAIN "name" (ext)	load a program from tape into the current page and run it, keeping the variables
CLEAR	clear all variables and arrays, the DATA pointer, the GOSUB / FOR stack, ON ERROR and the DEF types (no argument)
CLOSE [[#]n[,...]]	close a file (see section 14)
CLS	clear the screen
COLOR , SCREEN , CIRCLE , PAINT , DEF CHR\$ (ext)	external colour graphics unit, see section 18
CONT	continue after STOP or BREAK (Error 17 when not possible)
DATA item[,item...]	data for READ: numbers, or strings with or without "
DEF FNname[(p1,...,p10)]=expression (ext)	define a function, called by FNname(...) ; redefinition gives Error 10
DEFINT , DEFSNG , DEFDBL , DEFSTR letter[,letter...]	default type of the variables starting with these letters (single letters only, no ranges)
DEG	angles in degrees
DELETE first,last	delete program lines
DIM name(n1[,n2...])[,...]	declare arrays (indexes 0 to n)
EDIT [line]	edit a program line
END	end of the program, files closed
ERROR n	raise error n (0-255)
FIND "text"	list the lines containing text
FLD n	display mask (underline, inverse...)
FOR v=start TO end [STEP step] ... NEXT [v[,v...]]	loop; the body is skipped when start is already past end; Error 23 when no NEXT is found
GOSUB line / RETURN [line]	subroutine; RETURN line returns and jumps to line
GOTO line	jump (the line must be a number, not an expression)
GRAD	angles in grades
HCOPY	screen copy on the printer
IF condition THEN line statements [ELSE line statements]	condition, 0 = false; THEN may be omitted before a statement (IF A GOTO 100); the rest of the line belongs to the IF
INPUT ["prompt"; ,] v[,v...]	read values from the keyboard; ; adds ? , , does not; an empty answer keeps the variables

Statement	Description
INPUT #n, v[,v...]	read from a file
INPUT\$ [#]n,count,v\$ (ext)	read <i>count</i> characters from device n (0 = keyboard without echo, 1 = tape, 2 = serial)
KEY n,a\$ / KEY LIST / KEY 0,a\$	function keys, see section 13
LET v=expression	assignment (LET is optional)
LFIND "text"	FIND on the printer
LINE [(x1,y1)]-(x2,y2)[,PSET ,PRESET]	draw a line, see section 10
LINE INPUT ["prompt"; ,] v\$ (ext)	read a whole line, commas included, into a string variable (no ?)
LIST [range] / LLIST [range]	list the program on the screen / printer
LOAD ["name"][,R]	load a program from tape
LOCATE x,y	move the cursor
LPOUT a\$, INITP, LPT n (ext)	parallel printer, see section 18
LPRINT ...	PRINT on the printer
MERGE ["name"]	add an ASCII program from tape to the current page
MLOAD ["name"][,address][,R]	load machine code from tape
MON (ext)	machine-language monitor, see section 17
MSAVE "name",start,end[,exec]	save machine code on tape
MUSIC a\$, TEMPO n (ext)	play a melody, see section 11
NEW [ON address]	erase the current page / all pages
ON n GOTO GOSUB RESTORE RETURN RESUME line1,line2,...	n-th line of the list; nothing when n is out of range
ON ERROR GOTO line	error trap, see section 9 ; ON ERROR GOTO 0 cancels it
OPEN "I" "O",[#]n,"name"	open a data file
PAGE [n[,R]] / PAGE LIST	program pages
POKE address,byte[,byte...]	write bytes in memory
PRINT [#n,] [items] / ? ...	display; ; joins, , goes to the next 10-column zone; TAB(n) , SPC(n) ; a final ; or , keeps the line
PRINT [#n,] USING "format";n[,n...] (ext)	formatted numbers, see below
PRESET (x,y) / PSET (x,y)	clear / set a dot
RAD	angles in radians
RANDOMIZE n (ext)	change the random sequence
READ v[,v...]	read the next DATA items (Error 4 past the last one)

Statement	Description
REM text / ' text	comment
RENUM [new][,[old][,step]]	renumber
RESTORE [line]	DATA pointer to the start of the page or to <i>line</i>
RESUME line	end of an error trap, continue at <i>line</i>
RUN [line]	clear the variables and run the current page
SAVE ["name"][,A]	save the current page on tape (,A : as text)
SINIT n	serial port rate, see section 16
SLOAD / SSAVE [range]	program from / to the serial port
SOUND pitch,duration	tone, see section 11
STOP	stop, shows Break in line ; CONT continues
TITLE ["name"]	title of the current page
TRON / TROFF	trace: each program line shows its number in []
VERIFY ["name"]	compare a tape file with memory
WAKE "hh:mm"	alarm: switch off and on again at hh:mm
WIDTH columns,lines	virtual screen size
WLOCATE x,y	LCD window position
TIME\$ = "hh:mm:ss" , DATE\$ = "yy/mm/dd" , DAY\$ = "SUN" ... "SAT"	set the clock

```

TRON:RUN
[10][20] 1
[30][20] 2
[30]
Ok
TROFF
Ok

```

TRON

INPUT "NAME";N\$ asks NAME? , INPUT "AGE ",A prints AGE without ? :

```

RUN
NAME? JR
AGE 1983
JR IS 1983
Ok

```

INPUT

PRINT USING (ext)

PRINT USING "format";n1[,n2...] (also LPRINT USING , PRINT #n,USING) prints numbers with the fields of the format; the values are separated by , , the format is used again when it is exhausted, other characters are printed as they are.

Field	Meaning
#	one digit
.	decimal point (up to 19 decimals)
, after the last # of the integer part	thousands separator
+ first	sign always printed
+ or - last	sign after the number (- : only for negative numbers)
** first	leading blanks filled with *
¥¥ first	floating ¥ sign (¥ is the character 5Ch, SHIFT+^)
^^^^	exponent notation (E+ddd)

A number too wide for its field gives Error 5 ; there are no string fields.

```
PRINT USING "###.##";3.14159      → 3.14
PRINT USING "+###";5              → +5
PRINT USING "###-";-5             → 5-
PRINT USING "***###";12           → ***12
PRINT USING "#.##^"12345          → 0.12E+005
```

Functions

A function whose code is below AAh (see [Appendix A](#)) needs its argument in parentheses even when it does not use it: PI(1) , FRE(0) , POS(0) .

Function	Result
ABS(n)	absolute value
ACS(n) , ASN(n) , ATN(n)	arc cosine, arc sine, arc tangent, in the current angle unit
ASC(a\$)	code of the first character (0 for "")
CALC(a\$)	value of the expression written in a\$; only inside a program (Error 12 in direct mode)
CDBL(n) , CSNG(n) , CINT(n)	conversion to double, single precision, integer (CINT rounds; 40000 gives -25536)
CHR\$(n)	character of code n
COS(n) , SIN(n) , TAN(n)	trigonometry, in the current angle unit
CSRLIN	cursor line
CVI(a\$) , CVS(a\$) , CVD(a\$) (ext)	number from a 2-, 6- or 10-byte string made by MKI\$, MKS\$, MKD\$
DATE\$, TIME\$, DAY\$	"yy/mm/dd" , "hh:mm:ss" , "SUN" ... "SAT"
EOF(n)	-1 at the end of input file n, else 0
ERL , ERR	line and number of the last trapped error

Function	Result
EXP(n)	e to the power n
FIX(n)	integer part (towards 0)
FN name(...) (ext)	user function (DEF FN)
FRE(n)	free memory in bytes (n is not used)
HEX\$(n) , OCT\$(n)	hexadecimal, octal string (HEX\$(-1) = FFFF)
INKEY\$	key being pressed, once per press, "" if none; does not wait
INKEY\$(n)	n = 0: key being pressed, repeated while it is held; n = 1: waits for a key; n ≥ 2: one byte of the tape input file
INSTR([start,]a\$,b\$)	position of b\$ in a\$, 0 if not found
INT(n)	largest integer not above n (INT(-2.5) = -3)
LEFT\$(a\$,n) , RIGHT\$(a\$,n) , MID\$(a\$,start[,n])	parts of a string
LEN(a\$)	length
LN(n) , LOG(n)	natural, decimal logarithm
LPOS(n)	printer column
MEM\$(address,n)	string of the n bytes in memory from address
MKI\$(n) , MKS\$(n) , MKD\$(n) (ext)	number as a 2-, 6-, 10-byte string
PEEK(address)	byte in memory
PI(n)	$\pi \times n$ (PI alone gives Error 2)
POINT(x,y)	-1 if the dot is set, else 0
POS(n)	cursor column
RND(n)	random number from 0 to below 1 (the value of n is not used)
SCRN\$(x,y,n)	n characters read on the screen
SGN(n)	-1, 0 or 1
SPC(n)	n spaces
SQR(n)	square root
STR\$(n)	number as printed (leading space)
STRING\$(n,c) , STRING\$(n,a\$)	n times the character of code c / the first character of a\$
TAB(n)	in PRINT: go to column n
VAL(a\$)	number at the start of a\$ (VAL("&H10") = 16)

DSKF , LOC , LOF and FIELD\$ (ext) are for disk files, see [section 18](#).

9. Errors

An error stops the program with `Error n in line`, or `Error n` in direct mode; `BREAK` and `STOP` show `Break` or `Break in line`. The JR-800 shows only the number; the names below come from the code that raises each error.

```
OK
?SQR(-1)
Error 5
OK
RUN
Error 11 in 20
OK
```

Errors

Error	Meaning	Typical causes
1	NEXT without FOR	NEXT with no matching FOR
2	Syntax error	wrong statement, unknown keyword (<code>BEEP</code>), <code>5 XOR 3</code> , <code>PI</code> without argument
3	RETURN without GOSUB	
4	Out of DATA	READ past the last DATA item
5	Illegal function call	argument out of range (<code>SQR(-1)</code> , <code>CHR\$(256)</code> , <code>PAGE 9</code>), <code>RESUME</code> outside an error trap
6	Overflow	number too large, integer conversion above 65535
7	Out of memory	program, variables, strings, GOSUB / FOR nesting
8	Undefined line number	GOTO, GOSUB, RESTORE, DELETE... to a missing line; <code>RESUME</code> without a line number
9	Subscript out of range	index beyond DIM, array not dimensioned
10	Duplicate definition	DIM of an existing array, DEF FN twice
11	Division by zero	
12	Illegal direct	<code>CALC</code> in direct mode
13	Type mismatch	string where a number is expected, or the reverse
15	String too long	more than 255 characters
17	Can't continue	CONT after an error, a change of the program, END...
22	Missing operand	statement without its arguments (<code>KEY</code> , <code>WIDTH</code> , <code>SINIT</code>)
23	FOR without NEXT	
29	Device I/O error	no answer from the printer (no JR-P20 plugged), missing external unit
30	Tape read error	checksum error, VERIFY difference
31	Bad file mode	wrong file type (<code>LOAD</code> of a machine-code file...), wrong OPEN mode, PRINT # to an input file
50	Bad file number (ext)	file number not allowed for this statement (<code>OPEN "0",#2,...</code> , <code>LOC(1)</code>)
54	File already open	

Error	Meaning	Typical causes
55	Bad file name (ext)	disk file name without 1: or 2:
61	Input past end	reading past the end of a file
63	File not found (ext)	disk
64	File already exists (ext)	disk (RENAME)
67	Disk full (ext)	disk
69	Bad file structure (ext)	disk
70	Bad record number (ext)	disk random file
71	File not open	I/O on a file that is not open

ERROR `n` raises any error from 0 to 255.

Error trapping

```

10 ON ERROR GOTO 100
20 INPUT "N";N
30 PRINT 100/N:GOTO 20
100 PRINT "ERROR";ERR;"IN";ERL
110 RESUME 20

```

- `ON ERROR GOTO line` traps the errors of a running program (not in direct mode); `ON ERROR GOTO 0` cancels the trap. `END`, `NEW`, `RUN` and `CLEAR` cancel it too.
- When an error is trapped, `ERR` and `ERL` give its number and line, the active `FOR` loops and `GOSUBs` are forgotten, and the program goes on at the trap line. An error in the trap routine, before `RESUME`, stops the program.
- `RESUME line` ends the trap routine and continues at `line`. There is no `RESUME`, `RESUME 0` or `RESUME NEXT`: `RESUME` without a line number gives Error 8, which is trapped again — the program loops until `BREAK`.
- `BREAK` cannot be trapped.

10. Graphics

The 192×64 dots of the LCD have coordinates $x = 0$ to 191 (left to right) and $y = 0$ to 63 (top to bottom). Graphics and text share the screen: text output and scrolling overwrite drawings.

Statement / function	Use
<code>PSET (x,y)</code>	set a dot
<code>PRESET (x,y)</code>	clear a dot
<code>LINE (x1,y1)-(x2,y2)</code>	draw a line
<code>LINE -(x2,y2)</code>	line from the end of the previous <code>LINE</code>

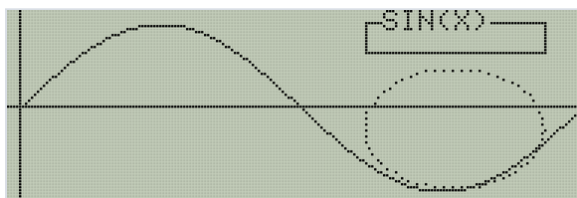
Statement / function	Use
LINE ...,PRESET	erase a line (,PSET draws, the default)
POINT(x,y)	-1 if the dot is set, 0 if not
HCOPY	copy the screen on the printer

Coordinates are 0 to 255 (Error 5 beyond); dots outside the screen are ignored. There is no box, circle or fill on the LCD (LINE ...,B , CIRCLE and PAINT are for the external graphics unit).

```

10 CLS:LINE (0,32)-(191,32):LINE (4,0)-(4,63)
20 FOR X=4 TO 191:Y=32-INT(28*SIN((X-4)/30)):PSET (X,Y):NEXT
30 FOR T=0 TO 62:A=T/10:PSET (150+INT(30*COS(A)),40+INT(20*SIN(A))):NEXT
40 LINE (120,4)-(180,4):LINE -(180,14):LINE -(120,14):LINE -(120,4)
50 LOCATE 21,0:PRINT "SIN(X)";
60 GOTO 60

```



Graphics

11. Sound

SOUND

SOUND pitch,duration plays a square wave on the speaker.

- *pitch* 1 to 65535: the frequency is about $1228800 / (8 \times \text{pitch} + 53)$ Hz, so the larger the value, the lower the tone.
- *duration* 0 to 255 (0 = 256): units of about 6.7 ms for pitch values below 1023 (150 is about one second); above, one unit is one period.
- BREAK stops the sound.

Note	Frequency	pitch
C4	262 Hz	580
A4	440 Hz	342
C5	523 Hz	287
C6	1047 Hz	140

```
SOUND 342,150
```

MUSIC and TEMPO (ext)

MUSIC a\$ plays the notes of a\$; spaces are ignored, BREAK stops.

Element	Meaning
C D E F G A B	note (lower case accepted)
R	rest
# before the note	sharp
+ or - before the note	one octave up or down, for this note only
digit 0-9 after the note	length, kept for the next notes: 0 = 1/32, 1 = 1/16, 2 = dotted 1/16, 3 = 1/8, 4 = dotted 1/8, 5 = 1/4 (default) , 6 = dotted 1/4, 7 = 1/2, 8 = dotted 1/2, 9 = whole

TEMPO n (1-255, 12 after a cold start) sets the speed: the length of the notes is proportional to n.

```
TEMPO 8:MUSIC "C4DEFGAB+C7"
```

Key click

Each key gives a short click (SOUND 90,1); POKE &H89,1 removes it.

12. Clock and alarm

Note: the alarm (WAKE) is still being tested in [PockEmul](#): it will only be available in a future version of [PockEmul](#).

The calendar clock keeps the time, the date and the day of the week while the machine is switched off.

Use	Syntax
read	TIME\$ = "hh:mm:ss" (24 hours), DATE\$ = "yy/mm/dd" , DAY\$ = "SUN" , "MON" , "TUE" , "WED" , "THU" , "FRI" , "SAT"
set	TIME\$="12:34:56" , DATE\$="83/12/13" , DAY\$="TUE" (exactly 8 or 3 characters, upper case)
keep a digit	? in place of a digit leaves it unchanged: TIME\$="?:?:00:?" sets only the minutes
alarm	WAKE "hh:mm"

The digits are not range checked. A cold start clears the clock to 00:00:00 00/00/00; set it again.

WAKE "hh:mm" sets the alarm and switches the JR-800 off; at hh:mm it switches on again and the program goes on after WAKE:

```
10 WAKE "07:30"  
20 MUSIC "CEG+C"
```

In **PockEmul** the clock runs with the emulated time, is saved with the session and moves on by the time elapsed while **PockEmul** was closed.

13. Function keys

PF1-PF10 and, with SHIFT, **PF11-PF20** type a string of up to 10 characters, defined by `KEY n,a$` ($n = 1$ to 20). `CHR$(13)` in the string acts as RETURN. The keys are empty after a cold start.

```
KEY 9,"LIST"+CHR$(13):KEY 10,"RUN"+CHR$(13)
KEY 11,"PRINT ":KEY 12,"GOTO "
KEY LIST
```

`KEY LIST` shows the 20 strings, two per line (PF1 PF2, PF3 PF4...); the control characters appear as their code + 60h (`CHR$(13)` as `m`).



KEY LIST

`KEY 0,a$` puts the string in the keyboard buffer, as if it was typed:

```
10 KEY 0,"PAGE 2,R"+CHR$(13)
```

The function keys act while a line is typed; `INKEY$` returns E0h-E9h for PF1-PF10 and EAh-F3h for PF11-PF20. The PF key card above the keys holds the user's labels.

14. Cassette tape

Note: the cassette tape is still being tested in **PockEmul**: it will only be available in a future version of **PockEmul**.

Connection

In **PockEmul**, plug a tape recorder (CE-152) into the **Line in / Rec / Rmt** jack of the JR-800. In the recorder, load a WAV file (an existing tape, or a new file to record), then press RECORD before `SAVE` or PLAY after `LOAD`. Real JR-800 tapes recorded as WAV files load in **PockEmul**.

Programs

Command	Use
<code>SAVE ["name"]</code>	save the program of the current page
<code>SAVE "name",A</code>	save it as text (ASCII), for MERGE

Command	Use
LOAD ["name"]	load a program into the current page (the old program of the page is erased, the variables cleared)
LOAD "name",R	load and run
MERGE ["name"]	add a text program (saved with ,A) to the current page: same line numbers replace
CHAIN "name" (ext)	load and run, keeping the variables
VERIFY ["name"]	compare a tape file with memory (Error 30 if different)
MSAVE "name",start,end[,exec]	save memory <i>start</i> to <i>end</i> (machine code)
MLOAD ["name"][, [address]] [,R]	load it back, at <i>address</i> (from &H2000) if given; ,R runs it from <i>exec</i>

- Names have up to 16 characters and must match exactly (upper / lower case); without a name the next file is taken.
- In direct mode the JR-800 shows `Writing name` , `Found name` (each file met) and `Loading name` .
- `SAVE` waits about 6 s (motor start) before writing.
- `BREAK` stops a tape operation.

```
SAVE "PROG"
Writing PROG
Ok
SAVE "PROGA",A
Writing PROGA
Ok
```

SAVE

```
VERIFY "PROG"
Found PROG
Ok
```

VERIFY

```
Found PROG
Loading PROG
Ok
LIST
10 PRINT "PROG"
20 PRINT "LINE 20"
Ok
```

LOAD

```
Found PROGA
Ok
LIST
10 PRINT "PROG"
20 PRINT "LINE 20"
30 PRINT "LINE 30"
Ok
```

MERGE

```
MLOAD "CODE",&H7100
Found CODE
Loading CODE
Ok
PRINT PEEK(&H7100);PEEK(&H7103)
1 4
Ok
```

MLOAD at another address

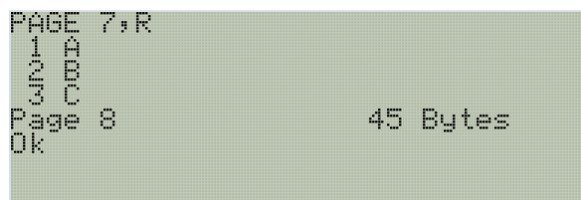
Data files

```
10 OPEN "0",#1,"DATA"
20 FOR I=1 TO 3:PRINT #1,I;" ";CHR$(64+I):NEXT
30 CLOSE #1
```

```

10 OPEN "I",#1,"DATA"
20 IF EOF(1) THEN 50
30 INPUT #1,A,B$:PRINT A;B$
40 GOTO 20
50 CLOSE #1

```



Reading a data file

Statement	Use
OPEN "O",#1,"name"	create a tape file; the header is written at once
OPEN "I",#1,"name"	open a tape file for reading (Found name)
PRINT #1,...	write (as PRINT; , writes one space)
INPUT #1,v[,v...]	read one line (Error 61 past the end)
INPUT\$ #1,n,v\$ (ext), INKEY\$(2)	read n characters, one byte
EOF(1)	-1 at the end of the file
CLOSE [#1]	close (an output file writes its last block)

One tape file is open at a time. Data is written in blocks of 256 bytes, each one preceded by a 2-second leader: keep the recorder running until CLOSE. File number 1 is the tape; 2 is the serial port (section 16).

15. Printer JR-P20

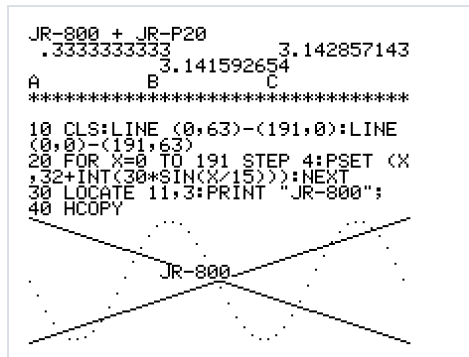
The JR-P20 is the graphic printer of the JR-800, plugged into the serial jack: 192 dots per line, 32 characters of 6×8 dots. In [PockEmul](#), add a JR-P20 and link it to the **Serial** connector of the JR-800.

Command	Use
LPRINT ...	print, as PRINT (, = next 10-column zone, TAB(n))
LPRINT ""	empty line
LLIST [range]	list the program
LFIND "text"	FIND on the printer
HCOPY , CTRL+2	copy of the screen (192 × 64 dots)
LPOS(0)	printer column

The JR-800 draws each character itself with its screen font, so the printout shows the same characters as the LCD (graphics and kana included). A line is printed when it is complete or at the end of the LPRINT. Without a printer, LPRINT, LLIST and HCOPY end with **Error 29** after a few seconds.



Error 29 without a printer



JR-P20 paper: LPRINT, LLIST, HCOPY

LPT 1 (ext) sends LPRINT, LLIST and HCOPY to a parallel printer of the external bus (not emulated) and **LPT 0** back to the JR-P20.

16. Serial port

Note: the serial port is still being tested in [PockEmul](#): it will only be available in a future version of [PockEmul](#). The JR-P20 printer ([section 15](#)) is available.

The serial jack (the one of the JR-P20) is an asynchronous port, 8 data bits, no parity, 1 stop bit, without control lines.

Command	Use
SINIT n	rate: 0 = 76800, 1 = 9600 (after a cold start) , 2 = 1200, 3 = 300 baud
SSAVE [range]	send the program as text (as LIST), lines ending with CR, then 1Ah
SLOAD	receive a program as typed lines (CR ends a line, 1Ah the transfer); the lines are added to the current page (NEW first for a clean load)
PRINT #2,...	send
INPUT #2,v... , INPUT\$ #2,n,v\$ (ext)	receive

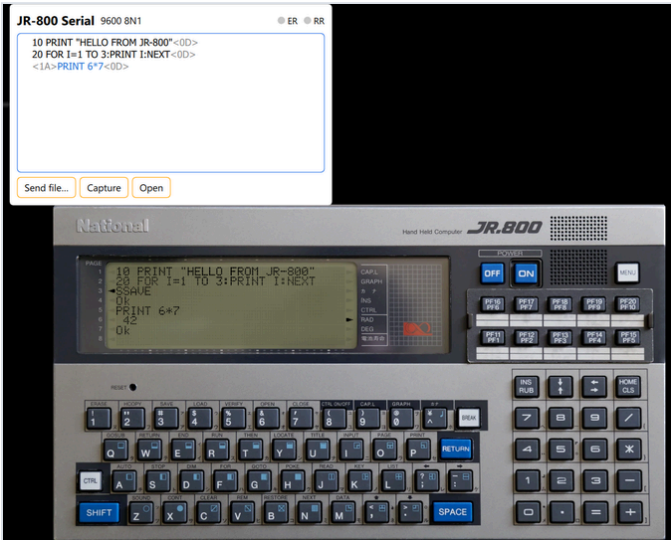
At the command prompt, a character received on the serial port is taken as a key: another computer can type commands on the JR-800.

Handshake. Each byte is acknowledged by one byte: the JR-800 sends each byte it receives back, and after each byte it sends it waits for one byte. Two JR-800s linked by their serial jacks work this way; a program on a PC talking to a real JR-800 must do the same.

PockEmul Serial Terminal

Open the terminal from the context menu of the JR-800 ("Serial port: open terminal") or the Serial tab of the advanced tools: it plugs into the serial jack (in place of the JR-P20). It does the handshake itself and hides the acknowledgements.

- `SSAVE` on the JR-800: the listing appears in the terminal; use Capture / Save capture to keep it as a text file.
- `SLOAD` on the JR-800, then "Send file..." (or drop a text file on the terminal) with a profile that ends lines with CR and appends 1Ah ("Sharp BASIC").
- A command typed in the terminal and ended by CR runs on the JR-800.



SSAVE and a remote command in the Serial Terminal

LPRINT and HCOPY need a JR-P20: to a terminal they give Error 29. The link between two JR-800s and the JH-600 voice unit are not emulated.

17. Machine language

The CPU is a Hitachi HD6301 (6801 instruction set plus AIM, OIM, EIM, TIM, XGDX, SLP).

Memory

`NEW ON address` reserves the memory from `&H2E00` to `address - 1` for machine code (all the program pages are erased). `FRE(0)` gives the free memory. See [Appendix C](#) for the map.

Statement / function	Use
<code>POKE address,byte[,byte...]</code>	write bytes
<code>PEEK(address) , MEM\$(address,n)</code>	read
<code>CALL address</code>	run the routine: it is called with <code>X = address</code> , and returns to BASIC with <code>RTS</code>
<code>MSAVE , MLOAD</code>	save, load on tape (section 14)

NEW ON &H3000

POKE &H2E00,&H39:CALL &H2E00

Monitor MON (ext)

MON starts a monitor with the prompt `*`; numbers are hexadecimal.

Command	Use
D[address]	dump 6 lines of 8 bytes (:AAAA hh hh ...); without address, the next ones
M[address]	modify: shows :AAAA hh- , type 2 digits or RETURN for the next byte; BREAK ends
:AAAA hh hh ...	a dump line changed on the screen and entered with RETURN writes its bytes
R	show and change the registers C, A, B, X, S, P
G[address][,break]	run from address (default P), with an optional breakpoint that returns to the monitor
B	back to BASIC (also BREAK)

```
*D8ABE
:8ABE 42 72 65 61 6B 00 45 72
:8AC6 72 6F 72 20 00 20 69 6E
:8ACE 20 00 50 61 67 65 20 00
:8AD6 20 42 79 74 65 73 00 27
:8ADE 17 81 86 27 2C BD 95 D3
:8AE6 BD 83 C6 4D 27 05 4A 81
*
```

MON

Useful addresses

Address	Content
&H82	current page - 1
&H89	key click off when not 0
&H91, &H92, &H93	kana, GRAPH, CAP.L modes (not 0 = on)
&H98, &H99	cursor column, line
&HA2	FLD mask
&HA3	insert mode
&H2000-&H20BF	user-defined characters E0h-FFh
&H20C0-&H20FF	keyboard buffer
&H2100-&H21DB	PF1-PF20 strings, 11 bytes each

18. The extended BASIC ROM and the external units

The JR-800 ROM image of [PockEmul](#) includes the 12 KB extended BASIC ROM (C000h-EFFFh, "JR-HuBASIC 2.0", V2.04 1983.12.13), installed at power on. It adds `'`, DEF FN / FN, PRINT USING, LINE INPUT, INPUT\$, MUSIC,

TEMPO, RANDOMIZE, CHAIN, MON, the conversions MKI\$ / MKS\$ / MKD\$ / CVI / CVS / CVD, string comparisons, and drivers for four units of the external bus. It uses 2800h-2DFFh as work area, so BASIC programs start at 2E00h.

The units are not emulated: the external bus of **PockEmul** is empty and their commands give **Error 29**, except where stated.

Note: the development of these external units is in progress in **PockEmul**, but it is an extremely complex task: no information is available on their technical specifications.

Colour graphics unit

320 × 200 dots, 8 colours, 40 × 25 characters.

Statement	Use
SCREEN m[,a[,b]]	m ≠ 0: use the unit (SCREEN 0 : back to the LCD)
COLOR f[,b]	drawing colour f (0-7), background b
CLS c	clear the screen of the unit to colour c
CIRCLE (x,y),r[,c[,p1[,p2]]]	circle or ellipse
PAINT (x,y),c[,border...]	fill
DEF CHR\$(n)=a\$	character pattern, n = 224-255, 8 bytes
PSET (x,y[,c[,OR XOR]]), LINE ...,PSET OR XOR[,c][,B BF], LOCATE, POINT	on the unit when SCREEN ≠ 0

COLOR f alone and SCREEN 0 work without the unit.

Floppy disk unit

5.25" double-sided disks, 2 drives, file names "1:NAME.EXT" / "2:NAME.EXT" (8 + 3 characters). No floppy unit is known for the JR-800.

Statement / function	Use
FILES [d], LFILES [d]	directory (types Bas, Asc, Bin, Rnd) and free clusters
LOAD, SAVE, MERGE, RUN, MSAVE, MLOAD with "d:name"	as on tape
KILL "d:name", RENAME "d:old" AS "d:new"	delete, rename
DSKINI d	logical format; asks Are you sure ? (y or n)
OPEN "I" "O" "R",#3 #4,"d:name"	sequential or random files 3 and 4
PUT [#]f[,r], GET [#]f[,r]	write, read record r (256 bytes)
LSET, RSET [#]f,pos,len,a\$, FIELD\$(#f,pos,len)	write, read a field of the record
LOC(f), LOF(f), DSKF(d)	current record, file size, free clusters
DSKI\$, DSKO\$ d,t,s,a\$,b\$	read, write a sector

Parallel printer and RS-232C

Statement	Use
LPT 1 / LPT 0	LPRINT, LLIST, HCOPY to the parallel printer / to the JR-P20
LPOUT a\$	send bytes to the parallel printer
INITP	reset the parallel printer
OPEN "I" "O",#5,"8N1"	RS-232C port of the unit (7 or 8 bits, N / O / E, 1 or 2 stop bits); in PockEmul PRINT #5 waits until BREAK

19. PockEmul specifics and limitations

- **Typing:** programs can be pasted in the keyboard simulator (Keyboard tab of [PockEmul](#)). The PC keys are described in [section 3](#).
- **Sessions:** saving a [PockEmul](#) session keeps the memory (programs, PF keys, variables), the clock and the linked JR-P20 and tape recorder.
- **ON** restarts the CPU with the memory kept; **RESET** is a cold start and erases everything.
- **Not emulated:** the LCD contrast slider, the 16 KB configuration ([PockEmul](#) always has 24 KB), memory packs, the external bus units (JR-U20, colour graphics, floppy disk, parallel printer, RS-232C), the link between two JR-800s, the JH-600 voice unit.

Appendices

A. Tokens

Program lines are stored with one byte per keyword. Words of the extended ROM are in *italics*; empty cells are unused codes.

Statements, operators and other words (one byte):

	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	GOTO	DATA	CHAIN	LFIND	FLD	FILES	INITP	<>
x1	GOSUB	READ	TRON	FIND	PSET	RENAME		=<
x2	RUN	DIM	TROFF	WIDTH	PRESET	KILL	LPOUT	<=
x3	RETURN	REM, '	END	TITLE	SINIT	LSET	LPT	=>
x4	RESTORE	STOP	PAGE	ERROR		RSET	MON	>=
x5	RESUME	CONT	SOUND	RENUM	HCOPY	DSKO\$	TO	=
x6	LIST	CLS	DEFINT	ELSE	SSAVE	DSKI\$	STEP	>
x7	LLIST	CLEAR	DEFSNG	CALL	SLOAD	PUT	THEN	<
x8	DELETE	ON	DEFDBL	WLOCATE	MSAVE	GET	USING	+

	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x9	EDIT	LET	DEFSTR	LOCATE	MLOAD	<i>DSKINI</i>	TAB	-
xA	AUTO	NEW	<i>DEF</i>	KEY	OPEN			
xB	FOR	POKE	LOAD	WAKE	CLOSE		<i>XOR</i>	
xC	NEXT	<i>COLOR</i>	SAVE	<i>TEMPO</i>	RAD		<i>OR</i>	/
xD	PRINT	<i>SCREEN</i>	MERGE	<i>RANDOMIZE</i>	DEG			*
xE	INPUT	<i>CIRCLE</i>	VERIFY	<i>MUSIC</i>	GRAD			^
xF	IF	<i>PAINT</i>	LPRINT	LINE	<i>LFILES</i>		> <	

Functions (two bytes: FF, then the code):

	8x	9x	Ax	Bx
x0	INT	FIX	<i>MKI\$</i>	SCRN\$
x1	ABS	PI	<i>MKS\$</i>	
x2	SIN	CDBL	<i>MKD\$</i>	STRING\$
x3	COS	CSNG	SPC	TIME\$
x4	TAN	CINT	ASC	DATE\$
x5	LOG	FRE	LEN	DAY\$
x6	EXP	<i>DSKF</i>	VAL	<i>FN</i>
x7	SQR	EOF	CVS	<i>FIELD\$</i>
x8	RND	<i>LOC</i>	<i>CVD</i>	ERR
x9	PEEK	<i>LOF</i>	<i>CVI</i>	ERL
xA	ATN	POS	LEFT\$	CSRLIN
xB	ASN	LPOS	RIGHT\$	POINT
xC	ACS	CHR\$	MID\$	
xD	LN	STR\$	INKEY\$	
xE	SGN	HEX\$	INSTR	
xF	CALC	OCT\$	MEM\$	

B. Character codes

Codes	Content
00h-1Fh	control codes (section 4)
20h-7Fh	ASCII, 5Ch = ¥
80h-9Fh	graphic characters

Codes	Content
A0h-DFh	katakana
E0h-FFh	user-defined characters

Key codes returned by `INKEY$` for the special keys:

Key	Code	Key	Code
BREAK	03h	RETURN	0Dh
MENU	04h	SHIFT+INS	12h
RUB	08h	→ ← ↑ ↓	1Ch 1Dh 1Eh 1Fh
SHIFT+HOME	0Bh	PF1-PF10	E0h-E9h
CLS	0Ch	PF11-PF20	EAh-F3h
CTRL+letter (CTRL mode)	01h-1Ah	CTRL+1 ... CTRL+^	F4h-FEh

C. Memory map

From the BASIC manual (appendix A.4) and the ROM.

Address	Content
0000-001F	CPU registers (ports, timer, serial)
0080-00FF	CPU internal RAM (BASIC work area)
0100-0FFF	I/O: calendar clock (0600), LCD drivers (0A00), keyboard (0C00-0FFF)
2000-20BF	user-defined characters (192 bytes)
20C0-20FF	keyboard buffer (64 bytes)
2100-21FF	PF key strings
2200-27FF	work area
2800-2DFF	work area of the extended BASIC (machine code area without it)
2E00-	machine code area (<code>NEW ON</code>), then the BASIC programs, the variables, the free area (<code>FRE</code>), the stack, the floating accumulator (256 bytes) and the virtual screen (<code>WIDTH</code>), up to the end of the RAM
5FFF / 7FFF	end of the RAM, 16 KB / 24 KB

Address	Content
8000-BFFF	BASIC ROM (16 KB)
C000-EFFF	extended BASIC ROM (12 KB)
F000-FFFF	CPU internal ROM (4 KB)

The BASIC manual also shows a second bank (#1) of RAM / ROM at 4000h-EFFFh, reserved for units such as floppy disk drives.

D. Sources

- 1983 catalogue specification, transcribed at <https://cms.matrix.jp/jr800.html>, translated at <https://pockemul.com/2020/08/28/national-jr-800-documentation/>.
- JR-800 BASIC manual, appendix A.4 (memory map).
- Disassembly of the 32 KB ROM and tests in [PockEmul](#).

© 2026 PockEmul. All rights reserved. This manual is published by [PockEmul](#), the emulator of pocket computers: <https://pockemul.com>.