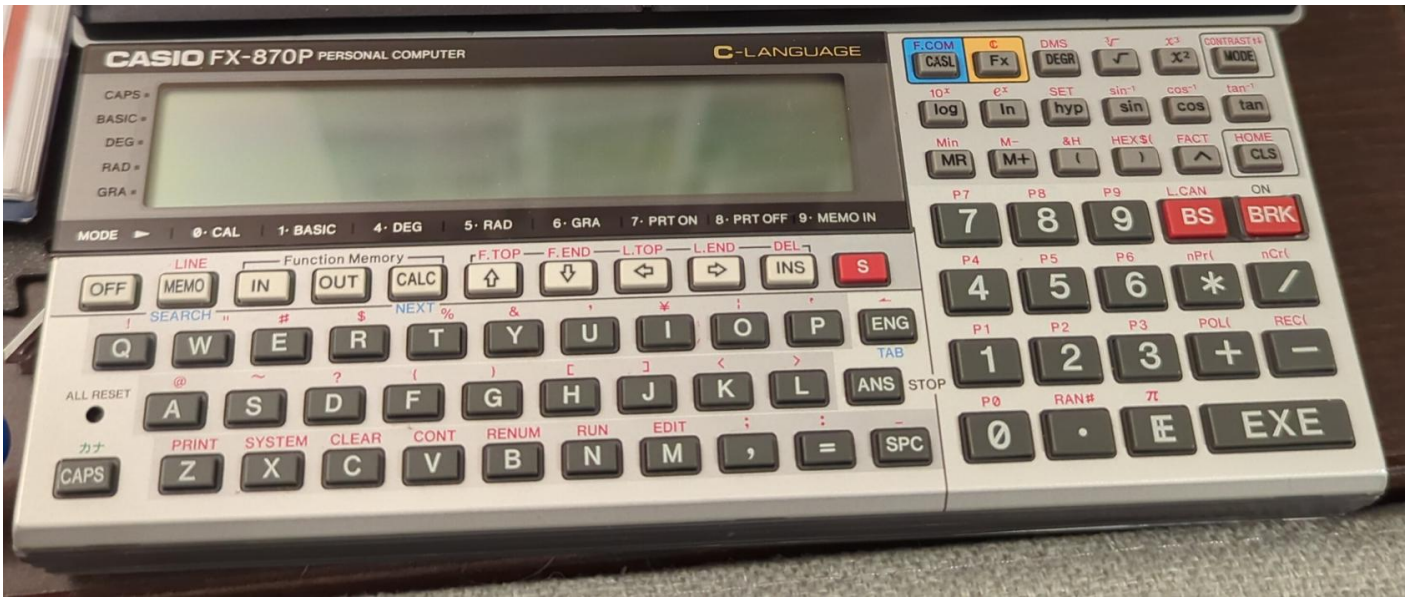
Calculators

- Casio FX-870P
- Casio FX-700P

Casio FX-870P



Manual (Japanese with English commentary): [CasioVX-4-Manual-Peter-Rost.pdf]

Programming in C

Navigation

`S` (red), `F.COM` (blue)	Enter C programming mode
`S`	Source - edit code
`R`	Run - compile and run code
Arrow keys	Select program
(after program run) `S`, `M` or type `edit`	Edit code
`S` (red), `MEMO` (line)	Go to line

Types

`char`	8 bit

`short`	16 bit
`int`	16 bit
`unsigned`	16 bit
`long`	32 bit
`float`	32 bit
`double`	64 bit

> `structs` or `unions` are not supported.

Flow control

```
if (C)
  expr;

if (C) {
  expr;
  expr;
}

if (C) {
  expr;
} else {
  expr;
}
```

```
while (C) {
  expr;
}

do {
  expr;
} while(C);

for (A; C; I) {
  expr;
```

```
}
```

```
goto LABEL;


LABEL;
```

> `switch` / `case` are not supported.

Functions

Entry function:

```
main() {
  expr;
}
```

Custom function:

```
int foo(buf,x,y)
  char buf[];
  int x;
  int y;
{
  int i;
  expr;
}
```

> Variables need to be declared first then assigned.

> Functions are declared in pre-ANSI style. Inner variables need to be defined at body top.

`int getchar()`	
`int getc(FILE)` / `int fgetc(FILE)`	For `stdin` file: `extern FILE *stdin;`
`int putchar(char)`	

`int putc(char, FILE)` / `int fputc(char, FILE)`	For `stdout` file: `extern FILE *stdout;` There is also printer(?): `extern File *stdprn`
`char *gets(char *)` (unsafe)	`char msg[8];` `gets(msg);` `pringf("Hello: %s",msg);`
`char fgets(char *, int buflen, FILE)`	`extern FILE *stdin;` `main() {` `  char msg[8];` `  fgets(msg,8,stdin);` `  pringf("Hello: %s",msg);` `}`
`int puts(char *)`	
`int fputs(char *,FILE)`	
`printf(format, args, ...)`	
`fpritf(FILE, format, args, ...)`	
`sprintf(char *buf, format, args, ...)`	
`int scanf(format, args, ...)`	
`int fscanf(FILE, format, args, ...)`	
`int sscanf(char *buf, format, args, ...)`	
`int fflush(FILE)`	
`int inport(int n)`	
`outport(int n, int i)`	
`clearerr(FILE in)`	
`breakpt()`	
`exit()` / `abort()`	

`char *malloc(unsigned size)`	
`char *calloc(unsigned n,unsigned size)`	
`int free(char *)`	
`int strlen(char *)`	
`char *strcpy(char *dest, char *src)`	
`char *?strcat(char *dest, char *src)`	
`int strcmp(char *, char *)`	
`char strchr(char *,char)`	
`int abs(int)`	
`double sin(double)` [`cos`, `tan`, `asin`, `acos`, `atan`, `sinh`, `cosh`, `tanh`, `asinh`, `acosh`, `atanh`]	
`double pow(double, double)`	
`double sqrt(double)`	
`double exp(double)`	
`double log(double)` / `double log10(double)`	
`angle(unsigned)`	mode for `sin` etc.: `0` - deg, `1` - rad, `2` - gra
`beep(unsigned)`	
`clrscr()`	
`gotoxy(unsigned x, unsigned y)`	`x` - between 0 and 31 (inc), `y` - between 0 and 3 (inc)

Casio FX-700P

Resources

- Manual: CASIO FX-700P.pdf
- Games: Cassio - Collection og games.pdf

Usage

Writing programs

1. Enter write mode: `MODE`, `1`
2. Select program `S`, `0` to `9`
3. Enter program lines: `<line no> <command> <args>`, `EXEC`
4. `LIST` to list the program (follow by line number to start from that line)
    1. Each listed line can be edited
    2. `AC` to cancel any changes made to a line
    3. `EXEC` save changes
5. Changing `<line no>` of a line makes a copy of it under that name (if already exist it overwrites existing one)
6. Putting just `<line no>` with no `<command>` removes that line

Running programs

1. Enter run mode with `MODE`, `0`
2. `S`, `<program number>` to run it
3. Alternatively type `RUN` to run currently selected program (last edited or run)

Typing

- Lower case letters: `MODE`, `.` (dot; `EXT` mode)

BASIC

Input	`INPUT <string var>`	Enter data from keyboard.
	`<char var>=KEY` ```FOR I=0 TO 20 K$=KEY NEXT I: PRINT K$```	Read a character and assign it to a variable. Program is not stopped. Empty string is read if no key is pressed.
Output	`PRINT [string | var | command] [: | ,] [string | var] ([: | ,] ...)` ```PRINT CSR 3; PRINT "HI" F=2 PRINT "FOO=";F```	If no arguments are given clears screen. If string or variable is given it is printed out. `;` separates arguments to print without clearing screen. `,` waits for any key and clears the screen. If terminated by `;` screen is not cleared, otherwise waits for any key and clears the screen. `PRINT CSR [var | num]` sets position of cursor (0 to 11).
Branching	`GOTO [line | var]`	Execution jumps to specified line number.
	`IF <comparison> [THEN <line> | ; <command>...]`	Jump to line or execute commands following `;` if comparison is true. Otherwise continue from next line.

	`GOSUB [line | var]`	Jump to line or line stored in a variable.
	`RETURN`	Returns from last `GOSUB` call to command next after it.
Loops	`FOR <var>=<val> TO <val> [STEP <val>]`	Starts loop counting from initial value to given value with step. Calling `NEXT <var>` will repeat loop incrementing `<var>` by 1 or `STEP <val>` value. Once `<val> >= TO <val>`, `NEXT <var>` will not jump and following command will be executed ending the loop.
	`NEXT <var>`	Repeat `FOR` loop for given `<var>` or continue from next command if loop is done.
Execution	`STOP`	Stop the execution of a program temporarily and wait for `EXEC` key.
	`END`	End of program.
	`RUN [line]`	Start program from given line number or form beginning.
Data	`VAC`	Clear all variable data for a program.
	`CLEAR`	Remove program.
	`CLEAR A`	Remove all programs (!).
Listing	`LIST [line>]`	Display program listing from beginning or given line.
	`LIST A`	List all programs and data.
Angular unit	`MODE [4 | 5 | 6]`	Sets trigonometric angular units: • `4` - degree • `5` - radian • `6` - gradient

Format	`SET [E <val>, F <val>, N]`	Set number of effective positions or decimal positions for displayed numerical value (0 to 9).
Character functions	`LEN <char var>`	Length of a string.
	`<char var>=MID (<n>[, <m>])`	Extract `n` characters from the `m`th character in `$`.
	`VAL <char var>`	Convert number in a string to number.
Numeric	`INT <val>`	Integer part of a number.
	`FRAC <val>`	Fractional part of a number.
	`SIN <val>, COS <val>, TAN <val>`	Trigonometric functions.
	`ASN <val>, ACS <val>, ATAN <val>`	Inverse trigonometric functions.
	`SQR <val>`	Square root of `<val>`.
	`EXP 1`	Call out the numerical value of exponential table (`e`).
	`LN <val>`	Natural logarithm.
	`LOG <val>`	Base 10 logarithm.
	`ABS <val>`	Absolute value.
	`SGN <val>`	Sign of a number: • `1` - `<val> > 0` • `0` - `<val> = 0` • `-1` - `<val> < 0`

	`RND (<x>,<y>)` `RND (123.456,-3) -> 123.46` `RND (123.456,-2) -> 123.5` `RND (123.456,-1) -> 123` `RND (123.456,0) -> 120` `RND (123.456,1) -> 100` `RND (123.456,2) -> 0`	Round number `<x>` to `<y>` significant digit place.
	`RAN#`	Random number between 0 and 1.

BASIC games

Spot and Stop

```
5 L=0
10 PRINT "Spot and Stop"
20 PRINT "HIGHEST=";H
30 PRINT "BY ";$
40 FOR G=1 TO 20
50 A=INT (RAN#*10)
60 GOSUB 300
70 INPUT C
80 IF A=C;L=L+20:PRINT "GOOD":GOTO 100
90 L=L/2:PRINT "MISSED"
100 NEXT G
110 L=INT L:PRINT :PRINT "SCORE=";L
120 IF L>H;H=L:PRINT "NEW HIGH":INPUT "NAME",$
130 END
300 PRINT "::::::::::";
310 PRINT CSR A;";";:FOR W=1 TO 40:NEXT W
315 PRINT
320 PRINT "0123456789";
330 RETURN
```

Gopher Trap

```
10 PRINT "HIGHEST:";H:G=0:$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 FOR C=0 TO 4
30 PRINT CSR 0;"            ";
40 A=INT (RAN#*26+1):B=INT (RAN#*12)
50 D$=MID(A,1)
60 FOR E=0 TO 25:PRINT CSR B;D$;:F$=KEY:IF F$="";NEXT E
70 IF D$≠F$;PRINT :PRINT "MISSED:0";:GOSUB 200:NEXT C:GOTO 100
80 PRINT :PRINT CSR 0;"TIME :";E;:GOSUB 200
90 S=(27-E)*10:PRINT CSR 0;"SCORE:";S;:G=G+S:GOSUB 200:NEXT C
100 IF H<G;H=G
110 PRINT :PRINT CSR 0;"FINAL:";G;" "
120 END
200 FOR W=1 TO 200:NEXT W:RETURN
```