Product Specification

JULY 1979

Zilog

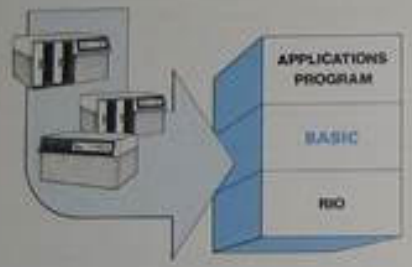
Z80® BASIC

The Zilog BASIC interpreter gives the MCZ or ZDS user an easy-to-use and learn, problem-solving language. The user works in a highly interactive environment where time-consuming separate editing and assembly are unnecessary. Programs may be interactively edited, entered, displayed, run and debugged all within the BASIC interpreter subsystem.

BASIC allows the user to manipulate several data types: real, integer, and string. BASIC lets the user take full advantage of the Z80 RIO Operating System's file capabilities, including random access. BASIC programs can be interfaced to PLZ or assembly language procedures.

Features

- PROGRAM ENTRY AND EDITING
  - Full syntax checking at each statement as entered.
  - Automatic output (listing) program formatting and indentation. All blanks are ignored on statement entry.
  - Single lines or groups of lines can be deleted or replaced.
  - Groups of lines (or the entire program) can be renumbered independently.
- LANGUAGE EXTENSIONS
  - IF-THEN-ELSE with DO-DOUNTIL nesting for structure improvement.
  - Multi-line user functions. Multiple arguments of any type are allowed.
  - Full string manipulation capabilities.
  - String arrays.
  - Escape, keyboard interrupt, error, and external event trapping or polling.
  - Chaining and common variables. Formatted output.
- I/O
  - Support for RIO file environment.
  - Semi-automatic buffer management.
  - Random files.
  - Binary and ASCII files.



- MATH PACKAGES
  - Choice of 7-digit binary floating math package or 13-digit BCD math package.
  - Integer data type for speed and reduced space requirements.
  - Trigonometric, logarithmic, and exponential functions.
  - Numeric-string and string-numeric conversions.
- DEBUGGING
  - Program can be interrupted, variables examined and modified, and the program resumed.
  - Single-step program execution.
  - Directly executable statements.
- ONLY 8K BYTES MEMORY REQUIRED

Zilog

BASIC INTERPRETER

Reference Manual

June 1979

Zilog

Zilog

Zilog

Zilog

Command	Description	Reference
RENUMBER	Renumbers any group of statements in the current program, optionally from a new first line number with a specified increment. By default, renumbering starts at 10 with increments of 10.	4.2.4
RUN	Executes the current program.	4.1.1, 4.1.3
RSAV	Stores a copy of the current program in a file that already exists.	4.3.3
SAV	Stores a copy of the current program on the user's disk in compiled form.	4.3.2
STEP	Resumes execution, completes an outer level statement (not part of a function) and then stops. Can be used to step through a program one line at a time.	4.1.3
SIZE	Gives status of: space available (bytes); program size (bytes); variable storage size (bytes); number of 512 byte reserved blocks.	4.2.5
XEQ	Gets and runs the specified program.	4.1.2, 4.3.5

APPENDIX C

SUMMARY COMMANDS OF ZILOG BASIC

Each command is listed by name in alphabetical order followed by a brief description and reference to the section or sections containing a complete description of the command. All commands may be abbreviated by their first three letters.

Command	Description	Reference
APPEND	Appends a specified program (which must be in ASCII form) to the current program.	4.3.6
ASAVE	Stores a copy of the current program on the user's disk in ASCII form.	4.3.1
CLEAR	Deallocates all variable space, closes files and resets function and subroutine calls. Frees space to save a program if there is not enough available.	4.2.6
CONTINUE	Resumes program execution after an interruption by ESCape or a STOP statement.	4.1.3
DELETE	Deletes one or a range of more than one statement from current program.	4.2.3
GET	Gets the specified Zilog BASIC program from the user's library, replacing the current program.	4.3.4
LIST	Lists all or part of the current program at the terminal.	4.2.1
NEW	Deletes entire current program.	4.2.2
QUIT	Terminates the current Zilog BASIC session.	4.1.4

1.9 Running A program

After the program is entered and, if desired, checked with LIST, it can be executed with the RUN command. RUN will be illustrated with two sample programs.

The first program has one line

```
>100 PRINT "5 * 10 =";5*10
```

When run, the result of the expression 5*10 is printed:

```
>RUN
5 * 10 = 50

READY
>
```

Because the program contains a PRINT statement, the result is printed when the program is run.

The second sample program adds two numbers. The numbers must be input by the user:

```
>10 INPUT A,B
>20 LET C=A+B
>30 PRINT
>40 PRINT A;" +";B;" =";C
```

The two letters following the word INPUT, and separated by a comma, name variables that will contain a value input by the user from the terminal. When the program is run, the interpreter signals that input is expected by printing a question mark. The user enters the values following the question mark. They are entered with a comma between each successive value.

The statement LET C=A+B assigns the value of the expression to the right of the equal sign to the variable C on the left of the equal sign. The expression adds the values of variables A and B together. The result is the value of C.

When the program is run, the user enters input values and the computer prints the result:

```
>RUN
?1078,5.3
1078 + 5.3 = 1083.3

READY
>
```

BASIC INTERPRETER

Reference Manual

June 1979

03-0077-02

June, 1979