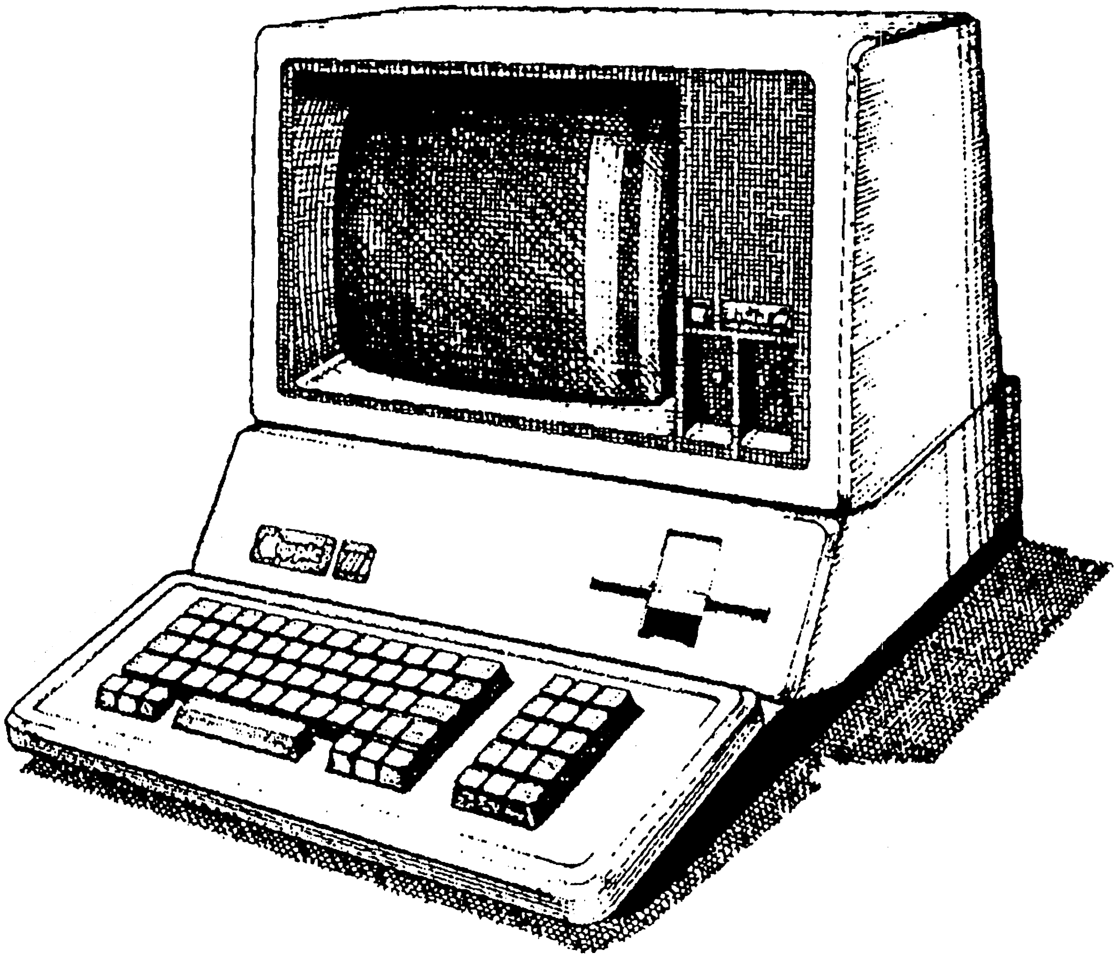
Apple /// Computer Information



DOCUMENT NAME

SPECIFICATION: APPLE /// BUSINESS BASIC

#

6

Apple Business BASIC

Apple *III's* Advanced BASIC for Business Applications

Languages and Utilities

Apple Business BASIC for the Apple III Computer System is one of the most powerful versions of BASIC ever developed for a microcomputer. Designed for those who want the flexibility, power, and ease of use of the popular BASIC language, Apple Business BASIC incorporates a number of innovative features which can be used to satisfy some of the most demanding business and scientific programming needs.

With Apple Business BASIC, programmers can address over 70K bytes of extended memory—the largest workspace available on any personal computer. In addition, Apple Business BASIC's special 64-bit, 18-digit data type handles the toughest accounting chores with "penny accuracy."

For producing reports, Apple Business BASIC offers some of the most comprehensive and flexibly formatted output capabilities found on any version of BASIC—even those used on large system mainframes. Apple Business BASIC also features: advanced file handling, with flexible file formats to match application needs; 16-bit binary integer, and 32-bit binary floating point data types; 64-character variable names (all characters significant) for documentation purposes; easy access to a wide variety of plug-in peripherals or auxiliary devices; and a trace mode for debugging.

If you prefer programming in BASIC, but require more power, flexibility, and precision than most BASICs can give you, turn to Apple Business BASIC for the Apple III—an advanced language for an advanced computer system. Apple Business BASIC is supplied with all Apple III system configurations.

Benefits

Apple Business BASIC…

- *makes it easier to generate reports, because of its flexible, extensive, formatted output capabilities, and versatile PRINT USING and IMAGE statements…*
- *increases system efficiency, because its large, user-available workspace (over 70K bytes) allows lengthy programs to be kept in memory…*
- *adds programming flexibility, because it lets you keep multiple (up to 10) files open simultaneously, with virtually no size constraints…*
- *allows you to display and calculate financial accounting data with "penny accuracy" through the use of a 64-bit, 18-digit data type and special functions…*
- *speeds up disk file storage, because its built-in facilities let programs read and write files more efficiently…*
- *simplifies system input/output (I/O) control, because just a few key words let you control a wide variety of peripherals, as well as machine language routines and graphics facilities…*
- *aids development and debugging of complex programs, because its optional ELSE statement and automatically indented listings encourage a structured approach to programming.*

Apple Business BASIC— A Closer Look

Apple Business BASIC is a general purpose, problem-solving language designed especially for business and scientific applications calling for: 1) an easy-to-use debugging environment; 2) fast, high-precision operations—such as accounting—on numbers with a range less than $\pm 10^{18}$; 3) a high degree of output formatting for printing reports; and 4) greater flexibility for file I/O operations. Apple Business BASIC's powerful features provide the programmer with a number of advanced capabilities and conveniences.

Extended User Program Memory Space

In an Apple *///* Computer System with 128K bytes of RAM, Apple Business BASIC presents you with a huge 70K-byte workspace—

more than is available with any other personal computer BASIC. This means you can write large programs more easily and run them more efficiently. Access time is reduced, because large files can be kept in memory—instead of on disk—and because many files can be opened for access simultaneously. In addition, large, high-resolution graphics areas can be used without fear of unduly restricting program size.

Flexible Output Formatting
Apple Business BASIC's versatile PRINT USING and IMAGE statements allow you to use a variety of format strings to prepare reports. Specifications are extremely flexible, and include string, literal, digit, scientific notation, and engineering notation categories.

Advanced File Handling
In Business BASIC, file handling is done through numbered file references defined within the program. You can make these file definitions perform chores simply by using generic terms. To print, for example, call out the generic term ".printer" within the program. Apple Business BASIC works with the Apple ///'s Sophisticated Operating System (SOS) to handle the details automatically, including which printer you're addressing (if your system has more than one), its I/O address, etc. For greater user convenience, disk files are referenced by directory, sub-directory, and file name, without regard to the storage device on which the file resides.

Long Variable Names
Apple Business BASIC allows you to use variable names (up to 64 characters in length) for documentation purposes, with each character significant. Furthermore, because spaces are used as delimiters, embedded BASIC keywords are allowed in variable names, permitting even more flexibility.

"Structured" BASIC Features
With Apple Business BASIC, you can add the ELSE statement to the usual IF...THEN statement provided by BASIC. And your ELSE clauses can themselves contain other IF...THEN...ELSE constructs. Apple Business BASIC also provides a LISTing feature that automatically indents the contents of FOR...NEXT loops. These "structured" BASIC features encourage programmers to use a more logical, structured approach to developing their programs, and greatly aid development and debugging of complex programs.

Powerful Interface to Assembly Language
Because Apple Business BASIC runs in the SOS environment, you don't have to worry about memory management, buffer allocation, or file handling. This freedom will save you a significant amount of programming time.

Additionally, Apple Business BASIC relates to SOS through a powerful Invoke/Perform interface mechanism. For example, if you have created assembly language routines using the disk assembler, you can specify the assembled routine by name in an Apple Business BASIC program. Once the routines are mentioned in the "INVOKE" statement, Apple Business BASIC works with SOS to find a residence in memory for the routines, and to establish—as entry points in the resultant linked module—any function or procedure names mentioned in the routines. All you have to do is PERFORM the previously invoked routine—specifying any variables to be passed—and Apple Business BASIC and SOS automatically handle all the operational details.

Apple Business BASIC makes life a lot easier for programmers by expanding system capabilities, reducing program development time, and adding greater flexibility to formatted output and file handling facilities.

Technical Specifications

Format
16-sector diskette

Variables
- 64 characters (max.), all significant
- Reserved variables:
  ERR, KBD, EOF, VPOS, ERRLIN, HPOS, FRE, PREFIX$

Apple Business BASIC

- Data Types
  16-bit binary integers (−32768 to +32767)
  64-bit binary integers (± 9223372036854775807 or $2^{63} - 1$)
  32-bit binary floating point (± 10^{38} with 6-digit precision)
  Character strings (0 to 255 characters, dynamic)
  String and numeric arrays (indexed starting with 0, no dimensional limits)

Operators

- General: +, −, *, /,   , DIV, mod
  (Note: DIV and MOD are only for long integer operations)
- Binary logical operators:
  AND, OR, =, <, >, <>, ><, >=, <=, =<, =>
- Unary logical operator:
  NOT
- String operator:
  + (concatenation)

Statements

(Note: No statement or statement list may exceed 254 characters, including delimiters.)
LET (optional)
REM
GOTO
IF...GOTO
IF...THEN
IF...statementlist: ELSE statementlist
FOR ctrl variable = expression TO expression STEP expression
NEXT ctrl variable (,other ctrl variable)
GOSUB
RETURN
POP
ON expression GOTO
ON expression GOSUB
ON ERR
ON KBD
ON EOF #
OFF ERR
OFF KBD
OFF EOF #
RESUME

Utility Statements:

NEW	LOAD	STOP
CLEAR	SAVE	END
FRE	DELETE	CONT
PREFIX$	RUN	CHAIN prog name, line number

User-Defined Functions:
FN functionname (argument)
DEF FN functionname (argument) = expression

Debugging:
TRACE
NOTRACE

Cursor and Screen:

LIST	VPOS	HOME	NORMAL
DEL	HPOS	INVERSE	TEXT

String, Numeric and File Functions:

LEN	TEN	CONV	TAN	SQR
STR$	MID$	CONV%	ATN	EXP
VAL	LEFT$	CONV&	INT	LOG
CHR$	RIGHT$	CONV$	RND	TYP
ASC	SUB$	SIN	SGN	REC
HEX$	INSTR	COS	ABS	

Program Resident Data Statements:
DATA
READ
RESTORE

Apple Business BASIC

Machine Statements and Functions:
INVOKE
PERFORM
EXFN
EXFN%

File I/O:
CATALOG
DELETE
RENAME
LOCK
UNLOCK
CREATE
OPEN # filenumber (AS INPUT, AS OUTPUT, AS EXTENSION)
CLOSE # filenumber
CLOSE
INPUT # filenumber, recordnumber
OUTPUT # filenumber
PRINT # filenumber, recordnumber
PRINT # filenumber, recordnumber USING
READ # filenumber, recordnumber
WRITE # filenumber, recordnumber

Console I/O:
INPUT
GET
TAB
SPC
SCALE
PRINT
PRINT USING
IMAGE specification(s)

String specifications:
A reserves a character position left-justified
C reserves a character position center-justified
R reserves a character position right-justified

Literal specifications:
X prints a space
/ prints a carriage return/line feed
"literal" prints whatever is in quotes

Digit specifications:
# reserves one numeric digit, leading zeros suppressed
Z reserves one numeric digit, leading zeros printed
& reserves one numeric digit or comma (comma fill every three digits)
. reserves a position for the decimal point
+ reserves a position for the sign
− reserves a position for the sign (if negative)
$ reserves a position for the dollar sign
** asterisk fill
+ + floating sign
− − floating sign (if negative)
$$ floating dollar sign

Scientific notation specification:
E reserves a position for the exponent (power of 10)

Engineering notation specification:
same as scientific notation, except the exponent is always a multiple of three

Apple Computer Inc. 10260 Bandley Drive Cupertino, California 95014 (408) 996-1010 TLX 171-576
©1981 Apple Computer Inc. Printed in U.S.A. 11/81 250M Order No. A2F0036