

BASIC 64

**complete compiler
for the Commodore 64**

By T. Helbig

A Data Becker product

Published by:

Abacus  Software

57655

COPYRIGHT NOTICE

Abacus Software makes this package available for use on a single computer only. It is unlawful to copy any portion of this software package onto any medium for any purpose other than backup. It is unlawful to give away or resell copies of any part of this package. Any unauthorized distribution of this product deprives the authors of their deserved royalties. For use on multiple computers, please contact ABACUS Software to make such arrangements.

WARRANTY

Abacus Software makes no warranties, expressed or implied as to the fitness of this software product for any particular purpose. In no event will Abacus Software be liable for consequential damages. Abacus Software will replace any copy of the software which is unreadable if returned within 30 days of purchase. Thereafter, there will be a nominal charge for replacement.

Fourth Printing, 1986

Printed in U.S.A. Translated by Gene Traas, Heidi Sumner

Copyright (C) 1984 Data Becker, GmbH

Merowingerstr. 30

4000 Dusseldorf, W.Germany

Copyright (C) 1984 Abacus Software, Inc.

P.O. Box 7211

Grand Rapids, MI 49510

ISBN # 0-916439-17-8

TABLE OF CONTENTS

0. INTRODUCTION	1
1. GETTING STARTED	3
2. THE COMPILER/OPTIMIZER	7
3. ADVANCED DEVELOPMENT FEATURES	11
4. THE OVERLAY FEATURE	15
5. COMPILER DIRECTIVES	17
6. HINTS FOR PROGRAMMING	21

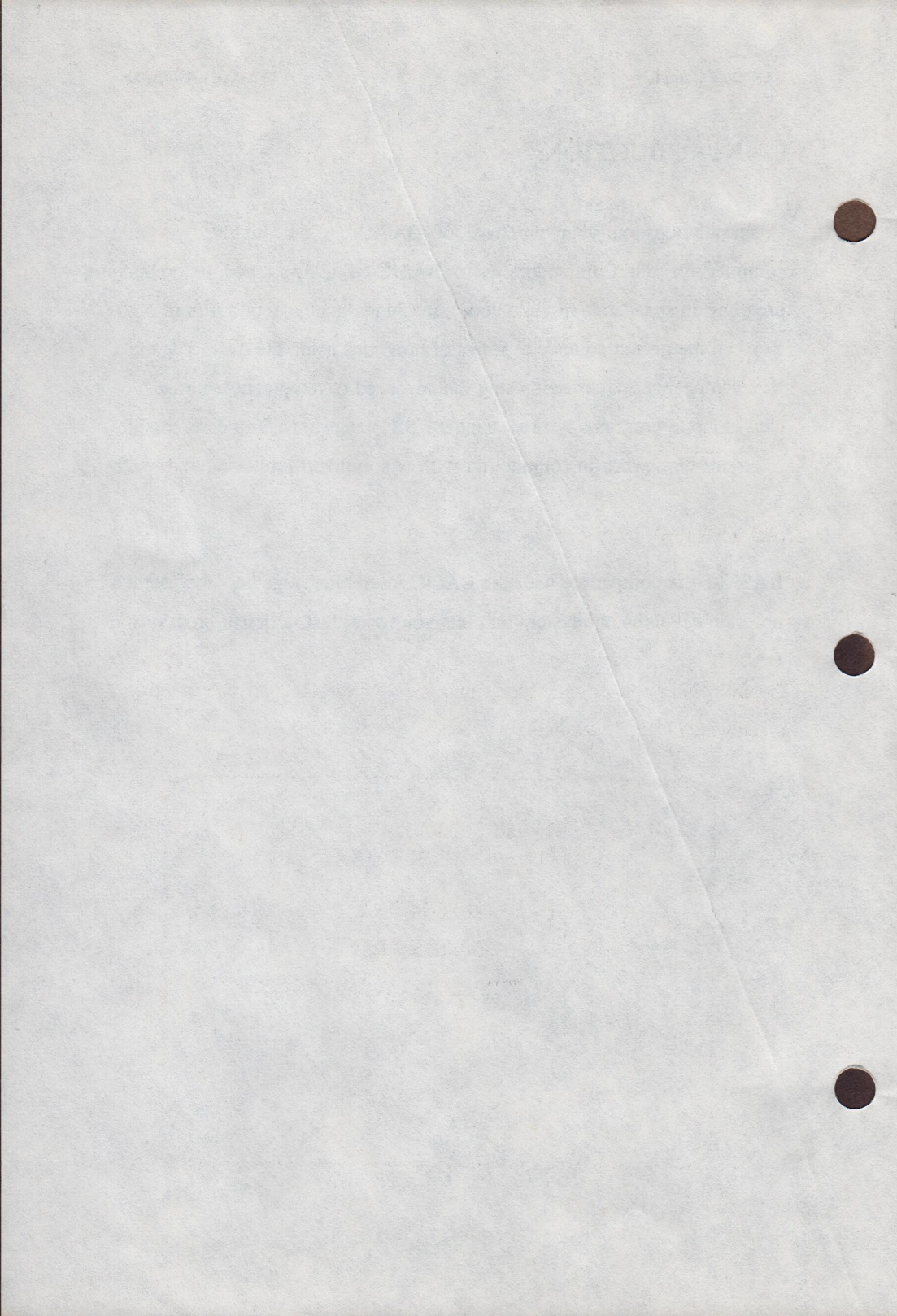
PROGRAMMER'S APPENDIX

COMPILER DETAILS	23
ARRAYS	25
INTEGER LOOPS	25
BASIC EXTENSIONS	26
SIMON'S BASIC	28
BASIC 4.0	30
COMPILING BASIC EXTENSIONS	30
OPERATION OF COMPILED EXTENSIONS	32
OTHER EXTENSIONS	33
ERRORS	34
CODE-START	35
MEMORY LAYOUT	35
MEMORY SWITCHING	37
FLOATING-POINT FUNCTIONS	38
RS-232 OPERATIONS	38
TECHNICAL NOTES	39

0. INTRODUCTION

Congratulations on your purchase of BASIC-64, our highly versatile compiler for the Commodore 64! BASIC-64 allows you to compile programs in speedcode (pseudo-code), machine language, or a mix of both. You can merge and compile a series of programs using the overlay feature. You can change parameters using the advanced development features. You can compile programs written using BASIC extensions. You can compile programs that work in conjunction with Assembler/Monitor-64 and much more.

BASIC-64 is compatible with the BASIC interpreter, together they form a program development system that lets you write fast, efficient programs in BASIC.



1. GETTING STARTED

SAVE your BASIC program onto a work diskette. Make sure that there is enough extra space on this diskette since larger compiled programs can require up to 300 blocks disk space. Remove the work diskette from the drive. Now, carefully place the BASIC-64 distribution diskette in the disk drive and type:

```
LOAD "BASIC 64",8 <RETURN>
```

After it has loaded, type:

```
RUN <RETURN>
```

The drive will run for a short time and the MAIN MENU is displayed on the screen which looks like this:

BASIC 64 COMPILER V1.03
(C)1984 DATA BECKER, T.HELBIG

- 1 = COMPILER/OPTIMIZER I**
- 2 = COMPILER/OPTIMIZER II**
- 3 = ADVANCED DEVELOPMENT PACKAGE**
- 4 = OVERLAY**

When the disk drive has stopped, remove the distribution diskette, and reinsert the work diskette containing the program to be compiled.

Press 1 or <RETURN> (<RETURN> defaults the compiler to option 1), which selects Compiler/Optimizer I.

Now enter the program name and press <RETURN>.

The compiler translates your program into a speedcode program. The line number of the line being compiled is displayed on the screen. If errors are detected, the compiler displays specific error messages.

Once the compiler is finished, it displays READY. At this point, pressing N tells the compiler that no other programs are to be compiled; pressing any other key restarts the compiler, returning you to the MAIN MENU.

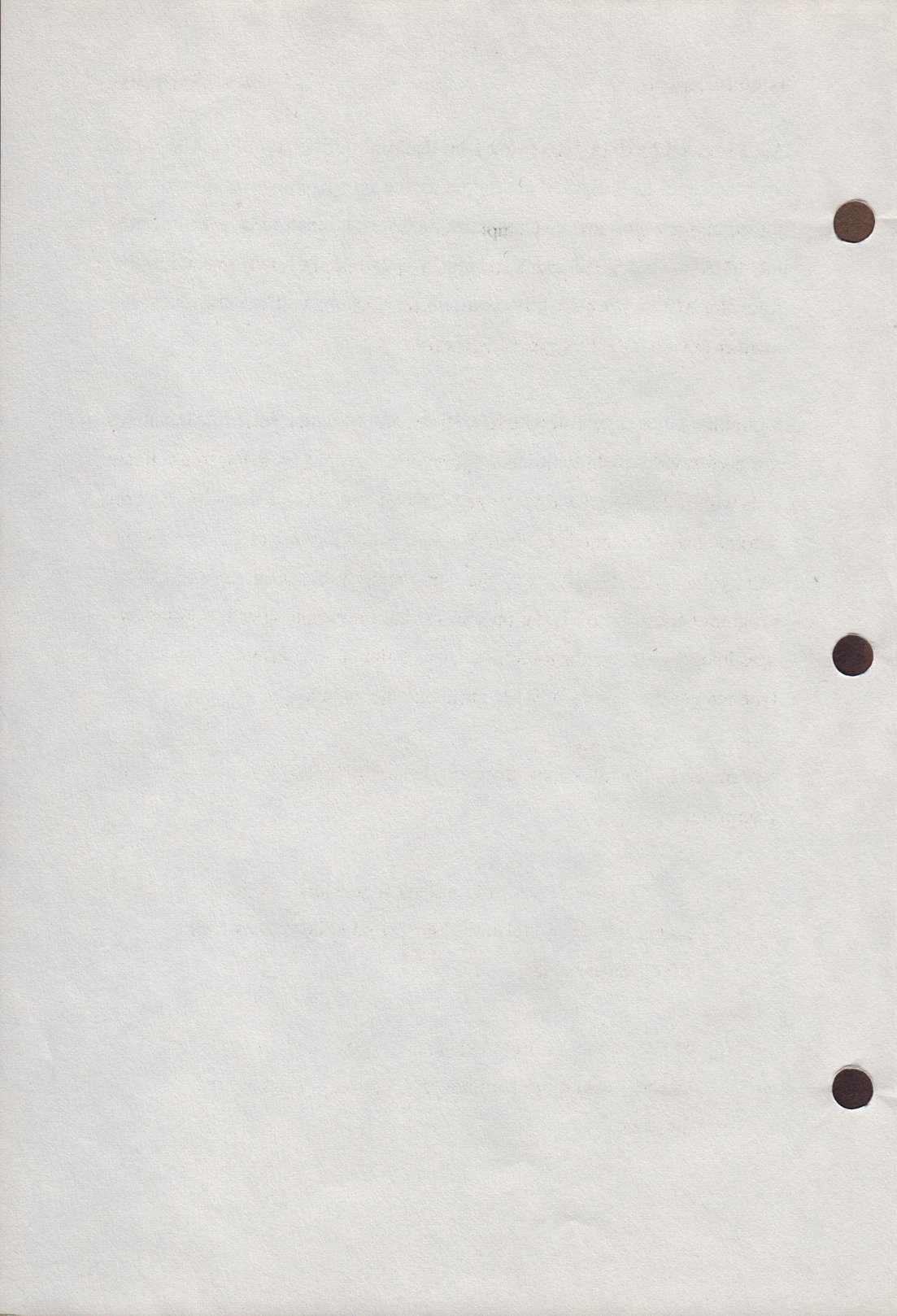
If errors were detected, you should correct the original program, and reRUN the compiler again to recompile. If no errors were detected, BASIC-64 saves the compiled program on the work diskette with a name P-progname. If your program name is "TEST", then the compiled program is named "P-TEST".

To run the compiled program type:

```
LOAD "P-progname", 8 <RETURN>
RUN <RETURN>
```


In a compiled program, errors may be detected during runtime. The same error messages as in BASIC are output to the screen. However, the address in memory at which the error occurred is displayed instead of a statement's line number. Using the address listing, you can easily find the error in the original program (see D in Advanced Development Features described later).

Any of the other capabilities of BASIC-64 , covered later in this text, can be used to decrease execution time.



2. THE COMPILER/OPTIMIZERS

The compiler defaults to Compiler/Optimizer I when you press either <RETURN> or 1. You can also run Compiler/Optimizer II (by pressing 2 from the MAIN MENU). These two selections are different only in the manner in which the program is optimized.

Optimizer I is totally compatible with the BASIC interpreter. Calculations are performed as whole number operations, so long as the whole number falls within the integer value range (-32768 to +32767). Otherwise, they are automatically changed to floating point. Since Optimizer I is completely compatible with BASIC 2.0, the optimizing procedures do not affect program behavior, and serve only to increase program speed. Optimizer I uses integer calculations for speed, so it helps if you change all variables to type integer (by adding a '%' to each variable name).

Optimizer II has functions different from Optimizer I and the BASIC interpreter:

All variables are normally treated as integers (by default), except for string variables, i.e. the compiler places a '%' after each variable.

The division of two integers is performed in whole number operations, unlike the normal floating point division.

Optimizer II ignores decimal places and converts numbers directly into integer data.

Optimizer II is best suited for programs which require mixed variables or which normally do not allow the use of integer variables in BASIC 2.0.

A typical application for Optimizer II:

```
10 A=INT(RND(1)*1000)
```

Variable A is assigned a whole number, even though it is associated with a floating point calculation (RND(1)*1000). Optimizer II will easily compile this program.

Programs to be compiled by Optimizer II should use as few floating point variables as possible. Since Optimizer II doesn't normally handle floating point variables, it is necessary to use the compiler directive REM @ R = so that variables are correctly handled by the compiler. This is discussed later.

Optimizer II has no effect on arrays. Arrays should be treated as whole numbers with a '%' suffix added. (NOTE: this will save a good deal of memory space.)

WARNING!

Use Optimizer II only with programs that you either have written yourself or that you understand completely in terms of operation and logic. This applies to all compiler features that don't rely on the BASIC interpreter (e.g. compiler directives).

3. ADVANCED DEVELOPMENT FEATURES

BASIC-64 offers special options for program development, which can be reached by pressing 3 from the MAIN MENU. All choices appear on the screen in alphabetical order:

- A- Gives you the option of producing 6502/6510 machine code, speedcode, or no code whatsoever. If you choose machine language, compile the program using Optimizer II (the compiled program will have the prefix "M-filename").
- B- Allows you to input the name of a symbol table to be loaded before compiling. A symbol table retains all variables and memory addresses. This is needed when multiple programs require the same variables (e.g. the Overlay Feature).
- C- Allows you to save a symbol table. You can list a symbol table to the screen or printer with the program SYMBOL which is found on the distribution diskette. You can also create a symbol table which is compatible with our Assembler/Monitor using SYMBOL.

- D- Generates an address list. After compiling a program, an optional address list is written to diskette. It can be loaded by typing:

LOAD "Z-filename",8 <RETURN>

and listed with:

LIST

Memory addresses are listed on the left side of the screen and the BASIC line numbers appears on the right (used for finding errors starting a program section with SYS).

- E- Lets you change the end-of-memory address (normal end address for the compiler is 65536, for BASIC = 40960).
- F- Allows you to raise or lower the starting address of a compiled program. The compiler gives you the option of removing the runtime module (see below) and loading it as a separate program. To start the program, you will have to change the starting address and SYS to the starting address (location 16384 is a good location).

- G- Controls the connection of the runtime module to the compiled program. Under some circumstances (for instance, when merging a series of programs with the Overlay Feature) a runtime module separate from the main program(s) may be necessary, which can save disk space and execution time.
- H- Gives you the ability to compile programs written using BASIC extensions. Options are SIMON'S BASIC, VICTREE (EXBASIC II), BASIC 4.0 (as found in MASTER-64) and OTHERS. When choosing OTHERS, you will have to input the starting address, and a few other items, check your manual for the extension. NOTE: Toolkit commands are not compilable. SUPERGRAPHICS extensions are not applicable.
- I- Lets you input the number of bytes per extended BASIC command (normally one byte, except SIMON'S BASIC).
- J - Locates the ELSE command and adjusts the compiler accordingly.
- K - Gives you the option of switching runtime error handling off or on (in other words, the compiler will not halt when an error is encountered). Putting a line 0 in your BASIC program means "error system on".

- L- Suppresses the Overlay Feature, which can disturb character string constants.
- M- Lets you send commands to the disk drive, such as scratching a program that has already been compiled (i.e. the original) to save disk space. After the drive performs its task, disk status is displayed (press <RETURN> to return to the menu).
- N- Displays the directory of the disk on the screen.

4. THE OVERLAY FEATURE

The Overlay Feature is used to compile a number of successive programs which can share the same set of variables.

Press 4 (Overlay Feature) from the MAIN MENU and then press 1 (Overlay Pass 1). The compiler will return to the MAIN MENU. Now compile the first program with Optimizer I, the compiler is just setting up a symbol table on these two passes, so it will not behave "normally". When it displays "READY.", press <RETURN>. Press 4 and then 2 (Overlay Pass 2), and do a "recompiling" of the same program. This second time around the program is compiled and you are returned to the MAIN MENU. Repeat this process with all remaining programs (the compiler will set up a separate variable table called "S-OVERLAY").

REMINDER:

- Since these programs are supposedly layered, be sure to include internal LOAD commands with the proper code prefix for the program name (e.g. 999 LOAD "P-NEXTPART",8).
- The first program must be longer than all subsequent program in the overlay group (do NOT alter the start-of-BASIC by POKEing locations 45 and 46).

- Character strings (e.g. A\$="DATA") are lost during setup of an overlay.

5. COMPILER DIRECTIVES

It is often necessary to inform the compiler of changes during compilation. To do this, you may use Compiler Directives. These directives are inserted into a program as a extra program line as a REM statement, followed by an @ symbol. Thus a compiler directive always being like this:

```
REM@ directive
```

TYPES OF DIRECTIVES:

- Arranging variable addresses:

```
REM@ A variable = address
```

The variable is placed at the address given. For example, you can put an integer variable into a sprite control register and the sprite can be quickly and easily controlled. NOTE: Addresses below 768 are not permitted.

- Switching error handling:

```
REM@ E line number
```


This can be changed manually by pressing K in the submenu. The error system defaults to 'ON' (placing a line 0 in your program also switches it on). Some BASIC extensions offer this, as does BASIC-64.

- Declaring integer variables:

```
REM@ I = variable, variable, ...
```

This is the most frequently used directive.

All named floating point (or other) variables are changed into integers by the compiler for faster execution. Additionally, integers can be used in FOR- NEXT loops (something which the BASIC interpreter normally doesn't allow). This command should only be used with Optimizer II.

- Switching Machine Code Generator:

```
REM@ M
```

From this line onward, the compiler produces machine language. Also, this instruction tells the program to switch to machine language during runtime.

- Switching Speedcode Generator:

REM@ P

From this line onward, the compiler produces speedcode and makes the proper runtime command. NOTE: @M and @P should be used carefully-- you can change between machine code and speedcode only if the individual program sections do not have any GOTOs and GOSUBs to locations outside of that section.

- Switching Optimizers:

REM@ 01 (for Optimizer I)
REM@ 02 (for Optimizer II)

This directive switches optimizers in mid-program, which affects only variables not previously used before these directives.

- Declaring floating point variables:

REM@ R = variable, variable, ...

All variables named are converted into floating point variables. NOTE: This step can only be used by Optimizer II.

EXAMPLE:

```
10 FOR I=1 TO 1000
20 A = SQR(I):PRINT A;
30 NEXT
```

You would have to insert the following line to compile this program with Optimizer II:

```
5  REM@ R=A
```

- Freeing cassette buffer memory:

```
REM@ S address
```

The compiler normally places all variables in the cassette buffer for easy access. You can use this range for you own purposes by supplying an address where you feel the buffer should end. For example:

```
REM@ S 1024
```

6. HINTS FOR PROGRAMMING

Here are a few items to help your programming before compiling:

- A GOTO/GOSUB executes slower than a RETURN.
- An IF-THEN executes slower than a FOR-NEXT loop.
- The speed of GOTO/GOSUB depends on the size of the jump.
- Spaces make the program easier to read, but take more time to execute.
- Structured BASIC programs execute more slowly than unstructured ones.
- When the limits of an array are unknown, such as:

```
10 INPUT X: DIM A(X)
```

the compiler displays the field name, a single parenthesis and a question mark; you respond with the maximum value and <RETURN>. This will occur only with unknown arrays.

- Try to dimension arrays in the first line whenever possible.

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

Programmer's Appendix

Compiler Details

Pass 1: Interprets and optimizes the program, then produces the corresponding code (SPEED-CODE or M/L). The computer displays the current line number and every end-of-line character (colon). Compiler instructions (**REM@**) will display an "R", commands not recognized by Commodore Basic (Basic extensions) are displayed with an "E".

PASS2: Generates the code, the runtime module is merged and the DATA inserted. The screen displays:

DATA - code start: The starting DATA line address in the compiled program.

OBJECT - code start: The starting address of the program.

STRINGS: This is the area for STRING storage. All variables and arrays are stored above the end of STRING range to the end of memory.

EXTENSIONS: If the program uses commands from a BASIC extension the compiler displays the number used.

ERRORS: Lists lines in which errors occurred.

Error Messages:

Normal BASIC errors are identical to interpreted BASIC

RUNTIME: Example is division by zero (A/0)

SYSTEM ERROR: Example: Disk Drive not turned on.

THE ADDRESS LIST:

When RUNTIME errors occur, error messages are displayed. The number presented is a memory location. The line in question can be determined by using the address list generated by OPTION D in the DEVELOPMENT PACKAGE.

To use the address list, note the memory address of the error, LOAD "Z-name",8, and LIST until you reach the proper location. The right side of the list contains the line number which matches the memory address on the left side. This line is normally the incorrect one, but in rare cases, the error may be at the end of the preceding line.

ARRAY Dimensioning:

Arrays must be explicitly dimensioned during compilation. Arrays can exist in memory from \$A000-\$FFFF. Arrays should be dimensioned in the first program line when possible. Sometimes the limits of an array are not known when compiling, in these cases the compiler displays the array name and a question mark. Enter the maximum size of the array and press return.

Example: 10 INPUT x: DIMA(x)

INTEGER LOOPS:

Integer variables are normally not allowed in BASIC loops (eg. FOR I%=1 TO 10: NEXT I%), it is possible to do so with the corresponding compiler instruction. Integer loops are not only faster, they use fewer stack locations and can be nested much deeper than in BASIC. There is a limitation in the use of INTEGER loops. No STEP-value is stored for the sake of speed. The increase is always by 1: STEP is not permitted in INTEGER loops. Faster loops can be made to imitate STEP values, examples:

```
10 I%=1
20 REM LOOP INTERIOR
30 I%=I%+2:IF I%<=1000 THEN 20: REM STEP 2
```



```
5 REM@ I=I
10 FOR I=1 TO 1000
20 REM LOOP INTERIOR
30 I=I+2:NEXT
```

BASIC EXTENSIONS:

BASIC 64 compiles most extensions using a common procedure. No "standard" extension exists for the '64, and commands exist in some extensions that can not be compiled. To compile most BASIC extensions please observe the following rules. A compilable command has the following format:

COMMAND value,value,value

Each value can be a constant, variable or formula of any sort. Here are some commands that comply:

```
PLOT 1,3*SIN(X) TO 4,A*B
GMODE 0,1
```

Many extensions have additional functions, which are used in the following format:

VARIABLE = FUNCTION(VALUE,VALUE,...)

The parenthesis are left off if no value is needed for the function. The compiler doesn't always understand the data resulting from a function. The best thing to do is pre-assign a variable to a function result, so that the data type can be recorded. Example:

```
X = EVAL(A$)
```

```
A$ = INSERT(B$,C$,2)
```

Some DIRECT MODE extension commands are not compilable. Examples: RENUM, TRACE, etc.

Commands that directly change the execution of a program can't be compiled. Examples include (CALL, EXEC, CGOTO, etc.) and program structures (REPEAT, UNTIL, etc.)

Commands that alter memory management can't be compiled. These commands include GLOBAL, LOCAL, etc.

Commands which assign variables values can't be compiled. Examples: INPUTLINE, etc.

Fortunately, most of these limitations are rare ones; useful commands, such as graphic commands, are compilable. The compiler accepts most of the popular BASIC extensions.

SIMON'S BASIC

All of the important commands in Simon's BASIC are acceptable to the compiler. This applies to the graphic commands and multiple functions. The commands listed below do not meet the above requirements and can't be compiled.

- Programmer's Aids (AUTO, RENUMBER, RESET, MERGE)
- INPUT Control (FETCH)
- Number Conversions (%=, \$=)
- Design Commands and PRINT AT
- Program Structures (ELSE is acceptable)
- Error Handling (ON ERROR, NO ERROR, OUT)

DESIGN:

There are several ways to replace this command; the simplest method is shown below:

```
10 A=64*13: REM BASE ADDRESS
20 FOR I=A TO A+62 STEP3
30 READ A$; FOR J =0 TO 2
40 W=0: FOR K=1 TO B
50 W=W*2: IF MID$(A$,J*8+K,1) ="B" THEN W=W+1
60 NEXT: POKE I+J,W:NEXT:NEXT
```


Now you can construct SPRITES as if you had the DESIGN command
(NOTE: the parenthesis replace the command DATA)

It is possible to use DESIGN as a SPRITE-EDITOR: after designing a sprite, the SPRITE can be read from its memory location, saved on disk, and loaded as needed. The advantage to this method is that a program using SPRITES will not have to be recompiled to change the SPRITE.

PRINT AT:

To replace the PRINT AT (X, Y) ; Z command use:

```
POKE211,X: POKE214,Y:SYS 58732:PRINT Z
```

PROGRAM STRUCTURES:

The programming structures of Simon's BASIC are easily replaced with the usual commands GOTO, GOSUB and IF.

BASIC 4.0 (MASTER-64)

Some BASIC extensions have syntax that can lead to errors after compiling:

```
COPY D0 TO D1
```

D0 and D1 represent disk drives, the compiler will view these as variables and attempt calculations in the program. A way to rework the syntax so the compiler can use these commands is to put the variables into parenthesis. Here is a BASIC 4.0 example:

```
10 INPUT "DRIVE 1 OR 0";X  
20 CATALOG D(X)
```

Some of the remaining commands in MASTER-64 are also compilable, as long as variables, formulas are set in parenthesis. Many MASTER commands interfere directly with memory management, and are not compilable (such as those that define screen-zones, data buffers, etc.)

COMPILING BASIC EXTENSIONS:

Remember, your program must not contain any non-compilable commands. BASIC extensions are constantly being updated, so whether commands can be compiled may depend on the version being used. When in doubt, experiment!

To compile the program, remove the BASIC extension package from memory (by removing the cartridge or resetting your '64) after saving your program.

Start the compiler and choose Option 3 from the main menu. Next choose the BASIC extension using option "H".

During PASS 1, when the compiler finds an extension command it will display an "E". The compiler does not recognize these extended commands from the codes stored in the standard BASIC interpreter.

After PASS 2 is completed, the compiler displays the number of BASIC extension commands used.

Before running the compiled program, load the BASIC extension (insert cartridge or load the extensions from diskette). If the program has uncompiled errors, either the extension will display an ERROR message or the program will not run. If you follow the rules above, regarding command compatibility, there should be no problems.

Memory locations 704 (sprite 11) to 767 should NOT be changed with POKE commands, mainly because Simon's BASIC uses these registers for purposes other than sprites.

OPERATION OF COMPILED COMMANDS FROM AN EXTENSION

To execute a compiled extension command, the program gives the command to the extended interpreter, which then executes the command. Extended compiled commands do not execute any faster, only the standard Commodore BASIC commands are speeded up. By using the rules below you can compile extensions and reach higher speeds.

Use as few extended commands as possible.

Use extended commands in program sections where time is not a critical factor.

When employing graphics for calculation displays, use them after and not during the calculations.

Use one complex command instead of several short ones.

When an extended command can be replaced by a standard BASIC command, do so. This is especially valuable for character string functions; complex character string formulas in BASIC work faster than corresponding single extended commands.

Graphic commands should use only whole numbers for coordinates. Calculating coordinates and other graphics should work best with integers variables.

OTHER EXTENSIONS

BASIC extension not listed in option "H", can also be compiled if the meet the criteria listed above. The best way to find out is to try and compile an extended program using each "H" option. You must enter the values for end- of-memory, number of bytes per command and the code for ELSE. These must be determined before running the compiler. If they are not given in the extension documentation use the following procedures.

END-OF-MEMORY: PRINT PEEK(55) +256 * PEEK(56)

BYTES PER TOKEN: TYPE NEW <RETURN> then enter 10
(EXTENSION COMMAND) <RETURN>

PRINT PEEK(2054) If this value is not 0 then enter 2 in BYTES PER
TOKEN

ELSE CODE: Enter NEW <RETURN> then enter 10 ELSE <RETURN>
PRINT PEEK(2053) This value gives the first byte of the command.
The second byte is only needed by extensions with two bytes per command
PRINT PEEK(2054)

Remember a colon (:) must proceed ELSE when using the IF...THEN...ELSE command

ERRORS

The number of an error in a compiled programs is stored in location 700 and can be read by PEEK(700). Below is the list of messages used to check for certain errors (PEEK (700) = 5; Device not present--i.e. Printer not turned on)

- | | |
|-------------------------|------------------------|
| 1 too many files | 16 out of memory |
| 2 file open | 17 undef'd statement |
| 3 file not open | 18 bad subscript |
| 4 file not found | 19 redim'd array |
| 5 device not present | 20 division by zero |
| 6 not input file | 21 illegal direct |
| 7 not output file | 22 type mismatch |
| 8 missing filename | 23 string too long |
| 9 illegal device number | 24 file data |
| 10 next without for | 25 formula too complex |
| 11 syntax | 26 can't continue |
| 12 return without gosub | 27 undef'd function |
| 13 out of data | 28 verify |
| 14 illegal quantity | 29 load |
| 15 overflow | |

CODE-START: Option "F" allows you to raise the start of the program. If the RUN-TIME module is loaded separately you must start the program with a SYS address command. A good location to set this to is 16384, this frees a graphic screen at 8192-16191 and 16192-16383 may be used for sprites. Programs using the graphic screen are also affected by the interpreter. You must enter `POKE44,64:POKE 16384,0:NEW`; now you can develop and compile graphic programs.

STOP KEY: `POKE 788,PEEK(788)+3` to disable the <STOP> key.

MEMORY LAYOUT

MEMORY MAP for compiled program:

0-1024	system memory
1024-2048	screen memory
2048-code start	Runtime module
code start-start of strings	Program code
start of strings-end of strings	Character strings
end of strings-top of memory	variables and arrays

Compiled program variables:

Integer: 2 bytes - low byte, high byte

Floating-point: 5 bytes - exponent, 4-byte mantissa

String variables: 3 bytes - length, low byte, high byte

Strings: 2 bytes plus 1 byte per character

Variable addresses are found in the symbol table produced in OPTION 3. The locations or length of the strings can't be altered after compiling the program.

Locations 144-828 are used by the operating system.

REM@ S 1024 will free the cassette buffer.

The range from \$2C0 to \$2FF is reserved for BASIC extensions; usually this memory is free.

To free the memory above \$C000, the top of memory must be set at 49152 in OPTION 3, selection "E".

LOAD "NAME",8,128 will LOAD data into the locations from which it was saved. Unlike the LOAD "NAME",8,1 the program will continue onward. This is useful for graphics and assembler routines.

MEMORY SWITCHING:

POKE 1, 52: allows access to \$A000-\$FFFF

POKE 1, 51: character generator in \$D000-\$DFFF.

POKE 1, 55: normal

Compiled programs can run without the BASIC ROM's but there are some limitations.

The interrupt must be switched off when changing location 1.

Use:

POKE 56334, PEEK(56334) AND 254 to switch off the interrupt.

Use:

POKE 566334, PEEK(56334) OR 1 to turn on the interrupt.

When the interrupt is off input/output commands and floating point operations should not be attempted.

FLOATING-POINT FUNCTIONS:

Complex floating-point functions (SIN, COS, TAN, LOG, etc.) work very slowly. Although these functions are accurate to 9 decimal places after compiling the program will run much faster if simpler equations are used. Below are a few examples:

$$\text{SIN}(X) = X - X^3/6 \quad (\text{for: } -\pi/2 < x < \pi/2)$$

$$\text{COS}(X) = 1 - X^2/2$$

$$\text{TAN}(X) = X + X^3/3$$

A small increase is possible by switching off the video processor with:

POKE 53265,PEEK(53265) AND 239: REM SCREEN OFF

POKE 53265,PEEK(53265) OR 16: REM SCREEN ON

RS-232 OPERATIONS

To use the RS-232 port on the Commodore 64 you must remember to lower memory to protect the RS-232 buffer area. Use Option 3 item E and lower to 39936. The cassette buffer can not be used for storage when using the RS-232 port with the compiler. You must remember to lower memory to protect the buffer area.

TECHNICAL NOTES

There are actually four different options:

1. Optimizer 1 - P code
2. Optimizer 1 - Machine code
3. Optimizer 2 - P code
4. Optimizer 2 - Machine code

The code output (P code or Machine code) is selected from the Advanced Development menu, item A.

The difference between the two optimizers is primarily that optimizer 1 assumes the ordinary variables (floating point) are to be handled as floating point variables and are allocated five bytes each. Optimizer 2 assumes that these variables should be treated as integer variables and the only way you can have a floating point variable in optimizer 2 is by telling the compiler about it through use of the REM@ statements.

To view it another way, suppose you have a program with both normal floating point variables (such as A) and integer variables (A%). In optimizer 1, A will be assigned as a floating point (five bytes internally) and treated as a floating point variable, and A% will be treated as an integer variable (two byte assignment). Note that this is better than the five bytes allocated in normal basic for a fixed point variable.

In optimizer 2, both will be treated as integer variables. The only way you can have A treated as a floating point is by doing a REM@ R=A.

The compiler starts assigning space to variables in the cassette buffer and then assigns them down from your top of memory address. You can change the start point in cassette buffer memory by a REM@ S command and give the starting address (838-1023). If the address is 1024 or above, the compiler assumes that you do not want any variables in buffer memory and therefore will only assign them down from the top of memory.

Note that some programs and extensions to BASIC use the cassette buffer memory as work space and therefore, with these programs, you will want to make sure that you use a REM@ S=1024, or your variables will be assigned in the work space with disastrous results.

You can see the assignment of variables in memory by using the symbol table output option. You do this by selecting option 3 from the Advanced Development feature. Then select option C. It will ask for a file name where you want the symbol table. This file name will be prefixed with an "S-", and stored on the disk during the compile process. When the compile process is completed, you must reset the 64 and then load the program "SYMBOL" from the BASIC 64 disk. You will be asked for the file name (do not key the S-), just the name you entered under option C). Then select option 2 for a listing to device 4 (the printer). The listing will show string variables and functions first, then arrays, then ordinary variables. They are sorted with the major sort on the second letter (blank in the case of single letter variables) and then the first letter. Each is followed by the decimal address of its assignment. If you want to see how the compiler assigns them, you must remember that addresses are generally assigned in encounter sequence (the sequence in which they are first used in your program). If you hunt for the first variable in the list you will see that it is either in the cassette buffer or starting down from the top of memory you allocated, then the next one down, and so forth.

You will notice that all strings are assigned a 3 byte space allocation. This is sufficient space for a pointer to the beginning of the string in the strings work area (2 bytes) and 1 byte for the length of the string. At the end of the compile, it prints out the amount of space allocated for strings. This is a string storage pool where all string variables are stored. Since all numerical variables are pre-assigned, the FRE() command only shows the amount of space remaining in this string storage pool.

Although this compiler works with almost any type of BASIC program a rule which will result in more free space is that all DIM statements should appear at the beginning of your program. This is because the compiler allocates space as it encounters variables. The first time it sees an array variable, it will allocate (just like BASIC) room for eleven entries (0-10). If later in your program you have a DIM statement for 21 entries (DIM(20)), the compiler will then allocate space for 21 entries of the variable. It will change the address pointer to this new 21 element space, but the 11 entry space previously allocated will be unused and wasted. Therefore all DIM statements should be moved up to the beginning of your program (it is not sufficient to have them as the first executed statements, even though they may be at the back of the program and executed with a GOTO).

The compiler stores the runtime routines starting at address 1025 (just like BASIC). There is no way to relocate this loader, it must always be at 1025.

The stack used for FOR/NEXT and GOSUB/RETURN is quite different from BASIC. First, it is larger, you can do about 70 GOSUBS without RETURNS before the stack will overflow. The system does not use the 6502 stack. Instead, it uses its own stack with the pointer located at address 64 (decimal). If you need to do the equivalent of a "pop" command to clear the stack, you simply poke the value 206 in location 64 [POKE 64,206]. This resets the stack pointer. This is useful in programs where you exit from GOSUB routines by the GOTO command, and this can be issued back at the main menu to clean up the stack.

If the program uses RS-232 operations, you must allocate 512 bytes at the top of memory for RS-232 buffer space. If the normal starting point was 40959 as the top of memory, and you are using RS-232, simply specify the top of memory as 40447 (40959-512) from the Advanced Development menu.

You can use BASIC extensions (such as VIDEO BASIC) with BASIC 64. Simply change any "pop" commands as mentioned above, and compile the program. As the system encounters VIDEO BASIC tokens, it will show you this with an E. It appears to operate all of these well except for the ones which require seeking a line number (such as sprite definitions and tone command setups). You have to change these by pre-compiling the sprites and loading them into memory yourself, since line numbers don't exist after the compilation is finished.

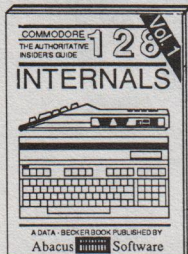
There is a difference in the way the compiler handles a "FOR/NEXT" loop which should not be executed at all. For example, for a factorial routine, you would normally do "FOR I=1 to FT: V=V*I:NEXT". This is fine as long as FT is 1 or greater. If you go in with FT = 0, it results in a FOR I=1 to 0. In standard interpreted BASIC, the expression will be executed once and then at the NEXT command it will finish. In BASIC 64, it does not execute at all.

Note that the top of memory means top of available memory, not top of memory plus 1 as we are used to specifying with BASIC program loading and saving.

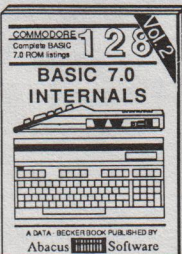
A REM statement takes no space at all in the compiled version. Therefore, you can make your programs much clearer by including REMs in the source code. The runtime routines take about 3K of memory, so you can have about about 3K of REMs in your program, and still compile it and it should fit into memory.

C-128[™] LIBRARY

and C-64[™]



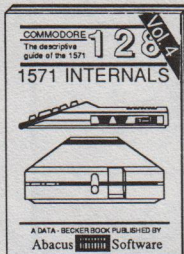
Detailed guide presents the 128's operating system, explains graphic chips, Memory Management Unit, 80 column graphics and commented ROM listings. **500pp \$19.95**



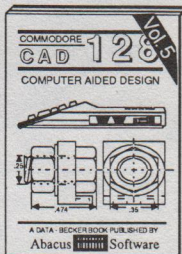
Get all the inside information on BASIC 7.0. This exhaustive handbook is complete with commented BASIC 7.0 ROM listings. Coming Summer '86. **\$19.95**



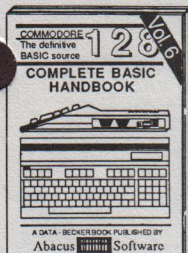
Filled with info for everyone. Covers 80 column hi-res graphics, windowing, memory layout, Kernel routines, sprites, software protection, autostarting. **300pp \$19.95**



Insiders' guide for novice & advanced users. Covers sequential & relative files, & direct access commands. Describes DOS routines. Commented listings. **500pp \$19.95**



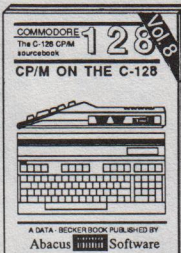
Learn fundamentals of CAD while developing your own system. Design objects on your screen to dump to a printer. Includes listings for '64 with Simon's Basic. **300pp \$19.95**



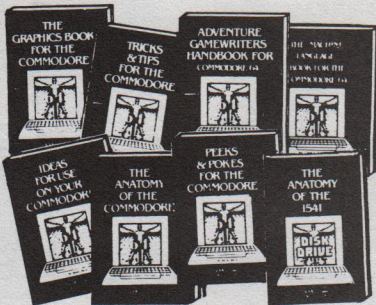
Introduction to programming; problem analysis; thorough description of all BASIC commands with hundreds of examples; monitor commands; utilities; much more. **400pp \$19.95**



Presents dozens of programming quick-hitters. Easy and useful techniques on the operating system, stacks, zero-page, pointers, the BASIC interpreter and more. **\$19.95**



Essential guide for everyone interested in CP/M on the 128. Simple explanation of the operating system, memory usage, CP/M utility programs, submit files & more. **\$19.95**



ANATOMY OF C-64 Insider's guide to the '64 internals. Graphics, sound, I/O, kernel, memory maps, more. Complete commented ROM listings. **300pp \$19.95**

ANATOMY OF 1541 Explains all. Many examples and tips. Includes commented 1541 ROM listings. **500pp \$19.95**

MACHINE LANGUAGE C-64 Learn 6510 code write fast programs. Many samples and listings for complete assembler, monitor, & simulator. **200pp \$14.95**

GRAPHICS BOOK C-64 - best reference covers basic and advanced graphics. Sprites, animation, Hires, Multicolor, lightpen, 3D-graphics, IRQ, CAD, projections, curves, more. **350pp \$19.95**

TRICKS & TIPS FOR C-64 Collection of easy-to-use techniques: advanced graphics, improved data input, enhanced BASIC, CP/M, more. **275pp \$19.95**

1541 REPAIR & MAINTENANCE Handbook describes the disk drive hardware. Includes schematics and techniques to keep 1541 running. **200pp \$19.95**

ADVANCED MACHINE LANGUAGE Not covered elsewhere: video controller, interrupts, timers, clocks, I/O, real time, extended BASIC, more. **210pp \$14.95**

PRINTER BOOK C-64/VIC-20 Understand Commodore, Epson-compatible printers and 1520 plotter. Packed: utilities; graphics dump; 3D-plot; commented MPS801 ROM listings, more. **330pp \$19.95**

SCIENCE/ENGINEERING on C-64 In depth into computers in science. Topics: chemistry, physics, biology, astronomy, electronics, others. **350pp \$19.95**

CASSETTE BOOK C-64/VIC-20 Comprehensive guide; many sample programs. High speed operating system fast file loading and saving. **225pp \$14.95**

IDEAS FOR USE ON C-64 Themes: auto expenses, calculator, recipe file, stock lists, diet planner, window advertising, others. Includes listings. **200pp \$12.95**

COMPILER BOOK C-64/C-128 All you need to know about compilers: how they work; designing and writing your own; generating machine code. With working example compiler. **300pp \$19.95**

Adventure Gamewriter's Handbook Step-by-step guide to designing and writing your own adventure games. With automated adventure game generator. **200pp \$14.95**

PEEK & POKES FOR THE C-64 Includes in-depth explanations of PEEK, POKE, USR, and other BASIC commands. Learn the "inside" tricks to get the most out of your '64. **200pp \$14.95**

Optional Diskettes for books For your convenience, the programs contained in each of our books are available on diskette to save you time entering them from your keyboard. Specify name of book when ordering. **\$14.95 each**

C-128 and C-64 are trademarks of Commodore Business Machines.

Abacus Software

P.O. Box 7219 Grand Rapids, MI 49510 - Telex 709-101 - Phone (616) 241-5510

Call now for the name of your nearest dealer. Or to order directly by credit card, MC, AMEX or VISA call (616) 241-5510. Other software and books are available—Call and ask for your free catalog. Add \$4.00 for shipping per order. Foreign orders add \$10.00 per book. Dealer inquiries welcome—1200+ nationwide.

Trusted Software and

Language Software

ADA TRAINING COURSE

Teaches you the language of the future. Comprehensive subset of language. Includes: editor; syntax checker, compiler; assembler; disassembler, hand-book. **\$39.95**



ASSEMBLER/MONITOR

Macro assembler and extended monitor. Supports all standard functions plus floating point constants. Monitor supports bank switching, quick trace, single step, more. **\$39.95**

LDA JSR DEC
INX INY TYA TAX
ROR BRK JMP INC
ROL STA STY
PLA RTS CMP STX
SEC PLP SED

BASIC COMPILER

Compiles complete BASIC language into either fast 6510 machine language and/or compact speedcode. Supports overlays and many extended BASIC commands. C-64 **\$39.95**
C-128 **\$59.95**



SUPER PASCAL

Full Pascal supports graphics, sprites, file management, more. Supports pointers, dynamic memory management, machine language. Compiles to fast 6510 machine code. C-64 **\$39.95**
C-128 **\$59.95**



FORTH LANGUAGE

Based on Forth 79 (+ parts of '83). Supports hires graphics and sound synthesizer. Full screen editor, programming tools, assembler, samples, handbook. **\$39.95**



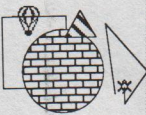
MASTER

Professional development package for serious applications. Indexed file system, full screen & printer management, programmer's aid, multi-programmer math, royalty-free runtime, more. **\$39.95**



VIDEO BASIC

Add 50+ graphic, sound and utility commands to your programs with this super development package. Free distribution of RUNTIME version - no royalties! **\$39.95**



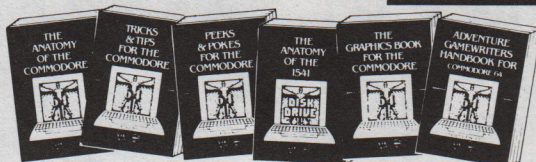
XREF-128 & XREF-64

BASIC cross-reference. Indispensable tool for BASIC programmers. Finds all references to variables, constants & line numbers. Sorts in alphabetical order. C-64 **\$17.95**
C-128 **\$17.95**



For Commodore Computers

Reference Books



ANATOMY OF C-64 Insider's guide to the '64 internals. Graphics, sound, I/O, kernel, memory maps, and much more. Complete commented ROM listings. **300pp \$19.95**

ANATOMY OF 1541 DRIVE Best handbook on this drive, explains it all with many examples programs and ROMs. Fully commented 1541 ROM listings. **500pp \$19.95**

MACHINE LANGUAGE FOR C-64 Learn 6510 code & write fast programs. Many samples and listings for complete assembler, monitor and simulator. **200pp \$14.95**

GRAPHICS BOOK FOR C-64 Best reference, covers basic and advanced graphics. Sprites, Hires, Multicolor, 3D-graphics, IRQ, CAD, projections, curves, more. **350pp \$19.95**

TRICKS & TIPS FOR C-64 Collection of easy-to-use techniques: advanced graphics, improved data input, enhanced BASIC, CP/M, data handling and more. **275pp \$19.95**

1541 REPAIR & MAINTENANCE Handbook on the drive's hardware. Includes schematics & techniques to keep 1541 running. Align drive w/ & w/o scope. Large handbook size. **\$19.95**

ADVANCED MACHINE LANGUAGE Subjects not covered elsewhere: video controller, interrupts, timers, I/O, extensions to BASIC. Tips for the serious programmer. **210pp \$14.95**

PRINTER BOOK C-64/VIC-20 Understand Commodore, Epson compatible printers & 1520 plotter. Utilities, screen dump, 3D-plot, commented MPS-801 ROM listings. **330pp \$19.95**

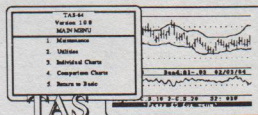
SCIENCE/ENGINEERING ON C-64 In-depth introduction to computers in science. Some topics covered are chemistry, physics, astronomy, electronics & others. **350pp \$19.95**

CASSETTE BOOK C-64/VIC-20 Make your cassette run faster than a disk drive! Cassette data-base, disk to tape backup, tape to disk, FastTape operating system. **225pp \$14.95**

Productivity Tools

TECHNICAL ANALYSIS SYSTEM

A sophisticated charting and technical analysis system for serious investors. By charting and analyzing the past history of a stock, TAS can help pinpoint trends & patterns and predict a stock's future. TAS lets you enter trading data from the keyboard or directly from online financial services. **\$59.95**



PERSONAL PORTFOLIO MANAGER

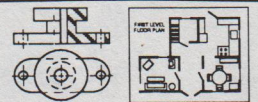
Complete portfolio management system for the individual or professional investor. Allows investors to easily manage their portfolios, obtain up-to-the minute quotes & news, and perform selected analysis. **\$39.95**

TAS Report
Portfolio and Box
Alpha Sector 1987 Performance 31 Oct and Regular USA 08/28
10/17/87

Symbol	Type	Vol	Price	Open	Close	High	Low	Vol	Cost	Gain	Loss
IBM	Stock	50	84.75	10.00	10.00	10.00	10.00	10.00	4237.50	0.00	0.00
MS	Stock	10	17.00	17.00	17.00	17.00	17.00	17.00	170.00	0.00	0.00
TOT									4407.50	0.00	0.00

CADPAK

A deluxe graphics design and drawing package. Use with or without an optional lightpen to create highly-detailed designs. With dimensioning, scaling, text, rotation, object libraries, hardcopy and more. **\$39.95**



DATAMAT

Powerful, easy-to-use data management package using menu selections. Free-form design, 50 fields/record, 2000 records/disk. Sort on multiple fields in any combination. Complete selection and formatting for printing reports. **\$39.95**



Authoritative Books

From Abacus Software
...a name you can count on



BOOKS COVERING THE C-128

IDEAS FOR USE ON C-64 Themes: auto expenses, calculator, recipe file, stock lists, diet planner, window advertising, others. Includes all program listings. **200pp \$19.95**

COMPILER BOOK C-64/C-128 All you need to know about compilers: how they work, creating your own and generating the final machine code. **300pp \$19.95**

Adventure Gamewriter's Handbook A step-by-step guide to designing and writing your own adventure games. Adventure game generator & four example games. **200pp \$14.95**

PEEKs & POKES FOR THE C-64 Includes in-depth explanations of PEEK, POKE, USR, and other BASIC commands. Learn the "inside" tricks about your 64. **200pp \$19.95**

OPTIONAL DISKETTES FOR BOOKS For your convenience, the programs contained in each of our books are available on diskette. All program thoroughly tested & error-free. Specify title of book when ordering. **\$14.95 each**

C-128 INTERNALS Detailed guide presents the 128's operating system, explains the graphics chips, Memory Management Unit, and commented listing of Kernal. **500+pp \$19.95**

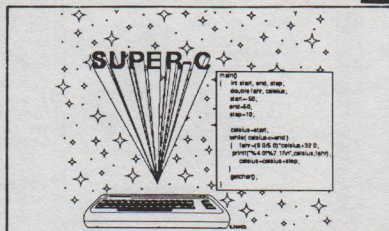
1571 INTERNALS Insiders' guide for novice and advanced users. Covers sequential & relative files, and direct access commands. Describes important DOS routines. Commented DOS listings. **500+pp \$19.95**

C-128 TRICKS & TIPS Check full of info for everyone. Covers 80 column hi-res graphics, windowing, memory layout, Kernal routines, sprites and more. **300 pp \$19.95**

CP/M ON THE C-128 Essential guide to using CP/M on your 128. Simple explanations of the operating system, memory usage, CP/M utility programs, submit files and more. **\$19.95**

COMPUTER AIDED DESIGN on your C-128 or 64. Create a CAD system using programs provided. Covers 3D objects & rotation, MACROS, hatching, zooming, mirroring, line widths, dashed lines, more. **300 pages \$19.95**

Special Feature



For school or software development, choose **SUPER C**. **SUPER C's** powerful screen editor is full-functioned with horizontal and vertical scrolling, copy and search/replace for easy editing. Source files may be up to 41K.

The **SUPER C** compiler is fast and creates link files. Up to seven separate modules may be linked into a ready-to-run object program. To maintain C's portability, **SUPER C** supports the Kernighan & Ritchie standard (without bit fields), making it very complete. **SUPER C** also includes a complete I/O library.

Other features of the **SUPER C** package:

- convenient hexadecimal and octal input
- error file listed to diskette
- supports conditional compiling
- complete strings and arrays
- full mathematical functions

C-64 \$79.95

C-128 \$79.95

Ordering Information

Abacus Software



P.O. Box 7211 Grand Rapids, Michigan 49510

For Postage and handling include \$4.00 per order. Foreign orders include \$10.00 per item. Money order and checks in U.S. Dollars only. MasterCard, VISA and American Express accepted. Michigan residents please include 4% sales tax.

For fast service call (616) 241-5510 Telex 709-101

For free catalog, please return this coupon or a copy to:
Abacus Software, P.O. Box 7211, Grand Rapids, MI 49510

PHONE: (616) 241-5510

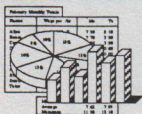
XPBR

Capture your information on **XPBR's** knowledge base and let this first expert system for Commodore computers help you make important decisions. Large capacity. Complete with editing & reporting. **\$59.95**



POWERPLAN

One of the most powerful spreadsheets with integrated graphics for your Commodore computer. Includes menu or keyword selections, online help screens, field protection, windowing, trig functions and more. PowerGraph lets you create integrated graphs and charts from your spreadsheet data. **\$39.95**



QUICKCOPY V2.0

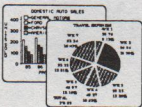
Back up your valuable data with the fastest disk copier we've seen to date. Copies an entire disk in two and a half minutes on two drives or three and a half on one. **\$19.95**



CHARTPAK

Make professional-quality pie, bar and line charts, and graphics from your data. Includes statistical functions. Accepts data from CalcResult and MultiPlan. C-128 has 3X the resolution of the C-64 version. Outputs to most printers.

C-64 \$39.95
C-128 \$39.95



Name _____
Address _____
City _____
State _____ Zip _____

'128™ and C-64™ SOFTWARE

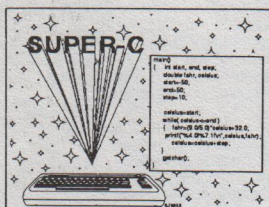


Our BASIC Compilers are the *complete* compiler and development packages. Speed up your programs from 5x to 35x.

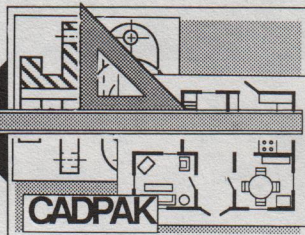
Our BASIC Compilers give you many options: flexible memory management; choice of compiling to machine code, compact p-code or a mixture of both. Also on the 128, 40 or 80 column monitor output and FAST-mode operation.

The 128 Compiler's extensive 80-page programmer's guide covers compiler directives and options, two levels of optimization, memory usage, input/output handling, 80 column hi-res graphics, faster, higher precision math functions, speed and space saving tips, more. A great package that no software library should be without.

BASIC 128 Compiler \$59.95
BASIC 64 Compiler \$39.95



For school or software development, select **SUPER C**. Learn to use one of today's most popular languages on your Commodore 128. Powerful screen editor, compiler, linker-link up to seven modules, I/O library—includes printf and fprintf. Compile your C programs into *fast* machine language. C-128 \$79.95 C-64 \$79.95



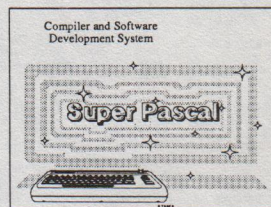
CADPAK is a remarkably easy to use drawing package for accurate graphic designs.

Using CADPAK's new dimensioning features you can create exact scaled output to all major dot-matrix printers.

This enhanced version of CADPAK allows you to input via the keyboard or a high quality lightpen.

Two graphic screens permit you to COPY from one screen to another. DRAW, LINE, BOX, CIRCLE, ARC, ELLIPSE are but a few of the many selections to choose from. FILL objects with preselected PATTERNS; add TEXT; SAVE and RECALL designs to/from disk. You can define your own library of intricate symbols/objects with the easy-to-use OBJECT MANAGEMENT SYSTEM—it will store up to 104 separate objects.

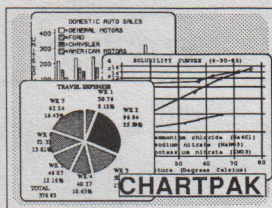
C-128 \$59.95
C-64 \$39.95



Compiler and Software Development System

Not just a compiler, but a complete system for developing applications in Pascal. Extensive editor with search, replace, auto, renumber, etc. Standard J & W compiler that generates fast machine code. If you want to learn Pascal or to develop software using the best tools available—**SUPER Pascal** is your first choice.

C-128 \$59.95
C-64 \$59.95



For the professional who wants to easily create high quality charts and graphs without programming. You can immediately change the scaling, labeling, axis, bar-filling, etc. to suit your needs. Accepts data from CalcResult and MultiPlan. C-128 version has 3X the resolution of the '64 version. Outputs to most printers.

C-128 \$39.95
C-64 \$39.95

PowerPlan

One of the most powerful spreadsheets with integrated graphics for your Commodore computer. Includes menu or keyword selections, online help screens, field protection, windowing, trig functions and more. Power-Graph, the graphics package, is included to create integrated graphs & charts. C-64 \$39.95

OTHER TITLES AVAILABLE:

Technical Analysis System

A sophisticated charting and technical analysis system for serious investors. By charting and analyzing the past history of a stock, TAS can help pinpoint trends & patterns and predict a stock's future. Enter data from the keyboard or from online financial services.

C-64 \$59.95

Personal Portfolio Manager

Complete portfolio management system for the individual or professional investor. Easily manage your portfolios, obtain up-to-the minute quotes and news, and perform selected analysis. Enter quotes manually or automatically through Warner Computer Systems.

C-64 \$39.95

Xper

XPER is the first "expert system" for the C-128 and C-64. While ordinary data base systems are good for reproducing facts, XPER can derive knowledge from a mountain of facts and help you make expert decisions. Large capacity. Complete with editing and reporting.

C-64 \$59.95

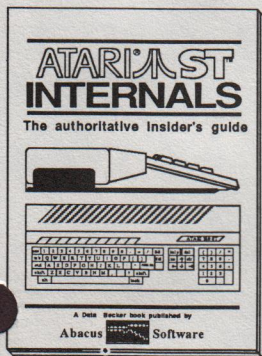
C-128 and C-64 are trademarks of Commodore Business Machines.

Abacus Software

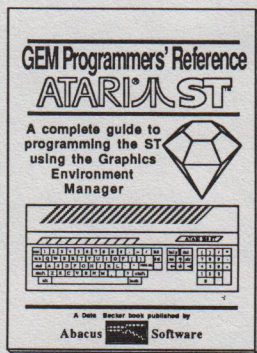
P.O. Box 7219 Grand Rapids, MI 49510 - Telex 709-101 - Phone (616) 241-5510

Call now for the name of your nearest dealer. Or to order directly by credit card, MC, AMEX or VISA call (616) 241-5510. Other software and books are available—Call and ask for your free catalog. Add \$4.00 for shipping per order. Foreign orders add \$12.00 per item. Dealer inquiries welcome—1200+ nationwide.

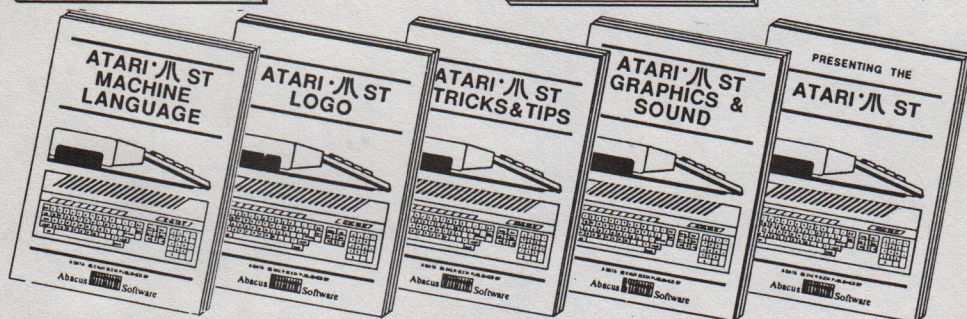
THE ATARI[®] ST[™] REFERENCE LIBRARY



Essential guide to learning the inside information on the ATARI ST. Written for the user who wants thorough and complete descriptions of the inner workings of the ST. Detailed descriptions of the sound and graphics chips, the Centronics and RS-232 ports, GEM, important system addresses and plenty more. Also included is a complete documented BIOS assembly listing. This indispensable reference is a required addition to your ATARI ST library. 450 pages. \$19.95



For the serious programmer in need of detailed information on the GEM operating system. Written especially for the Atari ST with an easy-to-understand format that even beginners will be able to follow. All GEM routines and examples are written in C and 68000 assembly language. Covers working with the mouse, icons, Virtual Device Interface (VDI), Application Environment Services (AES) and the Graphics Device Operating System. Required reading for the serious programmer interested in understanding the ST. \$19.95



Make full use of your ATARI ST. Learn the 68000 assembly language, its number system, use of registers, structure and details of the instruction set, and use of system routines. \$19.95

Take control of your ATARI ST by learning LOGO—the easy-to-use, yet powerful language. Topics include structured programming, graphic movement, file handling and more. \$19.95

Treasure trove of fascinating tips & tricks lets you make full use of the ST. Fantastic graphics, refining programs in BASIC, assembler, and C and advanced programming techniques. \$19.95

An in-depth book shows you how to create fascinating graphics and surprising music & sound from the ST. See and hear what visual and audio effects are possible on the ATARI ST. \$19.95

Describes the features of the ATARI ST to help you decide if you want to purchase this sensational new computer. Discusses GEM, the operating system, 68000 chip instructions. \$16.95

Abacus[®] Software

P.O. Box 7211 Grand Rapids, MI 49510 - Telex 709-101 - Phone (616) 241-5510

Call now for the name of your nearest dealer. Or order directly from ABACUS with your MasterCard, VISA, or American Express card. Add \$4.00 for postage and handling. Foreign add \$8.00 per item. Other software and books coming soon. Call or write for free catalog. Dealer inquiries welcome—over 1200 dealers nationwide.

