ST-LOG™

THE ATARI ST OPERATOR'S MAGAZINE

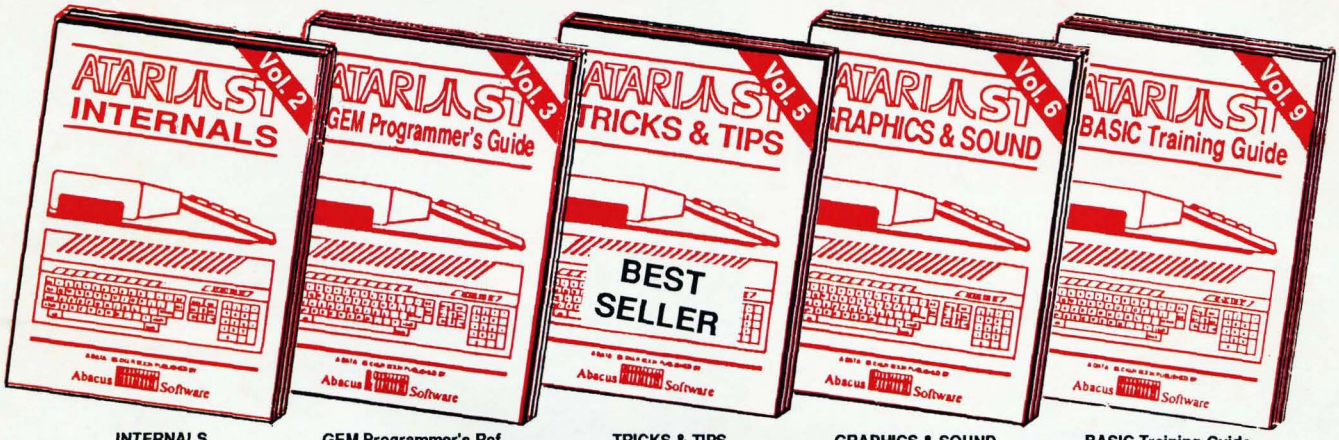
JUNE 1986 ISSUE 3

THIS ISSUE:

VDI Sampler

ATARI® ST™

REQUIRED READING



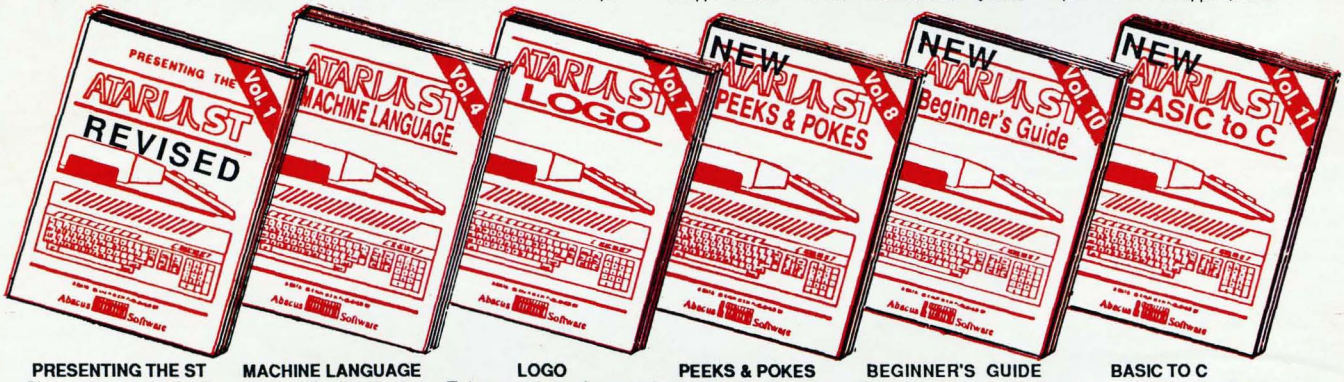
INTERNALS
Essential guide to learning the inside information of the ST. Detailed descriptions of sound & graphics chips, internal hardware, various ports, GEM. Commented BIOS listing. An indispensible reference for your library. 450pp. $19.95

GEM Programmer's Ref.
For serious programmers in need of detailed information on GEM. Written with an easy-to-understand format. All GEM examples are written in C and assembly. Required reading for the serious programmer. 450pp. $19.95

TRICKS & TIPS
Fantastic collection of programs and info for the ST. Complete programs include: super-fast RAM disk; time-saving printer spooler; color print hardcopy; plotter output hardcopy. Money saving tricks and tips. 200 pp. $19.95

GRAPHICS & SOUND
Detailed guide to understanding graphics & sound on the ST. 2D & 3D function plotters, Moiré patterns, various resolutions and graphic memory, fractals, waveform generation. Examples written in C, LOGO, BASIC and Modula2. $19.95

BASIC Training Guide
Indispensible handbook for beginning BASIC programmers. Learn fundamentals of programming. Flowcharting, numbering system, logical operators, program structures, bits & bytes, disk use, chapter quizzes. 200pp. $16.95



PRESENTING THE ST
Gives you an in-depth look at this sensational new computer. Discusses the architecture of the ST, working with GEM, the mouse, operating system, all the various interfaces, the 68000 chip and its instructions, LOGO. $16.95

MACHINE LANGUAGE
Program in the fastest language for your Atari ST. Learn the 68000 assembly language, its numbering system, use of registers, the structure & important details of the instruction set, and use of the internal system routines. 280pp $19.95

LOGO
Take control of your ATARI ST by learning LOGO–the easy-to-use, yet powerful language. Topics covered include structured programming, graphic movement, file handling and more. An excellent book for kids as well as adults. $19.95

PEEKS & POKES
Enhance your programs with the examples found within this book. Explores using the different languages BASIC, C, LOGO and machine language, using various interfaces, memory usage, reading and saving from and to disk, more. $16.95

BEGINNER'S GUIDE
Finally a book for those new to the ST wanting to understanding ST basics. Thoroughly understand your ST and its many devices. Learn the fundamentals of BASIC, LOGO and more. Complete with index, glossary and illustrations. +200pp $14.95

BASIC TO C
If you are already familiar with BASIC, learning C will be all that much easier. Shows the transition from a BASIC program, translated step by step, to the final C program. For all users interested in taking the next step. $19.95

Abacus Software

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

Optional diskettes are available for all book titles at **$14.95**

Call **now** for the name of your nearest dealer. Or order directly from ABACUS with your MasterCard, VISA, or Amex card. Add $4.00 per order for postage and handling. Foreign add $10.00 per book. **Other software and books coming soon.** Call or write for your **free** catalog. Dealer inquiries welcome–over 1400 dealers nationwide.

CIRCLE #124 ON READER SERVICE CARD

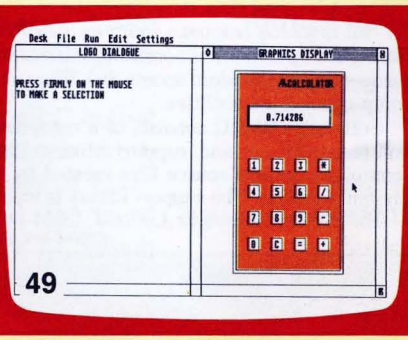
ST·LOG
THE ATARI ST OPERATOR'S MAGAZINE

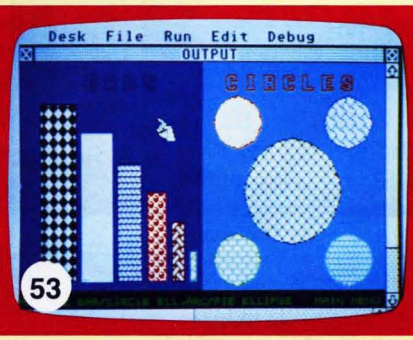
ISSUE 3
JUNE 1986

FEATURES

REVIEWS

COLUMNS

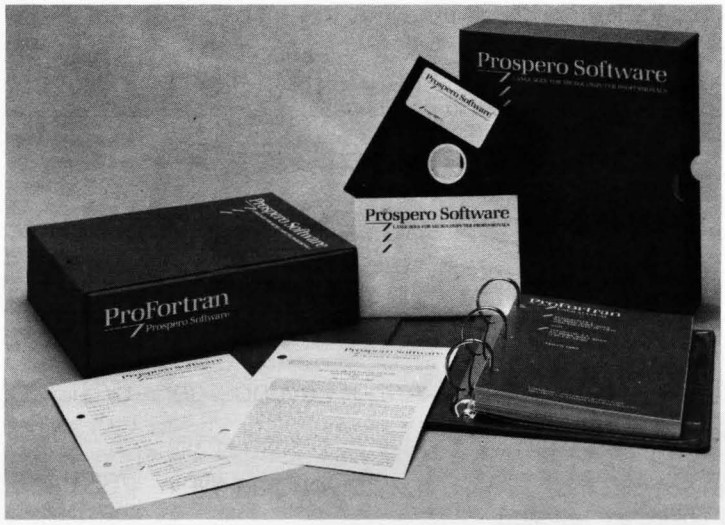




ST NEWS!

PRO FORTRAN-77

Prospero has announced **Pro Fortran-77**, a compiler that enables developers to recompile existing mini and mainframe software, to run on the ST line.



The original Fortran-77 is a very popular high-level programming language, often chosen by engineers, academics and scientists for their demanding work.

Pro Fortran-77 retails for $149.99. For more information, write: Prospero Software, 190 Castelnau, London, England SW13 9DH, or call 01-741 8531.

CIRCLE #168 ON READER SERVICE CARD

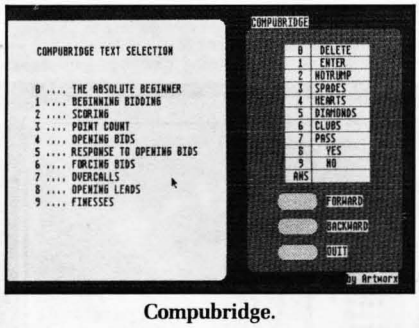
ARTWORX RELEASES FOR THE ST

Compubridge and **Bridge 4.0**, both popular programs for the 8-bit Ataris, are now available for the 16-bit machines. **Compubridge** is a basic bridge tutorial, compiled from the *Five Card Major Bridge Teacher's Manual* by Shirley Silverman. The program has ten chapters of text and has eight sections of quizzes. While you play "test" games, the computer analyzes your moves and suggests alternates.

Bridge 4.0 allows you and your computer partner to bid against two computer opponents, then play out the hand. This comprehensive bridge-playing game is similar to the newspaper column, with replay of selected hands, rotation of players' cards and a built-in referee.

Each bridge program retails for $29.95. Available from Artworx Software Company, Inc., 150 North Main Street, Fairport, NY 14450.

CIRCLE #169 ON READER SERVICE CARD



Compubridge.

SOFTWORKS BASIC

This new BASIC for STs is chock full of features. Advanced data structures, the ability to call machine language routines, superior string manipulation, full error detection, and sequential and random access disk filing are only *some* of its abilities.

Softworks BASIC consists of a compiler, runtime package and support library. The compiler converts source files created by the editor into tokenized files which the runtime system executes. The support library is made up of example programs and Atari interface files.

For $79.00, Softworks Limited, 2944 N. Broadway, Chicago, IL 60657 — (312) 975-4030.

CIRCLE #172 ON READER SERVICE CARD

OTHER NEWS

In **Championship Golf: The Great Courses of the World**, Gamestar has released **Volume 1: Pebble Beach**.

Actual topographical maps were used, to authentically recreate the course itself. All eighteen holes are placed, along with traps, roughs and trees. Even an ocean breeze is added into the play, in which up to four can compete. A driving range and putting green are available for practice and/or warmup.

For information, contact: Gamestar/Activision, Inc., 2350 Bayshore Frontage Road, Mountain View, CA 94043 — (415) 960-0410.

CIRCLE #170 ON READER SERVICE CARD

■ ■ ■

Introduction to Sound and Graphics on the Atari ST, by Tim Knight, has been published by COMPUTE! Books.

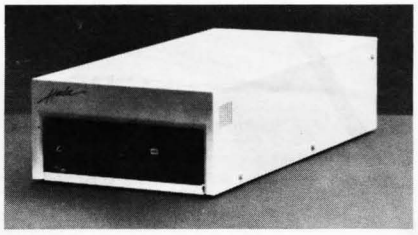
Included in this 195-page volume are sections on creating graphics using LOGO and FORTH, generating sound effects, drawing in BASIC, setting up your ST and using GEM.

The retail price is $14.95, from COMPUTE! Books, 324 West Wendover Avenue, Suite 200, Greensboro, NC 27408.

CIRCLE #171 ON READER SERVICE CARD

10-MEGABYTE DRIVE

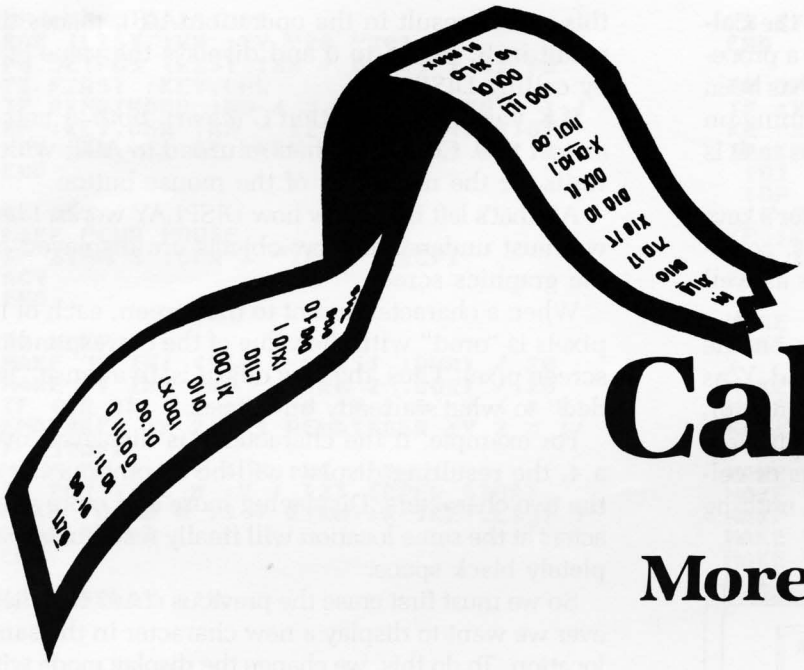
Haba is now shipping their external **10-megabyte hard disk drive**, which stores the equivalent of more than a dozen double-sided 800K disks.



Access time with the hard drive is much faster, using a data transfer rate of 5 megabits/second.

Complete with ST interface cable, the drive lists for $699.95. From Haba/Arrays, Inc., 6711 Valjean Avenue, Van Nuys, CA 91406 — (818) 994-1899.

CIRCLE #130 ON READER SERVICE CARD



ST Calculator

More on using LOGO

by Alain Birtz

The ST is one of the most powerful microcomputers currently on the market. Its low price, together with the GEM mouse-driven user interface, its exceptional graphics capabilities and its 68000 microprocessor, make it one of the most attractive machines available.

Its only defect (rapidly being remedied) is its lack of programming languages. The new owner of an ST, as a rule, has to settle for two languages: ST BASIC and DR LOGO. Although the LOGO itself is of high quality, the manual leaves something to be desired. It consists of a series of technical definitions of language statements, totally devoid of programming examples. This is a great obstacle for a LOGO novice. Program examples are absolutely necessary, and magazines like **ANALOG Computing** will have to provide the forum for any examples.

ST Calculator is written in LOGO for the ST. It allows numerical calculations similar to those of an ordinary hand-held calculator, which are carried out by clicking the mouse on the **Calculator**'s keys. It consists of six procedures and is activated by calling the procedure *C*.

The first thing to happen when you call C is that the text window is cleared and a message is printed. These are done by the first four lines of C.

Next, the graphics window is cleared, the word *Calculator* is printed (using the procedure *TT*, which allows one to print text in the graphics window), and the rectangles representing the **Calculator** and its window (where its numbers are displayed) are drawn (Lines 6-11). The variable *KEY.LIST* is defined in Line 12; it contains the symbols for each of the **Calculator**'s keys.

Next, the procedure *KEY* is called. KEY draws the **Calculator**'s keys. It takes four parameters: *NO*, a counter; *KEY.CHR*, which receives KEY.LIST as a value; and *XX* and *YY*, the coordinates of the rectangles forming the keys themselves.

KEY is a recursive procedure: it will call itself sixteen times, each time drawing one key and decrementing the counter *NO*. When the counter equals *0*, KEY terminates, and execution returns to Line 14 of C. The cursor is then positioned at the **Calculator**'s view window, and certain variables are initialized. C terminates by calling ACT.

ACT is an infinite loop. It calls itself after every click of the mouse's button. It determines the mouse state and transfers this information to the variable *CUR*. If the mouse's button is depressed (i.e., clicked), the third item in CUR has the value *TRUE*, and the procedure *KEEP* is called; if not, nothing happens. Then ACT is called again.

The MOUSE primitive is slow; the time it takes to respond to a click is great. Therefore, it's advisable to set the control panel response rate for the mouse button at maximum *(4)*.

ST Calculator *continued*

At this point in the program, we've drawn the **Calculator**, its window and its keys, and we have a procedure which tells us when the mouse button has been clicked. Now we need some way of determining on which key the mouse has been clicked. This task is accomplished by the procedure *KEEP*.

Imagine the area occupied by the **Calculator**'s keys as a grid of seven rows and seven columns, corresponding to the four rows of the actual keys as well as the three rows between the keys.

From the position of the mouse's arrow on the screen, KEEP calculates the variables *X* and *Y* as values along the row *(2-8)* and along the column *(0-6)*, respectively, where the arrow is located. If either X or Y is odd, then the arrow is between rows or columns (it cannot be pointing to a key), and nothing happens (line four of KEEP).

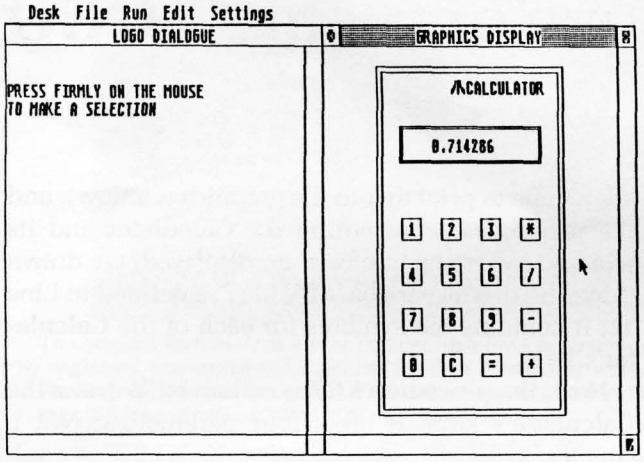


Figure 1.

Otherwise, KEEP calculates the value of the variable K in such a way that K gives the number of the item in KEY.LIST corresponding to the key selected. For example, if the / key (division operator) is selected, X and Y have the values *8* and *2*. K then gets the value *8*, and the eighth element of KEY.LIST is *(voila!)* /.

Two variables are used to perform calculations. B contains the "current" value, the number currently displayed in the **Calculator**'s window. A contains the second value, held in memory.

Initially A and B are set to *0*. If a numerical key *(0-9)* is clicked, KEEP modifies B to reflect the new value of the current number. It displays this updated number (via the procedure *DISPLAY*) in the **Calculator**'s window. If the key is not numeric, then KEY calls the procedure *OPER*.

OPER takes the values of A and B, carries out the operation determined by K (in the example above, this would result in the operation *A/B*), places the result in B, sets A to *0* and displays the value of B by calling DISPLAY.

If K yields the operation *C* (clear), both A and B are set to *0*. Control is then returned to ACT, which waits for the next click of the mouse button.

All that's left is to show how DISPLAY works. First, we must understand how objects are displayed on the graphics screen.

When a character is sent to the screen, each of its pixels is "ored" with the value of the corresponding screen pixel. Thus, the new object is, in a sense, "added" to what's already on-screen.

For example, if the character *3* is displayed over a *4*, the resulting display will be a combination of the two characters. Displaying more and more characters at the same location will finally result in a completely black space.

So we must first erase the previous character whenever we want to display a new character in the same location. To do this, we change the display mode with the primitive *PX* (PENREVERSE) and "reprint" the old character, effectively erasing it. After this, we can display the new character.

The number to be displayed is contained in TX, and the number previously displayed is in DP. DISPLAY erases DP with the sequence *PX TT :DP*, then assigns the value of TX to DP and, finally, displays DP.

Attention: **In this listing, the exclamation points at the end of program lines shouldn't be typed in. They are there to indicate that the statement wraps around to the next line.**

Listing 1.
LOGO listing.

```
TO C
CT PRINT [] PRINT []
PRINT [PRESS FIRMLY ON THE MOUSE]
PRINT [TO MAKE A SELECTION]
CS HT HOME ; BY ALAIN BIRTZ
PU SETPOS [-20 130] PD
TT (WORD CHAR 14 CHAR 15 "CALCULATOR!
)
BOX [-90 -150 180 300]
BOX [-95 -155 190 310]
BOX [-70 70 140 30]
BOX [-72 68 144 34]
MAKE "KEY.LIST [1 2 3 * 4 5 6 / 7 8 !
9 - 0 C = +]
KEY 16 :KEY.LIST -70 0
PU SETPOS [-40 78] PD
MAKE "DP []
MAKE "A 0 MAKE "B 0
ACT
END

TO KEY :NO :KEY.CHR :XX :YY
```

```
IF :NO = 0 [STOP]
BOX (LIST :XX :YY "20 "20)
PU SETPOS (LIST :XX + 6 :YY + 4) PD
TT FIRST :KEY.CHR
IF REMAINDER :NO 4 = 1 [KEY :NO - 1 !
BF :KEY.CHR :XX - 120 :YY - 40 STOP]
KEY :NO - 1 BF :KEY.CHR :XX + 40 :YY
END

TO ACT
MAKE "CUR MOUSE
IF ITEM 3 :CUR = "TRUE [KEEP]
ACT
END

TO KEEP
MAKE "X INT (110 + FIRST :CUR) / 20
MAKE "Y INT (20 - ITEM 2 :CUR) / 20
IF (OR :X < 2 :X > 8 :Y > 6 :Y < 0 R!
EMAINDER :X 2 = 1 REMAINDER :Y 2 = 1!
) [STOP]
MAKE "K (:X + 4 * :Y) / 2
IF (OR :X = 8 :K > 13) [OPER STOP]
MAKE "B 10 * :B + ITEM :K :KEY.LIST !
DISPLAY :B
END

TO DISPLAY :TX
PX TT :DP MAKE "DP :TX TT :DP
END

TO OPER
IF :X = 8 [MAKE "A :B MAKE "OP :K MA!
KE "B 0 DISPLAY []]
IF :K = 15 [IF :OP = 4 [MAKE "B :A *!
 :B] IF :OP = 8 [MAKE "B :A / :B] IF!
 :OP = 12 [MAKE "B :A - :B] IF :OP =!
 16 [MAKE "B :A + :B] DISPLAY :B]
IF :K = 14 [MAKE "A 0 MAKE "B 0 DISP!
LAY []]
END

MAKE "GFILL "FALSE
MAKE "CUR [-22 -111 FALSE FALSE TRUE!
]
MAKE "Y 6
MAKE "X 4
MAKE "KEY.LIST [1 2 3 * 4 5 6 / 7 8 !
9 - 0 C = +]
MAKE "K 14
MAKE "C "# # # # # # # #
MAKE "B 0
MAKE "A 0
MAKE "OP 4
MAKE "DP []
```

VIP Professional™

Finally – A Business Program that Brings Lotus 1-2-3® Functionality to Your *Atari ST*™!

VIP Professional is a state-of-the-art, integrated spreadsheet program which brings together a spreadsheet, a database and graphing capabilities. Professional was modeled after the powerful and best-selling Lotus 1-2-3® program which dominates the business world

Worksheet Magic

Nothing is left out of the workings of the worksheet. Ranges of cells can be named for convenience; column widths are variable; the screen can be split into two windows; titles can be frozen; contents of cells may be copied or moved; the worksheet may be altered as a whole or only partially; the list goes on and on. Perhaps most important, Professional can use and save Lotus 1-2-3 files for transfer between computers.

The worksheet includes over 45 special functions to simplify commonly used formulas, including powerful financial functions for the internal rate of return, present value, and future value. Of course Professional also has all mathematical, trigonometric, table, conditional and logical functions.

Database Power

The built-in database can handle up to 8192 records, with a possibility of up to 256 fields. The records can be searched, sorted and analyzed to find your best salesperson or your rarest stamp. Sorts can be done using multiple criteria, in ascending and descending order. And database functions can be used to do up to seven different kinds of statistical analyses of your database.

Graphs

The graphing capabilities of Professional are astounding. Not only are there six completely different types of graphs available, there are tens of ways to manipulate the data, titles, grids, colors, legends, keys, and scaling of the size of the graph.

Macros

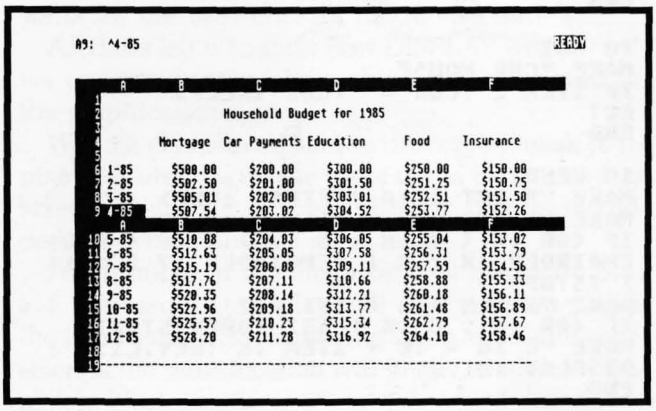
Professional also includes sophisticated macro programming commands. With several special macro commands, the user can actually *program* Professional to be dedicated to a specific task such as accounting.

Just Minutes to Learn

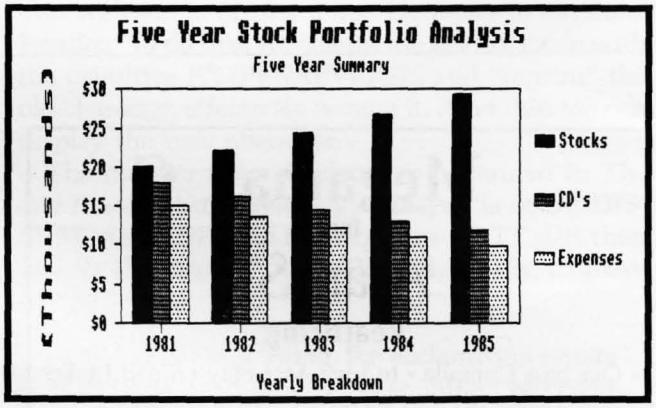
Professional is as easy to use as it is powerful. It comes with a user-sensitive tutorial for the newcomer. And help is built right into the program. With the handy tutorial, you will be able to create professional worksheets in just minutes.

Introducing Professional LITE™

For those of you who do not need the full power of Professional, we offer Professional LITE™. Though without the macros and the database features, and having a smaller sheet size (256 columns by 2048 rows, LITE still packs a powerful punch for only $99.95!



Integrated Spreadsheet Power



Easy-to-Use Graphs

The Power of Professional
Only $179.95
Or the Power of LITE
Only $99.95

If your dealer is out of stock, order direct. Send your check or money order to the address below, together with $3 for shipping and handling. In California add 6% sales tax. COD's and purchase orders not accepted. Personal checks will be held for three weeks to clear. All prices are subject to change without notice.

VIP TECHNOLOGIES

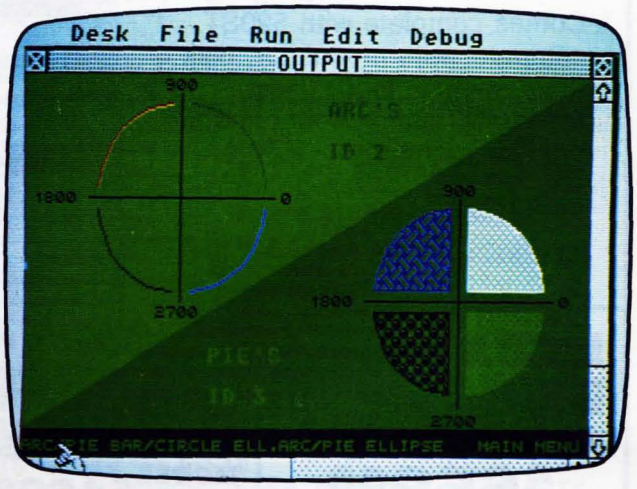
132 Aero Camino
Santa Barbara
California 93117

(805) 968-9567

SYSTEM REQUIREMENTS: Amiga with 512K; One disk drive; Monochrome or color monitor; Works with printers supported by the Workbench.

VDI Sampler

Functions from ST BASIC



by James Luczak

ST BASIC has a command to allow you to perform graphic operations—that used to take a lot of involved programming—quickly and easily. And it's *fast*. We're talking about the VDISYS() command.

The VDISYS() command gives you access to GEM's Virtual Display Interface (VDI) functions. Among other things, GEM's VDI has a full set of graphic functions to perform all sorts of graphic operations. Circles, ellipses, pie slices, rectangles, polygons and special effects text are only *a few* of the many operations you can utilize with the VDISYS() command.

The VDI Sampler program.

The **VDI Sampler** has three purposes. First, it contains the BASIC code required to call many of the VDI functions (Lines 3730 through 6340). Second, it demonstrates how VDI functions can be used in a program. Third, it displays many of the patterns, styles and special effects that can be put to use with VDI functions.

All graphics operations used in this program are performed with the help of the VDISYS() command.

Using the VDISYS() command.

VDISYS() is very easy to use; Figures 1, 2 and 3 contain all the necessary information to do so. The BASIC code required to employ the VDI functions is in the **VDI Sampler** program, Lines 3730 through 6340.

I found it easiest to set up the BASIC code for each function as a routine which can be called with the GOSUB command. Whenever you want the function, all you have to do is set whatever variables the VDI routine currently has to the desired value, then use a GOSUB to do the operation.

VDI COLOR INDEX

Index Number	Color	Index Number	Color
0	White	8	Low White
1	Black	9	Grey
2	Red	10	Light Red
3	Green	11	Light Green
4	Blue	12	Light Blue
5	Cyan	13	Light Cyan
6	Yellow	14	Light Yellow
7	Magenta	15	Light Magenta

Figure 1.

Figure 1, above, contains the COLOR INDEX values used by all the VDI functions. Figure 2, below, gives the DEFAULT values to which various functions are set.

VDI DEFAULT VALUES

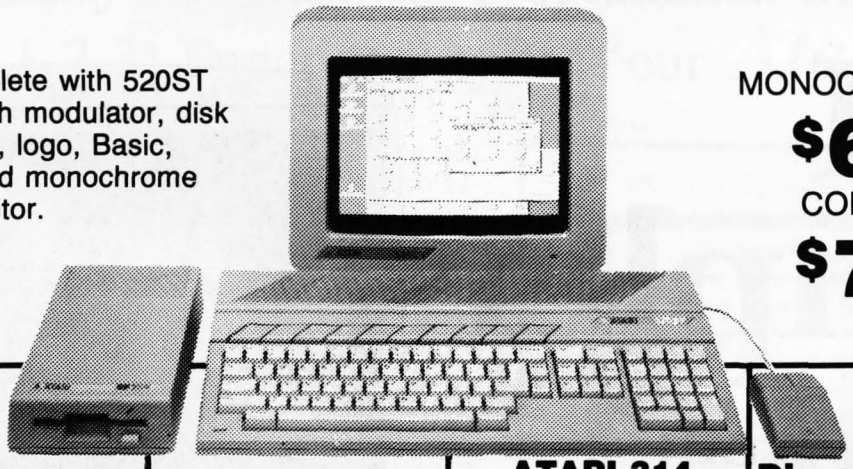
Attribute	Default Value
Character Height	Low Resolution 6 Medium Resolution 8
Character Baseline	0 Degree Rotation
Text Style	0 — Normal Intensity
Polymarker Height	1 — Smallest Size
Polyline Endstyle	0 — Squared
Writing Mode	1 — Replace
Perimeter Visibility	1 — Visible

Figure 2.

ATARI SPECTACULAR

ATARI 520ST SYSTEM PACKAGE

Comes complete with 520ST computer with modulator, disk drive, mouse, logo, Basic, 1st Word, and monochrome or color monitor.



MONOCHROME SYSTEM
$649.00
COLOR SYSTEM
$799.00

HABA DISK
10 Meg HARD DRIVE
$669.00

SUPRA MODEM
MODEL 1200ST $159.00

ATARI 314
1 Meg Double Sided DISK DRIVE
$219.00

ANCHOR 520
Direct Connect MODEM
300/1200 Baud
$129.00

— PRINTERS —

CITIZEN
MSP-10 (80 col.) $279.00
MSP-15 (132 col.) $389.00
MSP-20 (80 col.) $349.00
MSP-25 (132 col.) $509.00

C.ITOH
Prowriter 7500 ..... $179.00
Prowriter 1550P $349.00
Starwriter 10-30 ..... $399.00

EPSON
Homewriter 10, LX80 ..... CALL
FX85, FX286, RX100 ..... CALL
SQ2000, HI80, HS80, AP80 ..... CALL
LQ800, LQ1000 ..... CALL

JUKI
6000 Letter Quality ..... CALL
6100 Letter Quality ..... CALL
6200 Letter Quality ..... CALL
6300 Letter Quality ..... CALL
5510 Dot Matrix ..... CALL

LEGEND
808 Dot Matrix 100 cps $179.00
1080 Dot Matrix 100 cps $259.00
1380 Dot Matrix 130 cps $289.00
1385 Dot matrix 165 cps $339.00

NEC
3000 Series $799.00
8000 Series $1099.00
ELF 360 ..... $449.00
Pinwriter 560 ..... $999.00

OKIDATA
182, 183, 192, 193, 2410, 84 ..... CALL
Okimate 10 (Specify C64/Atari) $189.00
Okimate 20 (IBM) ..... CALL

PANASONIC
KX1091 ..... $259.00
KX1092 ..... $389.00
KX1093 ..... $479.00

STAR
SB/SD/SG/SR Series ..... CALL
Powertype Letter Quality ..... CALL

TOSHIBA
1340 (80 column) $399.00
P341 (132 column) $799.00
P351 (132 column) $1069.00

— SOFTWARE —

BATTERIES INCLUDED
D.E.G.A.S $27.99

INFOCOM
Cutthroats $29.99
Deadline $34.99
Enchanter $29.99
Hitchhiker's Guide $29.99
Infidel $34.99
Planetfall $29.99
Sea Stalker $29.99
Sorcerer $34.99
Starcross $34.99
Suspect $29.99
Suspended $34.99
Wishbringer $29.99
Witness $29.99
Zork I $29.99
Zork II $29.99
Zork III $29.99

MINDSCAPE
Deja Vu $37.99

O.S.S.
Personal Pascal $49.99

vip PROFESSIONAL

A jazzed-up 1-2-3 for your Atari 520ST!

- Spreadsheet
- Database
- Graphs

$129.00

HABA/ARRAYS
Hippo-C $44.99
Business Letters $29.99
Write Your Own Will $29.99
Haba Writer $44.99
Habadex Phonebook $29.99
Habamerge $12.99
Checkminder $39.99
Mail Room Manager $39.99

MUSE SOFTWARE
Final Word $99.99
Hex $27.99
PC Intercom $79.99

ACTIVISION
Hacker $29.99
Borrowed Time $32.99

MIRAGE
Express $34.99

SIERRA-ON-LINE
Ultima II $39.99
King's Quest $37.99

PENQUIN SOFTWARE
Crimson Crown $29.99

BLANK DISKETTES

ELEPHANT
(10) 3½" SS **$24.99**

MAXELL
(10) 3½" SS **$18.99**
(10) 3½" DS **$29.99**
(5) 3½" SS w/Flip'n File. **$9.99**

AMARAY
Disk Tub 3½" **$9.99**

CURTIS Surge Suppressors
CUSP2, Emerald **$39.99**
CUSPF2, Ruby **$59.99**
CUSP1, Diamond **$29.99**
CUSP3, Safety Strip **$19.99**

COMPUTER MAIL ORDER

CALL TOLL-FREE 1-800-233-8950

477 East Third Street, **Dept. B706,** Williamsport, PA 17701

VISA · MasterCard · American Express · Diners Club

POLICY: Add 3% (Minimum $7.00) shipping and handling. Larger shipments may require additional charges. Personal and company checks require 3 weeks to clear. For faster delivery use your credit card or send cashier's check or bank money order. Pennsylvania residents add 6% sales tax. All prices are subject to change and all items are subject to availability. Defective software will be replaced with the same item only. Hardware will be repaired or replaced at our discretion within the the terms and limits of the manufacturer's warranty. We *cannot guarantee compatibility.* All sales are final and returned shipments are subject to a restocking fee.

— CANADIAN ORDERS —

1-800-268-3974 Ontario/Quebec

1-800-268-4559 Other Provinces

All prices shown are for U.S.A. orders. Call the Canadian Office for Can. prices.

1-416-828-0866 In Toronto

Telex:06-218960
2505 Dunwin Drive,
Mississauga, Ontario
Canada L5L1T1

// VDI Functions *continued*

Our third figure, on the following page, provides a description of most of the VDI functions used by the program. Some VDI functions are self explanatory; these are not included in the description.

The attributes referred to in Figure 3 are VDI functions that can be used to modify the appearance of another VDI function. For example, if you're drawing a circle and want it to appear as a filled disk, you could use the "fill interior style" function with a value of *1* (solid) before drawing the circle. Now, every time you draw a circle it will be filled with a solid color.

Using the VDI Sampler.

The **Sampler** is written for a low-resolution color system. If you don't have TOS in ROM yet, remove all desk accessories and turn buffered graphics off before loading the program.

The **Sampler** opens with a title page that will remain on-screen for approximately 5 seconds. (If you want to keep the title page on for a shorter or longer period, change the value in Line 6370 of the program.)

After the title, the main menu screen will appear. At the bottom of this is a menu bar. Use the mouse to click on any of the menu options. You'll hear a "beep" when you click on a valid option. Each option will display its introduction page, with a menu bar. Click the mouse on one of the options, and that option will be displayed.

The values which appear on some of the displays correspond to those in Figure 3. To return to the main menu, click on the Main Menu option located on the right of the menu bar. To exit the program, select the quit option on the main menu—or click the right-hand mouse button.

I find this a handy program to preview styles, patterns, special effects, and so forth—before using them in a program. The data statements at the end of the program contain text (in ASCII form), X- and Y-coordinates, and some color information used in **VDI Sampler.**

James Luczak bought his first Atari in 1980 and has, since 1979, written programs in BASIC, C, LOGO, FORTH and Action!, plus 6502 assembly. He enjoys writing dedicated database programs.

Listing 1.
BASIC listing.

```
100 'A  V.D.I.  SAMPLER FROM ST BASIC
110 '******** by JIM LUCZAK ********
120 a#=gb:gintin=peek(a#+8):gintout=pe
ek(a#+12)
130 dim coord(25),char(50),ml(26)
140 a$(1)="P  O  L  Y  L  I  N  E  S"
150 a$(2)="P  O  L  Y  M  A  R  K  E
R  S"
160 a$(3)="T  E  X  T"
170 a$(4)="F  I  L  L     P  A  T  T
E  R  N  S"
180 a$(5)="P  R  I  M  I  T  I  V  E
S"
190 fx=1:fy=22:fx1=304:fy1=188:dth=6
200 '------- DO TITLE PAGE -------
210 fullw 2:clearw 2
230 rt=3:wm=1:fis=1:fci=1:cx=fx:cy=fy:
cx1=fx1:cy1=fy1
240 gosub DOSHAPE:wm=2:gosub WMODE:plc
=2:gosub PCOLOR
250 id=6:sa=2700:ea=900:cx=1:cy=105:ra
dx=304:rady=83
260 gosub ELLARCPIE
270 while rady>40
280 rady=rady-2:poke ptsin+6,rady:vdis
ys(1)
290 wend
300 plc=10:gosub PCOLOR
310 sa=900:ea=2700:cx=304:rady=83
320 gosub ELLARCPIE
330 while rady>40
340 rady=rady-2:poke ptsin+6,rady:vdis
ys(1)
350 wend
370 rt=5:wm=1:fis=1:fsi=0:fci=1:cx=152
:cy=47:radx=100:rady=25
380 gosub DOSHAPE
390 rt=5:wm=2:fis=2:fsi=3:fci=3:gosub
DOSHAPE
400 rt=5:wm=1:fis=1:fci=1:cy=163:radx=
125:gosub DOSHAPE
410 rt=5:wm=2:fis=2:fsi=19:fci=4:gosub
 DOSHAPE
430 rt=1:wm=1:fis=1:fci=3:cx=70:cy=163
:rad=20
440 gosub DOSHAPE
450 rt=1:wm=2:fis=2:fsi=16:fci=2:gosub
 DOSHAPE
460 rt=1:wm=1:fis=1:fci=2:cx=235:gosub
 DOSHAPE
470 rt=1:wm=2:fis=2:fsi=20:fci=6:gosub
 DOSHAPE
480 wm=2:gosub WMODE
500 tc=15:th=24:te=4:cx=110:cy=54:gosu
b DOTEXT
510 tc=5:th=28:te=16:cx=45:cy=115:gosu
b DOTEXT
520 tc=8:th=dth:te=4:cx=130:cy=148:gos
ub DOTEXT
530 tc=6:th=15:cx=97:cy=170:gosub DOTE
XT
540 tc=11:th=dth:cx=110:cy=185:gosub D
OTEXT
550 gosub TIMER:menu=0:gosub MENUES
560 '--- MAIN PROGRAM LOOP ---
570 poke gintin,3:gemsys(78)
580 poke gintin,257:gemsys(78)
590 poke systab+24,1
600 while mkey<>2
610 gemsys(79)
620 mx=peek(gintout+2):my=peek(gintout
+4):mkey=peek(gintout+6)
630 if mkey=1 then gosub CHECKM
640 wend
650 '--- CLEANUP AND END ---
660 poke systab+24,0
670 poke gintin,0:gemsys(78)
680 poke gintin,256:gemsys(78)
690 rt=1:tc=1:th=dth:te=0
700 gosub DOTEXT
710 sound 0,0,0,0,0:clearw 2:end
730 CHECKM:
740 if my<180 or my>189 then return
```

VDI FUNCTION DESCRIPTIONS

POLYMARKER — A polymarker is similar to the PLOT command. One or more polymarkers can be displayed simultaneously in different styles and colors.

Line 3730 Menu — Marker

ATTRIBUTES	ATTRIBUTE LINE #
Polymarker Type	3820
Polymarker Height	3890
Polymarker Color	3970
Writing Mode	5000

NOTE: Each polymarker needs an X- and Y-coordinate. For example, to display two polymarkers, you must give the X- and Y-coordinates for each marker, a total of four coordinates.

POLYMARKER TYPE — Identifies polymarker style.

Line 3820 Menu — Marker Item — Type

MARKER TYPE VALUE	DESCRIPTION	MARKER TYPE VALUE	DESCRIPTION
1	Dot	4	Square
2	Plus Sign	5	Diagonal Cross
3	Asterisk	6	Diamond

POLYLINE — Draws one or more lines simultaneously in different styles, colors, widths and end types.

Line 4040 Menu — Lines

ATTRIBUTES	ATTRIBUTE LINE #
Polyline Type	4130
Polyline Width	4200
Polyline End Style	4280
Polyline Color	4360
Writing Mode	5000

NOTE: Each polyline requires four coordinates: X- and Y-coordinates for the starting point of the line and those for the line's ending point. For example, to draw two polylines, indicate two X,Y pairs (Line 4060) and provide X- and Y-coordinates for the starting and ending points for each line, a total of eight coordinates.

POLYLINE TYPE — Identifies type of polyline.

Line 4130 Menu — Line Item — Type

POLYLINE TYPE VALUE	DESCRIPTION	POLYLINE TYPE VALUE	DESCRIPTION
1	Solid Line	4	Dash-Dot Line
2	Long Dash Line	5	Dash Line
3	Dot Line	6	Dash-Dot-Dot Line

POLYLINE ENDSTYLE — Identifies end style of polyline.

Line 4280 Menu — Lines Item — Endstyle

ENDSTYLE VALUE	DESCRIPTION
0	Squared
1	Arrow
2	Rounded

TEXT — Writes text to any X,Y coordinate on the display screen.

Line 4430 Menu — Text

ATTRIBUTES	ATTRIBUTE LINE #
Text Color	4540
Text Special Effects	4610
Text Height	4680
Text Baseline	4930
Writing Mode	5000

NOTE: Characters are referred to by their ASCII value.

TEXT SPECIAL EFFECTS — Identifies the text style to be used.

Line 4610 Menu — Text Item — Effects

SPECIAL EFFECTS VALUE	DESCRIPTION	SPECIAL EFFECTS VALUE	DESCRIPTION
1	Thickened	8	Underlined
2	Intensity - Light	16	Outlined
4	Skewed	32	Shadowed

NOTE: Any combination of special effects can be used. If, for example, you wanted skewed outlined text, add the values *4* (skewed) and *16* (outline) together, and use the result *(20)* as the special effects value.

WRITING MODE — Identifies how subsequent drawing operations will be performed.

Line 5000

WRITING MODE VALUE	DESCRIPTION	WRITING MODE VALUE	DESCRIPTION
1	Replace	3	XOR
2	Transparent	4	Reverse Transp.

NOTE: The writing mode specifies the operation performed between the current pixel color and the existing pixel color. The action of the writing mode is most easily observed when using the text or fill pattern functions.

ARC or PIE — Draws an arc or pie slice.

Line 5070 Menu — Shapes Item — Arc/Pie

ATTRIBUTES	ATTRIBUTE LINE #	
Polyline Type	4130	ARC
Polyline Width	4200	ARC
Polyline End Style	4280	ARC
Polyline Color	4360	ARC
Fill Interior Style	6080	PIE
Fill Style Index	6150	PIE
Fill Color	6220	PIE
Perimeter Visibility	6290	PIE
Writing Mode	5000	ARC/PIE

NOTE: This function will draw an arc or a pie, depending which primitive ID you specify *(ARC – 2, PIE – 3)*. Angles are referred to in tenths of a degree *(0-3600)*. The function draws in a *counterclockwise* direction. Zero degrees is 90 degrees to the right of vertical, with values increasing in a counterclockwise direction.

BAR — Draws a bar.

Line 5210 Menu — Shapes Item — Bar/Circle

ATTRIBUTES	ATTRIBUTE LINE #
Fill Interior Style	6080
Fill Style Index	6150
Fill Color	6220
Perimeter Visibility	6290
Writing Mode	5000

CIRCLE — Draws a circle.

Line 5320 Menu — Shapes Item — Bar/Circle

ATTRIBUTES	ATTR. LINE #	ATTRIBUTES	ATTR. LINE #
Fill Interior Style	6080	Perimeter Visibility	6290
Fill Style Index	6150	Writing Mode	5000
Fill Color	6220		

ELLIPTICAL ARC or PIE — Draws an elliptical arc or pie slice.

Line 5450 Menu — Shapes Item — Ell.Arc/Pie

ATTRIBUTES	ATTR. LINE #		ATTRIBUTES	ATTR. LINE #	
Polyline Type	4130	ARC	Fill Style Index	6150	PIE
Polyline Width	4200	ARC	Fill Color	6220	PIE
Polyline End Style	4280	ARC	Perimeter Visibility	6290	PIE
Polyline Color	4360	ARC	Writing Mode	5000	ARC/PIE
			Fill Interior Style	6080	PIE

NOTE: This function will draw an elliptical arc or pie slice, depending which primitive ID you specify *(Elliptical Arc – 6, Elliptical Pie – 7)*. Angles are referred to in tenths of degrees *(0-3600)*. The function draws in a *counterclockwise* direction. Zero degrees is 90 degrees to the right of vertical, with values increasing in a counterclockwise direction.

ELLIPSE — Draws an ellipse.

Line 5580 Menu — Shapes Item — Ellipse

ATTRIBUTES	ATTR. LINE #	ATTRIBUTES	ATTR. LINE #
Fill Interior Style	6080	Perimeter Visibility	6290
Fill Style Index	6150	Writing Mode	5000
Fill Color	6220		

ROUNDED RECTANGLE and FILLED ROUNDED RECTANGLE — Draws a rounded or filled rounded rectangle.

Line 5690

ATTRIBUTES	ATTRIBUTE LINE #	
Polyline Type	4130	Rounded Rectangle
Polyline Width	4200	Rounded Rectangle
Polyline Color	4360	Rounded Rectangle
Fill Interior Style	6080	Filled Rounded Rectangle
Fill Style Index	6150	Filled Rounded Rectangle
Fill Color	6220	Filled Rounded Rectangle
Perimeter Visibility	6290	Filled Rounded Rectangle
Writing Mode	5000	Rounded/Filled Rounded Rect.

FILLED AREA — Fills a complex polygon with the specified color or pattern.

Line 5800

ATTRIBUTES	ATTR. LINE #	ATTRIBUTES	ATTR. LINE #
Fill Interior Style	6080	Perimeter Visibility	6290
Fill Style Index	6150	Writing Mode	5000
Fill Color	6220		

EXAMPLE: To use this function to fill a triangle, enter *3* for the number of lines (Line 5820). Give the X- and Y-coordinates for the starting point, second point and ending point of the triangle. VDI will automatically connect the ending point to the starting point, to form a closed polygon. The function will then fill the triangle. VDI will not display a form with only one endpoint.

CONTOUR FILL — Fills an area until it finds the end of the screen or the color specified.

Line 5890

ATTRIBUTES	ATTRIBUTE LINE #
Fill Interior Style	6080
Fill Style Index	6150
Fill Color	6220
Perimeter Visibility	6290
Writing Mode	5000

NOTE: If you specify a *negative* value in Line 5930, VDI will search for any color other than the seed point.

FILL RECTANGLE — Fills the rectangular area specified.

Line 5980

ATTRIBUTES	ATTRIBUTE LINE #
Fill Interior Style	6080
Fill Style Index	6150
Fill Color	6220
Perimeter Visibility	6290
Writing Mode	5000

FILL INTERIOR STYLE — Identifies interior style.

Line 6080

INTERIOR STYLE VALUE	DESCRIPTION	INTERIOR STYLE VALUE	DESCRIPTION
0	Hollow	3	Hatch
1	Solid	4	User defined
2	Pattern		

FILL STYLE INDEX — Identifies style index.

Line 6150

See menu option FILL for example of style indices.

```
750 gosub CHECKMOUSELOC
760 if hc1=-1 then return
770 sound 1,0,10,4,0:wave 1,1,9,256,0
780 if hc1=n1/2 and menu=0 then mkey=2
:return
790 if hc1=n1/2 then menu=0:ft=0:gosub
 MENUE5:return
800 if menu=0 then menu=hc1:gosub MENU
E5:return
810 item=hc1:gosub ITEM5
820 return
840 CHECKMOUSELOC:
850 mc=0:hc=0:hc1=1
860 while mc=0
870 if mx>= ml(hc) and mx<= ml(hc+1) t
hen mc=1
880 if mc=0 then hc1=hc1+1
890 hc=hc+2:if hc)n1 then mc=1
900 wend
910 if hc1> n1/2 then hc1=-1
920 return
940 DOSHAPE:
950 gosub WMODE:gosub FILLSTYLE:gosub
FILLINDEX:gosub FILLCOLOR
960 on rt goto SHAPE1,SHAPE2,SHAPE3,SH
APE4,SHAPE5,SHAPE6
970 SHAPE1:gosub CIRCL:rt=0:return
980 SHAPE2:gosub BAR:rt=0:return
990 SHAPE3:gosub RECTFILL:rt=0:return
1000 SHAPE4:gosub FILLA:rt=0:return
1010 SHAPE5:gosub ELLIPS:rt=0:return
1020 SHAPE6:gosub ARCPIE:rt=0:return
1040 DOTEXT:
1050 gosub TCOLOR:gosub THEIGHT:gosub
TEFFECT
1060 if rt=1 then rt=0:return
1070 read n:for dt=0 to n-1:read char(
dt):next dt
1080 gosub GTEXT:return
1100 INTRO:
1110 restore INTRODATA
1120 wm=2:gosub WMODE
1130 rt=1:tc=4:th=20:te=16:gosub DOTEX
T
1140 gotoxy 9,2:?"W e l c o m e  T o"
1150 rt=1:tc=7:th=28:te=4:gosub DOTEXT
1160 gotoxy 6,7:?"V D I    S a m p l e
 r"
1170 rt=1:tc=9:th=dth:te=0:gosub DOTEX
T
1180 gotoxy 2,10:?"Use"
1190 rt=1:tc=2:te=16:gosub DOTEXT
1200 gotoxy 6,10:?"MOUSE"
1210 rt=1:tc=9:te=0:gosub DOTEXT
1220 gotoxy 12,10:?"to choose from men
u bar"
1230 rt=1:tc=1:te=16:gosub DOTEXT
1240 for x=12 to 16
1250 gotoxy 17,x:?chr$(7);chr$(2)
1260 next x:return
1280 MENUE5:
1290 poke gintin,256:gemsys(78)
1300 gosub FINDMENU
1310 rt=3:wm=1:fis=1:fci=1:cx=fx:cy=fy
:cx1=fx1:cy1=fy1
1320 gosub DOSHAPE:wm=2:gosub WMODE
1330 tc=3:th=dth:te=0:cx=1:cy=186:if m
enu=5 then th=4
1340 gosub DOTEXT
1350 read n:n1=n:for x=0 to n-1
1360 read ml(x):next x
1370 read cx2,cy2,fci,fci1,fsi1,tc
1380 rt=3:wm=1:fis=1:cx=fx:cy=fy:cx1=f
x1:cy1=176
1390 gosub DOSHAPE
1400 if menu=0 then gosub INTRO:goto 1
450
1410 rt=3:wm=2:fis=2:fci=fci1:fsi=fsi1
1420 gosub DOSHAPE
1430 rt=1:th=28:te=0:gosub DOTEXT
1440 gotoxy cx2,cy2:?a$(menu)
1450 poke gintin,257:gemsys(78):return
1470 FINDMENU:
1480 if menu=0 then restore MENU0DATA:
return
1490 on menu goto MENU1,MENU2,MENU3,ME
NU4,MENU5
1500 MENU1:restore MENU1DATA:return
1510 MENU2:restore MENU2DATA:return
1520 MENU3:restore MENU3DATA:return
1530 MENU4:restore MENU4DATA:return
1540 MENU5:restore MENU5DATA:return
1560 ITEM5:
1570 poke gintin,256:gemsys(78)
1580 on menu goto MITEM1,MITEM2,MITEM3
,MITEM4,MITEM5
1590 MITEM1:on item goto ITEM1A,ITEM1B
,ITEM1C
1600 MITEM2:on item goto ITEM2A,ITEM2B
1610 MITEM3:on item goto ITEM3A,ITEM3B
,ITEM3C
1620 MITEM4:on item goto ITEM4A,ITEM4B
1630 MITEM5:on item goto ITEM5A,ITEM5B
,ITEM5C,ITEM5D
1650 CLEARITEM:
1660 cx=fx:cy=fy:cx1=fx1:cy1=176
1670 rt=3:wm=1:fis=1:gosub DOSHAPE
1680 wm=2:gosub WMODE
1690 cx=100:cy=35:gosub DOTEXT
1700 cx=10:cy=50:return
1720 ITEM1A:
1730 restore ITEM1ADATA
1740 tc=6:th=8:te=16:fci=1:gosub CLEAR
ITEM
1750 tc=3:th=dth:te=0:plc=2:gosub PCOL
OR
1760 for x=1 to 6:plt=x
1770 gosub PTYPE:gosub DOTEXT:cy=cy+5
1780 for y=1 to 2
1790 n=2:coord(0)=10:coord(1)=cy:coord
(2)=300:coord(3)=cy
1800 gosub PLINE:cy=cy+5
1810 next y:cy=cy+5:next x
1820 plt=1:gosub PTYPE
1830 poke gintin,257:gemsys(78):return
1850 ITEM1B:
1860 restore ITEM1BDATA
1870 tc=5:th=8:te=16:fci=1:gosub CLEAR
ITEM
1880 tc=2:th=dth:te=0:plc=3:gosub PCOL
OR
1890 for x=1 to 7 step 3
1900 plw=x:gosub PWIDTH:gosub DOTEXT
1910 cy=cy+5:for y=1 to 3
1920 n=2:coord(0)=10:coord(1)=cy:coord
(2)=300:coord(3)=cy
1930 gosub PLINE:cy=cy+10
1940 next y:cy=cy+x
1950 next x:plw=1:gosub PWIDTH
1960 poke gintin,257:gemsys(78):return
1980 ITEM1C:
1990 restore ITEM1CDATA
2000 tc=0:th=8:te=16:fci=1:gosub CLEAR
ITEM
2010 tc=8:th=dth:te=0:plc=6:gosub PCOL
OR
2020 for x=0 to 2
2030 plsb=x:plse=x:gosub PSTYLE:gosub
DOTEXT
2040 cy=cy+7:for y=1 to 3
2050 n=2:coord(0)=10:coord(1)=cy:coord
(2)=300:coord(3)=cy
2060 gosub PLINE:cy=cy+10
2070 next y:cy=cy+7
2080 next x:plsb=0:plse=0:gosub PSTYLE
2090 poke gintin,257:gemsys(78):return
```

// VDI Functions *continued*

```
2110 ITEM2A:
2120 restore ITEM2ADATA
2130 tc=2:th=8:te=4:fci=6:gosub CLEARI
TEM
2140 tc=7:th=dth:te=0:pmc=4:gosub PMCO
LOR
2150 read n:n2=n:for x=0 to (n*2)-1
2160 read coord(x):next x
2170 for x=1 to 6:pmt=x:gosub PMTYPE
2180 cy=cy+10:gosub DOTEXT:cy=cy+6
2190 coord(1)=cy:coord(3)=cy:coord(5)=
cy:coord(7)=cy
2200 n=n2:gosub PMARKER:cy=cy+4:next x
2210 pmt=1:gosub PMTYPE
2220 poke gintin,257:gemsys(78):return
2240 ITEM2B:
2250 restore ITEM2BDATA
2260 tc=6:th=8:te=4:fci=4:gosub CLEARI
TEM
2270 tc=10:th=dth:te=0:pmc=5:gosub PMC
OLOR
2280 read n:for x=0 to n-1
2290 read coord1(x):next x:cy=40
2300 for x=1 to 5:cy=cy+8:gosub DOTEXT
:cy=cy+6
2310 for y=0 to 5:pmt=y+1:gosub PMTYPE
2320 n=1:coord(0)=coord1(y):coord(1)=c
y
2330 pmh=x*10:gosub PMHEIGHT:gosub PMA
RKER
2340 next y:cy=cy+4+(x*3):next x
2350 pmh=1:gosub PMHEIGHT:pmt=1:gosub
PMTYPE
2360 poke gintin,257:gemsys(78):return
2380 ITEM3A:
2390 restore ITEM3ADATA
2400 tc=2:th=8:te=4:fci=8:gosub CLEARI
TEM
2410 tc=1:te=0
2420 for x=4 to 12 step 2:cy=cy+5
2430 th=dth:cx=10:gosub DOTEXT
2440 char(0)=86:char(1)=68:char(2)=73:
char(3)=32:char(4)=83:char(5)=65
2450 char(6)=77:char(7)=80:char(8)=76:
char(9)=69:char(10)=82
2460 th=x:gosub THEIGHT:cx=120:n=11:go
sub GTEXT
2470 cy=cy+(x*3):next x:th=dth:gosub T
HEIGHT
2480 poke gintin,257:gemsys(78):return
2500 ITEM3B:
2510 restore ITEM3BDATA
2520 tc=4:th=8:te=4:fci=13:gosub CLEAR
ITEM
2530 rt=1:tc=7:th=dth:te=0:gosub DOTEX
T
2540 for x=1 to 4:read tbl,cx,cy,n
2550 for x1=0 to n-1:read char(x1):nex
t x1:gosub TBASE
2560 for y=1 to 3:gosub GTEXT
2570 if x=1 then cy=cy+10
2580 if x=2 then cx=cx+10
2590 if x=3 then cy=cy-10
2600 if x=4 then cx=cx-10
2610 next y:next x:tbl=0:gosub TBASE
2620 poke gintin,257:gemsys(78):return
2640 ITEM3C:
2650 restore ITEM3CDATA
2660 tc=6:th=8:te=4:fci=10:gosub CLEAR
ITEM
2670 rt=1:tc=4:th=8:te=0:gosub DOTEXT
2680 for x=0 to 6:read te,cx
2690 gosub DOTEXT:cy=cy+20
2700 next x:te=0:gosub TEFFECT
2710 poke gintin,257:gemsys(78):return
2730 ITEM4A:
2740 fc1=6:fci1=1:gosub SUBREFRESH
```

```
2750 if ft=0 then ft=1:fis1=3:fsi1=12
2760 fsi1=fsi1+1
2770 if fsi1>12 and fis1=3 then fis1=2
:fsi1=1:tl=9:b$="PATTERN":c$="24"
2780 if fsi1>24 then fsi1=1:fis1=3:tl=
10:b$="HATCH":c$="12"
2790 goto SUBREDRAW
2810 ITEM4B:
2820 fci=6:fci1=1:gosub SUBREFRESH
2830 if ft=0 then ft=1:fis1=2:fsi1=1
2840 fsi1=fsi1-1
2850 if fsi1<1 and fis1=3 then fis1=2:
fsi1=24:tl=9:b$="PATTERN":c$="24"
2860 if fsi1<1 and fis1=2 then fis1=3:
fsi1=12:tl=10:b$="HATCH":c$="12"
2870 goto SUBREDRAW
2890 SUBREFRESH:
2900 cx=fx:cy=fy:cx1=152:cy1=176
2910 rt=3:wm=1:fis=1:gosub DOSHAPE
2920 cx=153:cx1=304
2930 rt=3:fci=fci1:gosub DOSHAPE
2940 return
2960 SUBREDRAW:
2970 cx=fx:cx1=152:rt=3:wm=2:fsi=fsi1:
fis=fis1:fci=2:gosub DOSHAPE
2980 cx=153:cx1=304:rt=3:fsi=fsi1:fis=
fis1:fci=5:gosub DOSHAPE
2990 wm=1:gosub WMODE:rt=1:tc=1:th=dth
:te=0:gosub DOTEXT
3000 gotoxy tl,3:?b$;fsi" of "c$
3010 poke gintin,257:gemsys(78):return
3030 ITEM5A:
3040 restore ITEM5ADATA
3050 fis=1:for y=1 to 2:read fci,n
3060 for x1=0 to (n*2)-1:read coord(x1
):next x1
3070 rt=4:gosub DOSHAPE:next y
3080 fis=2:rad=40:sa=0:ea=900
3090 for x=1 to 4:read plc,fci,fsi,cx,
cy,cx1,cy1
3100 gosub PCOLOR:rt=6:id=2:gosub DOSH
APE
3110 cx=cx1:cy=cy1
3120 rt=6:id=3:gosub DOSHAPE
3130 sa=sa+900:ea=ea+900:next x
3140 plc=1:gosub PCOLOR
3150 n=2:for x=1 to 4:for y=0 to 3
3160 read coord(y):next y:gosub PLINE:
next x
3170 rt=1:tc=1:th=4:te=0:gosub DOTEXT
3180 for x=1 to 8:read cx,cy:gosub DOT
EXT:next x
3190 wm=2:gosub WMODE:rt=1:tc=7:th=dth
:te=0:gosub DOTEXT
3200 gotoxy 18,1:?"ARC'S":gotoxy 18,3:
?"ID 2"
3210 rt=1:tc=3:gosub DOTEXT
3220 gotoxy 11,13:?"PIE'S":gotoxy 11,1
5:?"ID 3"
3230 poke gintin,257:gemsys(78):return
3250 ITEM5B:
3260 restore ITEM5BDATA
3270 fci=4:fci1=11:gosub SUBREFRESH
3280 wm=1:gosub WMODE:cy1=170:fis=2
3290 for x=1 to 6:read fci,fsi,cx,cy,c
x1
3300 rt=2:gosub DOSHAPE:next x
3310 rad=20:for x=1 to 4:read fci,fsi,
cx,cy
3320 rt=1:gosub DOSHAPE:next y
3330 rt=1:rad=40:fis=3:fsi=3:fci=1:cx=
228:cy=109
3340 gosub DOSHAPE:wm=2:gosub WMODE
3350 rt=1:tc=5:th=8:te=16:gosub DOTEXT
3360 gotoxy 6,1:?"B A R S"
3370 rt=1:tc=2:gosub DOTEXT
3380 gotoxy 19,1:?"C I R C L E S"
3390 poke gintin,257:gemsys(78):return
3410 ITEM5C:
3420 restore ITEM5CDATA
3430 cx=fx:cy=fy:cx1=304:cy1=99
3440 rt=3:wm=1:fis=1:fci=1:gosub DOSHA
PE
3450 cy=100:cy1=176
3460 rt=3:fci=0:gosub DOSHAPE:id=6
3470 wm=2:gosub WMODE:rt=1:tc=0:th=dth
:te=0:gosub DOTEXT
3480 n=1:cy1=27:cy2=95:cy3=65:ry=68:sa
1=1800:ea1=0:id=6
3490 sa=sa1:ea=ea1:cx=15:cy=cy1:radx=7
:rady=ry
3500 for x=2 to 15:read char(0)
3510 if id=7 then fsi=x:fci=x:gosub FI
LLINDEX:gosub FILLCOLOR
3520 if id=6 then plc=x:gosub PCOLOR
3530 gosub ELLARCPIE
3540 cx1=cx:cx=cx-3:cy=cy3:gosub GTEXT
:cx=cx1
3550 if sa=sa1 then sa=ea1:ea=sa1:cy=c
y2 else sa=sa1:ea=ea1:cy=cy1
3560 cx=cx+21:next x:if id=7 then goto
 3590
3570 tc=1:gosub TCOLOR:fis=2:gosub FIL
LSTYLE
3580 sa1=1350:ea1=450:ry=30:cy1=138:cy
2=138:cy3=150:id=7:goto 3490
3590 cx=5:cy=27:tc=0:th=4:te=0:gosub D
OTEXT
3600 cy=105:tc=1:gosub DOTEXT
3610 poke gintin,257:gemsys(78):return
3630 ITEM5D:
3640 restore ITEM5DDATA
3650 tc=2:th=8:te=16:fci=1:gosub CLEAR
ITEM
3660 cx=152:cy=105:radx=150:rady=65
3670 for x=1 to 10
3680 rt=5:fis=1:fci=1:gosub DOSHAPE
3690 rt=5:fis=2:fsi=x:fci=x:gosub DOSH
APE
3700 radx=radx-10:rady=rady-5:next x
3710 poke gintin,257:gemsys(78):return
3730 PMARKER:
3740 poke contrl,7:'OPCODE
3750 poke contrl+2,n:'Number of marker
s
3760 poke contrl+6,0
3770 for lp=0 to (n*2)-1:'Enter coordi
nates
3780 poke ptsin+(lp*2),coord(lp)
3790 next lp
3800 vdisys(1):return
3820 PMTYPE:
3830 poke contrl,18:'OPCODE
3840 poke contrl+2,0
3850 poke contrl+6,1
3860 poke intin,pmt:'Marker type
3870 vdisys(1):return
3890 PMHEIGHT:
3900 poke contrl,19:'OPCODE
3910 poke contrl+2,1
3920 poke contrl+6,0
3930 poke ptsin,0
3940 poke ptsin+2,pmh:'Marker height
3950 vdisys(1):return
3970 PMCOLOR:
3980 poke contrl,20:'OPCODE
3990 poke contrl+2,0
4000 poke contrl+6,1
4010 poke intin,pmc:'Color index
4020 vdisys(1):return
4040 PLINE:
4050 poke contrl,6:'OPCODE
4060 poke contrl+2,n:'Number of X,Y pa
irs in line
4070 poke contrl+6,0
4080 for lp=0 to (n*2)-1:'Enter coordi
```

// VDI Functions *continued*

```
nates
4090 poke ptsin+(lp*2),coord(lp)
4100 next lp
4110 vdisys(1):return
4130 PTYPE:
4140 poke contrl,15:'OPCODE
4150 poke contrl+2,0
4160 poke contrl+6,1
4170 poke intin,plt:'Polyline type
4180 vdisys(1):return
4200 PWIDTH:
4210 poke contrl,16:'OPCODE
4220 poke contrl+2,1
4230 poke contrl+6,0
4240 poke ptsin,plw:'Polyline width
4250 poke ptsin+2,0
4260 vdisys(1):return
4280 PSTYLE:
4290 poke contrl,108:'OPCODE
4300 poke contrl+2,0
4310 poke contrl+6,2
4320 poke intin,plsb:'End style for be
gining of line
4330 poke intin+2,plse:'End style for
end of line
4340 vdisys(1):return
4360 PCOLOR:
4370 poke contrl,17:'OPCODE
4380 poke contrl+2,0
4390 poke contrl+6,1
4400 poke intin,plc:'Polyline color in
dex
4410 vdisys(1):return
4430 GTEXT:
4440 poke contrl,8:'OPCODE
4450 poke contrl+2,1
4460 poke contrl+6,n:'NUmber of charac
ters to display
4470 for lp=0 to n-1:'Enter text to di
splay (in ASCII)
4480 poke intin+(lp*2),char(lp)
4490 next lp
4500 poke ptsin,cx:'X coordinate
4510 poke ptsin+2,cy:'Y coordinate
4520 vdisys(1):return
4540 TCOLOR:
4550 poke contrl,22:'OPCODE
4560 poke contrl+2,0
4570 poke contrl+6,1
4580 poke intin,tc:'Text color index
4590 vdisys(1):return
4610 TEFFECT:
4620 poke contrl,106:'OPCODE
4630 poke contrl+2,0
4640 poke contrl+6,1
4650 poke intin,te:'Text effect word
4660 vdisys(1):return
4680 THEIGHT:
4690 poke contrl,12:'OPCODE
4700 poke contrl+2,1
4710 poke contrl+6,0
4720 poke ptsin,0
4730 poke ptsin+2,th:'Character height
4740 vdisys(1)
4750 charw=peek(ptsout):'Character wid
th
4760 charh=peek(ptsout+2):'Character h
eight
4770 cellw=peek(ptsout+4):'Cell width
4780 cellh=peek(ptsout+6):'Cell height
4790 return
4810 THEIGHTP:
4820 poke contrl,107:'OPCODE
4830 poke contrl+2,0
4840 poke contrl+6,1
4850 poke intin,th:'Cell height
4860 vdisys(1)
4870 charw=peek(ptsout):'Character wid
th
4880 charh=peek(ptsout+2):'Character h
eight
4890 cellw=peek(ptsout+4):'Cell width
4900 cellh=peek(ptsout+6):'Cell height
4910 return
4930 TBASE:
4940 poke contrl,13:'OPCODE
4950 poke contrl+2,0
4960 poke contrl+6,1
4970 poke intin,tbl:'Baseline angle
4980 vdisys(1):return
5000 WMODE:
5010 poke contrl,32:'OPCODE
5020 poke contrl+2,0
5030 poke contrl+6,1
5040 poke intin,wm:'Writing mode code
5050 vdisys(1):return
5070 ARCPIE:
5080 poke contrl,11:'OPCODE
5090 poke contrl+2,4
5100 poke contrl+6,2
5110 poke contrl+10,id:'Primitive ID
 2=ARC  3=PIE
5120 poke intin,sa:'Start angle in ten
ths of degrees (0-3600)
5130 poke intin+2,ea:'End angle in ten
ths of degrees (0-3600)
5140 poke ptsin,cx:'X coordinate of ce
nter point
5150 poke ptsin+2,cy:'Y coordinate of
center point
5160 for lp=4 to 10 step 2:poke ptsin+
lp,0:next lp
5170 poke ptsin+12,rad:'Radius
5180 poke ptsin+14,0
5190 vdisys(1):return
5210 BAR:
5220 poke contrl,11:'OPCODE
5230 poke contrl+2,2
5240 poke contrl+6,0
5250 poke contrl+10,1:'Primitive ID
1=BAR
5260 poke ptsin,cx:'X coordinate of ba
r
5270 poke ptsin+2,cy:'Y coordinate of
bar
5280 poke ptsin+4,cx1:'X coordinate of
 bar diagonally opposite
5290 poke ptsin+6,cy1:'Y coordinate of
 bar diagonally opposite
5300 vdisys(1):return
5320 CIRCL:
5330 poke contrl,11:'OPCODE
5340 poke contrl+2,3
5350 poke contrl+6,0
5360 poke contrl+10,4:'Primitive ID
4=CIRCLE
5370 poke ptsin,cx:'X coordinate of ce
nter point
5380 poke ptsin+2,cy:'Y coordinate of
center point
5390 poke ptsin+4,0
5400 poke ptsin+6,0
5410 poke ptsin+8,rad:'Radius
5420 poke ptsin+10,0
5430 vdisys(1):return
5450 ELLARCPIE:
5460 poke contrl,11:'OPCODE
5470 poke contrl+2,2
5480 poke contrl+6,2
5490 poke contrl+10,id:'Primitive ID
 6=Ell.ARC  7=Ell.PIE
5500 poke intin,sa:'Start angle in ten
ths of degrees
5510 poke intin+2,ea:'End angle in ten
ths of degrees (0-3600)
```

```
5520 poke ptsin,cx:'X coordinate of ce
nter point
5530 poke ptsin+2,cy:'Y coordinate of
center point
5540 poke ptsin+4,radx:'Radius of X ax
is
5550 poke ptsin+6,rady:'Radius of Y ax
is
5560 vdisys(1):return
5580 ELLIPS:
5590 poke contrl,11:'OPCODE
5600 poke contrl+2,2
5610 poke contrl+6,0
5620 poke contrl+10,5:'Primitive ID
5=Ellipse
5630 poke ptsin,cx:'X coordinate of ce
nter point
5640 poke ptsin+2,cy:'Y coordinate of
center point
5650 poke ptsin+4,radx:'Radius of X ax
is
5660 poke ptsin+6,rady:'Radius of Y ax
is
5670 vdisys(1):return
5690 RRECT:
5700 poke contrl,11:'OPCODE
5710 poke contrl+2,2
5720 poke contrl+6,0
5730 poke contrl+10,id:'Primitive ID
 8=Rounded rect  9=Filled
5740 poke ptsin,cx:'X coordinate of re
ctangle
5750 poke ptsin+2,cy:'Y coordinate of
rectangle
5760 poke ptsin+4,cx1:'X coordinate di
agonally opposite
5770 poke ptsin+6,cy1:'Y coordinate di
agonally opposite
5780 vdisys(1):return
5800 FILLA:
5810 poke contrl,9:'OPCODE
5820 poke contrl+2,n:'Number of lines
in ploygon
5830 poke contrl+6,0
5840 for lp=0 to (n*2)-1:'Enter coordi
nates
5850 poke ptsin+(lp*2),coord(lp)
5860 next lp
5870 vdisys(1):return
5890 CONTOUR:
5900 poke contrl,103:'OPCODE
5910 poke contrl+2,1
5920 poke contrl+6,1
5930 poke intin,cc:'Color index defini
ng contour
5940 poke ptsin,cx:'X coordinate of sa
trting point
5950 poke ptsin+2,cy:'Y coordinate of
satrting point
5960 vdisys(1):return
5980 RECTFILL:
5990 poke contrl,114:'OPCODE
6000 poke contrl+2,2
6010 poke contrl+6,0
6020 poke ptsin,cx:'X coordinate of re
ctangle
6030 poke ptsin+2,cy:'Y coordinate of
rectangle
6040 poke ptsin+4,cx1:'X coordinate di
agonally opposite
6050 poke ptsin+6,cy1:'Y coordinate di
agonally opposite
6060 vdisys(1):return
6080 FILLSTYLE:
6090 poke contrl,23:'OPCODE
6100 poke contrl+2,0
6110 poke contrl+6,1
6120 poke intin,fis:'Fill interior sty
le code
6130 vdisys(1):return
6150 FILLINDEX:
6160 poke contrl,24:'OPCODE
6170 poke contrl+2,0
6180 poke contrl+6,1
6190 poke intin,fsi:'Fill style index
code
6200 vdisys(1):return
6220 FILLCOLOR:
6230 poke contrl,25:'OPCODE
6240 poke contrl+2,0
6250 poke contrl+6,1
6260 poke intin,fci:'Fill color index
6270 vdisys(1):return
6290 PERMV:
6300 poke contrl,104:'OPCODE
6310 poke contrl+2,0
6320 poke contrl+6,1
6330 poke intin,pv:'Perimeter flag  0=
Invisible  1=Visible
6340 vdisys(1):return
6360 TIMER:
6370 poke gintin,5000:'5 Second wait
6380 poke gintin+2,0
6390 gemsys(24):'OPCODE
6400 return
6410 '--- PROGRAM DATA ---
6440 INTRODATA:
6450 data 5,86,32,68,32,73
6460 data 13,83,32,65,32,77,32,80,32,7
6,32,69,32,82
6470 data 4,70,82,79,77
6480 data 11,65,32,78,32,65,32,76,32,7
9,32,71
6490 data 9,67,79,77,80,85,84,73,78,71
6510 MENU0DATA:
6520 data 34,76,73,78,69,32,77,65,82,7
5,69,82,32,84,69,88,84,32
6530 data 70,73,76,76,32,83,72,65,80,6
9,83,32,32,81,85,73,84
6540 data 12,0,32,42,88,98,128,138,168
,178,224,250,280
6550 data 0,0,5,0,0,0
6560 MENU1DATA:
6570 data 31,84,89,80,69,32,87,73,68,8
4,72,32,69,78,68,83
6580 data 84,89,76,69,32,32,32,77,65,7
3,78,32,77,69,78,85
6590 data 8,0,32,42,80,90,152,186,256
6600 data 5,8,2,3,9,0
6610 MENU2DATA:
6620 data 23,84,89,80,69,32,72,69,73,7
1,72,84,32,32,32
6630 data 77,65,73,78,32,77,69,78,85
6640 data 6,0,32,42,88,122,192
6650 data 2,8,6,4,10,4
6660 MENU3DATA:
6670 data 33,83,73,90,69,32,66,65,83,6
9,76,73,78,69,32
6680 data 69,70,70,69,67,84,83,32,32,3
2,77,65,73,78,32,77,69,78,85
6690 data 8,0,32,42,104,114,168,202,27
2
6700 data 12,8,3,7,12,2
6710 MENU4DATA:
6720 data 25,78,69,88,84,32,80,82,69,8
6,73,79,85,83,32,32,32
6730 data 77,65,73,78,32,77,69,78,85
6740 data 6,0,32,43,104,130,200
6750 data 2,8,11,1,21,4
6760 MENU5DATA:
6770 data 50,65,82,67,47,80,73,69,32,6
6,65,82,47,67,73,82,67,76,69,32
6780 data 69,76,76,46,65,82,67,47,80,7
3,69,32,69,76,76,73,80,83,69,32
```

// VDI Functions *continued*

```
6790 data 32,32,77,65,73,78,32,77,69,7
8,85
6800 data 10,0,43,50,109,116,181,188,2
29,248,301
6810 data 4,8,10,4,20,5
6830 ITEM1ADATA:
6840 data 10,76,73,78,69,32,84,89,80,6
9,83
6850 data 6,84,89,80,69,32,49,6,84,89,
80,69,32,50,6,84,89,80,69,32,51
6860 data 6,84,89,80,69,32,52,6,84,89,
80,69,32,53,6,84,89,80,69,32,54
6870 ITEM1BDATA:
6880 data 11,76,73,78,69,32,87,73,68,8
4,72,83
6890 data 7,87,73,68,84,72,32,49,7,87,
73,68,84,72,32,51
6900 data 7,87,73,68,84,72,32,53,7,87,
73,68,84,72,32,53
6910 ITEM1CDATA:
6920 data 9,69,78,68,83,84,89,76,69,83
6930 data 10,69,78,68,83,84,89,76,69,3
2,48,10,69,78,68,83,84,89,76,69,32,49
6940 data 10,69,78,68,83,84,89,76,69,3
2,50
6950 ITEM2ADATA:
6960 data 12,77,65,82,75,69,82,32,84,8
9,80,69,83
6970 data 4,75,55,125,55,200,55,275,55
6980 data 6,84,89,80,69,32,49,6,84,89,
80,69,32,50,6,84,89,80,69,32,51
6990 data 6,84,89,80,69,32,52,6,84,89,
80,69,32,53,6,84,89,80,69,32,54
7000 ITEM2BDATA:
7010 data 13,77,65,82,75,69,82,32,72,6
9,73,71,72,84
7020 data 6,75,115,155,195,235,275
7030 data 9,72,69,73,71,72,84,32,49,48
,9,72,69,73,71,72,84,32,50,48
7040 data 9,72,69,73,71,72,84,32,51,48
,9,72,69,73,71,72,84,32,52,48
7050 data 9,72,69,73,71,72,84,32,53,48
7060 ITEM3ADATA:
7070 data 9,84,69,88,84,32,83,73,90,69
7080 data 11,84,69,88,84,32,83,73,90,6
9,32,52
7090 data 11,84,69,88,84,32,83,73,90,6
9,32,54
7100 data 11,84,69,88,84,32,83,73,90,6
9,32,56
7110 data 12,84,69,88,84,32,83,73,90,6
9,32,49,48
7120 data 12,84,69,88,84,32,83,73,90,6
9,32,49,50
7130 ITEM3BDATA:
7140 data 8,66,65,83,69,76,73,78,69
7150 data 0,110,55,10,66,65,83,69,76,7
3,78,69,32,48
7160 data 900,50,150,12,66,65,83,69,76
,73,78,69,32,57,48,48
7170 data 1800,200,150,13,66,65,83,69,
76,73,78,69,32,49,56,48,48
7180 data 2700,250,55,13,66,65,83,69,7
6,73,78,69,32,50,55,48,48
7190 ITEM3CDATA:
7200 data 8,32,69,70,70,69,67,84,83
7210 data 0,80,10,48,32,32,32,78,79,82
,77,65,76
7220 data 1,80,11,49,32,84,72,73,67,75
,69,78,69,68
7230 data 2,80,11,50,32,73,78,84,69,78
,83,73,84,89
7240 data 4,75,10,52,32,32,32,83,75,69
,87,69,68
7250 data 8,77,12,56,32,85,78,68,69,82
,76,73,78,69,68
7260 data 16,75,11,49,54,32,79,85,84,7
6,73,78,69,68
7270 data 32,75,11,51,50,32,83,72,65,6
8,79,87,69,68
7280 ITEM5ADATA:
7290 data 3,3,1,22,304,22,1,176
7300 data 7,3,304,22,304,176,1,176
7310 data 7,0,5,90,70,240,115,2,5,16,8
0,70,230,115
7320 data 1,1,19,80,80,230,125,4,3,7,9
0,80,240,125
7330 data 85,30,85,120,35,75,135,75,23
5,75,235,167,185,120,285,120
7340 data 139,77,1,48,76,28,3,57,48,48
,5,77,4,49,56,48,48
7350 data 72,127,4,50,55,48,48
7360 data 289,122,1,48,226,73,3,57,48,
48,155,122,4,49,56,48,48
7370 data 222,175,4,50,55,48,48
7380 ITEM5BDATA:
7390 data 1,23,17,50,47,0,8,52,70,77,5
,24,82,90,102
7400 data 2,19,107,110,122,6,16,127,13
0,137,3,10,142,150,147
7410 data 2,10,182,62,4,21,274,62,7,12
,182,156,10,24,274,156
7420 ITEM5CDATA:
7430 data 69,76,76,73,80,84,73,67,65,7
6,32,65,82,67
7440 data 69,76,76,73,80,84,73,67,65,7
6,32,80,73,69
7450 data 4,73,68,32,54,4,73,68,32,55
7460 ITEM5DDATA:
7470 data 9,32,32,69,76,76,73,80,83,69
```

●

ST-CHECKSUM DATA.

(see page 63ST)

```
 100 data  388, 683, 932, 753, 140, 62
0, 70, 47, 401, 399, 4433
 200 data  639, 501, 2, 245, 356, 514,
 241, 775, 73, 32, 3378
 310 data  872, 504, 231, 765, 63, 781
, 322, 923, 144, 40, 4645
 430 data  616, 312, 25, 999, 27, 741,
 904, 961, 250, 234, 5069
 540 data  608, 858, 532, 619, 882, 56
5, 249, 719, 454, 953, 6439
 640 data  66, 165, 555, 612, 882, 203
, 123, 894, 390, 370, 4260
 750 data  195, 93, 10, 582, 896, 462,
 212, 350, 409, 193, 3402
 860 data  790, 585, 87, 655, 60, 155,
 352, 537, 944, 424, 4589
 970 data  639, 421, 63, 740, 832, 816
, 519, 75, 773, 78, 4956
1080 data  112, 380, 911, 689, 47, 55
1, 3, 942, 231, 884, 4750
1190 data  300, 79, 229, 609, 294, 97
, 950, 393, 475, 849, 4275
1300 data  409, 163, 743, 703, 261, 7
78, 791, 31, 290, 277, 4446
1400 data  99, 869, 273, 313, 706, 98
3, 802, 871, 147, 827, 5890
1510 data  831, 835, 839, 843, 368, 8
56, 837, 857, 978, 870, 8114
1620 data  992, 839, 890, 114, 416, 7
10, 868, 954, 381, 79, 6243
1740 data  439, 275, 960, 44, 36, 947
, 993, 538, 849, 993, 6074
1850 data  390, 87, 444, 281, 879, 99
2, 959, 946, 91, 548, 5617
1950 data  82, 999, 399, 95, 410, 261
```

```
, 8, 229, 939, 924, 4346
 2060 data  69, 460, 149, 977, 366, 63
, 327, 497, 110, 232, 3250
 2170 data  830, 454, 466, 219, 43, 97
6, 375, 71, 333, 396, 4163
 2280 data  30, 20, 113, 997, 841, 452
, 130, 916, 983, 381, 4863
 2390 data  77, 337, 650, 7, 984, 43,
933, 388, 916, 988, 5323
 2500 data  382, 77, 443, 235, 289, 76
8, 844, 856, 856, 870, 5620
 2600 data  863, 801, 988, 392, 86, 44
3, 819, 367, 485, 877, 6121
 2710 data  990, 390, 390, 640, 864, 6
00, 13, 540, 394, 463, 5284
 2830 data  442, 867, 535, 104, 541, 5
5, 973, 421, 179, 248, 4365
 2940 data  470, 923, 666, 178, 589, 9
12, 971, 373, 67, 163, 5312
 3060 data  771, 896, 448, 936, 526, 2
76, 589, 716, 981, 925, 7064
 3160 data  937, 789, 185, 580, 380, 5
48, 600, 979, 384, 77, 5459
 3270 data  628, 463, 921, 895, 197, 8
97, 961, 749, 989, 67, 6767
 3370 data  554, 772, 988, 389, 81, 80
4, 152, 160, 391, 573, 4864
 3480 data  582, 811, 701, 309, 722, 6
07, 191, 249, 302, 16, 4490
 3580 data  720, 530, 762, 989, 400, 9
1, 437, 407, 79, 622, 5037
 3690 data  489, 592, 992, 576, 311, 9
77, 446, 274, 889, 452, 5998
 3800 data  729, 512, 433, 443, 451, 3
02, 736, 811, 435, 446, 5298
 3920 data  448, 131, 686, 737, 621, 4
26, 451, 424, 332, 709, 4965
 4040 data  333, 292, 311, 428, 256, 8
71, 427, 711, 401, 409, 4439
 4150 data  425, 433, 641, 718, 477, 4
11, 428, 430, 864, 258, 5085
 4260 data  719, 506, 496, 426, 437, 8
63, 333, 720, 492, 422, 5414
 4380 data  434, 442, 689, 720, 379, 3
05, 437, 965, 940, 656, 5967
 4490 data  445, 220, 467, 724, 500, 4
17, 438, 446, 686, 731, 5074
 4610 data  548, 495, 438, 446, 659, 7
31, 580, 423, 441, 443, 5204
 4720 data  126, 20, 725, 272, 610, 74
8, 952, 473, 797, 504, 5227
 4830 data  444, 452, 977, 730, 277, 6
15, 753, 950, 471, 324, 5993
 4940 data  429, 449, 4579 689, 742, 3
48, 401, 421, 429, 842, 5207
 5050 data  714, 418, 404, 440, 432, 4
4, 534, 374, 369, 689, 4418
 5160 data  295, 825, 401, 721, 997, 4
04, 434, 433, 234, 60, 4804
 5270 data  319, 392, 403, 718, 353, 4
08, 441, 437, 690, 378, 4539
 5380 data  698, 273, 273, 752, 400, 7
24, 877, 414, 444, 449, 5304
 5490 data  128, 744, 384, 379, 699, 2
69, 281, 730, 502, 420, 4536
 5600 data  443, 442, 813, 383, 703, 2
73, 285, 734, 379, 417, 4872
 5710 data  447, 446, 723, 982, 189, 3
72, 383, 738, 338, 317, 4935
 5820 data  113, 450, 278, 893, 456, 7
40, 677, 495, 450, 455, 5007
 5930 data  251, 883, 849, 742, 791, 5
09, 427, 426, 961, 168, 6007
 6040 data  351, 362, 717, 944, 412, 4
24, 432, 886, 717, 928, 6173
 6160 data  414, 431, 439, 385, 717, 9
29, 416, 431, 439, 739, 5340
 6270 data  724, 394, 482, 431, 439, 4
85, 724, 375, 369, 439, 4862
 6390 data  892, 459, 957, 919, 784, 1
75, 494, 576, 81, 828, 6165
 6520 data  464, 150, 513, 112, 835, 8
70, 862, 739, 165, 835, 5545
 6620 data  515, 752, 56, 264, 842, 55
5, 766, 935, 401, 842, 5928
 6720 data  154, 755, 125, 394, 849, 6
2, 84, 343, 853, 402, 4021
 6830 data  968, 351, 38, 29, 975, 656
, 632, 614, 975, 132, 5370
 6930 data  690, 347, 975, 967, 961, 4
4, 35, 948, 226, 489, 5682
 7030 data  787, 770, 15, 953, 59, 616
, 619, 615, 917, 913, 6264
 7130 data  956, 773, 184, 169, 354, 4
00, 965, 723, 913, 273, 5710
 7230 data  252, 935, 645, 227, 214, 9
65, 110, 441, 51, 23, 3863
 7330 data  664, 798, 76, 356, 151, 67
, 174, 963, 45, 971, 4265
 7430 data  261, 263, 784, 978, 49, 23
35
```

•

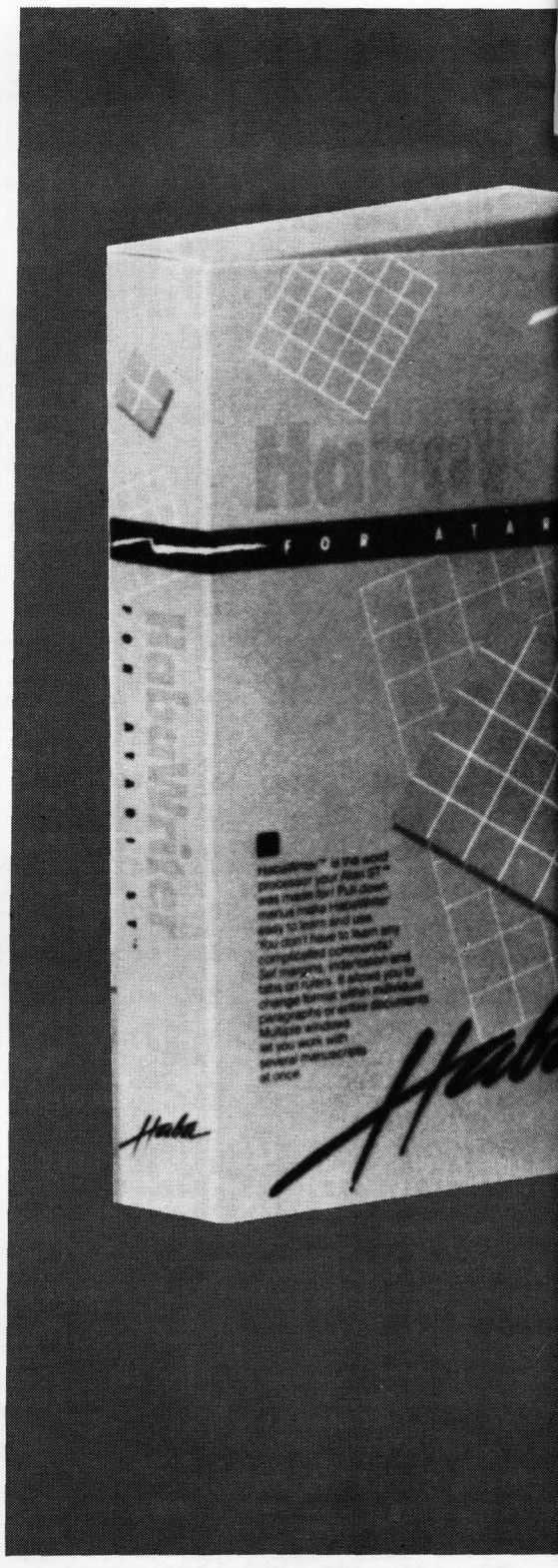
WHAT IS ST-CHECK?

Most program listings in **ST-Log** are followed by a table of numbers appearing as DATA statements, called "ST CHECKSUM DATA." These numbers are to be used in conjunction with **ST-Check** (which appeared in **ANALOG Computing/ST-Log** issue 41.

ST-Check (written by Clayton Walnum) is designed to find and correct typing errors when readers are entering programs from the magazine. For those readers who would like copies of the article, you may send for back issue 41 ($4.00).

ANALOG Computing/ST-Log
P.O. Box 625, Holmes, PA 19045

STylish Software



No question about it, the new Atari 520 ST™ is a remarkable computer. And nothing complements a great computer better than great software and great peripherals.

HabaWriter™. A full-function word processor, featuring windows for simultaneous multiple document editing as well as pull-down menus for fast access to program commands. Advantageous use of the mouse means never having to memorize cryptic commands again. **HabaWriter** is the word processor your 520 ST has been waiting for. If you do any writing at all, take a look at **HabaWriter**. Suggested Retail: $74.95

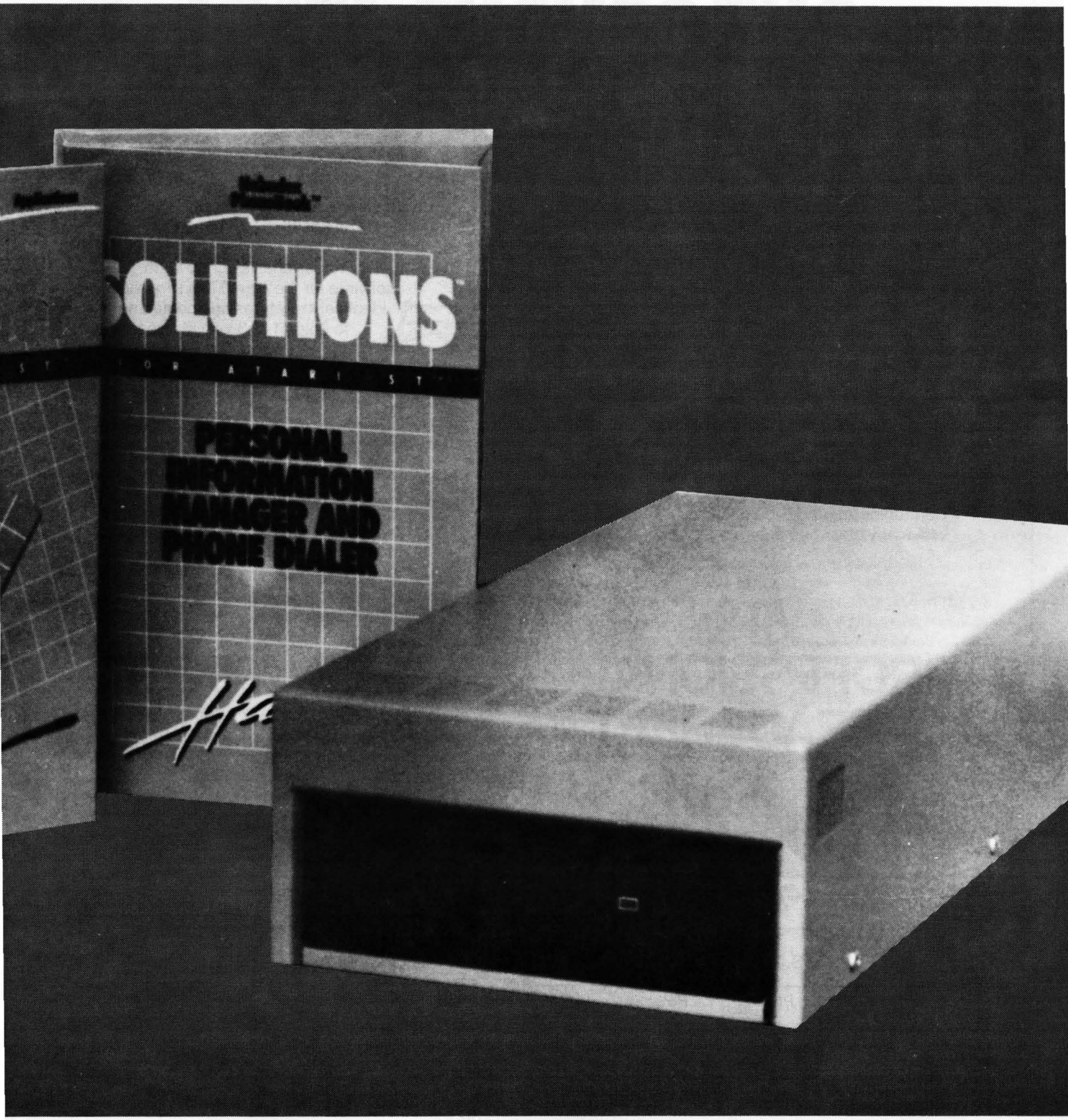
Habadex PhoneBook™ is the elegant way to store phone numbers. And it not only stores numbers, but it can dial them as well. It works and looks just like the flip-up phone book that you're used to. Long distance services like MCI and Sprint can be automatically dialed so you don't have to. The **PhoneBook** can sort on any field, is versatile enough to handle other types of information and can even print mailing labels. (Automatic dialing requires either a **HabaModem**™ or any Hayes™ compatible modem.) Suggested Retail: $49.95

The new **HabaDisk**™ **10 Megabyte** hard disk for the 520 ST is a Winchester plug-in hard disk that is capable of storing the equivalent of more than 12 dual-sided 800K diskettes and retrieves information in seconds (3 msec. track-to-track access time). It is self-powered and completely Atari ST compatible (including Atari Desktop and GEM™ DOS). Suggested Retail: $699.95

Also available for the 520 ST:

Haba Checkminder™—Suggested Retail: $74.95
Haba Mail Room™—Suggested Retail: $74.95
HabaMerge™—Suggested Retail: $39.95
Solutions: Wills™—Suggested Retail: $49.95
Solutions: Business Letters™—Suggested Retail: $49.95

STupendous Storage



Haba

6711 Valjean Avenue
Van Nuys, CA 91406
(818) 989-5822 • (800) HOT-HABA (USA) • (800) FOR-HABA (CA)

HOW TO GET THE MOST OUT OF YOUR ATARI.

The powerful new Atari ST is capable of extraordinary color graphics. Right now, there is only one full size, commercially available color printer that can screen dump in over 120 colors off the Atari ST, and we've got it. The Shanner SPC 700CI color printer.

Other features include:

- Fast and quiet operation
- Uses standard paper—inexpensive to operate
- Centronics interface—compatible with many other computers

- Million characters per color ribbon cassette
- Made by Seikosha

VIP Professional is our powerful new software for the Atari ST. Its program is identical in features and commands to Lotus 1-2-3 with Macros, 256 x 8092 columns, Data Base capabilities with Sort and Query Fields.

VIP Professional is a trademark of VIP Technologies, Lotus 1-2-3 is a trademark of Lotus Development Corporation. Atari ST is a trademark of Atari Corporation.

VIP PROFESSIONAL SPREADSHEET

$99.90

SPC-700CI COLOR PRINTER

$299.95

Full-Blown Original Version

GUMBALL EXPRESS ORDER FORM • For FAST delivery use this order form or call **TOLL FREE 800/423-9442**

Product Description	Price	P&L	TOTAL
VIP Professional for Atari ST	$ 99.90	$3.50	**$103.40**
Shanner SPC-700CI	$299.95	$7.50	**$307.45**

☐ Check here and add $14.95 to $399.95

All products will be shipped prepaid UPS ground.

☐ Check enclosed. (NOTE— order will be shipped when check clears).

Make check payable to:
Gumball Express
707 S.W. Washington Street Suite 200
Portland, Oregon 97205

☐ VISA ☐ MASTERCARD ☐ INTERBANK (MasterCard only)

Name on card _____

Account # _____

Expiration Date _____

Signature _____

SHIP TO:

Name _____

Address _____

City _____ State _____ Zip _____

C.O.D.'s and purchase orders will not be accepted by Gumball Express. Outside the USA add $10. and make payment by bank draft, payable in U.S. dollars drawn on a U.S. bank.

CIRCLE #131 ON READER SERVICE CARD

VIP Professional/Lite

VIP TECHNOLOGIES
132 Aero Camino
Santa Barbara, CA 93117
(805) 685-3948
520ST VIP Professional $180
VIP Professional Lite $100

by Arthur Leyenberger

When I first got the **VIP Professional**, I was eager to try it. I'd been a long-time user of Lotus 1-2-3, and I wanted to compare them head to head (or, more appropriately, cell to cell).

Lotus 1-2-3 was and is a breakthrough in the MS-DOS world. I was hoping the **Professional** would be the same for Atari. Here's where the tale begins.

In the carton, I found a thick reference manual. On skimming, it seemed easy to read. I also found the 3½-inch micro floppy disk. Eager to try the program on my 520ST, I booted up with TOS, then inserted the **Professional** disk in the drive. When I tried to get a disk directory, I got an error message.

Taking the disk out, I discovered what seemed to be a piece of adhesive tape on the disk's business end. Looking closer, I read a warning to the effect that, if I broke this seal, I was bound to the rules and regulations set forth in VIP's software license. Okay, I'd seen this kind of warning before. Typically, though, the disks would come wrapped in a sealed baggie; opening it would bind me to the license agreement.

I attempted to remove the sticker. It wouldn't come off—not easily. I soon realized the only way to remove it (and to use the disk) was to scrape it off.

I began doing so very carefully, trying to avoid getting crumbs inside the disk shell. I kept scraping, rubbing and, generally, getting nervous. I knew that if anything did get into the disk undetected, it might become history once accessed by the drive. Finally, I got the sticker off and inserted the disk.

VIP Professional booted fine. I didn't have time then for a full-length session. I just wanted to see how similar the screen and user interface were to the Lotus—and it looked very much the same. It *is* a Lotus clone, right?

My next surprise came on exiting the program. The message said, *insert TOS disk and turn the computer off and on*. I was slightly annoyed that I couldn't get back to the desktop. It was a nuisance, nothing more.

Next, I thought I'd better back the program up, so I wouldn't have to worry about those crumbs. Using MichTron's **M-Copy** program (the best currently available), I made a backup. When I tried to run the copy, I found it wouldn't work. It still needed the original disk in drive 1, using the key disk system. Surprise: **VIP** is copy protected.

I'm against copy protection for application and utility programs. If manufacturers want to protect games, that's okay with me. But a program used for serious work—especially at this price—shouldn't be copy protected. And, with those crumbs still causing nightmares, I get queasy every time I put the original in the drive.

So ends my preface. Right off the bat, I felt abused by VIP. The disk was stuck shut; I had to unstick it, causing possible disk damage— and I couldn't back it up completely. At least VIP could have included two disks.

It's been a month since that first experience with **VIP Professional**. I no longer feel angry. But I have seen several revisions of the program, each correcting previous bugs. I finally have a copy of **Professional** that's bug free, as far as I can tell. Now it's time for an *objective* review.

Professional by VIP Technologies is a spreadsheet for the ST, based upon the well-known, widely-used Lotus 1-2-3. As an integrated spreadsheet, **Professional** provides a sophisticated spreadsheet, database and presentation graphics capabilities, rolled into one program.

It really is a Lotus clone. As such, it will let you use the same keystrokes, applications, data files and templates as the original does. Worksheet files created on an IBM PC with Lotus 1-2-3 can be transferred to the ST, then accessed by the **VIP Professional**. No muss, no fuss.

The version of the **Professional** used for this review was the so-called text version, which doesn't use the GEM system

at all. No mouse control, drop-down menus or pointing and clicking. But that's how Lotus 1-2-3 works; many people have used this setup effectively for years.

Professional has two modes of operation. NAT is the native mode, used most of the time. The other is WKS and is Lotus compatible. In the latter mode, files in Lotus format can be read into and saved from **Professional**.

The native mode files load and save very quickly from the spreadsheet, whereas files in the WKS mode have to be translated when loaded. It's slightly slower. Pressing the ALT and UNDO keys on the ST toggle **Professional** into the two modes.

On a 520ST with TOS on disk, only about 40K of memory is available for your **Professional** spreadsheets. With TOS on ROM, a spreadsheet can be over 200K in size. If you have a 1-megabyte ST (either a 520 upgrade or a 1040), over ½ megabyte is available for your spreadsheet—more than the IBM PC allows Lotus 1-2-3.

The documentation is first rate. The comprehensive spiral bound handbook has over 200 pages, divided into tutorial and reference sections, plus a glossary, appendices and an index.

Well written and easy to understand, the tutorial gets you started with the program and introduces you to the basic concepts of **VIP Professional**. You're led through a sample budget worksheet with plenty of procedures and examples. The reference section gives details about the variety of available commands and functions.

The **VIP Professional** screen is divided into two areas, with a menu at the top line and the worksheet area filling the remainder. The menu line at the top gives the titles for commands accessed by pressing the / key.

For example, pressing */w* displays another set of menu titles, like column width, delete, erase, global, window, etc. Select an item, and you're prompted for an entry. Pressing ESC returns you to the previous menu level.

The worksheet area consists of a grid-like pattern of horizontal and vertical lines forming cells. Each cell may contain a piece of data. Columns are labeled *A, B, C. . .*, and rows are labeled by number, top to bottom.

The arrow cursor keys are used to move the cell pointer around the worksheet. Wherever the cell pointer rests indicates which cell will be affected by data entry or be a command's starting point. The current cell, as indicated by the pointer, is displayed at the top left of the screen.

Like other spreadsheets, any position in the sheet can be defined as a label, value or formula. Further, any formula can relate to any other positions (or combinations of positions) on the worksheet.

When a position's value changes, all items dependent on that value change automatically, without any effort on the user's part. This allows you to perform whatever calculations and manipulations of figures you wish—with amazing speed and accuracy.

There are a couple of negative aspects to **Professional**. The program currently only supports TOS-recognized printers, so Epson and Epson compatibles are the only printers that will work with the program's graphic output.

Unlike Lotus, **Professional** won't let you select output devices. Another problem, albeit minor, is that you can't format a disk from the program. Lotus 1-2-3 shares this snag.

Being copy protected, the **Professional** uses a key disk system. As mentioned earlier, this means you can copy the disk's contents to another disk, but the original program must be in drive A when it's run. However, the program can't be used with a hard disk. With the key disk in drive A, **Professional** simply will not run from drive C.

Many have complained about the **Professional**'s slow scrolling when redrawing the screen, as compared to Lotus 1-2-3. Scrolling in any direction in Lotus is instantaneous. With the former, a slight delay occurs if the screen is redrawn. VIP explains this by saying that the IBM PC only has 2K bytes of screen memory to update, whereas the ST has 32K bytes. That accounts for the slowness.

I understand the technicalities, but it's still no excuse. When a user is familiar with an excellent product (like **SynCalc** for the 8-bit Atari, which has no scrolling problems), they expect a more expensive, more sophisticated product to function as well as that, if not better.

Professional Lite is almost identical to **Professional**. Aside from a price $80 lower, **Lite** doesn't have the ability to use macros, has no database functions and has a matrix of "only" 256 by 2048 cells (**Professional** allows a whopping 8192 spreadsheet rows, while Lotus 1-2-3 will yield up to 2047 rows).

If you don't need these features, **Lite** is the one to buy. The discounted price will fall somewhere in the $79-$80 range, which makes it an excellent bargain.

The GEM version of **Professional** should be even better. It's due in the second quarter of 1986. Current VIP policy is that all registered owners of **Professional** will receive the GEM upgrade for free, instead of for the original $20 fee.

The bottom line is that **Professional** and **Professional Lite** from VIP Technologies are, as they claim, true imitations of Lotus 1-2-3. Both programs have better graph features than Lotus, and the 68000 processor of the ST makes calculations much faster than does the IBM PC's 8088.

The documentation is good, and the price is right for **Lite**—and for the full **Professional**, if you need it. Now that the bugs have been worked out, I have no problem with recommending either one.

C-MANSHIP

Part 5.

by Clayton Walnum

Okay, people. Pass your homework to the front of the class. What was that? Did I hear someone in the back say, "What homework?"

For those who need their memories refreshed, last month I suggested that you try writing a C version of a simple number guessing game. You were to have the computer pick a number from *1* to *100*, then allow a player to enter guesses. With each guess, the player was to receive a clue as to whether he was too high or too low.

My solution for this project is found in Listing 1. Does your program look something like this? Maybe, maybe not. At this early point in your C career, I think the following qualities are most important.

First of all, does it work? If you can give me an affirmative, you've earned 70 points. At this stage of the game, getting programs up and running is a good part of the battle.

Now, did you use a structured approach? Does the function main() concern itself with the major steps of the game, allotting details to other functions? If so, give yourself another 20 points. When you become more familiar with C, this category will be more pointworthy. In fact, eventually, an unstructured program will be an automatic zero. Strict, huh?

Finally, how readable is your code? Have you used indentation? Are there blank lines between each function? Did you use meaningful names for your functions and variables? Another 10 points to those who've added this touch of elegance.

Game time again.

Now that you've tallied up your homework score, type in Listing 1 and compile it. If you need help, see the sidebar accompanying Part 3 (issue 41) of this series.

To play the game, run the program and follow the prompts. When you're asked to input a number, end your response with the SPACE BAR (remember the strange way scanf() works). Everything work okay? Let's examine this program in a little more detail.

The function main() is written in a manner that makes the program's general operation clearly apparent. All the details are taken care of in other functions. In other words, the program is *structured*.

We start off by initializing the flag play to TRUE. This will get us into the while loop at Line 9. As long as play is true, this loop will repeat, allowing the user to play many games without rerunning the program each time.

Once in the loop, we must initialize some variables. The counter turns tallies the player's guesses. The flag win tells main() when the player has made a correct guess.

After initializing the variables, we call the function getnum(), which returns a random number between *1* and *100*. Next, since we had the forethought to initialize win to false, we enter the while loop at Line 12. This loop will repeat until win becomes true,

keeping the player guessing until he comes up with the right number.

In the body of the loop, we increment the turn counter, get the player's guess, then check if he's right. If not, win remains false and the loop repeats. If it's been guessed correctly, program execution drops through to Line 17, where the player is told how many guesses were made.

Line 18 calls play_again() to see if the player wants to continue. If so, the flag play remains true, and the outer while loop repeats. When play becomes false, the program ends, and the user's returned to the desktop.

Easy, right? You should've followed the above with little difficulty.

The other functions are just as simple. The function get_num() uses the same method we incorporated last month in our dice game to get a random number. The only difference is that now we're getting a number between *1* and *100* rather than between *1* and *6*.

The function get_guess() incorporates a while loop, forcing the player to enter a number within the proper range. The loop will repeat until the gamester bends to our will.

The function check_guess() checks if the player's guess was too high, too low, or right on the money, then prints the appropriate message. If the player has guessed right, then wn is set to TRUE (and, thus, win), and the game is over.

Finally, the function play_again() asks if the player wants another whack at it. Once again, we use a while loop to guarantee a proper response. The call to getchar() at Line 58 gets rid of the extra character scanf() likes to leave lying around.

Some classy information.

Before we take a look at the next two listings, we need to discuss a fun topic called "Storage Classes."

All storage you define in your C programs has a storage class, whether you're aware of it or not. In our previous program examples, the storage classes were set automatically. We didn't have to concern ourselves with the details. That's all fine and dandy for a beginner, but sooner or later we're going to have to know how our variables are treated by the system.

There are four C keywords that refer to the storage classes. They are: *extern*, *auto*, *static* and *register*.

The keyword *extern* stands for external. Any variable that's not defined within a function falls into this class. Both Listing 1 and Listing 2 contain examples. Notice the arrays week[] and weeks[].

Unlike local variables that disappear once we're through with them, external variables may be accessed anywhere within your program. The only rule to remember is that, if their declaration appears in another file or after a function that refers to them, they must be declared as external in the function where they're used. Here's a declaration example:

```
extern int numbers;
```

Automatic (or *auto*) variables are those declared within a function. They remain healthy and happy as long as we stay within the function where they were declared. The moment we exit, they vanish into that great CPU in the sky. It's not necessary to declare these variables by their storage class (we never have, right?)—but, if you wanted to, this is what it would look like:

```
auto int number;
```

Variables of the class *static* are similar to automatic variables, except their values aren't forgotten when the function is exited. Don't try to access them in other parts of your program, though. They're still strangers there. Look at this code fragment:

```
main()
{
for (x=0; x<5; ++x)
   counter();
}

counter()
{
static int count=1;
printf("%d ", ++count);
}
```

The output from this example would be:

```
2 3 4 5 6
```

Each time we call counter(), the variable count is incremented and printed out. If we hadn't declared count as a static variable, the output would have been a string of *2*s.

Do you see why? When a static variable is initialized as we did in counter(), it receives its initial value the first time we call the function. Thereafter, the declaration and initialization is ignored. This is only logical, since what good would a static variable be if it was reset each time we called the function?

By not declaring count as static, it automatically becomes automatic (no, I'm not being redundant). Each time we call the function, it gets set to *1*, then it's incremented and printed. This gives us that string of *2*s.

One last note on static variables. An interesting variation of this class can be created by defining it outside any function. This type of variable is called "external static." This class varies from regular ex-

ternal variables, in that it can be accessed only within the file where it appears, and only in functions following its declaration.

The last class we need to discuss are register variables. They're defined like this:

```
register int number;
```

When we declare a register variable, we're requesting that the value be stored in one of the ST's registers where processing is much quicker. Notice I used the word *requesting*. If there's no register free in which to store our variable, it becomes an automatic variable.

Hip, hip array!

We took a brief look at arrays when we wrote our sort program a couple of months ago. Now we're going to dig a little deeper.

First, let's tackle Listing 2. Suppose you're selling a peculiar product called a whamble (a what?) in your small business. At the end of the week, you want to write a quick and dirty program that'll print the number of units sold that week. Listing 2 is just such a program. When you run it, your output should look like this:

```
Sales for day 1: 5
Sales for day 2: 7
Sales for day 3: 2
Sales for day 4: 10
Sales for day 5: 7
Sales for day 6: 1
Sales for day 7: 6

Total sales: 38
```

The first thing we must do in this program is initialize an array. In our sorting program, we didn't worry about that. All we did was declare the array, then fill it, later in the program, with the numbers the user input. Sometimes, though, you'll need to have the array data stored and ready to process at run time. Line 2 shows you how to do this.

To initialize an array as part of its declaration, the array name is followed by an equal sign, which, in turn, is followed by the elements of the array, separated by commas and placed between brackets. Here are more examples:

```
int numbers[] = {1,2,4};
int numbers[3] = {1,2,4};
float numbers[] = {1.1,2.2,4.4};
```

The first is just like the declaration on Line 2, and the second example is, in this case, functionally the same as the first. However, it does present potential difficulties.

For instance, in the first example, the compiler automatically makes the array size the same as the number of values that follow. In the second example, we're telling the compiler that, no matter what, we want a three-element array. Here's a strange one:

```
int numbers[4] = {1,2};
```

What do you suppose happens here? Well, the compiler sets aside an array containing four elements, then looks to see what we've got stuffed between the brackets.

The first value goes into the first element, the second into the second. After that, if it's an external or static array, the remaining elements are initialized to *0*. Otherwise, whatever garbage happens to be in those locations filling out the remainder of the array becomes an authorized resident. Trouble, for sure. Here's another problem maker:

```
int numbers[2] = {1,2,4};
```

There's no way you're going to get away with this. Your compiler is sure to present you with some snide comments on your programming skills—and they'll be well deserved. You can't get three data items into a two-element array.

Continuing with Listing 2, after we've initialized our array, the program uses a for loop to access each element, add it to the total and print it out. Except for a little nuance with the way we've initialized the for loop, you've seen all this before. Just remember that an array starts at element *0*.

Now, how about that nuance I mentioned? Look at Line 6. I hope you remember about for loops. The first expression in the parentheses is the initialization, the second is the loop control, and the third is the loop's step value.

In this example, we've taken the opportunity to initialize not only the loop variable, but the accumulator total as well. This is a handy way to set variables used within a loop to their starting values.

Line 7 offers a new assignment operator for your inspection, one that's quite similar to the increment and decrement operators. Line 7 does the same work as this line of code:

```
TOTAL=TOTAL+WEEK[i]
```

The right side of the expression is added to the left.

Another dimension.

C is also capable of handling multi-dimensional arrays. You can think of these as arrays of arrays. Listing 3 illustrates how to handle them.

The declaration is similar to that of a one-dimensional array, except we've added another set of brackets, to tell the compiler how we would like the array set up.

Look at Line 2. Here we're declaring an array with

two sets of seven elements. You can think of this as a matrix, with two rows and seven columns.

When we initialize the array, each row of data is placed within its own set of braces. The rows, just like the data within, are separated by a comma. Finally, the entire array is enclosed with another set of braces. This tells the compiler how we want each element placed. Take a look at this:

```
int a[2][3] = {
    {1,2},
    {3,4,5}
    }
```

Here, we've declared an array which contains two arrays of three elements each. But wait a minute! In our initialization, we're missing a data element for the first subarray. How's this going to work out? Is the first element of the second row going to end up as the third element in the first?

Nope. The *1* will be placed in the first element of the first row. The *2* will go in the second. The third element of the first row will be initialized to *0*. (Remember that rule about external data?) The second row will be initialized just the way we want it. No mix-ups.

To tell you the truth, you don't need all those extra braces. We could've initialized weeks[][] by placing all the data between one set of brackets, like this:

```
{3,6,7,4,3,8,9,5,3,7,9,3,2,6}
```

The array will still function properly, but it's much harder to see how the data's divided up—and we've left ourselves open for possible errors. If we should accidentally (or deliberately, if you happen to enjoy that sort of thing) leave out one of the data elements, the compiler will no longer sort it out for us, making sure everything gets into its proper location.

It'll assign each element consecutively until it runs out of data, and then initialize the rest to *0*. Your program is then *sure* to act peculiarly. This type of error can be extremely difficult to locate.

Whambles for sale.

Okay, enough talk. Get Listing 3 compiled. A program run will look like this:

```
Sales for day 1: 3
Sales for day 2: 6
Sales for day 3: 7
Sales for day 4: 4
Sales for day 5: 3
Sales for day 6: 8
Sales for day 7: 9

Sales for week 1: 40

Sales for day 1: 5
Sales for day 2: 3
Sales for day 3: 7
Sales for day 4: 9
Sales for day 5: 3
Sales for day 6: 2
Sales for day 7: 6

Sales for week 2: 35

Total sales for month: 75
```

Two weeks. Wow, what a short month. Yes, I know there are usually four weeks in a month. I limited the output, so the data wouldn't scroll off your screen. What a nice guy!

This program is an example of indexing a two-dimensional array. Lines 9 and 10 set up nested for loops. The outer loop handles the indexing of the weeks; the inner loop indexes days.

The day loop is performed seven times for each iteration of the week loop. Line 11 shows how all this relates to our array.

The first subscript refers to each row of data (weeks). The second is the columns, or days. The first time we get to Line 11, w and d both equal *0*. So we're looking at weeks[0][0], that is, the data in row 0 and column 0. If we look at the array initialization, we see that this is the value *3*.

The day's total sales are printed, then the inner loop is repeated, incrementing d and advancing us to the row 0's next element. Looking at the data, we see that weeks[0][1] equals *6*.

This loop repeats until d is no longer less than *7*. At that point, we drop through to Line 14 and print the total for the week, and add to our monthly total.

Returning to the outer loop, w is incremented, and we re-enter the inner loop, resetting d to *0*. Now we're referencing weeks[1][0], row 1 and column 0, or the value 5. The inner loop continues through row 1 just as it did with row 0.

When we return to the outer loop, the value of w is incremented again, and thus is no longer less than *2*. The looping is completed, and program execution continues at Line 18, where the monthly total is printed.

Red and flustered.

That's it for this month. Sit back and relax. Put your feet up, massage your temples to get rid of that thundering headache (arrays are like that; yeah, they are).

Now that we've got all the work out of the way, it's confession time. It seems that a couple of the listings from issue 40's **C-manship** got a bit messed up. I'm still not sure how it happened, but if you were getting strange results, you can place the blame firmly on my shoulders.

The following corrections should be made (this includes those of you with disk subscriptions).

Line 8 of Listing 3 should be:

```
printf ( ">%010d<\n", num );
```

Line 11 of Listing 4 and Line 9 of Listing 6 should be:

```
ch = getchar();
```

Also, a couple of sentences got dropped from the end of page 75. The last part of the paragraph should read:

> In other words, in our program, every place the word *TEXT* appears, the string *Your full name is* will be substituted. Notice that there's no semi-colon at the end of a #define statement. It's a compiler directive and not subject to the semi-colon rule.

Happy trails.

Next month, we'll start developing our own input routines, so we won't be at the mercy of such functions as scanf() and gets(). Till then, fool around a bit with arrays. They're neat little critters.

Listing 1.
C listing.

```
#include <stdio.h>
#include <osbind.h>
#define TRUE 1
#define FALSE 0

main()
{
   int num, guess, win, turns, play;

   play = TRUE;
   while (play) {
      turns = 0; win = FALSE;
      num = get_num();
      while (!win) {
         ++turns;
         guess = get_guess();
         win = check_guess(num, guess);
      }
      printf("It took you %d turns.\n\n", turns);
      play = play_again();
   }
}

int get_num()
{
   int n;

   n = (int) Random();
   n = abs(n) % 99 + 1;
   return(n);
}

int get_guess()
{
   int g;

   g = 0;
```

```
    while (g<1 || g>100) {
       printf("Enter a number from 1 to 100: ");
       scanf("%d", &g);
       printf("\n\n");
    }
    return(g);
}

int check_guess(num, guess)
int num, guess;
{
    int wn=FALSE;

    if (guess < num)
       printf("Too low\n\n");
    else if (guess > num)
       printf("Too high\n\n");
    else {
       printf("You guessed it!\n");
       wn = TRUE;
       }
    return(wn);
}

int play_again()
{
    int ch, p;

    p = -1;
    ch = getchar();
    while ( (p!=TRUE) && (p!=FALSE) ) {
       printf("Play again? ");
       if ( (ch=getchar()) == 'y' || ch == 'Y')
          p = TRUE;
       else if (ch == 'n' || ch == 'N')
          p = FALSE;
    }
    printf("\n\n");
    return(p);
}
```

Listing 2.
C listing.

```
#include <stdio.h>

int week[] = {5,7,2,10,7,1,6};

main()
{
    int i,total,ch;

    for (i=0, total=0; i<7; i++) {
       total += week[i];
       printf("Sales for day %d: %d\n", i+1, week[i]);
       }
    printf("\n");
    printf("Total sales: %d", total);
    ch = getchar();
}
```

Listing 3.
C listing.

```
#include <stdio.h>

int weeks[2][7] = {
   {3,6,7,4,3,8,9},
   {5,3,7,9,3,2,6}
      };

main()
{
    int w,d,mtot,wtot,ch;

    for (w=0, mtot=0; w<2; w++) {
       for (d=0,wtot=0; d<7; d++) {
          wtot += weeks[w][d];
          printf("Sales for day %d: %d\n", d+1, weeks[w][d]);
          }
       printf("\n");
       printf("Sales for week %d: %d\n\n", w+1, wtot);
       mtot += wtot;
       }
    printf("\n\n");
    printf("Total sales for month: %d\n", mtot);
    ch = getchar();
}
```

InSoft, Corp. / InSoft ST NETWORK

For the low annual membership fee of $50, ST Network members can buy Hardware and Software for their ST's at Wholesale + 5%

As a limited time offer to new members, buy hardware and software for your ST at wholesale prices shown for orders placed before 7/15/86. After 7/15/86, add 5% to prices shown when ordering.

CALL TOLL FREE 1-800-556-5580 (orders and memberships) CALL FOR COMPLETE CATALOG
For technical questions about items offered and compatibility, call (617) 739-9012.

ST Software

(Please add $2 for shipping and handling for each software title ordered.)

ACTIVISION		**FIREBIRD**		**MARK OF THE UNICORN**		**MIRAGE**		Spell	$30
Borrowed Time	$29	Starglider	$27	Hex	$ 24	H & D Base	$58	Base	60
Hacker	26	The Pawn	27	Mince	100	H & D Forth	30	**SIERRA ON LINE**	
Mindshadow	29	**FTL**		The Final Word	85	**O.S.S.**		Black Cauldron	24
Music Studio	36	Sundog	24	**MEG**		Disk Kit—Disk Editor	24	King's Quest I	29
ADDISON/WESLEY		**HABA SYSTEMS**		Megafont	24	Personal Pascal	45	King's Quest II	29
Sorcerer of Claymorgue	12	call for latest prices		Rubber Stamp	24	Prolog	54	Ultima II	35
Spiderman	12	**HIPPOPOTAMUS**		Typesetter	24	**OTHER VALLEY SOFTWARE**		Universe II	41
ANTIC		Hippo Backgammon	24	**METACOMCO**		Delta Patrol—RGB	15	**SPINNAKER**	
A-Calc	36	Hippo Jokes and Quotes	21	Assembler	48	Monkey Business—RGB	15	Amazon	30
Lands of Havoc	12	call for latest prices		Lattice C	90	**PENGUIN**		Fahrenheit 451	30
Mom and Me—RGB	21	**INFOCOM**		Meta Pascal	60	Coveted Mirror	24	Nine Princes in Amber	30
Murray and Me—RGB	21	A Mind Forever Voy.	27	**MIC**		Crimson Crown	24	Perry Mason	30
ARTWORX		Bally Hoo	24	Silent Service	24	Do-Tops	24	Treasure Island	24
Bridge 4.0	18	Cutthroats	24	**MICHTRON**		Frank and Ernest	24	**SSI**	
Compubridge	18	Deadline	29	Bulletin Board Service	29	Sword of Kadash	24	Phantasie	24
Hole in One	18	Enchanter	24	Business Tools	30	The Quest	24	**SST**	
Strip Poker	24	Hitchhiker's Guide	24	DOS Shell	24	Transylvania	24	Chat 2.0	13
BECKEMEYER DEVELOPMENT		Infidel	27	Direct File Transfer	30	**PHYLON**		**SYNAPSE**	
TOOLS		Planetfall	24	Flip Side	24	Fast Basic	24	Mindwheel	75
call for latest prices		Seastalker	24	Flight Simulator	24	Henry's Fundamental	24	**TDI**	
BATTERIES INCLUDED		Sorcerer	27	Gold Runner	29	Basic	24	Andra	29
Degas	24	Spellbreaker	29	Kissed	29	**PRIORITY SOFTWARE**	24	Modula 2	48
BAYVIEW		Starcross	29	M-Disk	27	Forbidden Quest	24	UCSD Pascal	42
Word for Word	24	Suspect	29	Mud Pies	29	**PSYGNOSIS**	24	**UNISON**	
DRAGON GROUP		Suspended	24	Mi-Term	30	Bratacus	28	Art Gallery I	18
4 x Forth Level I	62	Wishbringer	24	Personal Money Manager	24	**QMI**		Print Master	24
4 x Forth Level II	93	Witness	24	Softspool	30	ST Talk	11	**VER.**	
ELECTRONIC ARTS		Zork I	24	Time Bandit	24	**QUICKVIEW**		dB Man	62
Chess Master	27	Zork II	27	**MIGRAPH**	24	Zoomracks	48	**VIP**	
Financial Cookbook	30	Zork III	27	Easydraw	27	**REGENT**	90	The Professional	108
						Word	30		

ST Hardware

(Please add shipping and handling charges found in brackets.)

Atari Hardware (call for 1040 ST)		
520 ST, Monochrome, ½ MEG Drive	$620	(40)
520 ST, Monochrome, 1 MEG Drive	700	(40)
520 ST, RGB, ½ MEG Drive	770	(40)
520 ST, RGB, 1 MEG Drive	850	(40)
1 MEG DRIVE	198	(15)
½ MEG DRIVE	160	(15)
Keyboard, mouse, ½ MEG Drive	Call	
Keyboard, mouse, 1 MEG Drive	Call	
Monitor, RGB	400	(25)
Monitor, Monochrome	175	(25)
Hippo EPROM Burner	105	(6)
Supra Modem 1200 ST	159	(10)
Soft Logic Logikhron Clock Radio	30	(3)

PRINTERS

Epson	Call
Toshiba	Call
Okidata	Call
STAR	Call
NEC	Call

DISKS 3½" (10)

	Single Sided	*Dual Sided*
Sony	$15 (3)	$25 (3)
Maxell	15 (3)	25 (3)
Nashua	14 (3)	

InSoft's Magazine on a Disk
12 months—$70
6 months—$45

InSoft's C Tool Boxes
Math Tool Box—$59
Search/Sort Tool Box—$59
Graphic Tool Box—$59

InSoft ST NETWORK
MEMBERSHIP APPLICATION

P.O. Box 180
Boston, MA 02123

1-800-556-5580
1-617-739-9012

Compatibility of ST configuration with software/hardware is the responsibility of buyer

Restocking fee is 15% for merchandise not accepted

Products Ordered	**Cost + 5%**	**S/H**	**Total**
Membership—one year			$50

Mass. residents add 5% sales tax _____

☐ Check or Money Order enclosed for . . . . . $ _____

Name _____
Address _____
City _____
Telephone _____ State _____ Zip _____

CIRCLE #133 ON READER SERVICE CARD

Abacus books

ATARI ST MACHINE LANGUAGE
ATARI ST GEM PROGRAMMER'S REFERENCE
ATARI ST INTERNALS
ABACUS SOFTWARE, INC.
P.O. Box 7211
Grand Rapids, MI 49510
(616) 241-5510
$19.95 each

by Douglas Weir

The Atari ST may have been "made for America," as the ads put it, but there are signs that it has also prompted a German invasion. A Dusseldorf, West Germany company called Data Becker has apparently had teams of experts busy since the ST was first introduced, writing a series of books to cover just about every programming aspect of the machine.

Presenting the Atari ST, the first volume of the series, translated into English and published in the United States by Abacus Software, appeared last fall. At the time, I thought it a bit long on copy and short on hard information. Now, however, it's been joined by three comrades, in the same distinctive field-gray paper covers. One of these is very good, and two are really excellent.

Atari ST Machine Language falls into the "very good" category. The authors advise you to "get a book on the 68000 processor and its instructions" (something like the official Motorola manual) as a companion to this volume, intended only as an introduction to assembly language programming on the ST. The 274-page book has eight chapters, covering the expected topics: binary representation, 68000 architecture, flow of control, data structures, etc.

Beware the discussion of assemblers in chapter 6; the authors have some sort of generic assembler in mind, not the (notorious) "AS68" supplied with the Atari Developer's Kit.

The two final chapters first take you, step-by-step, through the development of a simple decimal-to-binary conversion program; then eight sample ST assembly language programs are presented in chapter 8. Programs are accompanied by detailed descriptions of their operation, and each comes complete with its own keyboard and console I/O routines (all run under TOS—GEM is not discussed in this book).

Essentially, the book is arranged as a read-it-through tutorial, with no exercises. Written in a pleasant style, its occasional mistranslations only serve to make the prose rather endearing.

Despite their reliance on flowcharting as a program development tool, I found the authors' treatment of most topics superior to (for example) that in the well-known Kane, Hawkins and Leventhal book, *68000 Assembly Language Programming*, published by McGraw Hill.

You'll have to pick your way through a fair number of misprints (though most are obvious), and you'll have to put up with having no index (apparently one more result of the famous "No Index" law enacted for computer book publishers several years ago).

Still, I do think you'll find this book very useful if you are just starting assembly language programming on the ST—especially when used with the Motorola manual and your own assembler's documentation.

Programmers now have a handy one-volume GEM guide in the ***Atari ST GEM Programmer's Reference*** (414 pages). Anyone programming with GEM should have a copy of this book. The bulk of its contents—which we find in chapters 3 and 4—cover the VDI and AES functions.

Each function is listed separately, with at least one page to itself, in a format similar to (but more readable than) that used in the Digital Research manuals in the Atari Developer's Kit. There are two short sections, "Sample Programs using the VDI" and "Sample Programs using the AES," containing short C and assembly language programs.

An interesting aspect of the assembly language programs is that they're completely self-contained. All the *contrl* and *intin*, etc. arrays are declared explicitly, and calls are made directly through the entry point in the ST BDOS, rather than by linking to AESBIND and VDIBIND. It's nice to know how to do this, if one has to.

Where appropriate, the authors have included ST-specific information on particular functions. For example, we're told that *vst_load_fonts()* will always return a null, since no additional character sets are now available for the ST.

There are, however, cases where we get less information than the DRI manuals offer: for example, pages 5-19 of the VDI manual go into quite a bit of detail on *vst_point()* (a function to set the current graphic text character height), while the Abacus entry on page 133 confines itself to a few vague sentences.

Chapter 1 is a fairly cursory overview of GEM on the ST (I'm starting to think that GEM/TOS on the ST could be a Japanese art film, where everyone's account of the same subject is unrecognizably different from everyone else's). Chapter 2 includes short introductions to programming in C and assembly language. Also in chapter 2 is a description of the main parts of the Atari Developer's Kit, with some helpful tips on using the C compiler, assembler and linker. As seems usual with these Abacus books, there are misprints, but most are obvious, and none of those I noticed would lead to serious misunderstanding.

In the appendices, you'll find a list of VDI and AES functions, as well as a list of the 68000 instruction set. There's also (surprise!) an index, but the authors have managed to maintain partial compliance with the above-mentioned law by *not* putting the VDI and AES function lists in alphabetical order.

All joking aside, I found this a very useful book, one I think no serious GEM programmer should do without.

Finally, with 446 pages, ***Atari ST Internals*** is the biggest and the best of the current lot. This book is filled with indispensable information.

You might also want to acquire a copy of the most recent version of the Atari *Hitchhiker's Guide to the Bios*, included in the Developer's Kit. The authors of the Abacus book seem to have used an earlier version of the *Guide*, so that (for the most part) anything discussed beyond page 43 of the August 26 version of the *Guide* is not covered here. This includes such things as Cartridge Support, ROM Initialization, Boot Sectors, etc., which a lot of programmers can probably do without.

The book has two main divisions, the first covering hardware aspects of the ST; the second, software. There's a short section on the 68000, 7 pages on the four custom chips, and sections on the floppy disk controller chip, 68901 multifunction chip, 6850 ACIA chips and the sound chip. The latter sections consist of short introductions and fairly detailed descriptions of pinouts and chip architecture. Addresses and descriptions of all the ST's I/O registers are given.

Next comes an 18-page section on the keyboard interface, with descriptions of the mouse and the keyboard processor commands, and a chart of ST key codes. A 2-page assembly language program, which the authors used to read the 6301 (keyboard processor) ROM and output it to a printer, is included. It provides, incidentally, an example of how to use some of the GEMDOS and XBIOS calls.

Shorter sections on the video connection, the Centronics interface, the RS-232 and MIDI interface, the cartridge slot, and the floppy disk and DMA interface follow. These are all hardware descriptions and consist, for the most part, of pinout discussions. All in all, the hardware section of the book occupies the first 100 pages.

Software aspects take up the rest of the book. GEMDOS, BIOS and XBIOS calls receive detailed treatment on pages 105-205. The GEMDOS functions are the TOS "system-level" calls—used, for example, to access the disk, get keyboard input, etc. Many of them resemble the IBM PC MS-DOS system calls.

Each function is described, and most are accompanied by assembly language examples to show how they're used. The ST BIOS and XBIOS functions are allotted a page each. Both have their own explanatory text and assembly code samples. (I noticed an important misprint on page 199: XBIOS function number 34—*kbdvbase*—returns its pointer in *d0*, not *a0*.)

There is a good discussion of the "line a" codes on pages 206-233. Pages 234-254 deal with the ST's exception vectors, the VT52 emulator and the ST system variables. There is (yet another) summary of the 68000 instruction set on pages 255-267.

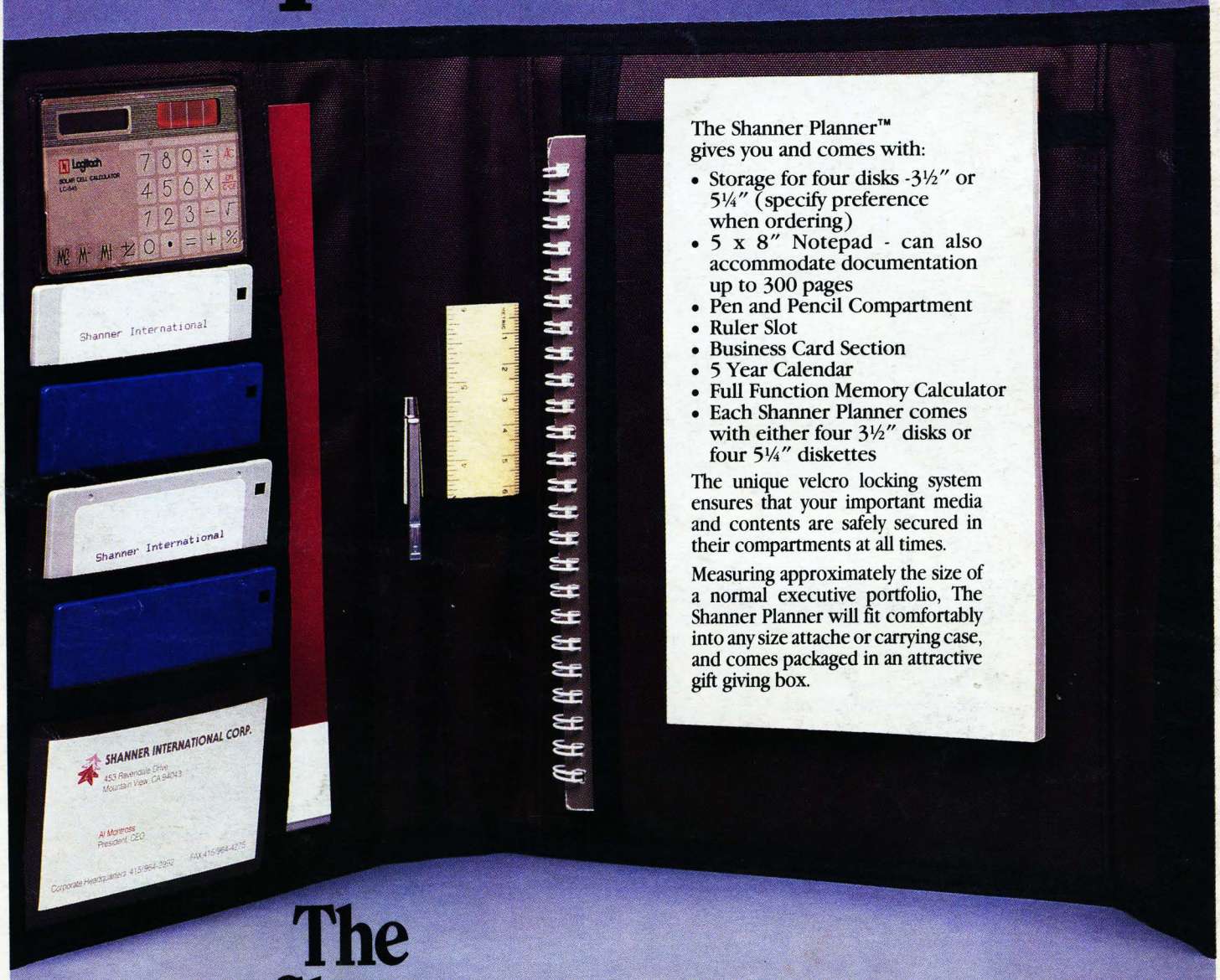
Finally, on pages 268-442, you'll find a complete (fully commented) assembly-language source code listing of the ST BIOS. This includes the code for all the GEMDOS, BIOS and XBIOS calls, the 68901, keyboard and VT52 routines, and the screen dump.

Even if your BIOS version is different from that given here (you can check, by comparing the date-of-creation bytes close to the beginning of the listing with yours), you will certainly find this section extremely useful, if you're at all interested in the ST's internal workings.

I highly recommend this book. But I feel bound to mention that—you guessed it—there's no index, and the wealth of information presented can make this a handicap in the early stages of using the book.

ST INDEX TO ADVERTISERS

It speaks for itself.



The Shanner Planner™ gives you and comes with:

- Storage for four disks -3½" or 5¼" (specify preference when ordering)
- 5 x 8" Notepad - can also accommodate documentation up to 300 pages
- Pen and Pencil Compartment
- Ruler Slot
- Business Card Section
- 5 Year Calendar
- Full Function Memory Calculator
- Each Shanner Planner comes with either four 3½" disks or four 5¼" diskettes

The unique velcro locking system ensures that your important media and contents are safely secured in their compartments at all times.

Measuring approximately the size of a normal executive portfolio, The Shanner Planner will fit comfortably into any size attache or carrying case, and comes packaged in an attractive gift giving box.

The Shanner Planner™

$39.95

Dealer Inquiries Accepted
Call 800/828-6637

Remember the expression "A picture is worth a thousand words?" Take a second look. The Shanner Planner™ is the first ever full system portfolio created specifically for the computer user. Designed to address the individual needs of the computer industry, The Shanner Planner is tastefully constructed of long lasting, durable, textured nylon *(it's the in thing today)* to maintain its smart appearance over time.

SHANNER INTERNATIONAL CORP.

The Shanner Planner™ is a creation of, marketed worldwide by, and is a Trademark of SHANNER INTERNATIONAL CORP.

Please send me _____ Shanner Planner(s).

I am enclosing $ _____ .
(CA residents add $2.60 tax per Planner)
Please make check or money order payable to:
Shanner International Corp.
453 Ravendale Drive
Mountain View, CA 94030

Canadian Orders:
Canadian cost per Planner is $49.95. (Ontario residents add $3.50 PST per Planner) Please direct orders to:
Shanner Sales & Marketing, Inc.
450 Garyray Drive
Weston, Ontario

Name _____

Address _____

City _____ State _____ Zip _____

For all orders: Please add $3.00 per Planner for shipping and handling. C.O.D.'s and purchase orders will not be accepted. Please allow 21 days for delivery.

CIRCLE #131 ON READER SERVICE CARD