# FORTH Dimensions

Volume 5, Number 4 November/December 1983 \$2.50

# Multi-Tasking



#### FEATURES

So Many Variables	Michael Ham5
Yet Another Number Utility	David McKibbin7
Manufacturing Cost Program	Marc Perkel9
Menu-Driven Software	John Bowling10
Vocabulary Tutorial, Part II	Evan Rosen14
Forth Froth	Wil Baden16
Vectored Execution and Recursion	Roy W. Sommers17
Apple Forth a la Modem	R. D. Ackerman19
Forth-83 Loop Structure	Bill Stoddart22

#### DEPARTMENTS

Letters	3
Editorial: Fifth Forth Fest	3
Techniques Tutorial:	
Multi-Tasking, Part I	Henry Laxen26
New Product Announcements	
FIG Chapter News	John D. Hall31

INVF-FORTH Cross Computer to Public Common - Tools Volume 1, and the Computer Common - Tools Volume 1, and the Computer Common - Tools Volume 1, and the Common - Tools Volume 1, and the Commo	THE FOR	
State:         Image definition control for the state of the sta	MVP-FORTH	MVP-FORTH Cross Compiler for CP/M Programmer's Kit.     Can also generate beaderless code for POM or torget
The region rule is any other. This public domain order is a subset of	Stable - Transportable - Public Domain - Tools	CPU \$300
audety to a varety of computer. MPH-OPTH gives you both these relatives and analysis of the second or many other diverses public.         \$15           bit mask of a varety of computer. Methods with an edity of the diverse of many other diverses. Define the diverse of many other diverses public.         \$15           bit mask of a varety of computer. Methods with an edity of the diverse of many other diverses. Define the diverse of many other diverses of the MPF-ORTH provides with an edity of the diverse of many other. Methods with an edity of the diverse of the MPF-ORTH provides with an edity of the diverse of the MPF-ORTH provides with an edit of the diverse of the d	stable operating system and the ability to move programs easily and	MVP-FORTH Meta Compiler for CP/M Programmer's kit. Use
<ul> <li>Hours P, Arabins Barnbard, Josef Marker Marke</li></ul>	quickly to a variety of computers. MVP-FORTH gives you both these features and many extras. This public domain product includes an editor,	for applicators on CP/M based computer. Includes public domain source \$150
FORT Hit for a "under of computing"     Other MAP-CRTH (pockurds with market the about FORT Ho Standard, 2PC at pockast about FORT Ho FORT Ho FORT Ho pockast about FORT Ho pochabout FO	book "Starting FORTH". The Programmer's Kit provides a complete	MVP-FORTH Fast Floating Point Includes 9511 math chip on
HVP Botts - A Series     NUR-FORTH Signal and FORTH Signal Activation by decompliance with and FORTH Assembly South of the Second South	FORTH for a number of computers. Other MVP-FORTH products will simplify the development of your applications.	board with disks, documentation and enhanced virtual MVP-FORTH for Apple it and lie \$450
□ Wulme 1, <i>M</i> about POINT by Heydon. MVP-FORTH         S55           □ Wulme 1, <i>M</i> about POINT by Heydon. MVP-FORTH         S55           □ Wulme 1, <i>M</i> about POINT by Heydon. MVP-FORTH         S55           □ Wulme 1, <i>M</i> about POINT by Heydon. MVP-FORTH         S55           □ Wulme 1, <i>M</i> about POINT by Heydon. MVP-FORTH         S55           □ Wulme 1, <i>L</i> about POINT by Heydon. MVP-FORTH         S55           □ Wulme 1, <i>L</i> about POINT by Heydon. MVP-FORTH         S55           □ Wulme 1, <i>L</i> about POINT by Heydon. MVP-FORTH         S55           □ Wulme 1, <i>L</i> about POINT by Heydon. MVP-FORTH         S55           □ WUP-FORTH States Term Manual Call Company         MVP-FORTH States Term Manual Call Company           □ MVP-FORTH Hy Heydon. MVP-FORTH         S55           □ WVP-FORTH Hy Heydon. MVP-FORTH         S55           □ MVP-FORTH Hy Manual Call Company         MVP-FORTH Hy Heydon. MVP-FORTH           □ MVP-FORTH Hy MES (Call Company         MVP-FORTH Hy MES (Call Company           □ MVP-FORTH Hy MES (Call Company         MVP-FORTH Hy MES (Call Company           □ MVP-FORTH Hy MES (Call Company         MVP-FORTH Hy MES (Call Company           □ MVP-FORTH Hy MES (Call Company         MVP-FORTH Hy MES (Call Company           □ MVP-FORTH Hy MES (Call Company         MVP-FORTH Hy MES (Call Company           □ MVP-FORTH Hy MES (Call Company         MVP-	MVP Books - A Series	MVD.FORTH Programming Aide for OP/M IPM or ADDI 5
■ Vol. III V3 Sublack 2 + Color         Submark 2 + Color - Submark 2	Volume 1, All about FORTH by Haydon. MVP-FORTH glossary with cross references to fig-FORTH, Starting FORTH SORTH 20 Overland and FORTH, Starting FORTH	<ul> <li>www.rconterrogrammer's Kit. Extremely useful tool for decompiling, califinding, and translating.</li> <li>\$150</li> </ul>
Volume 3. Reating Point Glossay by Springer         510           Outsme 4. Report System vith source code by Park         525           With 5. Reating Point Glossay by Springer         510           Outsme 4. Expert System vith source code by Park         525           MP-FORTH Forth Horsmanner K II including disk, document         555           MP-FORTH Horsmanner K II including disk, document         555           MP-FORTH Horsmanner K II including disk, document         555           FORTH Michael System Code and Stringer         510           Computer, Comment C, Marcobeciaons, II Konger, Computer System Code velopment of knowledge based programs for Apple. Ellow Cor PAN.         5200           FORTH With adiron, assembler, and manual.         APPLE by MM         510         5200           APPLE by MM         510         520         520           FORTH With adiron, assembler, and manual.         APPLE by MM         510         520           APPLE by MM         510         520         525           FORTH With adiron, assembler, and manual.         520         521         525           FORTH With addres, assembler, and manual.         525         525         525           FORTH Michael Store Manual Point, Brock Store Manual Point, Brock Store Point, B	and FORTH - 79 Standard. 2 <sup>™</sup> Ed.     ≥0       □ Volume 2, MVP-FORTH Assembly Source Code. Includes       CP/M®     IBM-PC®       Appl F®     listing for kernel       \$20	MVP-FORTH by ECS for IBM-PC or ATARI® . Standalone with     action ditor. License required.
• Marriele Management System with internued security by the mean of the Management System with internued security by the Management System with internued security by the Market Security History Market Management System (Market Market Markt Market Markt Market Markt Market Market Market Market Market	<b>Volume 3,</b> Floating Point Glossary by Springer \$10	MVP-FORTH by ECS for IBM-PC or ATARI. With color
Moreten         S25         MVP-FORTH Programme's Lystem         MVP-FORTH Programme's System           MVP-FORTH Programme's K1 including disk, documentation, with complete documentation. Complete system S400         MADE VERTH Expect System I or development of knowledge. Complete documentation. Complete System (FMS) with interrupt security CPM. BC, CEAN. B	Volume 5, File Management System with interrupt security by	animation, multitasking sound, utilities, and license. \$175
MVP-FORTH Software - A Transportable FORTH       (PADS) for IBM-PC, or APPLE with complete documentation. Complexe system 5:3400         MVP-FORTH Programmer's Kill including disk, documentation. Values 1:8:2 of MVP-FORTH System 1or development of knowledge.       MVP-FORTH Expert System 1or development of knowledge.         FORTH With editor, assembler, and manual.       MVP-FORTH DSKS       MVP-FORTH With editor, assembler, and manual.       MVP-FORTH MANUALS, GUIDES & DOCUMENTS       S200         FORTH With editor, assembler, and manual.       APPLE by MM.       S100       NOVA by CCI 8" DSIOD5100       S200       S200         Compupro.       Compupro.       Computor.       Nova by CCI 8" DSIOD5100       S200       S200       S200         FORTH With Editor, assembler, and manual.       APPLE by MM.       S00       S00       COMPUT Molex.       S200         Compupro.       Compupro.       Computor.       S00       S00       Computor.       S00       S00       S00         Computor.       S00       S00       Computor.       S00	Moreton \$25	MVP-FORTH Professional Application Development System
Horner 1.4.2 of MVP-FORTH Series (All About FORTH, Specify _ Group M_ CPM B_ CPM B_ CPM + _ IA PP LE Computer, _ CPM B_ CPM B_ CPM + _ IA PP LE Computer, _ CPM B_ CPM B_ CPM B_ CPM + _ IA PP LE Computer, _ CPM B_	MVP-FURTH Software – A Transportable FORTH  MVP-FORTH Programmer's Kit including disk, documen-	(PADS) for IBM-PC, or APPLE. A three level integrated system with complete docurrentation. Complete system \$400
EIM PC:              INFEC:              MOVE-PORTH File Management System (FMS) with interrupt security for IBM. Victor 9000, or CPM.              S20                 FORTH MEXTS               S150               FORTH MALLS, GUDES & DOCIMENTS               S20                 FORTH with clitts, assembler, and manual.               APPLE by MM             S100               S00               S00             S00	tation, Volumes 1 & 2 of MVP-FORTH Series ( <i>All About</i> FORTH, 2 <sup>nd</sup> Ed. & Assembly Source Code), and Starting FORTH, Specify □ CP/M □ CP/M 86 □ CP/M + □ APPI F	NC <sup>4</sup> MVP-FORTH Expert System for development of knowledge- based programs for Apple, IBM, or CP/M. \$80
Computo         Computo         Stol         Security for IBM. Victor Bood, or CPM         Stol           FORTH with clitor, assembler, and manual.	□ IBM PC, □ MS-DOS, □ Osborne, □ Kaypro, □ H89/Z89, □ Z100, □ TI-PC, □ MicroDecisions, □ Northstar	ー MVP-FORTH File Management System (FMS) with interrupt
FORTH DISKS         FORTH With editor, assembler, and manual.         APPLE by MM       S100       NOVA by CCI 8" DS/DS150         APPLE by MM       S100       Isso of DS/DS150         APPLE by Kuntze       S90       Z80 by LM         BH P-55 by Lange       S90       Gardinge       S50         BM PC2 by LM       S100       C64 by HS Commodore 64         BM PC2 by LM       S100       C64 by HS Commodore 64         BM PC2 by LM       S100       C64 by HS Commodore 64         BM PC3 by Lange       S100       C64 by HS Commodore 64         BM PC3 by LAM       S100       C64 by HS Commodore 64         BM PC3 by Lange       S100       C64 by HS Commodore 64         CPM North with FORTH - 27 by Cassady       S100       C64 by HS Commodore 61         CPM Hy MM F & 79       S100       S00       C64 by HS Commodore 61         CPM Hy MM F & 79       S140       BM Z80, or 8086       S100         CPM Hy MM F & 79       S140       S00 regarmics       S120         CPM Hy MM F & 79       S140       C64 by HS Commodore 61       S100         CPM Hy MM F & 79       S140       S150       Comoral and the 10 by Commodore 61         CPM Hy MM F & 79       S140       C	Compupro, Cromenco \$150	security for IBM, Victor 9000, or CP/M \$200
Image: Point with editor, assembler, and manual: <ul> <li>Image: Point Point</li></ul>	FORTH DISKS	FORTH MANUALS, GUIDES & DOCUMENTS
_ APPLE by MM       \$100       NOVA by CCI 8" OSHODSTUD         _ APPLE by MM       \$100       NOVA by CCI 8" OSHODSTUD       They construction of the sector of the	FORTH with editor, assembler, and manual.	
Arr Le by Munites           acc           Book 36 by Ling           Sol             Arr Le by Munites           Sol               Sol           Sol           Sol           Sol           Sol           Sol           Sol           Sol           Sol           Sol                                 Sol           Sol	APPLE by MM \$100 NOVA by CCI 8" DS/DD\$150	LI ALL ABOUT FORTH by LI 1980 FORML Proc. \$25 Havdon, See above. \$25 I 1981 FORML Proc. 2 Vol \$40
□ CPMP by MM       \$100       ∪ CF ORTH by HES, VIC20       □ Standard CA by HES Commodore 64         □ BM-PC* by LM       \$100       _ CF ORTH by IMS       □ CF ORTH by IMS, ISO IC CA by HES Commodore 64       □ Standard CA by HES Commodore 64       □ Standard CA by HES Commodore 64         □ Standard CA base       MAM       □ CF ORTH by IMS, ISO IC CA by HES Commodore 64       □ Standard CA by HES Commodore 64       □ S		□ FORTH Encyclopedia by □ 1981 FORML Proc. \$25
□ PHP-85 by Lange       \$30       □ Cartridge       \$50       □ Cartridge       \$500       □ Date Date Date       \$510       □ Date Date       \$510	□ CP/M <sup>®</sup> by MM \$100 □ VIC FORTH by HES VIC20	Derick & Baker.   1981 Rochester FORTH
PAPE PS by Cassady S150 C64 by HES Commodore 64 BM+PC* by LM S100 C64 by HES Commodore 64 BM+PC* by LM S100 C64 by HES Commodore 64 S16 Cm PC* D1 Line 1982 Rochester FORTH pc. S25 PM Cassady S150 C64 by HES Commodore 64 S16 Cm PC* D1 Line 1982 Rochester FORTH pc. S25 PM Cassady S150 Cm PC* D0 PM Cassady S150 Cm PC* D1 PM PM Cassady S150 Cm PC* D1 PM PM Cassady S150 Cm PC* D1 PM PM PC* D1 PM Cassady S150 Cm PC* D1 PM Cassady S15	□ <b>HP-85</b> by Lange \$90 cartridge \$50	Programmer's manual to fig- EORTH with EORTH 79
IBM-PC* by LM       \$100       \$60       Proc.       \$25         Enhanced FORTH with: F-Floating Point.       G-Graphics, T-Tutorial,       Point       \$25         AFPCHTH-73.       Image: Second Alone, M-Mathic Tasking, X-Other Strasting, Y-Orth-79       A FORTH Fundamentals, Vol.       \$25         AFPCHTH-79.       Extensions for LM Specify       FORTH Fundamentals, Vol.       Threaded Interpretive Languages       \$23         AFPCHTH-79.       Software Floating       Point       \$100       Point       \$25         APple, GraFORTH by SL,       Point       \$100       Point       \$25         Moltinashing FORTH by SL,       \$100       Point       \$25         Mapple, GraFORTH by SL,       \$251       MEAFORTH by SL,       \$25         POINT Purdamentals, Vol.       Point       \$25       FORTH Encyclopedia       \$27         Mapple, GraFORTH by SL,       \$25       \$25       FORTH Encyclopedia       \$27       Point Horgramming Valis for decompiling, Califinding, and translating, CAM, BM-PC, 280, or Apple       \$28       FORTH Porgramming Valis for decompiling, califinding, and translating, CPM, BM-PC, 280, or Apple       \$28       FORTH Porgramming Valis for decompiling, for Site of Apple IMH Porgramming Valis for decompiling, for Site of Apple IMH Porgramming Valis for decompiling, for Site of Apple IMH Porgramming Valis for decompiling, for Site Site of Apple IMH Porgramming Valis Site O	N HP-75 by Cassady \$150 C64 by HES Commodore 64	references. Flow charted, 2 <sup>nd</sup>
Enhanced FORTH       with:       F-Reading PORT, MT-Multi-Tasking, X-Other Surfas, 79-FORTH-79.       Understanding FORTH by MC       Understanding FORTH by MC       Understanding FORTH by MC       Uses Accurate Stras, 79-FORTH-79.         APPLE by MM, C, G, X79       Extensions for LM Specify IBM, Z80, or 8066       Extensions for LM Specify IBM, Z80, or 8066       FORTH Fundamentals, Vol. Iby McCabe       Iby McCabe       S13         CPM M by MM, FA 79       S140       B087 Support       IBM, Z80, or 8066       Iby McCabe       S13         CPM M, & & 79       S33       Software Floating       Point       S100       B087 Support       S33         CPM M, & & A79       S33       Color Graphics       Color Graphics       S100       Chillian       S10         CPM M, & & A79       S45       Management       S200       S26       FORTH of Languages       S10         Specia and memory savings, can also produce ROMable code + Requires FORTH disk.       S300       BMA       S300       S30       S10         Sector MMELENS Allow extending, notrinstarf       S300       BMA       S300	<sup>₩</sup> IBM-PC <sup>®</sup> by LM \$100 N <sup>E<sup>N</sup> cartridge</sup> \$60	Ed. \$25 Proc. \$25
S-Stand Alone, M-Math Chip Support, MT-Multi-Tasking, X-Other       Heymann       33       A FORTH Primer       525         Extras, 79-FORTH-79.       Extensions for LM Specify       F, G, & X. S90       Threaded Interpretive       Languages       523         APPLE by MM,       Extensions for LM Specify       BM. Z80, or 8086       10       I by McCabe       \$13       I anguages       \$23         APPLE by MM,       F, G, & X. S90       Software Floating       Noth Mither Action       \$10       METAFORTH by       Cassady       \$30       Extensions for LM Specify       I by McCabe       \$13       I anguages       \$23         CPIM by MM, F & 79       \$140       Bog Support       Software Floating       \$10       Invitation to FORTH by       Cassady       \$30       Cassady       \$30       Systems Guide to fig-FORTH by       Color Graphics       \$10       Invitation to FORTH \$20       Systems Guide to fig-FORTH by       Software Floating       \$10       FORTH Programming J dis for decompling, califinding, and franslating. CPM, IBM-PC, 280, or Apple       \$10       FORTH Programming J dis for decompling, califinding, and translating. CPM, IBM-PC, 280, or Apple       \$10       Software \$10       FORTH-79 Standard       \$10       Timpeaci fig-FORTH \$10       Software \$10       Noth Arge \$15       Noth Arge \$15       Noth Arge \$10       Noth arge \$15       Noth Arge \$15	Enhanced FORTH with: F-Floating Point, G-Graphics, T-Tutorial	NEW Dunderstanding FORTH by NET 1983 Rochester FORTH
APPLE by MN,       Extensions for LM Specify         F, G, & 79       \$140         ATARI by PNS, F, G, & X, S90       Software Floating         Point       \$100         CP/M by MM, F, 879       \$140         Bogs routing       Bogs routing         Point       \$100         MUIT-Tasking FORTH by I.       \$177         Multi-Tasking FORTH by SL,       9511 Support         (BM-PC or 8066)       \$100         CP/M, X & 79       \$195         Times by FD, tape G, X,       magement         A 79       \$45         Management       \$200         Requires LM FORTH Hogs and memory savings, can also produce ROMable code, *Requires       \$17         PORTH Comparison       \$170         PORTH Programming Adds for decompling, califinding, and finalizable. (soft cover)       \$18         and translating. CP/M, IBM-PC, Z80, or Apple       \$150         Bogs compling, califinding, and finalizable.       \$100         Bogs compling, califinding, and finalizable.       \$100         Bog compling, califinding, and finalizable.       \$100         Bog compling, califinding, and finalizable.       \$100         Bog compling, califinding, and finalizable.       \$100         Starting FORTH by Brodie.       \$100	S-Stand Alone, M-Math Chip Support, MT-Multi-Tasking, X-Other	AND FORTH Fundamentals Vol A FORTH Primer \$25
F. G. 8, 79       \$140       IBM, 280, or 6086       \$23         I DATARI by PNS, F.G. 8 X, S30       Bottware Floating       Bottware Floating       Iby McCabe       \$30         I DATARI by PNS, F.G. 8 X, S30       Bottware Floating       Bottware Floating       Iby McCabe       S30         I DATARI by PNS, F.G. 8 X, S30       Bottware Floating       Bottware Floating       Iby McCabe       S30         I DATARI by MNS, F.G. 8 X, S30       Bottware Floating       Bottware Floating       FORTH Encyclopedia       FORTH Horespteadia         I Res-801 or III by MMS       Color Graphics       Color Graphics       Imitation to FORTH by 200       Imitation to FORTH as 200         I TIMS by FD, tape G.X.       A 79       \$130       Data Base       Scanlon       \$17       FORTH Horespteadia       \$150         I fig-FORTH Programming Alds for decompiling, califinding, and translating. CP/M. IBM-PC. 280. or Apple       \$150       Starting FORTH by Brodie.       \$8       Timy Pascal IIg-FORTH \$10         Bo86+       \$300       Z80       S300       S300       S00       Bost Hilles       \$251       NOVA by CCI User's Manual includes editor, assembler.       S256       NOVA by CCI User's Manual includes editor, assembler.       S256       NOVA by CCI User's Manual includes editor, assembler.       S257         FORTH COMPUTER       S300	APPLE by MM.     APPLE by MM.     Constructions for LM Specify	Ner by McCabe \$16 Threaded Interpretive
ATARI by PNS, F.G. &X. \$290       ⇒ ostiware Hoaing       Point       \$100       Cased y       \$30       Cased y       \$30         Apple, GraFORTH by IST.       Boginning FORTH by IST.       Boginning FORTH by IST.       Systems Guide to fig-FORTH by IST.         CP/M, X & 79       \$330       Cased y       \$30         TRS-801 or II by MMS       Color Graphics       FORTH by Hoang, A       PDP-11 User Man.       \$20         F.X. & 79       \$130       Caller of Cased with a stating of PORTH by Hoang, A       PDP-11 User Man.       \$20         Monters by FD, tape G, X,       Management       \$200       FORTH Pogramming by       College level text.       \$25, where FORTH Py Hoang, A       PDP-11 User Man.       \$20         Monters by FD, tape G, X,       Management       \$200       FORTH Pogramming by       Conversion       \$10         Speed and memory savings, can also produce ROMable code + Requires       Stating FORTH by Brodie.       NOVA ig-FORTH by CCI       Source Listing \$15       NOVA by CCI User's Manual by Vickers       \$25         Goode a state or onduce ROMable code - Requires       \$10       Source Listing \$15       NOVA by CCI User's Manual by Vickers       \$25         Monters by FD, FORTH Area       \$300       Apple II/II+ \$300       Source Listing \$15       Source Listing \$25       Uniters & Manual by Vickers <t< td=""><td>F, G, &amp; 79 \$140 IBM, Z80, or 8086</td><td>FORTH Fundamentals, Vol. Languages \$23</td></t<>	F, G, & 79 \$140 IBM, Z80, or 8086	FORTH Fundamentals, Vol. Languages \$23
L CPIM Dy MM, F & 79       3140       B 807 Support       Chirlian       \$17       Systems Guide to fig- FORTH         A pole, GreFORTH by IS, CPIM, X & 79       3395       (BM-PC or 8086)       \$100         B 507       B	□ ATARI by PNS, F,G, & X. \$90 □ Software Floating Point \$100	II by McCabe \$13 Limetaron in by Cassady \$30
Multi-Tasking PORTH by SL, CP/M, X& 79       (IBM-PC or 8066)       \$100       FORTH Encyclopedia       FORTH       \$220         CP/M, X& 79       \$395       (Z80 or 8086)       \$100       Initiation to FORTH by SL (Z80 or 8086)       \$100         FX. & 79       \$130       (IBM-PC)       \$100       PDP-11 User Man.       \$20         FX. & 79       \$130       (IBM-PC)       \$100       PORTH by Huang. A       PDP-11 User Man.       \$20         FX. & 79       \$130       (IBM-PC)       \$100       PORTH by Klang. A       PDP-11 User Man.       \$20         FX. & 79       \$100       Data Base       \$200       PORTH by Klang. A       PDP-11 User Man.       \$20         FX. & 79       \$100       Data Base       \$200       PORTH by Klang. A       PDP-11 User Man.       \$20         FX. & 79       \$100       Data Base       \$200       PORTH by Klang. A       PDF-11 User Man.       \$20         FX. & 79       \$100       Data Base       \$200       PORTH by Klang. A       PDF-11 User Man.       \$20         FX. & 79       \$100       Data Base       \$200       FORTH by Klang. A       PDF-11 User Man.       \$20         FX. & 79       \$300       Stating FORTH by Klang. A       PDFORTH by Klang. A       PDFO		Chirlian \$17 Systems Guide to fig-
CP/M, X & 79       S100       13011 or ubyoft       S100       13011 or ubyoft       S200         F, X, & 79       S130       Color Graphics       Color Graphics       Color Graphics       Color Graphics       Color Graphics       S200         Min Magement       S200       S201       Color Graphics       Color Graphics       S200         Min Magement       S200       Sandard       Conversion       S110         F, X, & 79       S45       Requires LM FORTH disk.       Color Graphics       FORTH-79 Standard       S15         Grad translating. CP/M, IBM-PC, Z80, or Apple       S150       Starting FORTH by Brodie.       Source Listing       NOVA tig-FORTH by CCI         Speed and memory savings, can also produce ROMable code.       Peote 1st.       S225       NOVA tig-FORTH, by CCI         Speed and memory savings, can also produce ROMable code.       Peote 1st.       S225       NOVA tig-FORTH, by CCI         Speed and memory savings, can also produce ROMable code.       Save	$\square Multi-Tasking FORTH by S \square 0511 Support$	☐ FORTH Encyclopedia FORTH \$25
□ TRS-80/10r III by MMS       □ Color Graphics	CP/M, X & 79 \$395 (Z80 or 8086) \$100	Pocket Guide \$7 Invitation to FORTH \$20
F, X, & 79       \$130       (IBM-PC)       \$100         Immex by FD, tape G, X,       Data Base       Management       \$200       FORTH Programming by       FORTH-79 Standard       \$15         \$79       \$45       Management       \$200       FORTH Programming by       Scale       FORTH-79 Standard       \$15         \$166       FORTH Programming Alds for decompiling, califinding, and translating. CP/M. IBM-PC, Z80 or Apple       \$150       FORTH of the ATARI by E.       Forth met ATARI by E.       Forth met ATARI by E.       Forth Programming by       Conversion       \$10         CROSS COMPILERS Allow extending, modifying and compiling for speed and memory savings, can also produce ROMable code. • Requires FORTH disk.       Starting FORTH hy Brodie.       Source Listing       \$15         \$200       \$300       IBM •       \$300       Apple II/II +       \$300       Apple II/II +       \$300       Starting FORTH, for specific CPU ser's Manual includes editor, asembler, and utilities       \$22         \$200       \$306 •       \$300       Apple II/II +       \$300       Apple II/II +       \$300       Starting FORTH, for specific CPU ser's Manual includes editor, asembler, and utilities       \$22       Supplet ACE Manual by Vickers       \$15         \$200       FORTH disk.       \$25       Starting of fig-FORTH, for specific CPU seand computers. The Installation Manual is required for	TRS-80/I or III by MMS Color Graphics	□ And So FORTH by Huang. A □ PDP-11 User Man. \$20 college level text \$25 cm EOPTH \$2 Standard \$15
Maragement       \$200         Requires LM FORTH disk.       Scanlon       \$17       □ FORTH-79 Standard         Crowsrsion       fig-FORTH Programming Alds for decompiling, califinding, and translating. CP/M, IBM-PC, Z80, or Apple       \$150         Scanlon       \$17       □ FORTH-79 Standard         Crowsrsion       \$10         Installation       CPORTH Programming Alds for decompiling, califinding, and translating. CP/M, IBM-PC, Z80, or Apple       \$150         Speed and memory savings, can also produce ROMable code.       • Requires         FORTH disk.       Sono       280         Cover)       \$23       NOVA big-FORTH by CCI source Listing         Source Listing FORTH (hard cover)       \$18         Source Listing FORTH (hard cover)       \$23         Notristar®       \$300       280 * \$300         Notristar®       \$300       Apple II/II + \$300         Hoft Computer       \$25         Jupiter ACe       \$150         Statting FORTH, for specific CPU's and computers       \$15         Cl Caastone Computing Inc.       MM MicroMotion         DE Date Systems       MS Nauliuk Systems         FD Forth Dimension       NS Nauliuk Systems         FD Forth Dimension       NS Nauliuk Systems         FD Forth Dimension	F, X, & 79 \$130 (IBM-PC) \$100	Conlege level text. 923 20 Contines Standard \$15
Wictor 9000 by DE,G,X \$150       Requires LM FORTH disk.       □       FORTH on the ATARI by E.       Conversion       \$10         □       figf-ORTH Programming Alds for decompiling, callfinding, and translating. CP/M, IBM-PC, Z80, or Apple       \$150       □       Thry Pascal fig-FORTH \$10         CROSS COMPILERS Allow extending, modifying and compiling for speed and memory savings, can also produce ROMable code. • Requires FORTH disk.       □       ONVA fig-FORTH by CCI       □       NOVA fig-FORTH by CCI         □       CP/M       \$300       □       BM*       \$300       ≥ Starting FORTH with axaliable. (soft cover)       \$16         □       00866 •       \$300       ⊇ Z80 •       \$300       ≥ Z80 •       \$250         □       Most RAM Pack       \$150       Starting FORTH, with assembler       \$25       □       Jupiter ACE Manual by Vickers       \$15         □       Jupiter ACE       \$150       \$150       \$25       □       Installation Manual for fig-FORTH, for specific CPU's and computers. The Installation Manual is required for implementation.       Each \$15         □       B080       0806 0       8080       000       AlphaMicro       Each \$15         □       MM MicroMotion       MM MicroMotion       MM MicroMotion       B020       Each \$15       B080       AlphaMicro       Each \$15	K <sup>N</sup> □ Timex by FD, tape G,X, □ Data base & 79 \$45 Management \$200	Scanlon \$17
Fig-FORTH Programming Aids for decompiling, califinding, and translating. CP/M, IBM-PC, Z80, or Apple       \$150         CROSS COMPILERS Allow extending, modifying and compiling for speed and memory savings, can also produce ROMable code. • Requires       \$150         CROSS COMPILERS Allow extending, modifying and compiling for speed and memory savings, can also produce ROMable code. • Requires       \$18         CP/M       \$300       IBM •       \$300         8086 •       \$300       Z80 •       \$300         8086 •       \$300       Z80 •       \$300         9       Thotage of tig-FORTH, that       \$22         •       68000 fig-Forth with assembler       \$25         •       16K RAM Pack       \$150         16K RAM Pack       \$150         CI capstone Computing Inc.       MM MicroMotion         De Dai-E Systems       MM MicroMotion         DE Dai-E Systems       MM MicroMotion         No.), VISA, MasterCard, American Express.       Starting solutions visual subject to change or wind subject to	Requires LM FORTH disk.	□ FORTH on the ATARI by E. Conversion \$10
and translating. CP/M, IBM-PC, Z80, or Apple \$150 CROSS COMPILERS Allow extending, modifying and compiling for speed and memory savings, can also produce ROMable code. •Requires FORTH disk. CP/M \$300   IBM• \$300 0 8086• \$300   Z80• \$300 0 Northstar® \$300   Apple II/II + \$300 FORTH COMPUTER 1 Jupiter Ace \$150 0 48K RAM Pack \$155 1 G K RAM Pack \$155 1 G K RAM Pack \$155 1 G C Cl Capstone Computing Inc. DE Dai-E Systems I insoft I nosoft De Dai-E Systems S, Shaw Labs Cordering information: Check, Money Order (payable to MOUNTAIN VIEW PRESS, No SNautilus Systems S, Shaw Labs Cordering information: Check, Money Order (payable to MOUNTAIN VIEW PRESS, No Killing or unpaid PO's. California residents add sales tax. Shipping costs in US include of in price. Foreign orders, pay in US funds on US bank, include for handling MOUNTAIN VIEW. CA 94040 (415) 961–4103	fig-FORTH Programming Aids for decompiling, califinding.	Floegel 38 Tiny Pascal fig-FORTH \$10
CROSS COMPILERS Allow extending, modifying and compiling for speed and memory savings, can also produce ROMable code. •Requires FORTH disk.       available. (soft cover)       \$18       NOVA by CCI User's Manual includes editor, assembler, and utilities       NOVA by CCI User's Manual includes editor, assembler, and utilities         CP/M       \$300       IBM•       \$300       Starting FORTH (hard cover)       \$18       NOVA by CCI User's Manual includes editor, assembler, and utilities       \$22         Nover of the cover of the co	and translating, CP/M, IBM-PC, Z80, or Apple \$150	Best instructional manual Source Listing \$15
Speed and memory savings, can also produce normable code. • nequires       Starting FORTH (hard cover)       includes editor, assembler, and utilities         FORTH disk.       \$23       and utilities       \$25         Booke \$300       Z80 • \$300       \$300       BM • \$300         Northstar® \$300       Apple II/II + \$300       Image: Speed and the speed and th	CROSS COMPILERS Allow extending, modifying and compiling for	available. (soft cover) \$18
□ CP/M       \$300       □ IBM•       \$300       □ Z80 +       \$300       □ Z80 +       \$300       □ Z80 +       \$300       □ Z80 +       \$300       □ Jupiter ACE Manual by Vickers       \$15         ■ OPERATE COMPUTER       \$300       □ Apple II/II +       \$300       □ Apple II/II +       \$300       □ Installation Manual for fig-FORTH, for specific CPU's and computers. The Installation Manual is required for implementation.       Each \$15         □ ABK RAM Pack       \$150       \$180       \$180       □ ABR/BER AME       \$15         □ ABK RAM Pack       \$150       \$125       \$120       □ G6502       □ 6800       □ AlphaMicro       Each \$15         □ Bai-E Systems       MMS Miller Microcomputer Services       NS Nautilus Systems       □ B080       B0806/88       9900       □ APPLE II         □ Insolt       PNS Pink Noise Studio       □ B080       □ 8080       □ 8086/88       9900       □ APPLE II         □ Boiler       Statulates Sistems       MMS Suiler Microcomputer \$15       □ 68000       □ Eclipse       VAX       □ Z80         Ordering information: Check, Money Order (payable to MOUNTAIN VIEW PRESS, No billing or unpaid PO's. California residents add sales tax. Shipping costs in US       □ and shipping by Air: \$5 for each item over \$100. All prices and products subject to change or withdrawal without notice. Single systems and/or single user license a	speed and memory savings, can also produce ROMable code. •Hequires FORTH disk.	Starting FORTH (hard includes editor, assembler,
□       8086•       \$300       □       Z80•       \$300         □       Northstar®       \$300       □       Apple II/II +       \$300         □       Northstar®       \$300       □       Apple II/II +       \$300         □       Jupiter Ace       \$150         □       □       □       □       □       State       \$150         □	□ CP/M \$300 □ IBM● \$300	and utilities \$25
FORTH COMPUTER       Installation Manual for fig-FORTH,       \$15         Image: Installation Manual for fig-FORTH,       Image: Installation Manual for fig-FORTH,         Image: Installation Manual for fig-FORTH,       Image: Installation Manual for fig-FORTH,         Image: Installation Manual for fig-FORTH,       Image: Installation Manual for fig-FORTH,         Image: Installation Manual for fig-FORTH,       Image:	□ 8086• \$300 □ Z80• \$300 □ Northstar® \$300 □ Apple I//II + \$300	Action of the second se
□ Jupiter Ace       \$150         □ 16K RAM Pack       \$50         □ 48K RAM Pack       \$125         Key to vendors:       \$125         CCI Capstone Computing Inc.       MM MicroMotion         DE Dai-E Systems       MMS Miller Microcomputer Services         FD Forth Dimension       NS Nautilus Systems         Linsoti       PNS Pink Noise Studio         Included in price. Foreign orders, pay in US funds on US bank, include for handling       and shipping by Air: \$5 for each item under \$25, \$10 for each item between \$25 and \$99 and \$20 for each item over \$100. All prices and products subject to change or withdrawal without notice. Single system and/or single user license agreement required on some products.         DeALER & A AUTHOR INQUIRIES INVIED            PO BOX 4656       MOUNTAIN VIEW, CA 94040       (415) 961-4103		Installation Manual for fig-FORTH, \$15
Installation Manual is required for implementation.       Each \$15         ABK RAM Pack       \$125         Key to vendors:       CCI Capstone Computing Inc.       MM MicroMotion         DE Dai-E systems       MMS Miller Microcomputer Services       8080       8086/88       9900       APPLE II         DE Dai-E Systems       NS Nautilus Systems       PNS Pink Noise Studio       PACE       6800       DOVA       PDP-111/LSI-11         Linsoti       PNS Pink Noise Studio       St. Shaw Labs       and shipping by Air: \$5 for each item under \$25, \$10 for each item between \$25 and \$99 and \$20 for each item over \$100. All prices and products subject to change or withdrawal without notice. Single system and/or single user license agreement preducts.         No billing or unpaid PO's. California residents add sales tax. Shipping costs in US include for handling       DEALER & AUTHOR INQUIRIES INVITED         MODUNTAIN VIEW, CA 94040       Letter & AUTHOR INQUIRIES INVITED	□ Jupiter Ace \$150	Source Listings of fig-FORTH, for specific CPU's and computers. The
Key to vendors:       Image: State of the s	□ 48K RAM Pack \$125	Installation Manual is required for implementation. Each \$15
Octored applicitie Control unity into inclored on puter Services       B 8080       B 8086/88       9900       APPLE II         De Daie: Systems       NS Nautilus Systems       NS Nautilus Systems       PACE       6809       NOVA       PDP-11/LSI-11         Linsoft       PNS Pink Noise Studio       E 6800       E clipse       VAX       Z80         Ordering information: Check, Money Order (payable to MOUNTAIN VIEW PRESS, INC.), VISA, MasterCard, American Express. COD's \$5 extra. Minimum order \$15.       and shipping by Air: \$5 for each item under \$25, \$10 for each item between \$25 and \$99 and \$20 for each item over \$100. All prices and products subject to change or withdrawal without notice. Single system and/or single user license agreement tequired on some products.         DeALER & AUTHOR INQUIRIES INVITED         DeALER & AUTHOR INQUIRIES INVITED	Key to vendors:	□ 1802 □ 6502 <b>□</b> 6800 □ AlphaMicro
FD Forth Dimension Linsoft       NS Nautilus Systems PNS Pink Noise Studio       PACE       6809       NOVA       PDP-11/LSI-11         Linsoft       PNS Pink Noise Studio       Edipse       VAX       Z80         Ordering information: Check, Money Order (payable to MOUNTAIN VIEW PRESS, INC.), VISA, MasterCard, American Express. COD's \$5 extra. Minimum order \$15. No billing or unpaid PO's. California residents add sales tax. Shipping costs in US included in price. Foreign orders, pay in US funds on US bank, include for handling       and shipping by Air: \$5 for each item under \$25, \$10 for each item between \$25 and \$99 and \$20 for each item over \$100. All prices and products subject to change or withdrawal without notice. Single system and/or single user license agreement required on some products.         DEALER & AUTHOR INQUIRIES INVITED         DADA A656       MOUNTAIN VIEW, CA 94040       (415) 961-4103	DE Dai-E Systems MMS Miller Microcomputer Services	
In the lab s     In the lab s     Shaw Lab s     Ordering Information: Check, Money Order (payable to MOUNTAIN VIEW PRESS,     INC.), VISA, MasterCard, American Express. COD's \$5 extra. Minimum order \$15.     No billing or unpaid PO's. California residents add sales tax. Shipping costs in US     included in price. Foreign orders, pay in US funds on US bank, include for handling     MOUNTAIN VIEW, CA 94040     (415) 961-4103	FD Forth Dimension NS Nautilus Systems	
Ordering Information: Check, Money Order (payable to MOUNTAIN VIEW PRESS, INC.), VISA, MasterCard, American Express. COD's \$5 extra. Minimum order \$15.       and shipping by Air: \$5 for each item under \$25, \$10 for each item between \$25 and \$99 and \$20 for each item over \$100. All prices and products subject to change or withdrawal without notice. Single system and/or single user license agreement required on some products.         MOUNTAIN VIEW PRESS, No billing or unpaid PO's. California residents add sales tax. Shipping costs in US included in price. Foreign orders, pay in US funds on US bank, include for handling       and shipping by Air: \$5 for each item under \$25, \$10 for each item between \$25 and \$99 and \$20 for each item over \$100. All prices and products subject to change or withdrawal without notice. Single system and/or single user license agreement required on some products.         DEALER & AUTHOR INQUIRIES INVITED         MOUNTAIN VIEW, CA 94040       (415) 961-4103	LM Laboratory Microsystems SL Shaw Labs	
No billing or unpaid PO's. California residents add sales tax. Shipping costs in US included in price. Foreign orders, pay in US funds on US bank, include for handling MOUNTAIN VIEW, CA 94040 (415) 961-4103	Ordering Information: Check, Money Order (payable to MOUNTAIN VIEW PRESS,	and shipping by Air: \$5 for each item under \$25, \$10 for each item between \$25 and \$20 for each item over \$100. All prices and products subject to change or
included in price. Foreign orders, pay in US funds on US bank, include for handling required on some products. <b>DEALER &amp; AUTHOR INQUIRIES INVITED</b> <b>DEALER &amp; AUTHOR INQUIRIES INVITED</b> <b>DEALER &amp; AUTHOR INQUIRIES INVITED</b> <b>OBALER &amp; AUTHOR INQUIRIES INVITE</b>	NO.), VISA, Masterclard, American Express. COD's \$5 extra. Minimum order \$15. No billing or unpaid PO's. California residents add sales tax. Shipping costs in US	withdrawal without notice. Single system and/or single user license agreement
MOUNTAIN VIEW PRESS, INC. PO BOX 4656 MOUNTAIN VIEW, CA 94040 (415) 961-4103	included in price. Foreign orders, pay in US funds on US bank, include for handling	required on some products. DEALER & AUTHOR INQUIRIES INVITED
PO BOX 4656 MOUNTAIN VIEW. CA 94040 (415) 961-4103	MOTINITAIN VI	FW PRESS INC
PO BOX 4656 MOUNTAIN VIEW. CA 94040 (415) 961-4103		
	PU BOX 4656 MOUNTAIN	VIEW, CA 94040 (415) 961-4103

#### **FORTH Dimensions**

Published by FORTH Interest Group Volume V, No. 4 November/December 1983 Editor Marlin Ouverson Publisher Roy C. Martens Typesetting/Production LARC Computing, Inc. Cover Art Al McCahon

FORTH Dimensions solicits editorial material, comments and letters. No responsibility is assumed for accuracy of material submitted. Unless noted otherwise, material published by the FORTH Interest Group is in the public domain. Such material may be reproduced with credit given to the author and the FORTH Interest Group.

Subscription to FORTH Dimensions is free with membership in the FORTH Interest Group at \$15.00 per year (\$27.00 foreign air). For membership, change of address and/or to submit material, the address is: FORTH Interest Group, P.O. Box 1105, San Carlos, CA 94070.

# Letters to the Editor

#### A Friend in Need

Sirs:

I need help. I have used my computer (Kaypro 10) for word processing only. I was not interested in learning a language till I began reading about Forth in the journals. I picked up Starting Forth and leafed through it. I said to myself, "I can learn this!" I bought the book and ordered the language disk. The computer store ordered SL5 from SuperSoft. I started to learn the language using Starting Forth as my documentation. I soon discovered that they were not mated to one another. Valiantly, I struggled with the six pages of SuperSoft tutorial, then back to Starting Forth, trying to change the words that wouldn't work. After many

hours, I gave up. After all that, here is my question:

What should I buy, as a beginner, that would let me use *Starting Forth* as my documentation?

Awaiting your information, I am Yours sincerely,

Duane Windemiller 367 Ocean Boulevard Hampton Beach, NH 03842

#### **Breakpoint Revisited**

#### Dear Editor,

The breakpoint tool described in Forth Dimensions (Vol. V, No. 1)

(Continued)

# Editorial Fifth Forth Fest

Another October has come and gone, and with it the Fifth Annual Forth Convention. Hats off to the FIG board of directors, especially to Robert Reiling and Gary Feierbach, who created an informative, streamlined event for the 1200 attendees and the thirty exhibitors. It is a sign of the maturation of the Forth community and of the diligence of the organizers that the convention earned coverage in the public media as well as in the trade press.

At each of the annual conventions, one FIG member is named "Figgie of the Year." The recipients are those have made exceptional who contributions to Forth and its growth in the industry. This year, the whimsy of the title was surpassed only by the surprise of John D. Hall when his name was announced. John's work as coordinator of local chapters of the Forth Interest Group has been diligent, thorough and unselfish. Through his efforts, many chapters have been guided into formation and now serve the world-wide Forth programming community. This has greatly enhanced

Forth's growth and status. Thanks, John, and congratulations!

Mountain View Press, a major vendor of Forth products, presented a prize at the convention this year. The company had held a contest for those who receive its newsletter. The challenge was to describe Forth in twenty-five words or less for non-Forth people. Charles Moore judged the entries and had this to say before announcing the winner: "Forth resists analysis. It's a right-side of brain function .... Language is practical. Forth is a language, not an operating system." The winner of the contest was Michael Ham of Iowa City, Iowa. His entry reads, "Forth is like the Tao: it is a Way, and is realized when followed. Its fragility is its strength; its simplicity is its direction."

Your editor spent the two-day meeting with authors, authors-to-be and readers. I hope we learned a lot about each other. From what I saw and heard, all of us can look forward to a great deal of exciting material in upcoming issues of *Forth Dimensions*. As the magazine grows, so does the number of ways in which we can serve you. Especially vital is that this publication always remain an open forum. There must always be room for you to voice your support and concerns, not just of the magazine, but about the language and community in which we are immersed.

A small commercial: the ways we can serve you are limited only by our resources. Help *Forth Dimensions* and the Forth programming community to grow by joining as an individual member of the Forth Interest Group. Sure, it's nice to share reading materials. But why not get your own membership and begin sharing this magazine with new people who haven't yet been exposed to Forth. Besides, aren't you tired of getting that wellthumbed issue after everyone else has read it?

In the next issue we will bring tidings from the November FORML conference. Until then, accept our heartfelt wishes for a pleasant holiday season and a new year of peace.

> -Marlin Ouverson Editor

I



FORTH programmer, this is the one you have been waiting for! If you are a beginning FORTH programmer, this will get you started right, and quickly too!

Available on 8 inch disk under CP/M 2.2 or greater also various 51/4" formats and other operating systems

#### FULLY WARRANTIED, DOCUMENTED AND SUPPORTED



Shaw Laboratories, Ltd. 24301 Southland Drive, #216 Hayward, California 94545 (415) 276-5953

needs to be slightly modified for versions of Forth, such as MVP FORTH, that use a vectored **INTERPRET**. This results in three levels of return instead of two in going into and leaving the new interpreter. In the enclosed code, I used **RESUME** because GO has a different function in MVP FORTH (it is a code word that permits one to directly load the program counter). The changes are to replace the four in line seven with six, and to add an additional  $\mathbf{R}$  > **DROP** in line eleven (line numbers refer to the original listing on page nineteen of Forth Dimensions). The version for my Forth is shown in figure one.

#### File Fan Dear Sir:

After laying off for a couple of years, I have rejoined FIG; and I just treated myself to an (extended) evening of catching up on Volumes III and IV of *Forth Dimensions*. So many (mostly pleasurable) reactions are reverberating in my mind that it has taken an act of utmost discipline to restrict my comments to the few below.

First, let me say that I am *not* a Forth fanatic. I have been driven somewhat reluctantly to Forth after careful study of a number of languages for personal computing, including, most recently, C.

The most important issue on my mind is that of operating systems. I see, over a two-year period, some significant evolution away from the Forth screen system and towards the file handling systems of the host environment, e.g. CP/M. This evolution is being driven by the vendors, and my impression is that it is being quietly resisted by FIG.

As a personal computerist, I do not see this as primarily an issue of being able to share disk space with the resident system. Rather, for me, it is simply an issue of being able to use with Forth the power of whatever operating system is available.

Editors and word processors are among the programmer's most personal tools. On the face of it, there just can be no comparison between a Forth screen-based editing system and a good file-based system. On top of that, there is a whole industry out there devoted to developing editors and word processors; and many of us own several examples of each, some of which we find extremely useful.

The design of my personal Z-80, CP/M Forth system includes file variables for sequential and random CP/M files, and allows an arbitrary number of open channels. It also includes a file control disk stack which makes it possible for any Forth source file to load another, to any depth, and pick up where it left off. Some of the possibilities this opens up for the organization of libraries should be clear.

Let me make a radical proposal. Abandon the screen system and, in

#### (Continued on page 28)



# Why Novices Use So Many Variables

#### Michael Ham Iowa City, Iowa

Forth programmers often find themselves tutoring Forth novices. The more experienced the tutors, the harder it is for them to recall their own early difficulties and to understand the sources of the beginner's problems. Without knowing the source of the problem, one cannot attack it at its root, and instead must correct the errors one by one, as they occur. This paper discusses a possible source of a common novice error: using unnecessary variables.

I recently found a conceptual block when I examined the reasons I had coded a program awkwardly (see listing). The listed definition works — it does, in fact, display the primes less than 1000 — but an experienced Forth programmer will instantly see that some changes are in order:

- 1. Eliminate the variable **PRIME**.
- 2. Replace 2 **PRIME** 1 in line five with 1.
- 3. Replace **O PRIME !** in line eight with **DROP 0**.
- 4. Eliminate 1 PRIME ! in line nine.
- 5. Eliminate line eleven altogether.

The word still works with these changes, but more efficiently. (The actual difference in execution time is onetenth of a second on my Forth: 29.0 vs. 28.9 seconds.) What caused the superfluous code?

I discovered the redundancies by chance: I picked up the listing two or three days after writing it, and on glancing at it, suddenly saw that the careful replacement of the two by a one (line eleven) was unnecessary, since two would serve as a true flag as well as one.

But why had I even put a two into **PRIME** in the first place? Its origin seemed to lie in my unexamined feeling that, on coming out of the loop, it would be good to know how I discovered the nature of the number: from within the loop (**PRIME** contains zero or one), or by exhausting all possible divisors (**PRIME** contains two). Perhaps I also had wanted to avoid declaring, before even starting the loop, whether the number was prime or not: two was a way of not taking a position.

I decided that I might as well leave the two on the stack as a true flag and, given that, no reason it shouldn't be a one from the start. So I dropped

0	( EXAMPLE	OF BEGINNER	R CAUTION	M Ham	8/23/83 )
23	VARIABL	E PRIME ( ST	ART zeroes	system clock; TIME	prints time)
4 5 6 7 8 9 10 11 12 13 14 15	: PRIMES	( ) CR 1001 1 DO LOOF TIME ;	START 2 FRIME ! 501 2 DO LOOP PRIME @ DU IF I 5 .R	J I > J I MOD O= AND IF O PRIME J I = IF 1 PRIME P 2 = IF DROP 1 THE THEN	! LEAVE THEN ! LEAVE THEN EN
Listing					

#### DUP 2 🖬 IF DROP 1 THEN

from line eleven, and in line five initialized **PRIME** with **1 PRIME** !.

Then I realized that when exiting the loop early (via one of the LEAVES), I was fetching from **PRIME** the value just put into it. What was going on? Looking at the code again, I saw that **PRIME** was unnecessary, and that all the changes listed above should be made.

Why had I created **PRIME** in the first place? I thought about it and concluded that I had not fully understood Brodie's cautionary remark in *Starting Forth* (page ninety-three) about making the stack effect of a word be the same, regardless of which part of the word was executed. I had thought that he meant you should be careful about the stack effect only to prevent stack underflow or overflow, which would crash the program.

It is that, of course, but with this example I saw something more — something undoubtedly so familiar to those who are used to Forth and the stack that they accept it unthinkingly: if words keep properly to themselves, using the stack only for their expected input and output, and cleaning up after themselves, then they can be looked on as sealed systems, having no effect on anything that might already be on the stack but outside their sphere of influence.

In my prime number routine I had pulled the flag off the stack and tucked it into a variable to keep it safe. No telling *what* might happen with all that thrashing about on the stack. The flag might get hit by a wild shot or ricochet. So off it went for safekeeping, to be fetched from its cubbyhole when needed.

I immediately recalled another example of the same behavior in a program to compute quartiles. The program collects the scores for each group, sorts them, computes the quartiles and prints the results, cycling in an endless loop to get the figures for the next group. After every eight groups I wanted to do a form feed to avoid printing on the perforation. At first I had put the count of the groups in a variable, once again to keep it safe from all the activity taking place on the stack: gathering data, sorting, computing, printing.

But then I had seen that the sequence of words constituted a closed system, and the count could rest safe and sound on the stack the whole time, totally unaware of the storm of activity raging right over its head. The count could go on the stack at the beginning of the loop, and remain through the complete routine for each group. At the end of each, the count would emerge, back on top of the stack, not a hair out of place.

It still somewhat amazes me that merely by making sure words practice

good stack hygiene vou can be so confident about what is happening on the stack. If you investigate the unnecessary variables that beginning Forth programmers use, I bet you'll find them unaccustomed to using a stack and nervous about unforseen by-products of all that frantic stack activity. They become overprotective and use a mechanism familiar from their experience with other programming languages to shield their counts and intermediate results from possible harm. They see the whirlwind of activity that will descend on the stack, but they fail to recognize that it is harmless, enclosed within the inviolable sphere defined by the words' net stack effect.

You need only convince novices that, if they write their definitions with a careful eye on the stack effects, they can drop enormous clusters of words on top of a lonely little number on the stack and, after the words have done their work and gone and the dust has settled, that number will stand there, once more at the top of the stack, absolutely unharmed.



OTHERS

\$99.95

\$ 49.95

\$139.95

# **Yet Another Number Utility**

#### David McKibbin Timonium, Maryland

In any program or environment, there is typically one number base that predominates. Just as typically, there are also times when another number base for input or output would be put. I still use < BUILDS in my system as clearer or easier. The following word the complement to **DOES**>. For those set allows for these single occurrences

of numbers which are not in the current number base and can be used in either compile or execution mode.

There is one defining word (BASE.) for the class of words that output in the various number bases, and another (BASE') for the class of words that inwho don't, replace the **BUILDS** with

SCR # 30 0 ( NUMBERS -BASE., B., O., X., H. DTM 03Aug81 ) 2 : BASE. ( base print defining word \*) <BUILDS . 3 DDES> BASE @ >R @ BASE ! U. R> BASE ! ş 4 5 2 BASE. B. ( binary print \*) 8 BASE. O. 6 print \*) ( octal 7 10 BASE. X. ( decimal print \*) 8 16 BASE. H. ( hey print \*) Ģ 10 11 12 13 14 15 SCR # 31 0 ( NUMBERS - BASE', B', O', X', H' DTM 25Feb82 ) 1 2 : BASE' З <BUILDS . IMMEDIATE 4 DDES> BASE @ >R @ BASE ! 5 0.0 BL WORD CONVERT 2DROP [COMPILE] LITERAL R> BASE ! 6 ; 7 8 2 BASE' B' ( binary input \*) 9 8 BASE' D' ( octal input \*) 10 10 BASE' X' ( decimal input \*) 11 16 BASE' H' ( hex input \*) 12 13 14 15 0K DECIMAL 1234 DK DUP B. 10011010010 DK DUP 0. 2322 DK DUP X. 1234 OK DUP H. 4D2 OK . 1234 OK H' 1234 . 4660 OK . 1234 OK . 668 OK X' 1234 0' 1234 . 69 OK B' 1000101 : strip-parity H' 7F AND ; OK 193 strip-parity . 65 OK 00:00:00 00/00/00

CREATE. Words defined by BASE. and BASE' have their new, temporary base stored in their parameter fields. When executed, they simply fetch the current number base and save it on the return stack, set **BASE** to the new value stored in their parameter field, perform the input or output, and finally restore the old base from the return stack. Additionally, words defined by BASE' are made immediate so that they can be used inside colon definitions. For me, this has been very useful inside **DO...LOOP** where the count is clearer if expressed in decimal but the body of the loop contains bit masks, *etc.*, that are more clearly expressed in hex.

Inside **BASE**' the code fragment

#### 0 0 BL WORD CONVERT 2DROP

is used to get a sixteen-bit number from the input device to the stack. Then **LITERAL** is used either to do nothing (in execution mode) or compile the number as a literal (in compile mode). This is presuming upon the "state smartness" of LITERAL. With the current move away from state smartness, I recommend checking your implementation of LITERAL.

In writing programs, I will usually leave the current number base in **DECIMAL** and use these words for any departures from that. This has solved several recurring problems. First, when I see a number, I know that it is in decimal if not preceded by a modifier. I don't have to reverse-scan the code looking for the last base change. Second, I can painlessly insert hex bit masks regardless of the current number base. And third, I can enter numbers in whatever base most clearly conveys my intent or purpose without bulking up the code with explicit HEX. DECIMAL or OCTAL words.



#### 8080/Z80 FIG-FORTH for CP/M & CDOS systems FULL-SCREEN EDITOR for DISK & MEMORY

\$50 saves you keying the FIG FORTH model and many published FIG FORTH screens onto diskette and debugging them. You receive TWO diskettes (see below for formats available). The first disk is readable by Digital Research CP/M or Cromemco CDOS and contains 8080 source I keyed from the published listings of the FORTH INTEREST GROUP (FIG) plus a translated, enhanced version in ZILOG Z80 mnemonics. This disk also contains executable FORTH.COM files for Z80 & 8080 processors and a special one for Cromemco 3102 terminals.

The 2nd disk contains FORTH readable screens including an extensive FULL-SCREEN EDITOR FOR DISK & MEMORY. This editor is a powerful FORTH software development tool featuring detailed terminal profile descriptions with full cursor function, full and partial LINE-HOLD LINE-REPLACE and LINE-OVERLAY functions plus line insert/delete, character insert/delete, HEX character display/update and drive-track-sector display. The EDITOR may also be used to VIEW AND MODIFY MEMORY (a feature not available on any other full screen editor we know of.) This disk also has formatted memory and I/O port dump words and many items published in FORTH DIMENSIONS, including a FORTH TRACE utility, a model data base handler, an 8080 ASSEMBLER and a recursive decompiler.

The disks are packaged in a ring binder along with a complete listing of the FULL-SCREEN EDITOR and a copy of the FIG-FORTH INSTALLATION MANUAL (the language model of FIG-FORTH, a complete glossary, memory map, installation instructions and the FIG line editor listing and instructions).

This entire work is placed in the public domain in the manner and spirit of the work upon which it is based. Copies may be distributed when proper notices are included.

							USA	Foreign
D FIG-FORT	H & Full Scre	en EDIT	OR package					AIR
Minimun	n system reau	irements	5. 5.					
80x24 vi	deo screen w	cursor :	addressability					
8080 or 2	Z80 or compa	tible cpu	1					
CP/M or	compatible c	perating	system w/ 32K o	or more user	RAM			
Select di	isk format bel	ow, (soft	sectored only).				\$50	\$65
🗆 8″ SS	SD for CP/M	(Single	Side, Single De	nsity)				
		Cromen	nco CDOS forma	ts, Single Sid	le, S/D Density			
🗆 8″ SS	SD 🗆 8″ S	SSDD	5¼" SSSD	□ 5¼″ SSD	D			
		Cromem	co CDOS format	s. Double Sid	de, S/D Density			
🗆 8″ DS	SD 🗆 8"	DSDD	□ 5¼″ DSSD	□ 5¼″ DS	DD			
	C	ther for	mats are being c	onsidered. tel	ll us vour needs	5.		
□ Printed Z8	0 Assembly li	sting w/	xref (Zilog mnen	nonics)			\$15	\$18
Printed 808	80 Assembly I	isting					\$15	\$18
					т	OTAL \$_		
Price inc	ludes postage	. No pur	chase orders with	nout check. Ar	rizona residents	add sales	tax. M	lake checl
or money or	der in US Fun	ds on U	S bank, payable	to:				
•			6	Dennis Wilson	ı c/o			
			A	vristotelian Lo	ogicians			
			2	631 East Pind	hot Avenue			
			F	hoenix A7	85016			

(602) 956-7678

# **Manufacturing Cost Program**

#### Marc Perkel Springfield, Missouri

This program demonstrates how simply cost analysis can be done in Forth. In this example, Forth is used not only as a compiler, but as a job control language. In other words, source screens do not have to contain code to be compiled. They may contain lists of commands to be executed. This execution replaces typing the same commands from the keyboard. In this

way, disk files of command strings to be executed can be changed at will.

#### What the Program Does

This program calculates the cost of manufacturing a fruit basket. The user first compiles screen eighty, which is the program. Then the user types the word **PRICES**, which causes the prices to compile. Then the user types **FIGURE FRUIT-BASKET**, which causes the cost of the fruit basket to be printed on the screen. Any time the user wants to

```
Screen 80 (*) 128
      ( Pricing Program ) : TASK ; DECIMAL
   ۵
   1
      2VARIABLE TOTAL
   2
   3
                 <# # # ASC . HOLD #S ASC $ HOLD #> TYPE SPACE #
      : .MONEY
   4
        +TOTAL
                 TOTAL 20 D+ TOTAL 2! ;
   5
      1
                 CREATE , , DOES> 20 >R OVER U* ROT R> * + +TOTAL ;
      : PRICE
   6
        COST
                 CREATE , , DOES> 20 +TOTAL ;
   7
      :
        LOCATES CREATE , DOES> @ LOAD ;
   9
      :
   9
      :
        DOZEN
                 12 * ;
                0. TOTAL 2! ECOMPILED ' EXECUTE TOTAL 20 .MONEY #
   A
      : FIGURE
   в
   С
      ( Directory )
      129 LOCATES PRICES
   D
      130 LOCATES FRUIT-BASKET
   F
Screen 81 (*) 129
   Ö
     ( Friut Basket Costs )
                   APPLES
      .34
           PRICE
   2
                   BANANAS
   3
      .26
           PRICE
           PRICE
                   ORANGES
   4
      .47
   5
                   CHERRIES
      .03
           PRICE
      +54
                   GRAPEFRUIT
   6
           PRICE
   7
      •32
           PRICE
                   PEARS
   8
                   PEACHES
      .29
           PRICE
   9
      .02
           PRICE
                   GRAPES
   Α
                   BASKET
      2.45
            COST
   В
                   PACKAGING
   С
      1.08
            COST
                   SHIPPING
   Τı
      3.67
            COST
      1.00
            COST
                   HANDLING
   E
Screen 82 (*) 130
   0
     ( Fruit Basket materials )
   1
      5 PEARS
   2
      8 ORANGES
   3
      3 GRAPEFRUIT
   4
      2 DOZEN CHERRIES
   5
      3 DOZEN GRAPES
   6
      6 PEACHES
      4 APPLES
   8
   9
      7 BANANAS
   A
      BASKET
   В
      SHIPPING
   С
   D
      PACKAGING
   E
      HANDLING
```

change prices, he brings in the editor and types over the old data. Or, if the user wants to change the materials, he likewise types in the new materials by using the editor.

#### How the Program Works

The variable TOTAL accumulates the total costs. .MONEY is used to print the total in dollars and cents format. +TO-TAL adds the thirty-two-bit number on top of the stack to TOTAL. PRICE is a defining word; at compile time, it creates a definition (containing the price) that, when executed, multiplies the price by the number on the stack and adds it to the total. For example, 34 PRICE APPLES creates the word APPLES and assigns the value .34 each. When 4 APPLES executes, APPLES multiplies four times thirty-four cents and adds it to TOTAL.

**COST** works like **PRICE** except that it doesn't multiply. It assumes a quantity of one. **DOZEN** merely multiplies the quantity by twelve. **LOCATES** is a defining word used here as a crude disk directory. **LOCATES** creates a word (*e.g.* **PRICES**) and stores 129 into it. When **PRICES** executes, it loads screen 129. The word **FIGURE** executes the following word first and displays the **TOTAL**.

The interesting thing to note is that the words created by **PRICE**, **COST** and **LOCATES** become part of the Forth vocabulary. Thus, typing any word created by **LOCATES** will cause a predetermined screen to load. Any word created by **COST** will cause an amount to be added to **TOTAL**, and likewise with **PRICES**. The Forth outer interpreter is used as a big, text-driven case statement and eliminates the need to write one as part of the program.

In conclusion, I challenge anyone to write such an elegant program in any other language. This program is in use by a local manufacturer, and I hope others will expand on this unusual technique in real-world applications.

# **Menu-Driven Software**

John Bowling Phoenix, Arizona

Menu-driven software has always been easier for all but the most sophisticated of users. The programmer puts a list on the screen with a brief description of the option, along with a key code. The user enters the simple key code, and is off and running in a new section of the program. If the new section has options, up comes another menu. If the purpose of the software requires it, a menu tree can have hundreds of levels, with some menus able to jump tens of levels per key code.

Menus make a user's job very easy, but can be a headache for a programmer. They require that the software write an entire page out to a terminal, pick up a user's response, index into a vector table, and jump to a new section of code. If the software is very large, overlays or some type of virtual memory scheme may be required. Ideally, the language selected should support menus without having to code each one separately. Using a subroutine for the menu code requires passing a pointer to the character data for the terminal, and a pointer to the vector table. Somewhere in memory reside pages of text for the terminal and more pages of vector tables.

There is a simple solution to the problem in Forth. The Forth listing shown here is quite simple, and is designed to work with any standard FIG-Forth system. The two primary words, M" and MENU-UP, are supported by a mode variable (MUMODE), a pointer variable (MP), and two arrays (MNU1 and MNU2). MENU-UP controls the use of M" by setting MUMODE true and sets MP to zero.

When a screen is loaded and an M" is encountered, M" checks MUMODE. If MUMODE is true, two numbers are taken from the input stream (the disk), and saved in the array positions pointed to by MP. MP is incremented once

1.30 0 ( Sample Directory Screen for MENU jlb October 12, 1983) 1 2 DECIMAL ( Header at top of page ) 3 CR ." Disk Directory" 20 Spaces " October 12, 1983" CR 24 SPACES ." Drive " 4 CR ." Disk Menu:" DR @ . CR CR 5 6 ( TYPE out directory according to MUMODE ) 7 8 M" 12 19 Assembler (6502)" 9 M" 131 131 Directory Menu" M" 22 23 Disking" M" 10 M" 20 21 Documentor" 25 39 Editor" 11 M" 180 185 Permanent Extensions" 12 M" 10 11 Startup Loader" 13 14 15 131 0 ( Non-Standard words used for MENU jlb October 12, 1983 ) 1 FORTH DEFINITIONS DECIMAL 2 3 **0 VARIABLE DR** 4 ( DR is a variable that contains the drive number last ) 5 ( specified by DRO or DR1 ) 6 7 8 : DR0 EMPTY-BUFFERS DRO 0 DR ! ; ) 9 : DR1 EMPTY-BUFFERS DR0 1 DR ! ; ) 10 ( Adjust a number between Low and High values, inclusive ) 11 12 : LIMITS ROT MIN MAX ; ( n low high ---- n ) ( PAGE should clear the screen ) 13 : PAGE OC EMIT ; 14 15 ---> 132 jlb October 12, 1983 ) O ( MENU variables ?NUMB 1 2 DECIMAL ( number storage arrays ) 3 4 0 VARIABLE MNUL 52 ALLOT MNU1 52 BLANKS 5 0 VARIABLE MNU2 52 ALLOT MNU2 52 BLANKS 6 7 ( Array Pointer Mode ) 8 9 0 VARIABLE MP **0 VARIABLE MUMODE** 10 11 ---> 12 13 14 15 copyright 1983 by John Bowling Starlight-FORTH OSI V sl.10

133 O ( MENU M" jlb October 12, 1983 ) 1 2 ( Menu display function ) 3 : M" MUMODE @ IF (Save numbers) 4 (Print out a code letter for the MENU  $\rightarrow A$ ) 5 MP @ 65 + ( Adjust to Alpha ) 6 . 52 EMIT 2 SPACES ( Print out Menu number ) 7 MP @ DUP 2\* DUP ( Adjust array pointer ) 8 ( Pickup the numbers following M" and place them in array ) 9 32 WORD HERE NUMBER DROP SWAP MNU1 + ! ( Pull # ) 10 32 WORD HERE NUMBER DROP SWAP MNU2 + ! ( Pull # ) 11 0 26 LIMITS MP ! THEN ( Increment pointer ) 1+12 34 WORD HERE COUNT TYPE CR ; (Type remainder) 13 14 ---> ( M" should be used as you would use ." ) 15 134 ( MENU MENU-UP jlb October 12, 1983) 0 1 2 ( Set variables and printout the specified screen ) 3 : MENU-UP 0 MP ! 1 MUMODE ! 0 MUMODE ! CR 4 LOAD 5 ( Request a number from the user ) ." Your Selection {1 to " MP @ 65 + . ." }? " 6 7 ( Adjust number within LIMITS ) 0 MP @ 1 - LIMITS 8 KEY 65 -2\* DUP 9 ( Call saved numbers from arrays ) 10 MNU2 + @ SWAP MINU1 + @ 0 MP ! : 11 12 --> 13 14 15 135 0 ( MENU Sample uses jlb October 12, 1983 ) 1 2 ( Sample uses of MENU menu -> editor ) 3 : EM PAGE ." Edit Menu" 130 MENU-UP EDITOR E ; 4 5 ( menu -> compiler ) : LM 6 PAGE ." Load Menu" 130 MENU-UP SWAP DROP LOAD ; 7 8 ( Print out MENU screen without saving numbers ) 9 ( M" works just as ." does ) 10 : DIR PAGE 0 MUMODE ! 0 MP ! 130 LOAD ; 11 12 ;S ( See text about EDITOR modifications) 13 (Replace all 130's with the screen number you select for ) ( your directory screen ) 14 15

for each execution of M'' when MU-MODE is true. Following the extraction of the two numbers, a letter corresponding to MP value plus sixty-five is printed, followed by > and the remainder of the data in the input stream up to the next occurrence of ".

After the screen load is completed, the user is requested to press a key between A and MP value plus sixty-five. This upper letter, and the maximum value in MP, is limited by the space available in the arrays, Z or twenty-six. This limit is enforced in M" by MP, and in MENU-UP by the letter keyed in.

There are three sample methods of using the menu function. **DIR** prints the specified screen as if **M**'' were equivalent to .". **LM** will load starting at the screen number in **MNU1**, pointed to by the pressed key. **EM** calls the editor after displaying the menu.

I use a full-screen editor which employs control keys and function keys to perform various functions. It uses two variables called FIRST and LAST that contain the LIMIT values for screen numbers that are allowed to be edited. When using EM, the values on the directory screen are placed into FIRST and LAST. Control-Q will move to the previous screen, but not prior to FIRST. Control-W will move to the next screen, but not after LAST. Function key five (F5) will move to the FIRST screen, and F6 moves to the LAST. EDITOR sets to vocabulary and E enters the editor.

(See figures on page 13)

Starlight-FORTH OSI V sl.10

copyright 1983 by John Bowling

**End Listing** 

#### FORTH for Z-80<sup>®</sup>, 8086, 68000, and IBM<sup>®</sup> PC

FORTH Application Development Systems include interpreter/compiler with virtual memory management and multi-tasking, assembler, full screen editor, decompiler, utilities, and 130 + page manual. Standard random access files used for screen storage, extensions provided for access to all operating system functions.

<b>Z-80 FORTH</b> for CP/M <sup>®</sup> 2.2 or MP/M II	\$ 50.00
8080 FORTH for CP/M 2.2 or MP/M II	\$ 50.00
8086 FORTH for CP/M-86 or MS-DOS	\$100.00
PC/FORTH <sup>™</sup> for PC-DOS, CP/M-86, or CCPM	\$100.00
68000 FORTH for CP/M-68K	\$250.00

83 - Standard version of all application development systems available soon. All registered users will be entitled to software update at nominal cost.

FORTH + Systems are 32 bit implementations that allow creation of programs as large as 1 megabyte. The entire memory address space of the 68000 or 8086/88 is supported directly for programs and data.

PC/FORTH + for PC-DOS or CP/M-86	
8086 FORTH + for CP/M-86	
68000 FORTH + for CP/M-68K	\$400.00
Extension Packages for FORTH systems	
Software floating point (Z-80, 8086, PC only)	\$100.00
Intel 8087 support (8086, PC only)	
AMD 9511 support (8086, Z-80 only)	\$100.00
Color graphics with animation support (PC only)	\$100.00
Symbolic interactive debugger (PC only)	\$100.00
Cross reference utility	\$ 25.00
PC/GEN <sup>TM</sup> (custom character sets, PC only)	\$ 50.00
PC/TERM communications program for PC and Smartmodem	\$ 60.00
Hierarchical file manager	\$ 50.00
B-tree index manager	\$125.00
B-tree index and file manager	\$200.00
OTE L Sereen editor and text formatter by Lee Bredie	
for IDM DC with IDM or Encore printer	\$100.00
for IBM PC with IBM of Epson printer	
<b>Nautilus Cross Compiler</b> allows you to expand or modify the FORTH nucleus, recomp a different target computer, generate headerless and ROMable code. Supports forw executable image in RAM or disk file. No license fee for applications. Prerequisite:	bile on a host computer for rard referencing. Produces Application Development
System for host computer.	

Hosts: Z-80 (CP/M 2.2 or MP/M II), 8086/88 (CP/M-86 or MS-DOS), IBM PC (PC-DOS or CP/M-86), 68000 (CP/M-68K)

09 6500 1 61 11 60000 100

(argets: 6666, 2-66, 6666/66, 6562, 231-17, 66666, 1662, 2-6	
Cross-Compiler for one host and one target	\$300.00
Each additional target	\$100.00
<b>AUGUSTA<sup>™</sup></b> ADA subset compiler from Computer Linguistics, for Z-80 computers under CP/M 2.2 <b>LEARNING FORTH</b> computer-assisted tutorial by Laxen and Harris for CP/M, includes Brodie's	? \$ 90.00
"Starting FORTH" (8" format only)	\$ 95.00
2-80 Machine Lests Memory, disk, printer, and console tests with all source code in standard 2 mnemonics	110g \$ 50.00

8080 and Z-80 application development systems require 48 kbytes RAM and 1 disk drive, 8086 and 68000 require 64 kbytes. Prices include shipping by UPS or first class mail within USA and Canada. California residents add appropriate sales tax. Purchase orders accepted at our discretion. Master Charge and Visa accepted

Disk formats available: Standard CP/M 8" SSSD, Northstar 51/4 " QD, Micropolis 51/4 " QD, Sage 51/4 " DD, Apple 51/4 ", Victor 9000 51/4 ", Kaypro 51/4 ", Osborne 51/4 " DD, Micromate 5¼ ", IBM PC 5¼ ", Standard MS-DOS 5¼ " SSDD. Most other formats can be special ordered

#### Laboratory Microsystems, Inc. **4147 Beethoven Street** Los Angeles, CA 90066 (213) 306-7412

Z-80 is a registered trademark of Zilog, Inc CP/M is a registered trademark of Digital Research, Inc. IBM is a registered trademark of International Business Machines Corp.

Augusta is a trademark of Computer Linguistics dBASE II is a trademark of Ashton-Tate PC/FORTH and PC/GEN are trademarks of Laboratory Microsystems Inc. Disk Directory

Disk Menu:

Drive O

A> Assembler (6502)
B> Directory Menu
C> Disking
D> Documentor
E> Editor
F> Permanent Extensions

G> Startup Loader

Your Selection {A to G}? G

Stack after key press

TOS 10 11 BOS

OK

Sample of 130 DIR

Disk Directory

Disk Menu:

Drive 0

12 19 Assembler (6502) 131 131 Directory Menu 22 23 Disking 20 21 Documentor 25 39 Editor 180 185 Permanent Extensions 10 11 Startup Loader OK

Sample of 130 MENU-UP



When in-house staff can't solve the problem, make us a part of your team. As specialists in custom designed software, we have the know-how to handle your application from start to finish.

Call us for some straight talk about:

- Process Control
- Automated Design
- Database Management
- System Software & Utilities-
- Engineering
- Scientific Applications
- Turn Key Systems

Inner Access Corporation P.O. Box 888, Belmont, CA 94002 PHONE (415) 591-8295

# Vocabulary Tutorial, Part II

Evan Rosen East Setauket, New York

A commonly recognized problem with the **CONTEXT/CURRENT** scheme arises when programming a code definition for some vocabulary (call it **VOCAB**) which is not **FORTH**. Then **ASSEMBLER** is **CONTEXT** and **VOCAB** is **CURRENT**. If some needed data is in a third vocabulary, the programmer has to declare that vocabulary, get the data to the stack, and then declare **ASSEM-BLER** again to continue assembling machine code. This problem and its cousins are at the nuisance level.

More serious is the fact that **CON**-**TEXT/CURRENT** makes implementing "front-end" routines like algebraic parsers very difficult. One has only to write out the solution to the quadratic equation in Forth to realize the need for such a utility. Yet, although parsing schemes in Forth have been proposed many times (see the recent one by Stolowitz<sup>1</sup>), few have found their way into widely distributed systems. This is largely attributable to the difficulty of having more than two vocabularies, CONTEXT and CURRENT, in the search order. For instance, for an algebraic parser's vocabulary, redefined words would include at least the following: + - <> /()

These words would have to be reached by the search before their counterparts in the **FORTH** vocabulary, since they are redefinitions. In FIG, this means that the parser vocabulary would have to be **CONTEXT**, and the next searched vocabulary would have to be put in **CURRENT**, producing inelegant and misleading code. Accessing any other vocabulary during parsing, for data pickup or another reason, would become awkward to the point of impracticality.

Still other uses for extra, transient vocabularies, such as named, local arguments, which would make life so

<sup>1.</sup> Michael Stolowitz, "Algebraic Expression Evaluation in Forth," *Forth Dimensions*, Vol. IV No. 6.

Actio	ac	$\sim \sim \sim$	<u>Resulting</u>	search lis	t		
		(	os. 0) first/conte:	xt)	,	() (last/pro	pos. 4) tected)
ONLY		$\sim \sim \sim$	ONLY	Ø	Ø	Ø	ONLY
	ONLY (poin list ("pro the e will twice preve	erases nters 1 so that tected and as be sk: since ant the	s the 10 by (c) itself at it will ( d") list po- more vocabl (pped during a "twice second second second second	tes of sea in the fir show as CC sition so ularies ar g searches in a row" arch.	inch list a st (conte) INTEXT, and that it wo that it wo added. . ONLY wi detector i	and then (t) posit i at the on't be p The Ø's .11 not be .n -FIND (	installs last ushed off in the list e searched will
FORTH	ļ	~~~	FORTH	Ø#1	0#2	Ø#3	ONLY
	We've which the f norma imple good	tagge ones irst ( l func mentat sign.	d the zeroe disappear. context) po tion of a v ions, and i	es in the FORTH pl osition of ocabulary s preserv	search lis aces a poi the list. name in v ed in this	t so you nter to j This is irtually one. Th	can watch tself into the <u>all</u> FORTH nat's a
ALSO		~~~	FORTH	FORTH	@#1	0#2	ONLY
	Here' Ø, 1, The e left next room	s part and 2 mtry a untouc vocabu for th	of the mag of the sea t position hed. FORTH lary call. e next voca	ic. ALSO arch list 3 is lost 1 is now s The effe abulary, i	, a new wo into posit . The <b>ONL</b> afe from o ct of <b>ALSO</b> n this cas	rd, moves ions 1, 2 Y at posi verwritir is simpl e ASSEMBL	s entries 2, and 3. Ition 4 is ng by the Ly to make L <b>ER</b>
ASSEM	BLER	$\sim \sim \sim$	ASSEMBLER	FORTH	Ø#1	0#2	ONLY
	The v list. ONLY.	ocabul The	ary <b>ASSEMBL</b> search orde	ER is put er is now	at the to ASSEMBLER	p of the FORTH (s	search (ip) (skip)
ALSO		~~~	ASSEMBLER	ASSEMBLER	FORTH	0#1	ONLY
	ASSEM Vocat	BLER i mulary	s slid dowr to the list	) preparat :. Note t	ory to add hat 0#2 ha	ing anoth s been lo	ner Dst.
PARSE		~~~	PARSE	ASSEMBLER	FORTH	Ø#1	ONLY
	PARSE and t there vocat then	is ad o pars 's roc ularie FORTH	ded. We ar e algebraic m for anoth s were to b would be pu	re now pre r expressi ner vocabu ne added, ushed off	pared to d ons while lary. Not e.g. <b>ALSO</b> the end.	o assemb) doing do. e that if VOC1 ALS	ly code, And <sup>5</sup> two more 60 VOC2

much easier, also have been impeded by the vocabulary search problem.

#### The ONLY Solution?

Probably the best and most obvious approach to the outlined situation is simply to allow the programmer to specify a search order for as many vocabularies as are required, and be done with it. Indeed, over the last few years many authors have proposed and/or implemented systems which do just this. And the STOIC language, a variant of Forth, has long had a vocabulary stack for the same reason. But no general agreement on specifics has arisen for Forth.

Recently, however, due to some campaigning by Bill Ragsdale on behalf of his own rather clever scheme<sup>2</sup>, a welcome concensus on the next milestone in Forth vocabulary structures may be in sight.

(An important thing to remember when promoting ideas to the rag-tag Forth community is to give your code a short, catchy name like **ONLY** and then emblazon the name on your t-shirt at Forth conferences. It also doesn't hurt if the solution is a good one.)

Following is a description and illustration of the form of Ragsdale's vocabulary structure.

At the heart of the **ONLY** scheme are three words: ONLY, ALSO, and a new version of **-FIND**. The Forth-79 version of the word vocabulary is used instead of the FIG version. (The difference makes each one stand-alone or "sealed," e.g. with its last link field containing zero, similar to the way the vocabulary FORTH is in FIG-Forth.) These words and a few bytes of memory appended to the data area of the variable **CONTEXT** are about all the implementation requires to function, though several other programmerfriendly words are included for convenience.

When the variable **CONTEXT** is defined, it is followed by, say, **8** ALLOT which reserves some additional memory where the list of (pointers to) vocabularies in the search order will reside. With eight more bytes allotted after **CONTEXT**, there is a total of ten, or room enough for five vocabularies in the list at once. Five entries is an arbitrary but practical list size.

At compile time, the new -FIND searches through each of the vocabularies in turn, using the usual (-FIND) primitive. But the real innovation is in how the search list is set up and manipulated. We examine this process by starting with the word ONLY.

**ONLY** is a very small vocabulary containing only a few words and having an additional feature not shared with other vocabularies: when executed, ONLY zeroes the search list and then installs pointers to itself in the first and last positions in the list. (During search, zeroes in the list are skipped over, and duplicate successive entries do not result in duplicate searches.) The first position is the normal data area of the variable CONTEXT, and the last position in the search list is a protected one which is not normally altered by anything except ONLY. Saying **ONLY** is how one starts a new search list

The few words in the **ONLY** vocabulary allow a larger list to be built. The main words are: **ONLY** itself, which always allows you to start over; **ALSO**, which makes room for more vocabularies on the list (see figure one); **FORTH**, which adds the Forth vocabulary to the search list; and the null word, a system detail necessary for interpretation of source code.

Obscure? Probably only at first. Figure one shows a walkthrough of a typical sequence.

Now, because appropriate search orders can usually be set up outside a word to be defined, there is no particular need for vocabularies to be immediate any more, and so in a particular implementation you may find that they are not. Beyond this, if you are only programming in the Forth vocabulary, you'll see very little difference in code. However, all sorts of additional programming utilities are made possible with the new structure. Expect to see them coming up soon.

The next and last installment in this vocabularies series will discuss the marked similarity between the vocabulary structure in Forth (especially FIG-Forth) and the "object" structure in the Xerox-developed language, Smalltalk.

#### FOR TRS-80 MODELS 1, 3 & 4 IBM PC, XT, AND COMPAQ

#### The MMSFORTH System. Compare.

- The speed, compactness and extensibility of the MMSFORTH total software environment, optimized for the popular IBM PC and TRS-80 Models 1, 3 and 4.
- An integrated system of sophisticated application programs: word processing, database management, communications, general ledger and more, all with powerful capabilities, surprising speed and ease of use.
- With source code, for custom modifications by you or MMS.
- The famous MMS support, including detailed manuals and examples, telephone tips, additional programs and inexpensive program updates, User Groups worldwide, the MMSFORTH Newsletter, Forth-related books, workshops and professional consulting.



# A World of Difference!

- Personal licensing for TRS-80: \$129.95 for MMSFORTH, or "3+4TH" User System with FORTHWRITE, DATA-HANDLER and FORTHCOM for \$399.95.
- Personal licensing for IBM PC: \$249.95 for MMSFORTH, or enhanced "3+4TH" User System with FORTHWRITE, DATAHANDLER-PLUS and FORTHCOM for \$549.95.
- Corporate Site License Extensions from \$1,000.

If you recognize the difference and want to profit from it, ask us or your dealer about the world of MMSFORTH.

MILLER MICROCOMPUTER SERVICES 61 Lake Shore Road, Natick, MA 01760 (617) 653-6136

<sup>2.</sup> William F. Ragsdale, "The 'ONLY' Concept for Forth Vocabularies," *Forth Modification Laboratory Conference Papers*, 1982.

### C64-FORTH for the **Commodore 64**

#### FORTH SOFTWARE FOR THE **COMMODORE 64**

C64-FORTH (TM) for the Commodore 64 -\$99.95 • Fig Forth-79 implementation with extensions

- Full feature screen editor and macro assembler
- Trace feature for easy debugging
- 320x200, 2 color bit mapped graphics
- 16 color sprite and character graphics Compatible with VIC peripherals including
- disks, data set, modem, printer and cartridges • Extensive 144 page manual with examples and
- application screens "SAVETURNKEY" normally allows
- application program distribution without licensing or royalties

#### C64-XTEND (TM) FORTH Extension for C64-FORTH - \$59.95

- (Requires original C64-FORTH copy) Fully compatible floating point package including arithmetic, relational, logical and transcendental functions
- Floating point range of 1E+38 to 2E-39 String extensions including LEFT\$, RIGHT\$, and MID\$
- BCD functions for 10 digit numbers including multiply, divide, and percentage. BCD numbers may by used for DOLLAR.CENTS calculations without the round-off error inherent in BASIC real numbers.
- Special words are provided for inputting and outputting DOLLAR.CENTS values
- Detailed manual with examples and applications screens

(Commodore 64 is a trademark of Commodore)

TO ORDER - Specify disk or cassette version Check, money order, bank card, COD's add \$1.50

- Add \$4.00 postage and handling in
- USA and Canada - Mass. orders add 5% sales tax
- Foreign orders add 20% shipping
- and handling
- Dealer inquiries welcome



(617) 828-1209



# **Forth Froth**

Wil Baden Costa Mesa, California

Forth is unusual among programming languages in that it uses ) and ( for the same purpose as natural language. Indeed, "parenthesis" means "remark" in Greek, and ) and ( are properly called "parenthesis marks," just as we say "quotation mark" and "exclamation mark."

Other analogies with natural language can be made.

In natural language, a noun is the name of a person, place or thing. In Forth, this corresponds to something whose stack diagram is (-n1, ..., ni). Nouns can be singular or plural; this corresponds to how many values are put on the stack. A proper noun, in Forth, is a constant.

An adjective modifies a noun. In Forth, the stack diagram is (n1,...ni -- n1',...ni'). 1+ is an adjective; so are + and COUNT.

A verb is the name of an action. In Forth the stack diagram is (n1,...,ni --).

Intransitive verbs use up one value from the stack (e.g. LIST) and transitive verbs consume more than one value from the stack. The stack element corresponding to n1 is the subject of the verb, the other elements are objects. The value corresponding to n2 is the direct object; any others are indirect objects. A verb without a subject, e.g. the stack diagram would be (--), is an impersonal verb; others are personal verbs.

Prepositions must be followed by another word. Some examples in Forth are :, VARIABLE, FORGET and '.

A pronoun takes the place of a noun. In Forth, these are the stack operators, DUP and OVER. It is convenient to consider the other stack operators as pronouns also, e.g. ROT, SWAP, and DROP. (Note that by our previous definitions, **DROP** could also be a verb and the rest could all be adjectives.)

Interjections are the words in a parenthesis.

Conjunctions are used to join words and phrases. The Forth conjunctions are BEGIN. WHILE, REPEAT, UNTIL, IF, ELSE, THEN, DO, LOOP, and +LOOP.

Adverbs tell how, when or where. They modify adjectives, verbs and other adverbs. They are something like adjectives and something like verbs. In Forth they are words with stack diagram

(n1, ..., ni, n[i+1], ..., nk -- n1, ..., ni)

They take items off the stack, but leave some unchanged.

Thus, all eight parts of speech are found in Forth. Other grammatical features are also present.

A phrase is a sequence of words which could be used as a colon definition. A phrase has a stack diagram which tells what kind of phrase it is. Thus, if the stack diagram is (--n), it is a noun phrase.

A clause is a verb phrase or an adverb phrase. Dependent clauses are adverb phrases; independent clauses are verb phrases.

In English, just the first letter of the name of a language is capitalized (e.g. English, French, Fortran, Forth). Thus, **FORTH** is a Forth word. The name of the compiler will depend upon your system, but will probably be "forth" or "FORTH."

When I first thought of these correspondences, I thought they were amusing but not very useful. A year later, I think they have a definite value. They suggest a way of choosing names for words, a convenient classification when describing a word and a rationale for punctuating phrases in definitions. A clause, for instance, should be separated by end-of-line or by extra spacing.

# Vectored Execution and Recursion

( Used to store PFA of (LBR) ) O VARIABLE 'LBR O VARIABLE 'RBR ( Used to store PFA of (RBR) ) ( Value of calling level ) 8 VARIABLE LEVEL 20 VARIABLE ANGLE ( Angle between stems ) ( Returns execution to calling level ) : RETURN R> DROP ; : LBR 'LBR @ CFA EXECUTE ; ( Define LBR in terms of 'LBR ) ( Define RBR in terms of 'RBR ) RBR 'RBR @ CFA EXECUTE ; : NODE LEVEL @ 1 < IF RETURN THEN -1 LEVEL +! ANGLE @ LEFT ( Adjust level and turn ) I BR ( Draw left branch ) ANGLE @ 2\* RIGHT (Turn) ( Draw right branch ) RBR ANGLE @ LEFT 1 LEVEL +! ; ( Reset ) : (LBR) DUP 2\* PENDOWN FORWARD ( Draw 2x stem ) ( Do next level ) NODE DUP 2\* PENUP BACKWARD ; ( Reset Cursor ) : (RBR) DUP PENDOWN FORWARD ( DRAW 1x stem ) ( Do next level ) NODE DUP PENUP BACKWARD ; ( Reset Cursor ) (LBR) 'LBR ! ( Store PFA of (LBR) in 'LBR ) ( Store PFA of (RBR) in 'RBR ) (RBR) 'RBR ' : SETUP 8 LEVEL ! 10 LBR DROP ; ( Type SETUP to draw tree )



#### Roy W. Sommers Pennsville, New Jersey

Previous articles on recursion (e.g. Vol. IV, No. 2) utilized the words MY-SELF and RETURN to effect recursion. However, in cases where the procedures call each other, this approach does not work. Recursion can still be accomplished, though, by using vectored execution.

This type of recursion can be illustrated in terms of the binary tree described in *Turtle Geometry* (Abelson and diSessa, page 83). The reference program was written in a form of LO-GO and used separate procedures to draw left and right stems of different lengths, and a third procedure to control depth of recursion and cursor positioning. Each procedure called one or more of the other procedures.

An analogous program can be written in Forth by using vectored execution (see accompanying code and illustration). Since NODE contains LBR and RBR, these words must be defined before NODE is loaded. However, if they were written in the form (LBR) and (RBR), they could not be loaded before NODE since they contain NODE. Vectored execution solves this problem by defining LBR and RBR in terms of the variables 'LBR and 'RBR which, at the time of execution of NODE, contain the PFAs of (LBR) and (RBR).



IBM, COMPAQ, MPC, and FORTH-32 are trademarks of IBM, COMPAQ, Columbia Data Products, and Quest Research, respectively.

# Apple Forth á la Modem

```
SCR # 120
  O ( TERMINAL MODEM ROUTINE
                                -RDA 8/06/83 )
  1
  2
    HEX
  3 COB5 CONSTANT CTRL2
  4
    COB6 CONSTANT CTRL1
                            ( WRITING )
  5
    COB6 CONSTANT STATUS
                            ( READING )
  6
    COB7 CONSTANT DATA
  7
      3CO CONSTANT BYTES/SCR ( APPLE SCREEN)
  8
  9
    : IN?MOD
               ( --- BITO ) STATUS C@ 1 AND #
 10 : OUT?MOD ( -- BIT1 ) STATUS C@ 2 AND #
 11
 12
    : OUT.MODEM ( CHAR --- )
 13
      BEGIN OUT?MOD UNTIL DATA C! ;
 14
 15
    : INIT.MODEM ( 1=ORIG 0=ANS
                                    --- )
 16
      3 CTRL1 C! 11 CTRL1 C!
 17
      IF 8F ELSE 8B THEN CTRL2 C! ;
 18
 19
     : GET.MODEM ( -- CHAR )
 20
      BEGIN INTMOD UNTIL DATA CO ;
 21
 22
     DECIMAL
 23
          ۶S
 0K
SCR # 121
  0 ( TERMINAL ROUTINE FOR MODEM PAGE 2 )
  1
  2
     HEX
  3
    : TRANS#
                ( FROM.LOCAL# TO.REMOTE# -- )
  4
      CR 2 ( CONTROL B )
  5
      OUT.MODEM OUT.MODEM BLOCK
  6
      5000 0 DO LOOP BYTES/BLK 0
  7
      DO DUP I + CO DUP EMIT OUT.MODEM
 8
      LOOP DROP ;
  9
 10
   : RTRANS# ( F.LOCAL# T.REMOTE #BLKS --- )
 11
      O DO OVER I + OVER I + TRANS#
 12
           LOOP 2DROP ;
 13
 14
    : RECV GET. MODEM BLOCK BYTES/BLK Q
 15
      DO GET.MODEM DUP EMIT OVER I + C!
 16
      LOOP DROP UPDATE ;
 17
 18
 19
      DECIMAL
                 ∮S
 20
```

#### R. Dudley Ackerman San Francisco, California

These words will allow Forth users with Hayes Micromodems and Apples to send screens back and forth. Minor modifications will allow use on other systems.

Execute **MODEM** after connect is made. A one should be on the originator's stack, a zero on the stack of the computer in answer mode. Both parties should be able to see entries from both keyboards.

When the modem program sees a control-B from its keyboard, it inputs a line of Forth. By executing **RTRANS#**, a set of screens can be sent from one system to the other. Notice the stack setup for **RTRANS#** : source screen number, destination screen number, and number of screens.

**TRANS#** sends a control-B out and a screen number, then waits to give the receiver time to get the screen into its buffer. When the modem program gets a control-B across the line, it executes the Forth word to receive a screen.

**@KEY** is Apple specific and gets a character from the keyboard. Hex eleven in **INIT.MODEM** sets the number of bits per character to eight, with no parity, one start bit, and two stop bits for the Hayes modem. Hex three initializes the Hayes modem.

The program will also function as a simple terminal program. One desirable feature would be a way to capture text into free memory, then save the text to disk.

(Listings Continued)

```
53
                                              22
                                               17
                                         S#
                                               50
                                   DECIMAL
                                               16
                                               18
                                               11
                                  REPEAT ;
                                               91
                                               12
                                    THEN
     IE DKOE KECA EFRE EWIL THEN
                                               ŧ١.
                                               13
    IF DATA CO DUP 2 = \langle CTRL 'B' \rangle
                             MHILE INTROD
                                               15
                ( LIND=/0/ ) - 15 NHL )
                                               TI
 ELSE DUP EMIT DUP OUT. MODEM THEN
                                               10
           IF DROP QUERY INTERPRET
                                               6
     IE DKOB GKEX DAB 5 = ( CIKF , B,
                                               8
   (
                           BECIN SKEX DOL
                                               ۷
                         HODEM INIT, MODEM
                                               9
                                               G
          f
            COOO CG COIO CG DROF YF AND
                                               ŧ
                          : GKEA ( -- KEA )
                                              Σ.
                                               7
                                         XBH
                                               Ţ
(28/90/8
           ⊎пи
                                     WENTOW ) O
                                        2CK # 155
```



iearn. FORTH workshop catalogue). transportability, then FORTH is a language you should

#### Join the FORTH Revolution!

- Intensive 5-day workshops
- Small classes
- Experienced protessionals

ЮК

9628-169 (914)

P.O. Box 888, Belmont, CA 94002

Inner Access Corporation

gnuzia bad

# PIECE OF MIND

#### The System 816. The fastest, most cooperative computer you can buy.

OEMs and systems integrators are busy people. Too busy to waste time with an uncooperative computer system. That's why every System 816 from **CompuPro** is built to work long and hard without a whine or a whimper.

#### More Dependable.

With ten years of pioneering successes built into it, the System 816 is backed by the industry's longest warranty coverage. Depending on your needs, our warranties range from 12 to 36 months. Most other computer manufacturers expect you to be satisfied with 90 days, which typically covers parts only.

You can also depend on complete hardware and software support, flexible configurations and upgrades, and system training.

#### More Powerful.

The System 816 squeezes more performance out of the IEEE-696 S-100 bus than any other system you can buy. A choice of CPUs—and up to 4 Mbytes of our exclusive M-Drive/ $H^{TM}$  RAM disk—give multiple work stations all the speed and power they can ask for. Standard RAM memory is expandable to one megabyte or more.

Disk storage capacity ranges up to 4.8 Mbytes on floppy drives and as much as 320 Mbytes per controller on hard disk.



A GODBOUT COMPANY 3506 Breakwater Court, Hayward, CA 94545

See us at COMDEX/FALL'83 in Las Vegas.

Prices and specifications subject to change without notice. System 816 front panel design shown is available from Full Service CompuPro System Centers only.

CP/M and CP/M-86 are registered trademarks and CP/M-68K, MP/M-86, Concurrent CP/M-86 and FORTRAN 77 are trademarks of Digital Research. CP/M and MP/M 8-16 are compound trademarks of Digital Research and CompuPro.



©1983 CompuPro

#### More Versatile.

All family members share a common modular architecture. So it's a simple matter to upgrade or reconfigure any of them to keep up with your needs. All the while maintaining complete software compatibility up and down the line.

And the S-100 bus allows you the flexibility to plug in any compatible board to add graphics capabilities or boards for your own unique applications.

You also get your choice of operating environments, including CP/M<sup>®</sup> CP/M<sup>\*</sup>86<sup>®</sup> Concurrent

CP/M®, CP/M®, CP/M-86®, Concurrent CP/M®, CP/M-86®, Concurrent CP/M® 8-16<sup>™</sup> and CP/M-68K<sup>™</sup>, and our own CP/M® 8-16<sup>™</sup> and MP/M<sup>™</sup>8-16<sup>™</sup>. At the programming level, the System 816 family supports Pascal, C, map – FORTH, BASIC, COBOL, PL/1, and ANSI FORTRAN 77<sup>™</sup> and more.

#### More Information.

Your customer's satisfaction is important to both of us, so don't get stuck with a system that's more of a hindrance than a help. Send in the coupon and find out what peace of mind is all about.

- □ Send me your free System 816 brochure.
- Send me the name of my nearest Full
- Service CompuPro System Center or dealer.

NAME \_\_\_\_\_\_

ADDRESS .....

- CITY\_\_\_\_\_STATE \_\_\_\_ZIP\_\_\_\_
- Mail to: CompuPro, Attn: Sales Dept. 3506 Breakwater Court, Hayward, CA 94545

Volume V, No. 4

# Forth-83 Loop Structure

Bill Stoddart Middlesbrough, England

#### The story so far...

Bob Berkey has suggested a new loop structure capable of covering a full 64K range, of handling positive or negative increments, or even increments which switch sign. The internal implementation is based on the fact that an overflow condition occurs when a sixteen-bit addition or subtraction traverses the boundary between 7FFF and 8000 hex in either direction. By using 8000 hex as a universal loop limit and performing a corresponding transformation on the loop index, we can test for completion of the loop by checking whether adding the increment to the transformed index causes an overflow.

The new loop has been accepted into the 83-Standard, but requires careful thought if its advantages in terms of generality and speed are to be accompanied by simplicity of implementation.

The main complication of the new loop is in the implementation of **LEAVE**. which has traditionally equated the loop limit and index, forcing termination on the next occurrence of LOOP or +LOOP. This technique is no longer available, as there is no longer an explicit loop limit, and there is no value the index can be set to which will ensure termination for both positive and negative increments. Setting the limit to 7FFF hex will ensure termination for LOOP and +LOOP with a positive increment, since adding a positive value to 7FFF will always cause an overflow. However, there will be no overflow if the index is decremented by + LOOP.

With this in mind, the 83-Standard specifies that **LEAVE** should straight away transfer execution to just beyond the end of the loop structure. Various ways of achieving this have been suggested. Bob Berkey's original suggestion was that the runtime operation compiled by **DO** should push an exit address onto the return stack. This im-

```
180
( Nucleus )
1
2 HEX
 4 CODE (DO)
                                  CX POP ( limit )
                                                      AX CX CMP
 5
      AX POP ( initial index )
                                           AH AH SUB
                                                       AX SI ADD
      O= IF ( bypass null loop ) LODS
 6
             BP SP XCHG 8000 # DX MOV
                                           CX DX SUB
                                                        DX PUSH ( × )
 7
      ELSE
        AX DX ADD
                   DX PUSH ( i )
                                     SP BP XCHG
                                                   SI INC
 8
                    END-CODE
 9
      THEN
             NEXT
10
11 DECIMAL
12
13
14
15
181
0 ( Nucleus )
                                           4 # BP ADD
                                                        SI INC
2
   CODE (LOOP)
                 BP9 WORD INC
                                 OFL IF
                                                             END-CODE
                                              THEN
      ELSE
             LODS
                    AH AH XOR
                                 AX SI SUB
                                                     NEXT
 3
 4
5 CODE (+LOOP)
                             OFL IF
               AY BER ADD
                                      4 # RP ADD
                                                    SI INC
6
      AY POP
                                               THEN
                                                              END-CODE
                                  AX SI SUB
                                                      NEXT
7
      ELSE
              LODS
                      AH AH SUB
8
9
10
11
12
13
14
15
182
0 ( Nucleus )
1
2 CODE I ( -- n
                   leave loop index
      BPQ AX MOV
                   2 DISP8 BP3 AX SUB
                                          AX PUSH
                                                    NEXT
                                                            END-CODE
3
4
                   leave outer loop index )
5 CODE J ( -- n
                                                  AX PUSH
                                                             NEXT
                            6 DISP8 BP2 AX SUB
      4 DISP8 BPQ AX MOV
6
7
      END-CODE
8
9
   CODE (LEAVE)
                           AH AH SUB
                                       AX SI SUB
      4 # BP ADD
                   LODS
10
      LODS
            AX SI ADD
                          NEXT
                                 END-CODE
11
12
13
14
15
```

```
183
 0 ( System word set, high level )
 2 : <MARK ( -- addr )
                          HERE
 З
 4 : <RESOLVE ( addr --.)
      HERE SWAP - 1+
 5
                        с.
                            ;
 6
    →MARK ( --- addr )
                          HERE
                                  ос,;
 7.
 я
 Q
     >RESOLVE ( addr -- )
10
      HERE
            OVER - 1-
                          SWAP C! :
11
12
13
14
```

15

```
184
0 ( Program structures )
 1
 2 VARIABLE CLUE
 3
  : DO ( -- dest )
 4
      CLUE @ COMPILE (DO)
                               >MARK ( forward branch past loop )
 5
      DUP CLUE !
                   <MARK ( backward branch from loop ) ; IMMEDIATE
6
 7
8 : LOOP ( dest -- )
      COMPILE
               (LOOP)
                         <RESOLVE
                                     >RESOLVE
                                                CLUE ! ;
                                                            IMMEDIATE
 Q
10
11 :
     +LOOP ( dest --- )
12
      COMPILE (+LOOP)
                         <RESOLVE
                                     >RESOLVE
                                                CLUE !
                                                            IMMEDIATE
13
14 : LEAVE ( addr1 addr2 -- addr1 addr2 )
                                                 TMMEDIATE
15
      COMPILE (LEAVE)
                         CLUE 3
                                    <RESOLVE
                                              .
```

End Listing

	DO	TEST	IF CONNEC	T L	EAVE THE	N LOOP			
This com	pile	s as:							
(DO)	14	TEST	?>BRANCH	5	CONNECT	(LEAVE)	11	(LOOP)	13
Figure One									

plies a runtime penalty whether or not LEAVE is included in a loop, and subsequent suggestions from Klaus Schleisiek and Bill Ragsdale have avoided any runtime penalty by using LOOP or +LOOP to resolve forward branch addresses left by an IMMEDIATE version of LEAVE.

Now read on...

The implementation presented here runs loops at maximum speed and also avoids any complexity in the compiletime behavior of **LOOP** or **+LOOP**. These words do not need to know about the existence of **LEAVE**. There is a minimal runtime penalty when the runtime operator for **LEAVE** is executed. The essential idea of the implementation is that the runtime operator (**LEAVE**) compiled by **LEAVE** calculates its continuation address by locating an offset which follows the runtime operator (**DO**) compiled by **DO**.

Consider figure one. The runtime operators (DO), (LOOP) and (LEAVE) are followed by unsigned eight-bit displacements. Thus, (DO) has an offset to beyond the loop, and (LOOP) has an offset back to the start of the loop.

The offset that follows (LEAVE) is back to the location following (DO). (LEAVE) performs its function by:

a) subtracting its offset from IP, leaving IP pointing to the byte that follows (DO), and

b) adding the offset that follows (**DO**) to IP, leaving IP pointing just beyond the loop.

At compile time, **DO** saves the loop start address in the variable **CLUE**, having first saved the previous value of **CLUE** on the parameter stack. **LEAVE** uses the contents of **CLUE** to calculate the offset back to the start of the loop. **LOOP** or **+LOOP** will restore the previous value of **CLUE**, which will either be its initial value or the start address of an outer loop.

The code presented here is without compiler protection, but setting the initial value of **CLUE** to zero provides a simple test for being inside a loop.

The existence of an offset following (DO) means that including a test for a

null loop is very simple and efficient. On grounds of functional correctness, too, I think this should be included in the standard, and the test is included here

The code definitions of the runtime operators are given for an 8086 implementation. For those who wish to ponder the details. SI is a sixteen-bit index register used as Forth's IP. Next is post incrementing, so that during execution of a code-level word, IP points to the following byte.

BP is a sixteen-bit index register used as Forth's RP. The return stack grows toward low memory, so that

#### 4 # BP ADD

drops two items from the return stack.

AX is a sixteen-bit accumulator with low byte AL and high byte AH. The @ symbol is used in the assembler to denote indirection, so

#### AX BP@ ADD

will add AX to the contents of the location indicated by BP, which is the top of the return stack. The instruction

2 DISP8 BP@ AX ADD adds the second return stack item to AX.

2

equivalent to

SI@ AL MOV SI INC

The offsets compiled by DO, LEAVE, **LOOP** and **+LOOP** are calculated using the 83-Standard system words > MARK. >RESOLVE. < MARK and < RESOLVE. The definitions of these words are included for completeness. They are specific to a system using eight-bit unsigned offsets.

The definitions of DO, LEAVE, LOOP and +LOOP will be usable on any system, since they use the system words to hide details of the underlying implementation.

#### Appendix: DO...LOOP Algebra

It is fascinating that a discovery such as the new loop can be made after so many programmers and compiler writers have been implementing loops for so many years. One reason why this type of thing is tricky is that sixteen-bit integer arithmetic is really two arithmetic systems combined, signed and unsigned, and in using the overflow flag to test for loop termination we are

LODS is a one-byte instruction using a facility from signed arithmetic to test for the completion of an unsigned operation. In a hardware realization of a Forth machine, of course, the internalized limit could be zero, with a "loop" flag which comes up when a sixteen-bit arithmetic operation crosses zero

> Let L and I represent the loop limit and index. We define the range R of the loop as R = L - I.

> We transform L and I to internalized values I and i, where in most systems 1 = 8000 hex.

> This transformation does not affect the range of the loop, so we can write R = 1 - i, and thus i = 1 - L + I.

> As the loop executes, there is a constant difference between i and I which can be expressed as I = i - x, which by eliminating i gives x = l - L.

> The values of x and i are calculated by the runtime operator for **DO** and are pushed onto the return stack. (See comments in (DO).) The Forth word 1 calculates the loop index using I = i - x.

### proFORTH COMPILER

#### 8080/8085, Z80 VERSIONS

- SUPPORTS DEVELOPMENT FOR DEDICATED APPLICATIONS
- INTERACTIVELY TEST HEADERLESS CODE
- IN-PLACE COMPILATION OF ROMABLE TARGET CODE
- MULTIPLE, PURGABLE DICTIONARIES
- FORTH-79 SUPERSET
- AVAILABLE NOW FOR TEKTRONIX DEVELOPMENT SYSTEMS \$2250

#### MICROPROCESSOR-BASED PRODUCT DESIGN

- SOFTWARE ENGINEERING
- DESIGN STUDIES COST ANALYSIS
- ELECTRONICS AND PRINTED CIRCUIT DESIGN
- PROTOTYPE FABRICATION AND TEST
- REAL-TIME ASSEMBLY LANGUAGE/proFORTH
- MULTITASKING
- DIVERSIFIED STAFF

#### **MICROSYSTEMS, INC.**

(818) 577-1471

2500 E. FOOTHILL BLVD., SUITE 102, PASADENA, CALIFORNIA 91107

## What Do All Have In Common?

**Hewlett Packard** AT & T Long Lines **General Electric Hughes Aircraft** Motorola **Rockwell International** U.S. Army ET & D Labs **U.S. Navy NOSC University Of California** and over 200 others . . .

Over the past three years, each has bought professional

# 68000 FORTH

systems from Creative Solutions Inc.

## Why?

MATURE RELIABLE PRODUCT First Multi-FORTH<sup>TM</sup>installation in December 1979 - installed base of over 200 sites

#### MULTITASKING

Since the beginning, Multi-FORTH™ has supported multiple background tasks and optional multiple users.

**16 OR 32 BIT IMPLEMENTATIONS** 16 bit 79 - Standard or 32 bit unlimited program size implementations available.

#### FAST ...

Eratosthenes Sieve Benchmark in under 18 seconds for 10 passes in high level.

#### **IN-LINE ASSEMBLER**

**BUILT-IN TRACE, DEBUG FEATURES** 

CORE IMAGE SNAPSHOT FEATURES Saves and restores current system image without recompiling (for turnkey appli-

cations). **EXTENSIVE DOCUMENTATION** 

Current user manual is over 350 pages:

ONLINE CAI COURSE, HELP FEATURES

GRAPHICS AND FLOATING POINT, SCREEN EDITORS, ON HP SERIES 200 **OR MOTOROLA VME/10** 

#### MOST SINGLE BOARD COMPUTERS ARE SUPPORTED

VME110, KDM, ECB, VM01, VM02, OB68K, BRI, DUAL, ERG, CP/M68K installations - 8" media.

PRICES START at \$895.00 for a SINGLE COMPUTER LICENSE (CP/M Version)

<b>OK!!!</b> I'm interested! Please send me more	information about the	e <b>Multi-FORTH</b> ™ system.
Name	_ Company	· · · · · · · · · · · · · · · · · · ·
Address		
Phone	_ Hardware Type	
Creative Solutions	Inc.	4801 Randolph Road Rockville, Maryland 20852 (301) 984-0262

# Techniques Tutorial **Multi-Tasking, Part I**

#### Henry Laxen Berkeley, California

Multi-tasking has long been one of the biggest benefits of Forth and one of its most closely guarded secrets. The fact that even crippled processors like the 8080 can be made to run four or five tasks simultaneously with little performance degradation is a testament to the efficiency both of Forth itself and of the techniques involved in implementing a multi-tasking kernel. It is time to reveal the techniques used in most Forth multi-tasking systems and to allow the user to benefit from the power such knowledge can bestow.

Now for the disclaimers. First, I am only going to discuss multi-tasking, and not multi-user, Forth systems. The difference is that in a multi-tasking Forth there is but one terminal attached to the system, hence only one person at a time is interpreting or compiling. This is considerably simpler than a multi-user Forth system with several terminals (each perhaps with its own unique characteristics), all interpreting and compiling at the same time. In our multi-tasking system, the user will be able to have several tasks running simultaneously, perhaps communicating with each other and with the terminal. but I will not get into the subtleties associated with turning it into a true multi-user environment. The second disclaimer is that in order to get some efficiency out of the system, the techniques used to implement multi-tasking are generally very machine dependent. Since my machine is an 8080, and it is me writing this article, you will either have to bear with me or ignore the article. The choice is yours.

This first installment will deal with the low-level mechanism involved in task switching, and the structures that must be in place in Forth in order to make multi-tasking possible. The second article on this subject will talk about creating and manipulating tasks. While the code I present is oriented toward an 8080, I will describe its function. You should be able to translate it into code for your processor without much pain.

Now then, let me first describe the philosophy behind multi-tasking in Forth. Unlike traditional multi-tasking systems, which interrupt the currently running task at a completely arbitrary time and initiate another task unbeknownst to the first one. Forth requires tasks to cooperate. While each task does not know details about the other tasks, it must at least be aware of them, or else the system will revert to a singletask existence. Each task must explicitly give up control of the CPU at certain points while it is running. The Forth kernel does so whenever it is about to perform an I/O operation, such as reading or writing to the terminal or mass storage device. If the user creates a task that does no I/O of its own, then he must explicitly give up control or else as soon as that task is activated, it will take total control of the machine. Each of the two approaches mentioned above has merit, and I will briefly discuss the good and bad points of each.

The main advantage of traditional multi-tasking systems is that the programmer does not need to be aware of their existence. As far as he is concerned, the program he writes is just running slower. He does not need to modify it in any way from a single-user environment. There is, of course, a cost associated with this benefit, and that is one of performance. Since the operating system must absolutely guarantee that the state of the system is undisturbed between one running of the task and another, an extremely complex process usually is required to save and restore a task. Since the task is unaware that it is being removed from control, the operating system may grab it at a particularly inopportune moment, and the amount of information that must be saved and restored can be staggering. This is why performance on such systems typically degrades rather dramatically as soon as several tasks are running concurrently. The main advantage of the Forth approach to multi-tasking is that the overhead of task switching is extremely small.

Thus, many tasks can be run simultaneously with little performance degradation. The disadvantage is that an additional burden is placed on the programmer. He must follow some rules that apply in multi-tasking systems, as well as, perhaps, modify his code to take advantage of multi-tasking. My conclusion from this synopsis is that traditional multi-tasking is great on very large systems where tens or hundreds of tasks are running simultaneously and the computer hardware helps you. On microcomputers and small systems, the traditional approach simply does not apply, and the benefits of

VARIABLE UP ( Points to the currently active user base address )

;

: USER CREATE , DOES> @ UP @ +

**Figure One** 

0 #USER ! ( Holds the size of the user area ) VARIABLE #USER VOCABULARY USER USER DEFINITIONS CREATE #USER @ , @ UP @ + DOES> : CREATE ; ALLOT #USER +! VARIABLE CREATE 2 ALLOT ; FORTH DEFINITIONS **Figure Two** 

Volume V, No. 4

the Forth approach greatly outweigh the drawbacks.

The basic mechanism Forth uses is simply to define an ordinary Forth word, usually called **PAUSE**, which relinquishes control of the CPU from the currently running task and gives it to the next task that is ready to run. **PAUSE** takes nothing from the stack and returns nothing, and disturbs the system in a well-defined way of which the user must be aware. What PAUSE actually does is to examine a linked list of tasks that may or may not be ready to run. The first ready task it finds is given control of the CPU and is allowed to run until it executes a PAUSE of its own. The linked list is circular, so eventually we will get back to the first task in the list and run it again, with execution continuing immediately after the **PAUSE** word. By agreement, tasks shall not disturb the state of the system except with regard to block buffers. Thus, each task may not assume that the buffer it is using is still present after a **PAUSE** has been executed. This minor restriction greatly simplifies the job of saving and restoring the state of the system between task activations.

Wait a minute, you say, what about the many system variables that tasks may use while they are running. For example, if a background task is doing print spooling while you are editing a screen, both tasks are accessing variables such as **OUT**, **BASE**, **HLD**, etc.

each task could change these and affect other tasks. Fortunately, there is an extremely elegant way to prevent this which has traditionally been known as USER VARIABLE in Forth. The idea is simple, namely, just group together those variables which each task must have to itself. These variables become offsets from some base address. When these variables are executed, they must add their offset to the base address of the current task. Thus, to switch tasks one need only change the base address from whence these variables originate, instead of copying the values themselves to some safe area. The portion of memory reserved for these variables is called the USER area. There are many different ways of implementing this concept, and I would like to present a new one here which I believe has great merit. Traditionally. USER was a defining word which took as an argument an offset from the base address and assigned a name to that offset. At runtime, the offset was added to the base of the current user area, which was contained in a regular variable, and that address was placed onto the parameter stack. This is simple, but has some disadvantages. It is difficult to insert a new USER variable into the middle of existing ones with this implementation. It also forces the user to be aware and do arithmetic in order to maintain the user area. The old im-

Things would get very confusing if plementation was as shown in figure each task could change these and affect one.

A much more flexible approach is to make **USER** a vocabulary, and redefine those words which may be needed on a task level. Consider the implementation in figure two.

Now you need no longer keep track of where each variable is going and how much space has been used. Also, arrays are much easier to create, and I think it reads much more nicely. With the old approach, you would have to say **34 USER BASE** to define a user variable called "base," and you must know that location 34 is available for use. With the new scheme, you simply type **USER VARIABLE BASE**, which reads very nicely indeed.

Now then, suppose we have such things as **USER** variables, regardless of exactly how they were defined. In particular, I will need three such variables as follows:

**USER VARIABLE TOS** (Holds top of stack when switching tasks.)

**USER VARIABLE ENTRY** (Contains machine code and task status.)

**USER VARIABLE LINK** (Points to next task in a circular list.)

Let's examine the role of each of these a little more closely. **TOS** is simply going to hold the value of the top of the parameter stack for this task, when it gives up control of the CPU to the next task. Since each task must have its own local stack in order to do just about anything, this value must be saved and restored between successive activations of a task. ENTRY in our implementation will contain machine code that will either jump to the next task in the list if the current one is not ready to run yet, or it will jump to some activation code that will bring this task to life once again. Finally, LINK points to the ENTRY field of the next task in the circular list. The only tricky part of this is how to fit the code that decides whether or not to activate this task and either continue or restore all of the task's parameters, in the two bytes reserved for ENTRY. It just so happens that, on the 8080, two bytes is more than enough and, in fact, one would suffice.

CODE PAUSE (S -- ) IP PUSH ( Push the current interpreter pointer onto stack ) ( and the current return stack pointer ) RP LHLD H PUSH 0 H LXI SP DAD XCHG ( Stack pointer now in DE ) ( Points to TOS, which is first entry ) H INX D M MOV H INX ( Move stack po UP LHLD ( Move stack pointer to TOS ) E M MOV H INX PCHL ( Jump to next task ) C:

Pause on the 8080 Figure Three

CODE RESTART (S -- )
 ( Since a RST instruction has just been executed, the address
 UP + 3 is now on the stack )
-3 H LXI D POP D DAD UP SHLD ( Set up new USER area )
M E MOV H INX M D MOV XCHG SPHL ( Restore parameter stack )
H POP RP SHLD ( Restore Return Stack Pointer )
IP POP ( Restore the IP ) NEXT JMP C;

**Figure Four** 

The 8080 has several one-byte instructions called RST instructions. When these are executed, they push the value of the program counter on the stack and jump to a specified location in low memory. Thus, the trick on the 8080 is to put either an RST or a JMP into the ENTRY point. An RST instruction will cause this task to be activated. while a JMP instruction will jump immediately to the ENTRY point of the next task in the list. Remember that the contents of LINK point to the ENTRY point of the next task in the list. So to make a task active, an RST instruction is placed into ENTRY while to deactivate a task an NOP instruction is placed into ENTRY. The JMP instruction is always present in ENTRY + 1. This is wasteful, I know, but what the hell. Now then, all we have to do is understand what exactly happens when we do a **PAUSE** and a task activation. Let's first look at what **PAUSE** does on the 8080. (See figure three.)

**PAUSE** is in charge of saving the current task's status and jumping to the next task in the circular list. Notice how little information needs to be

saved during a task switch. Only the current value of the IP, the return stack depth, and the parameter stack depth is saved. Note that the IP and the return stack depth have been pushed onto the parameter stack, so it will be the duty of the **RESTART** word to pop these off so that the stack depth is unchanged. Now let's take a look at **RESTART** in figure four, which must restart a task where it left off, namely just after executing a **PAUSE**.

Remember that the RST instruction is a one-byte call to a fixed address. Thus, it pushes the address of the current user area plus three onto the current stack. This information is used to restore the user area for the task that is now being restarted. Once the base of the user area is computed, the parameter stack is restored and then the return stack and the interpretive pointer. Thus, **RESTART** has undone what was done by **PAUSE**, and resumed execution with the word following **PAUSE**, as though nothing has happened.

I hope this has shed a little light on what goes on in a multi-tasking system.

Next time, we will explore how to create and manipulate tasks, now that we understand the task-switching mechanisms involved. Until then, good luck, and may the Forth be with you.

Copyright © 1983 by Henry Laxen. All rights reserved. The author is Chief Software Engineer for Universal Research, 150 North Hill Drive #10, Brisbane, CA 94005, specializing in the development of portable computers.

#### **Letters** (Continued from page 4)

fact, any reference to an operating system at all (except for terminal interfacing words like **EMIT**), as far as the standard is concerned. That is most definitely not to say that the screen system is not a good one. In fact, it might be the best system for particular applications. But I think Forth should emulate the attitude of C and Modula II in that respect. Separate the design and implementation of the operating system from the language standard.

The next issue on my mind is that of strings. Possibly one of the FIG study groups is taking care of my concerns; but it would make me feel more comfortable to see some debate, at the level of fundamentals, in *Forth Dimensions* before things get too far along.

BASIC has taught us that good string handling is one of the very important components of interactiveness. The secret of (Microsoft) BASIC's friendliness with strings is that you never have to specify the length of a string, and the secret of that is BASIC's string-space garbage collector. Here is an opportunity for Forth, because, however they are implemented, Forth's string variables are likely to be dispersed randomly in the code, rather than being assigned to a well-defined memory region; and that makes garbage collection impossible without both forward and backward links between string variables and string values in string space. This is extra memory overhead, compared to BASIC, but the advantage is that it makes very fast garbage collection possible.

String handling is very important, and Microsoft BASIC has shown us that it can be done extremely well in small-scale systems. We should accept nothing less for Forth.

Finally, let me comment on the editor's reply to a letter in *Forth Dimensions* (Vol. IV, No. 6, page 25), pointing out a design bug in a FIG release. The response was to the effect that the bugs were known, and had been fixed in the implementations of various vendors, who have more resources for that kind of maintenance than FIG.

I have to chide you a little for that. We all appreciate the tremendous generosity of the people behind FIG, who put their public-domain philosophies where their mouths are, at the cost of that most valuable commodity, personal time. But the situation is surely analogous to that of publishing research in scientific journals. As a theoretical physicist, it is my professional and ethical responsibility to publish an erratum if I become aware of a significant mistake in any of my published work. I think the same should apply to FIG publications. If you know of a bug, you should publish an erratum.

I hope that, in the interest of pruning my remarks to a few, I have not conveyed an overly critical impression. I really get a sense, from two years' worth of *Forth Dimensions*, that the community is making solid advances. Although not a Forth fanatic, let me

(Continued on next page)

```
Screen # 32
 0 ( Calander Development, Screen 1 of 3 )
   : CTABLE (BUILDS 0 DO C, LOOP DOES) + C3;
31 30 31 30 31 31 30 31 30 31 29 31 0
31 30 31 30 31 31 30 31 30 31 28 31 0
 1
 з
 4 26 CTABLE DAYS-IN-MONTH ( month -- days in month )
 5
   : IS-LEAP-YEAR? ( year -- flag )
 6
     DUP DUP 400 MOD 0=
                                     \overline{(} if year divisable by 400 )
 7
               100 MOD 0= NOT OR
 8
      SHAP
                                    <
                                          or not divisable by 100 )
 9
      SWAP
                 4 MOD 0= AND ;
                                        and divisable by 4 then leap )
                                     ζ.
10
11
12
13
14
15 --->
   Screen # 33
 Ø
   ( Calander Development, Screen 2 of 3 )
   ; DAY-DF-YEAR ( day, month, year -- day of year )
 1
     IS-LEAP-YEAR?
 2
 з
     IF 13 + 14
                               ( if leap year convert offsets )
 4
     ELSE 1 ENDIF
                               ( else start at month 1 )
 5
     2DUP = 1F 2DROP
                               ( if month is january, return day )
             ELSE DO I DAYS-IN-MONTH + LOOP ( else cal day of year )
 6
   ENDIF ;
: DAY-OF-WEEK ( day, month, year -- day of week, sunday is 8 )
 8
     DUP >R DAY-DF-YEAR 0 ( get current day of year, current day )
R> 1980 DD ( start at year 1988)
 9
10
        I IS-LEAP-YEAR? IF 366 + ELSE 365 + ENDIF
11
     LOOP +
12
                               ( add in day of this year )
     7 MOD ;
13
                               ( mod by 7 for day of week )
14
      ( Note: works from year 1901 to 2100 )
15
   -->
   Screen # 34
 0
   ( Calander Development, Screen 3 of 3 )
   : CALANDER ( month, year -- ) ( print months calander )
 1
                            . CR
     2DUP SWAP . ."
 2
```

Sun Mon Tue Wen Thr Fri Sat" CR

13 1 DO DUP I SWAP CALANDER LOOP DROP ;

2DUP 1 ROT ROT DAY-OF-WEEK

DUP 2\* 2\* SPACES

ROT ROT

1+ 1 DO

I4.R

7 1983 CALANDER 7 - 1983 Sat Sun Mon Tue Wen Thr Fri 2 1 З 4 5 7 8 9 6 10 11 12 13 14 15 16 17 19 20 23 18 21 22 24 25 26 27 28 29 30 31

IS-LEAP-YEAR? IF 13 + ENDIF DAYS-IN-MONTH ( get days in month )

1+ DUP 7 = IF CR ENDIF 7 MOD ( if sat then cr )

11 LOOP 0= NOT IF CR ENDIF CR ; ( output ending crs )
12 : CALANDER-YEAR ( year -- ) ( print calander for whole year )

end by saying that I find FIG a uniquely valuable enterprise. David N. Williams 1238 Westport

Ann Arbor, MI 48103

#### **Code for All Seasons**

Dear FIG,

At last year's Forth convention in San Jose, one speaker mentioned that functions to print out the calendar would be useful. Though I am not an expert in Forth yet, these functions seemed simple, and I have developed the code shown in the accompanying screens. I am using a Forth developed from the FIG model; it should be simple to convert these functions to Forth-79.

I would personally like to see some routines that would display a comment associated with a Forth word. I find it very awkward to look up uncommon definitions each time I need to use one. Currently, I am just not familiar enough with Forth to implement this. Does anyone have some ideas?

Thank you very much,

Jesse Jay Wright 164 N. Oak Knoll #8 Pasadena, CA 91101

#### FIG Aficionado

Dear Editor,

Re: the contents of Forth Dimensions, you find me delighted to see true FIG-Forth screens reappearing. After all, FIG-Forth is our group's own version of Forth, and though the "standard" (unfortunately, I think) gets changed every four years or so (this is bound to scare off many a potential commercial user worried about software maintenance — a standard that changes is a contradiction in terms!), FIG-Forth escaped unscathed. In contrast, a Forth-79 screen written only one year ago won't necessarily load on a Forth-83 system. Try to explain this to a business user who just got coaxed into spending a few thousand bucks on a Forth-83 system with promises of a wealth of ready-made software at large, and who discovered that it will require considerable amounts of rewriting before he can use it. The same

з

4

5

6

8

9

10

13

14

15

( find day of 1st day of month )

( stack: dayofweek, month, year )

( space over in output )

( out day )

applies to Forth-79 users who won't be able to load all these '83 screens which, no doubt, will appear before long. If new "standards" keep popping up like this, there will soon be as many Forth dialects as there are BASICs, and the intended source portability will be nothing but a hollow phrase.

A while ago, I obtained a Z-80 FIG-Forth listing from Dennis Wilson of Aristotelian Logicians. To do better justice to the Z-80 bit, I re-worked parts of it. I got lots of fun out of rewriting code definitions, ending up with fewer bytes used and a faster execution time to boot. As a result, the Primes benchmark (*Forth Dimensions*, Vol. II, No. 4) executes 1000 primes in fifty-four seconds, and the *BYTE* test (September 1981) runs in just eight seconds. I'd very much like to hear from anyone who can do better than this on a 4Mz, Z-80 system. I will gladly supply the source text to any interested person who sends a 5 1/4 '' double-sided, single-density diskette (or \$5) and return postage.

Yours Forthfully,

Edmund Ramm Postfach 38D-2358 Kaltenkirchen, West Germany

# **New Product Announcements**

Forth Dimensions welcomes press releases and product announcements, as well as reader letters regarding product performance. Addresses of the distributors and manufacturers mentioned in this column may be found in the Vendors List.

Perkel Software Systems has announced that its **Marx Forth v1.4** is now a public-domain product. Included is its target compiler system which allows applications to be run as stand-alone machine code files that don't require a Forth system to run. Source listings and forty-page manual cost \$35, disk version \$150; for North-Star, CP/M, Atari, Radio Shack. Those interested in developments related to a "universal compiler" with the ability to compile code from Forth, Pascal, C and BASIC may write to Perkel Software Systems.

The **Pocket Guide to Forth** lists Forth words in ASCII order, along with definitions and stack diagrams. Gives FIG-Forth and Forth-79 correlations. Available for \$7.00 from Mountain View Press (\$7.25 elsewhere).

Forth Inc. has released a schedule of Forth classes for the period from November 1983 to February 1984. Included are an intensive introduction covering vectored execution, array handling, sealed vocabularies, data typing, data formatting and management; advanced instruction detailing multi-tasking, serial device drivers, interrupt routines, **BLOCK** device drivers and target compile applications culminating in a running target image; and data-base design concentrating on the techniques needed to store data on disk and to design reports, user interface and security, indexing methods, and data description and access tools. Courses are five days in length, consist of both lecture and hands-on learning, and cost between \$750 and \$950 for individual enrollment.

Several printed listings which include the **83-Standard** are available from MicroMotion for \$15 each. Also available are 6502, 8080 and 8086 source listings which support both the new standard and the 83 model by Laxen and Perry. Call or write them for a free **83-Standard Programming Reference Card**.

Ziggurat Software announced applications for HES' FIG-Forth for the VIC-20. Included are additional FIG-Forth words not in the HES implementation, printer utilities for the VIC printer, a case statement, arrays and strings, and disk utilities. Available on cassette or disk.

Innovatia Laboratories offers three products. FMS is a **text formatter** which permits storage of text or direct output (typewriter-like) to printer, and permits access to all upper- and lowercase Greek letters and forty-six math and special symbols. FLH provides **LISP-like list-handling** in Forth and a small expert system written in FLH. FWG, a firmware generator, creates **ROMable Forth code**, with or without headers, for target compilers.

Forth for the **Texas Instruments** 99/4A can be purchased from Wycove Instruments Ltd. It requires at least 32K of additional memory and one of the following: Editor/Assembler, Mini Memory or Extended BASIC. A cassette version is available. Sprites, sound, floating-point arithmetic and peripheral access are supported. The \$50 price includes a short introduction to Forth, complete description of the included Forth words, hardware notes and a sample game. Beginners are referred to *Starting Forth* by Leo Brodie.

# **Chapter News**

John D. Hall Oakland, California

We have three new chapters! They are:

Fox Valley FIG Chapter Batavia, Illinois Philadelphia Area FIG Chapter Philadelphia, Pennsylvania Houston FIG Chapter Houston, Texas

The following areas do not have FIG chapters, even though there are sufficient FIG members to form them. I know that many members in these areas are interested, but someone will have to make an effort!

Atlantic City, NJ
Corvallis, OR
Eugene, OR
Columbus, OH
Cincinnati, OH
Norfolk, VA
Roanoke, VA
Richland, WA
Madison, WI
Milwaukee, WI

#### Australia Chapter

At the meeting on July 1, after a question and answer session, the discussion was devoted to screen transfer, with some heavyweight thinking about the ISO seven-layer model. On August 5, the same topic was featured, and at the end of the meeting, several people decided to actually do something about writing code and making cables to actually transfer some data.

#### Los Angeles Chapter

There was a talk on August 27 about the current state and future trends of software and hardware development for personal and business computers in Japan. Martin Tracy discussed techniques and implementations for ROMable Forth systems. Barry Cole explained how he implemented a quick booting Forth on a portable computer.On September 24, Nathaniel Grossman described his method for

implementing nine decimal place, binary logarithms in fixed-point Forth. John Hall, FIG Chapter Coordinator, gave a report on the activities of other FIG chapters. There was some discussion on the way to increase communication between chapters. Later, the group talked about implementation of Forth-83, which may lead to a model. Martin Tracy demonstrated and described the MicroMotion implementation of Forth-83. Greg Stevenson discussed a method of speeding dictionary searches independent of the existing method. At the October 22 meeting, Nathaniel Grossman spoke on finding 16-bit square, cube, etc. roots by Newton's method, in fixed-point Forth. Bob Jaffray presented a simple method to provide execution security by testing for valid CFAs as a patch to NEXT. Steven Lewis talked about **VERIFY**, a word to document effects of the execution of a Forth word. Jim White described an implementation of LISP in Forth. Ken Inouye demonstrated Forth on the new Sharp 16-bit CPU computer. There were also reports from members who attended the National Forth Convention. About thirty-five people attended this chapter meeting and enjoyed it very much.

#### Iowa Chapter (in formation)

August 23, the group saw a presentation of a simple but fast LIFE implementation on the Commodore 64 (using C64–Forth from Performance Micro Products) by Robert Benedict, Assistant Professor of Mechanical Engineering. One generation in Forth was 1.6 seconds vs. 126 seconds in BASIC. Another demonstration by Scott Evans, an electrical engineering student, showed some Forth utilities developed for the Commodore 64, used as a controller to move a hydrophone in a plane, for data acquisition.

Eugene Johnson, a professor of mathematics, presented a series of words on September 27 to do matrix operations. He uses these in his courses

on linear algebra. Michael Ham showed some words that simply ignore invalid input. These are useful when the range of valid input is obvious, as from a menu. The effect for the user is that the only keys that work are those for valid input.

#### **Orange County Chapter**

Roy Martens paid a brief visit at the August 24 meeting, and discussed what was happening with the Northern California Chapter, and the state of the Forth-83 standard. Lee Jordan presented a paper on a 6502 disassembler. Lee is a beginning Forth programmer and was unaware of the present state of the art of Forth disassemblers. He put a lot of work into it; one learns by doing. At a short meeting on September 7, Wil Baden presented his text formatter, which was a model done in Pascal.

#### Northern California Chapter

At the October 22 morning FORML meeting, Doug Dillon presented code for a Modem-7 environment. Doug's code was written in Forth-83, using the Laxen and Perry implementation. Michael Stolowitz presented code on a B-tree and file-indexing method, also using F83. In the afternoon, Bob Reiling, Forth Convention Chairman, gave a review of the convention and reported an attendance of over 1200. Larry Forsley and Thea Martin, from Rochester, New York, spoke about the latest state of affairs at the Institute for Applied Forth Research, which they direct. Larry mentioned the next conference in Rochester, to be held June 5 through 9. The topic is to be Forth Applications, with one day devoted to real-time systems. Mike Perry gave a demonstration of a Forth-based CP/M BIOS generator, and discussed the virtues of Forth-83 in combination with the F83 implementation.

Support your local chapter!

John D. Hall is the Chapter Coordinator for the Forth Interest Group and is a consulting programmer.

#### **Chapters in Formation**

Here are more of the new chapters that are forming. If you live in any of these areas, contact one of these people and offer your support in forming a FIG chapter.

Contact:

Charles Samuels 7805 Linda Lane Anchorage, AK 99502

Doug Dillon California Cedar Products P.O. Box 8449 Stockton, CA 95208 209/931-2448

Robert McFarland DIGILOG Corp. Box 3315 Ventura, CA 93006

### **Fig Chapters**

#### U.S.

#### ARIZONA

Phoenix Chapter Call Dennis L. Wilson 602/956-7678

#### • CALIFORNIA

Los Angeles Chapter Monthly, 4th Sat., 11 a.m. Allstate Savings 8800 So. Sepulveda Boulevard Los Angeles Call Phillip Wasson 213/649-1428

Northern California Chapter Monthly, 4th Sat., 1 p.m. FORML Workshop at 10 a.m. Palo Alto area. Contact FIG Hotline 415/962-8653

#### Orange County Chapter

Monthly, 4th Wed., 7 p.m. Fullerton Savings Talbert & Brookhurst Fountain Valley Monthly, 1st Wed., 7 p.m. Mercury Savings Beach Blvd. & Eddington Huntington Beach Call Noshir Jesung 714/842-3032

#### San Diego Chapter

Weekly, Thurs., 12 noon. Call Guy Kelly 619/268-3100 ext. 4784 Alan B. Cohen 14 Candlelight Dr. Danbury, CT 06810

Alan H. Lake Bankers First P.O. Box 1332 Augusta, GA 30913

Michael Ham 3110 Alpine Ct. Iowa City, IA 53340 319/337-1353

Manfred Peschke Intersystems Mgmt. & Consult. Story Hill Rd. RFD 3 Dunbarton, NH 03045

Gary Bergstrom 191 Miles Rd. Chagria Falls, OH 44022 Steve Buffone 621 Center Ave. Cuyahoga Falls, OH

Joel A. Neely Interface Solutions, Inc. Box 11167 Memphis, TN 38111

Jim Watson 801 Orleans Corpus Christi, TX 78418

Pete Koza 9671 NE 122 Place Kirkland, WA 98033

Zafar Essak P.O. Box 46263 Vancouver, BC V6R 4G6 Canada

MISSOURI

816/363-1024

314/867-4482

• NEVADA

Suite 900

Las Vegas, NV

702/452-3368

Call Gerald Hasty

• NEW JERSEY

**New Jersey Chapter** 

Call George Lyons

201/451-2905 eves.

New York Chapter

212/261-3213 eves.

**Rochester Chapter** 

Univ. of Rochester

Call Thea Martin

716/235-0168

Hutchison Hall

Monthly, 2nd Wed., 8 p.m.

212/432-1414 ext. 157 days

Monthly, 4th Sat., 2 p.m.

• NEW YORK

Queens College

Call Tom Jung

Kansas City Chapter

Call Terry Rayburn

St. Louis Chapter

Thornhill Branch of

Call David Doudna

Monthly, 3rd Tue., 7 p.m.

St. Louis County Library

Southern Nevada Chapter

101 Convention Center Drive

Syracuse Chapter Call C. Richard Corner 315/456-7436

Norbert Heindl

Berlichingstrasse 9

8540 Schwabach

West Germany

P.J. Reynolds

P.O. Box 4853 Cape Town 8000

South Africa

reflecta-electronic gmbh

Murray & Roberts Bldg.

#### • OHIO

Athens Chapter Call Isreal Urieli 614/594-3731

Dayton Chapter Twice monthly, 2nd Tues & 4th Wed., 6:30 p.m. CFC, 11 W. Monument Ave. Suite 612 Dayton, OH Call Gary M. Granger 513/849-1483

#### • OKLAHOMA

Tulsa Chapter Monthly, 3rd Tues., 7:30 p.m. The Computer Store 4343 South Peoria Tulsa, OK Call Art Gorski 918/743-0113

#### • OREGON

**Greater Oregon Chapter** 

Monthly, 2nd Sat., 1 p.m. Computer & Things 3460 SW 185th, Aloha Call Timothy Huang 503/289-9135

#### PENNSYLVANIA

Philadelphia Chapter Monthly, 3rd Sat. LaSalle College, Science Bldg. Call Lee Hustead 215/539-7989

#### • COLORADO

**Denver Chapter** Monthly, 1st Mon., 7 p.m. Call Steven Sarns 303/477-5955

#### • ILLINOIS

Fox Valley Chapter Call Samuel J. Cook 312/879–3242

Rockwell Chicago Chapter Call Gerard Kusiolek 312/885-8092

#### KANSAS

Wichita Chapter (FIGPAC) Monthly, 3rd Wed., 7 p.m. Wilber E. Walker Co. 532 S. Market Wichita, KS Call Arne Flones 316/267-8852

#### • MASSACHUSETTS

Boston Chapter Monthly, 1st Wed., 5 p.m. Mitre Corp. Cafeteria Bedford, MA Call Bob Demrow 617/688-5661 after 7 p.m.

#### MINNESOTA

MNFIG Chapter Monthly, 1st Mon. 1156 Lincoln Avenue St. Paul, MN Call Fred Olson 612/588-9532

#### • TEXAS

Dallas/Ft. Worth Metroplex Chapter Monthly, 4th Thurs., 7 p.m. Software Automation, Inc. 14333 Porton, Dallas Call Marvin Elder 214/392-2802 or Bill Drissel 214/264-9680

Houston Chapter Call Dr. Joseph Baldwin 713/749-2120

#### • VERMONT

Vermont Fig Chapter Monthly, 4th Thurs., 7:30 p.m. The Isley Library, 3rd fl. 3rd Floor Meeting Room Middleburynes, VT Call Hal Clark 802/877-2911 days 802/452-4442 eves

#### • VIRGINIA

Potomac Chapter Monthly, 1st Tues., 7 p.m. Lee Center Lee Highway at Lexington St. Arlington, VA Call Joel Shprentz 703/437-9218 eves.

#### FOREIGN

#### • AUSTRALIA

Australia Fig Chapter Contact: Ritchie Laird 25 Gibsons Road Sale, Victoria 3850 051/44-3445

FIG Australia Chapter Contact: Lance Collins 65 Martin Road Glen Iris, Victoria 3146 03/29-2600

Sydney Chapter Monthly, 2nd Fri., 7 p.m. John Goodsell Bldg., Rm LG19 Univ. of New South Wales Sydney Contact: Peter Tregeagle 10 Binda Rd., Yowie Bay 02/524-7490

#### • BELGIUM

Belgium Chapter Contact: Luk Van Loock Lariksdreff 20 2120 Schoten 03/658-6343

#### • CANADA

Nova Scotia Chapter Contact: Howard Harawitz P.O. Box 688 Wolfville, Nova Scotia B0P 1X0 902/542-7812

#### Southern Ontario Chapter

Monthly, 1st Sat., 2 p.m. General Sciences Bldg, Rm 312 McMaster University Contact: Dr. N. Solntseff Unit for Computer Science McMaster University Hamilton, Ontario L8S 4K1 416/525-9140 ext. 2065

#### • COLOMBIA

Colombia Chapter Contact: Luis Javier Parra B. Aptdo. Aereo 100394 Bogota 214-0345

#### • ENGLAND

Forth Interest Group -- U.K. Monthly, 1st Thurs., 7 p.m., Rm. 408 Polytechnic of South Bank Borough Rd., London Contact: Keith Goldie-Morrison 15 St. Albans Mansion Kensington Court Place London W8 5QH

#### • ITALY

FIG Italia Contact: Marco Tausel Via Gerolamo Forni 48 20161 Milano 02/645-8688

#### • SWITZERLAND

Contact: Max Hugelshofer ERNI & Co. Elektro-Industrie Stationsstrasse 8306 Bruttisellen 01/833-3333

#### • TAIWAN

Taiwan Chapter Contact: J.N. Tsou Forth Information Technology P.O. Box 53-200 Taipei 02/331-1316

#### **SPECIAL GROUPS**

Apple Corps FORTH Users Chapter Twice Monthly, 1st & 3rd Tues., 7:30 pm 1515 Sloat Boulevard, #2 San Francisco, CA Call Robert Dudley Ackerman 415/626-6295

Baton Rouge Atari Chapter Call Chris Zielewski 504/292-1910

FIGGRAPH Call Howard Pearlmutter 408/425-8700

#### Vendors (Continued from page 35)

Triangle Digital Services, Ltd. 100A Wood St., Walthamstow London E17 3HX England 01-520-0442 Telex 262284

Application Packages Only See System Vendor Chart for others

Curry Associates P.O. Box 11324 Palo Alto, CA 94306 415/322-1463

InnoSys 2150 Shattuck Ave. Berkeley, CA 94704 415/843-8114

**Consultation & Training Only** See System Vendor Chart for others

Bartholomew, Alan 2210 Wilshire Blvd. #289 Santa Monica, CA 90403 213/394-0796

Boulton, Dave 581 Oakridge Dr. Redwood City, CA 94062

Brodie, Leo 17714 Kingsbury St. Granada Hills, CA 91344 213/368-3677

Eastgate Systems Inc. P.O. Box 1307 Cambridge, MA 02238

Girton, George 1753 Franklin Santa Monica, CA 90404 213/829-1074

Go FORTH 504 Lakemead Way Redwood City, CA 94062 415/366-6124

Harris, Kim R. Forthright Enterprises P.O. Box 50911 Palo Alto, CA 94303 415/858-0933

Intersystems Management Computer Consultancy Story Hill Rd. RFD3 Dunbarton, NH 03045 603/774-7762

Laxen, Henry H. 1259 Cornell Ave. Berkeley, CA 94706 415/525-8582

McIntosh, Norman 2908 California Ave., #3 San Francisco, CA 94115 415/563-1246

Metalogic Corp. 4325 Miraleste Dr. Rancho Palos Verdes, CA 90274 213/519-7013 Peschke, Manfred Intersystems Mgmt. & Consult. Story Hill Rd. RFD 3 Dunbarton NH 03045 603/774-7762

Petri, Martin B. Computer Consultants 16005 Sherman Way Suite 104 Van Nuys, CA 91406 213/908-0160

Redding Co. P.O. Box 498 Georgetown, CT 06829 203/938-9381

Schleisiek, Klaus Eppendorfer Landstr. 16 D 2000 Hamburg 20 West Germany (040)480 8154

Schrenk, Dr. Walter Postfach 904 7500 Karlstruhe-41 West Germany

Software Engineering 6308 Troost Ave. #210 Kansas City, MO 64131 816/363-1024

Softweaver P.O. Box 7200 Santa Cruz, CA 95061 408/425-8700

Timin, Mitchel 3050 Rue d'Orlean #307 San Diego, CA 92110 619/222-4185

Technology Management, Inc. 1520 S. Lyon St. Santa Ana, CA 92705 714/835-9512

#### **FORTH System Vendors**

#### (by Category)

(Codes refer to alphabetical listing e.g., A1 signifies AB Computers, etc.)

#### **Processors**

1802	C1, C2, F3, F6, L3
6502 (AIM, KIM, SYM)	R1, R2, S1
6800	C2, F3, F5, K1, L3, M6, T1
6801	P4, T1
6809	C2, F3, L3, M6, S11, T1
68000	C2, C4, D1, E1, F3, K1, T1
68008	P4, T1
8080/85	A5, C1, C2, F4, I5, L1, L3, M3,
	M6, R1, T3
Z80/89	A3, A5, C2, F3, F4, I3, L1, M2,
	M3, M5, N1, T3
Z80000	13
8086/88	C2, F2, F3, L1, L3, M6
9900	E2. L3

#### **Operating Systems**

A3, A5, C2, F3, I3, L3, M1, M2,
M6, T3
C2, F3
T1
T1
F3

#### Computers

Alpha Micro	P3, S3
Apple	A4, F3, F4, I2, I4, J1, L4, M2,
	M6, M8, 02, 03
Atari	M6, P2, Q1, V1
Compaq	F3, M5
Cromemco	A5, M2, M6
DEC PDP/LSI-11	C2, F3, L2, S3
Heath-89	M2, M6, M7
Hewlett-Packard 85	
Hewlett-Packard 9826/36	C4
IBM PC	A8, C2, F3, L1, M5, M6, Q2, S9
	W2
IBM Other	L3, W1
Kaypro II/Xerox 820	M2
Micropolis	A2, M2, S2
North Star	15, M2, P1, S7
Nova	C5
Ohio Scientific	A6, B1, C3, O1, S6, T2
Osborne	M2
Pet SWTPC	A1, A6, B1, C3, O1, S6, T2, T5
Poly Morphic Systems	A7
TRS-80 I, II, III, XVI	15, M2, M5, M6, S4, S5, S10
TRS-80 Color	A3, A8, F5, M4, S11, T1
Vector Graphics	M2

#### **Other Products/Services**

Applications	F3, P4
Boards, Machine	F3, M3, P4, R2
Consultation	C2, C4, F3, N1, P4, T3, W1
Cross Compilers	C2, F3, I3, M6, N1, P4, T1
Products, Various	A5, C2, F3, I5, S8, W2
Training	C2, F3, I3, P4, W1

#### FORTH Vendors (Alphabetical)

The following vendors offer FORTH systems, applications, or consultation. FIG makes no judgment on any product, and takes no responsibility for the accuracy of this list. We encourage readers to

FORTH Systems	В
A 1. AB Computers 252 Bethlehem Pike	<ol> <li>Blue Sky Products 729 E. Willow Signal Hill, CA 90806</li> <li>Business Computing Press</li> </ol>
Colmar, PA 18915 215/822-7727	Suite 289
2. Acropolis 17453 Via Valencia San Lorenzo, CA 94580 415/276-6050	Santa Monica, CA 90403 213/394-0796
<ol> <li>Applied Analytics Inc. 8910 Brookridge Dr., #300 Upper Marlboro, MD 20870</li> </ol>	с
5. Aristotelian Logicians 2631 E. Pinchot Ave. Phoenix, AZ 85016	1. Capstone Computing, Inc. 5640 Southwyck Blvd., #2E Toledo, OH 43614
7. Abstract Systems, etc. RFD Lower Prospect Hill Chester, MA 01011	419/866-5503 2. Chrapkiewicz, Thomas 16175 Stricker
8. Armadillo Int'l Software	East Detroit, MI 48021
P.O. Box 7661 Austin, TX 78712 512/459-7325	3. CMOSOFT P.O. Box 44037 Sylmar, CA 91342

Vendors may send additions and corrections to the Editor, and must include a copy of sales literature or advertising.

keep us informed on availability of the products and services listed.

- 4. COMSOL, Ltd. Treway House Hanworth Lane Chertsey, Surrey England KT16 9LA
- 5. Consumer Computers 8907 La Mesa Blvd. La Mesa, CA 92041 714/698-8088
- 6. Creative Solutions, Inc. 4801 Randolph Rd. Rockville, MD 20852 301/984-0262
- 7. Curry Associates P.O. Box 60324 Palo Alto, CA 94306

E

1. Elcomp Publishing, Inc. 53 Redrock Lane Pomona, CA 91766 714/623-8314 Telex 29 81 91

- 2. Elcomp-Hofacker Tegernseerstr. 18 D-8150 Holzkirchen West Germany 08024/7331 Telex 52 69 73
- 3. Emperical Research Group P.O. Box 1176 Milton, WA 98354 206/631-4855
- 4. Engineering Logic 1252 13th Ave. Sacramento, CA 95822
- 5. Eco Technologies 1100 Larkspur Landing Circle #275 Larkspur, CA 94939 415/461-6121

F

1. Fantasia Systems, Inc. 1059 The Alameda Belmont, CA 94002 415/593-5700

- FORTH, Inc. 2309 Pacific Coast Highway Hermosa Beach, CA 90254 213/372-8493
- 4. FORTHWare 639 Crossridge Terrace Orinda, CA 94563
- 5. Frank Hogg Laboratory 130 Midtown Plaza Syracuse, NY 13210 315/474-7856
- 6. FSS P.O. Box 8403 Austin, TX 78712 512/477-2207

#### Н

1. HAWG WILD Software P.O. Box 7668 Little Rock, AR 72217

#### I

- 1. IDPC Company P.O. Box 11594 Philadelphia, PA 19116 215/676-3235
- 2. IUS (Cap'n Software) 281 Arlington Ave. Berkeley, CA 94704 415/525-9452
- 3. Inner Access 517K Marine View Belmont, CA 94002 415/591-8295
- 4. Innovatia Laboratories 5275 Crown St. West Linn, OR 97068
- 5. Insoft 10175 S.W. Barbur Blvd. Suite #202B Portland, OR 97219 503/244-4181
- Interactive Computer Systems, Inc.
   6403 Di Marco Rd. Tampa, FL 33614

J

1. JPS Microsystems, Inc. 361 Steelcase Rd., W. Markham, Ontario Canada L3R 3V8 416/475-2383

#### K

1. Kukulies, Christoph Ing. Buro Datentec Heinrichsallee 35 Aachen, 5100 West Germany

#### L

1. Laboratory Microsystems 4147 Beethoven St. Los Angeles, CA 90066 213/306-7412

- Laboratory Software Systems, Inc. 3634 Mandeville Canyon Los Angeles, CA 90049 213/472-6995
- Lynx
   3301 Ocean Park, #301
   Santa Monica, CA 90405
   213/450-2466
- Lyons, George 280 Henderson St. Jersey City, NJ 07302 201/451-2905

#### Μ

 M & B Design 820 Sweetbay Dr. Sunnyvale, CA 94086

- MicroMotion

   12077 Wilshire Blvd., #506
   Los Angeles, CA 90025
   213/821-4340
- Microsystems, Inc.
   2500 E. Foothill Blvd., #102 Pasadena, CA 91107
   213/577-1477
- 4. Micro Works, The P.O. Box 1110 Del Mar, CA 92014 714/942-2400
- 5. Miller Microcomputer 61 Lake Shore Rd. Natick, MA 01760 617/653-6136
- 6. Mountain View Press P.O. Box 4656 Mountain View, CA 94040 415/961-4103
- MCA
   8 Newfield Ln.
   Newtown, CT 06470
- 8. Metacrafts Ltd. Beech Trees, 144 Crewe Rd. Shavington, Crewe England CW1 5AJ

#### Ν

1. Nautilus Systems P.O. Box 1098 Santa Cruz, CA 95061 408/475-7461

#### 0

- 1. OSI Software & Hardware 3336 Avondale Court Windsor, Ontario Canada N9E 1X6 519/969-2500
- 2. Offete Enterprises 1306 S "B" St. San Mateo, CA 94402
- 3. On-Going Ideas RD #1, Box 810 Starksboro, VT 05487 802/453-4442

#### P

1. Perkel Software Systems 1636 N. Sherman Springfield, MO 65803

- 2. Pink Noise Studios P.O. Box 785 Crockett, CA 94525 415/787-1534
- Professional Mgmt. Services 724 Arastradero Rd., #109 Palo Alto, CA 94306 408/252-2218
- 4. Peopleware Systems Inc. 5190 West 76th St. Minneapolis, MN 55435 612/831-0827

#### Q

- 1. Quality Software 6660 Reseda Blvd., #105 Reseda, CA 91335
- 2. Quest Research, Inc. P.O. Box 2553 Huntsville, AL 35804 800/558-8088

#### R

2. Rockwell International Microelectronics Devices P.O. Box 3669 Anaheim, CA 92803 714/632-2862

#### S

- 1. Satellite Software Systems 288 West Center Orem, UT 84057 801/224-8554
- Saturn Software, Ltd. P.O. Box 397 New Westminister, BC Canada V3L 4Y7
- Shaw Labs, Ltd.
   P.O. Box 3471
   Hayward, CA 94540
   415/276-6050
- Sierra Computer Co.
   617 Mark NE Albuguergue, NM 87123
- 5. Sirius Systems 7528 Oak Ridge Highway Knoxville, TN 37921 615/693-6583
- Software Federation 44 University Drive Arlington Hts., IL 60004 312/259-1355
- 7. Software Works, The 1032 Elwell Ct., #210 Palo Alto, CA 94303 415/960-1800
- Spectrum Data Systems 5667 Phelps Luck Dr. Columbia, MD 21045 301/992-5635
- 9. Stearns, Hoyt Electronics 4131 E. Cannon Dr. Phoenix, AZ 85028 602/996-1717
- 10. Stynetic Systems, Inc. Flowerfield, Bldg. 1 St. James. NY 11780 516/862-7670

- 11. Supersoft Associates P.O. Box 1628 Champaign, IL 61820 217/359-2112
- 12. Sylmar Software P.O. Box 44037 Sylmar, CA 91342

#### т

- 1. Talbot Microsystems 1927 Curtis Ave. Redondo Beach, CA 90278 213/376-9941
- 2. Technical Products Co. P.O. Box 12983 Gainsville, FL 32604 904/372-8439
- Timin Engineering Co. C/o Martian Technologies 8348 Center Dr. Suite F La Mesa, CA 92041 619/464-2924
- 4. Transportable Software P.O. Box 1049 Hightstown, NJ 08520 609/448-4175

#### V

1. Valpar International 3801 E. 34th St. Tucson, AZ 85713 800/528-7070

#### W

- 1. Ward Systems Group 8013 Meadowview Dr. Frederick, MD 21701
- 2. Worldwide Software 2555 Buena Vista Ave. Berkeley, CA 94708 415/644-2850
- 3. Wycove Systems, Ltd. P.O. Box 499 Dartmouth, NS B2Y 3Y8 Canada 902/469-9897

#### Z

- Zimmer, Tom
   292 Falcato Dr.
   Milpitas, CA 95035
- Ziggurat Software
   P.O. Box 100
   N. Salem, NH 03073

#### **Boards & Machines Only** See System Vendor Chart for others

Controlex Corp. 16005 Sherman Way Van Nuys, CA 91406 213/780-8877

Datricon 7911 NE 33rd Dr., #200 Portland, OR 97211 503/284-8277

Golden River Corp. 7315 Reddfield Ct. Falls Church, CA 22043

#### FORTH INTEREST GROUP

#### MAIL ORDER

	LISA	FOREIC
Membership in FORTH Interest Group and Volume V of FORTH DIMENSIONS	\$15	\$27
Back Volumes of FORTH DIMENSIONS. Price per each.	\$15	\$18
] fig-FORTH Installation Manual, containing the language model of fig-FORTH, a complete glossary, memory map and installation instructions Assembly Language Source Listings of fig-FORTH for specific CPU's	\$15	\$18
Check appropriate box(es). Price per each.	\$15	\$18
1802         6502         6800         6809         VAX         z80           8080         8086/8088         9900         APPLE II         ECLIPSE           PACE         NOVA         PDP-11         68000         ALPHA MICRO		
"Starting FORTH, by Brodie. BEST book on FORTH. (Paperback)	\$18	\$22
" "Starting FORTH" by Brodie. (Hard Cover) " PROCEEDINGS: FORML (FORTH Modification Conference) [] 1980, \$25USA/\$35Foreign [] 1981, Two Vol., \$40USA/\$55Foreign [] 1982, \$25USA/\$35Foreign ROCHESTER FORTH Conference	\$23	\$28
1981, \$25USA/\$35Foreign 1982, \$25USA/\$35Foreign		
1983, \$25USA/\$35Foreign Total	\$	
<ul> <li>/ STANDARD: FORTH-79, [ FORTH-83. \$155A/\$1070121gh 10121</li> <li> Kitt Peak Primer, by Stevens. An in-depth self-study book.</li> <li> MAGAZINES ABOUT FORTH: BYTE Reprints 8/80-4/81</li> <li> Dr Dobb's Jrnl, 9/81, ( 9/82, 9/83)</li> <li> Ponlar Computing. 9/83 \$3.500SA/\$5Foreign EACH. Total</li> </ul>	\$ <u>25</u>	\$35
☐ FIG T-shirts: / Small / Medium / Large X-Large 7 Poster, BYTE Cover 8/80, 16"x22"	\$10 \$3	\$12 \$ 5
FORTH Programmer's Reference Card. If ordered separately, send a stamped, self addressed envelope.	Fı	cee
TOTAL	\$	
NAMEMS/APT		
DRGANIZATION PHONE (	)	
ADDRESS		
CITYSTATECO	UNTRY	
/ISA#MASTERCARD#		
MERICAN EXPRESS#Card Expiration Date (Minimum of \$15.00 on Charge Cards)		
Make check or money order in US Funds on US Bank, payable to: FIG. All postage. No purchase orders without check. California residents add s	prices i ales tax	include . 10/83

### FORTH INTEREST GROUP

P.O. Box 1105 San Carlos, CA 94070

> ROBERT JULE 2300 ST. FRANCIS DR PALO ALTO, CA 94303

BULK RATE U.S. POSTAGE PAID Permit No. 261 Mt. View, CA