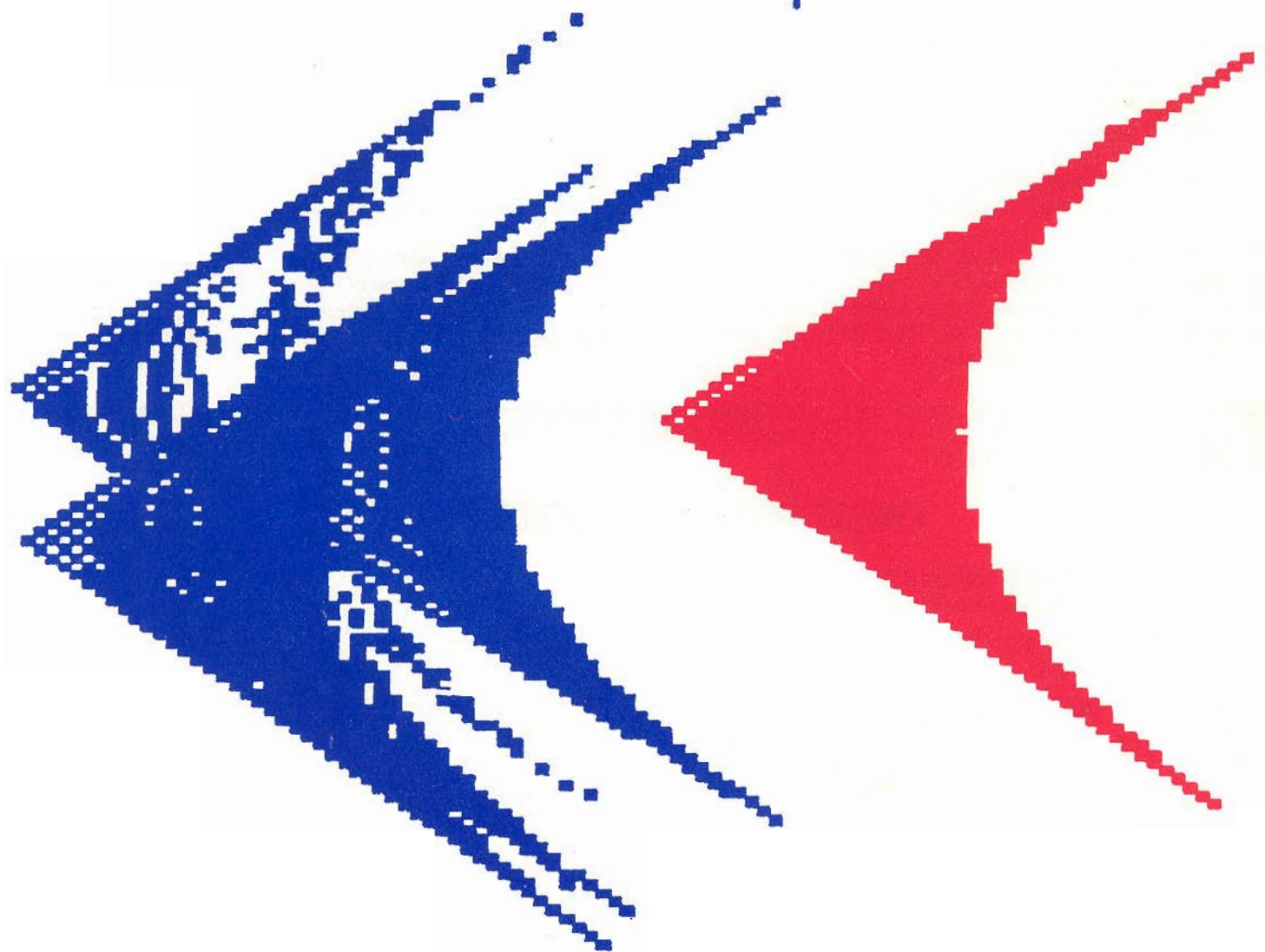


Color Graphics

for

INTECOLOR 3651 and
COMPUCOLOR II computers



by David B. Suits

COLOR GRAPHICS for Intecolor 3651 and Compucolor II Computers

by David B. Suits

COLOR GRAPHICS for Intecolor 3651 and Compucolor II Computers is intended for those who want to learn the ins and outs of color graphics for the Intecolor 3651 or Compucolor II computer. An introductory knowledge of BASIC programming is assumed.

David Suits has unravelled the mysteries of graphics on the Intecolor 3651 and Compucolor II Computers better, perhaps, than anyone else. Suits guides the reader through the graphics capabilities of these computers with cleverness, style, and touches of genius. His appreciation of these computers infects the reader—and a wonderful infection it is! His writing is clear, understandable, witty, and engaging. The teaching ability of a good teacher shines through.

The depth and subtleties of the graphics capabilities shared with the reader must be experienced to be appreciated. There is so much more to be had if one knows how! The topics covered include:

Cursor control	Cursor & plot coordinates	CRT mode
Hiding the cursor	Removing plotted lines	CRT plotting
Page/scroll/vertical modes	Cursive writing	Special function keys
CCI codes	Bar graphs	Saving & duplicating displays
Color keys	Moving bar graphs	Editing displays
Double blinking	Vector plotting	Four-directional scrolling
Blind cursor addressing	Incremental plotting	Large Chess graphics
Double height blind cursor	Plotting character strings	Real-time lunar lander
Special characters	"Plot-English"	Radar simulation
Larger characters	Screen refresh memory	Animated displays
Character & point plotting	Peeking & Poking screen RAM	Fast dice

COLOR GRAPHICS contains over eighty tutorial programs and supplements the above topics with nine appendices and an extensive index. It takes up where **BASIC Training for Compucolor Computers** leaves off.

David B. Suits received his Ph.D. in Philosophy in 1977 from the University of Waterloo, Waterloo, Ontario, Canada. He presently teaches Logic and Philosophy at Rochester Institute of Technology, Rochester, New York. He has published articles in both philosophy and computer journals and is the author of a number of Intelligent Systems Corporation's programs, including BOUNCE, MAZEMASTER, LINKO and others. His special interests include artificial intelligence, science fiction, games, and music.

COLOR GRAPHICS for Intecolor 3651 and Compucolor II Computers. D.B. Suits. Joseph J. Charles Publishing Company, 130 Sherwood Dr., P.O. Box 750, Hilton, NY 14468. 8.5" x 11" (21.6 cm x 28 cm), ix + 152 pp. (1981). (Shipping wt. 1 lb. 10 oz. (740 g).).

ORDERING INFORMATION

_____	ea.	COLOR GRAPHICS for Intecolor 3651 and Compucolor II Computers*	@ \$15.00	✓ _____
_____	ea.	Page Mode/Plot Mode Chart, No. 101, pads (50-sheets/pad)** This is a to-scale layout form for graphics and output.	@ \$ 4.00	x _____
_____	ea.	BASIC Training for Compucolor Computers , J. J. Charles* Also applies to Intecolor 3600 series computers.	@ \$14.95	x _____
			Subtotal:	_____
			New York State residents add sales tax:	_____
			Total:	_____

Please remit payment with order. Satisfaction guaranteed or money refunded if returned within 14 days.

ORDER FROM: Joseph J. Charles Publishing Co., 130 Sherwood Dr., Dept. F, P.O. Box 750, Hilton, NY 14468.

Your Name _____

Your Address _____ City _____ State _____ Zip _____

Note: *Surface postage included. Extra charge for Air Mail if desired: Mexico, Central America-\$3.50; S. America, Western Europe, N. Africa-\$6.50; Eastern Europe, Africa, Middle East, Pacific, Australia, New Zealand-\$9.00.

**Pads prohibitively expensive to air mail to most countries, also frequently subject to duty.

COLOR GRAPHICS

FOR

INTECOLOR 3651 AND

COMPUCOLOR II COMPUTERS

BY **DAVID B. SUITS**

49 KARENLEE DRIVE

HENRIETTA

JOSEPH J. CHARLES PUBLISHING

130 SHERWOOD DR. P.O. BOX 750

HILTON, NY 14468

UNIVERSITY OF MICHIGAN

LIBRARY

UNIVERSITY OF MICHIGAN
LIBRARY

Copyright © 1981 by David B. Suits

Library of Congress Catalog Card Number: 81-69547

ISBN 0-9607080-1-4

Printed in the United States of America

JOSEPH L. CHARLES

LIBRARY

NAME: Dave Suits

ADDRESS: Rochester Institute of Technology
General Studies
Rochester, New York 14623

or 95 Castle Road,
Rochester, N.Y. 14623

PHONE: (Day) (716) 475-6665, (Night) (716) 359-2179

SUBJECTS TAUGHT: Philosophy, especially symbolic logic at the university level. Interested in, but do not teach, economics.

PROGRAMMING LANGUAGES: BASIC, a little FORTRAN, PASCAL, and Assembly. Personally owns a 32K Compucolor (purchased in October 1978), uses it for games, word processing, and hobby applications.

SCHOOL INFORMATION: Rochester Institute of Technology. Formal courses in most computer languages are taught. School owns 1 (maybe 2) 16K Compucolors (purchased in Fall 1979). Primary application is instruction in the microcomputing laboratory. Other instructors use the Compucolor in the microcomputing laboratory. Student population is 10,000.

PROJECT INTEREST: Interested in participating in software exchange; small group software development; development and writing training materials, AV aids, workbooks; development of a formal newsletter or other communication link; and research into software availability.

SOFTWARE AVAILABILITY FOR EXCHANGE OR SALE: Not available yet; working on some programs in logic and economics.

PREFACE

This book was written for two kinds of people:

- (1) Those who have not had much programming experience and who wish to learn more about color graphics on the Intecolor 3651 or Compucolor II computer.
- (2) Those who are more experienced programmers and who wish to learn more about color graphics on these computers.

The book has two main functions. First, and foremost, it is a tutorial to help you become acquainted with what your computer can do and what you can do with it. Second, the book is meant to be a reference to return to again and again in order to refresh your recollection of all the nitty-gritty of the material presented here. It is just not possible, in one reading, to become familiar with all the tools you might make use of in your own work. If you come back to the book now and then, you will no doubt rediscover interesting tidbits which you had forgotten and which could now make your programming tasks easier.

This is **not** a beginning manual in the BASIC programming language. You are assumed to know at least the rudiments of BASIC programming. Nor is this a book on the mathematics of computer graphics. On occasion there is a little math or trigonometry, of course. But there are no discussions of hidden line routines, co-ordinate translations, and so on. Nor is it the intention of the book to teach programming style. (Most versions of BASIC are not very well suited to the kind of "structured programming" which has become the darling of writers within the last decade.) Indeed, you will find several styles of putting together BASIC programs in the book. What you find to be an easily understood computer program will depend in large part on how well-versed you are in the particular language used. BASIC on Intecolor/Compucolor computers is often rather cryptic, allowing for the concatenation of mysterious numbers such as these:

FROM CRT MONITOR
WITH SEWAGE KEYBOARD

```

10 PLOT 3,64,0,2,0,0,242,127,0,127,127,0,127,0,0,255

```

BLINK: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000

Still, anyone familiar with this dialect of BASIC will understand that statement's function almost immediately. Nevertheless, at the expense of speed and memory, I have usually chosen to break up such statements into smaller logical units and to add numerous REMarks:

```

9 REM  DRAW A BORDER AROUND THE SCREEN
10 PLOT 3,64,0:REM  HIDE THE CURSOR
20 PLOT 2:REM  GENERAL PLOT MODE
30 PLOT 0,0:REM  FIRST POINT AT X=0, Y=0
40 PLOT 242:REM  VECTOR PLOT SUBMODE
48
49 REM  DRAW EACH OF THE FOUR LINES OF THE BORDER
50 PLOT 127,0
60 PLOT 127,127
70 PLOT 0,127

```



```
80 PLOT 0,0
89
90 PLOT 255:REM EXIT PLOT MODE
```

You are advised to follow suit when entering the book's programs on your own computer so that subsequent changes are easily made.

Most of the BASIC programs in the book are program "fragments"--short sets of instructions which demonstrate some aspect of the computer's abilities. They are to be entered, run and **studied** so that why they do what they do will be clearly understood. Try variations. Fiddle around. Explore. There is a skill to be learned--a skill which develops with both experience and reflection. You ought to learn to appreciate not only the fundamental rules for using your machine, but also--and very importantly---the **aesthetics** of the output.

Some of the longer programs in the book build upon other programs. You might judiciously save them on disk in case additions to them are offered later in the book, and in case, after having learned new techniques, you wish to go back and try an earlier program in a slightly different way. Disk space is relatively cheap. Besides, what you have once saved, you can later delete.

All the programs were written in Disk BASIC 8001 V6.78, often with the aid of Intelligent Systems Corporation's BASIC Text Editor. (By the way, that editor allows for including blank lines, indentations in FOR-NEXT loops, and other pretty-printing devices used in the listings in this book.) All programs are compatible with V8.79 and V9.80.

Machine language opens up a host of new possibilities for graphics, especially for real time applications. However, except for a few helpful machine language patches (in Programs 9.2, 10.1, 10.2, 12.8, 12.9 and 12.12), 8080 machine language is neither discussed nor used.

I owe special thanks to Christine Bryant, to Fred and Nancy Calev of Compuworld in Rochester, NY, and to Bruce Williams of ISC for their encouragement and help.

PREFACE

This book was written for two kinds of people:

- (1) Those who have not had much programming experience and who wish to learn about graphics on the Compucolor II.
- (2) Those who are more experienced programmers and who wish to learn about graphics on the Compucolor II.

This is not a beginning manual in the BASIC programming language. The reader is assumed to know at least the rudiments of BASIC programming on the Compucolor II--the assignment of variables, loops, conditional branches, and so on. Nor is this a book on the mathematics of computer graphics. On occasion there is a little math or trigonometry, of course. But there are no discussions of hidden line routines, co-ordinate translations, etc.

Rather, this is a book of instructions and ideas. Although much of the information is available in the Compucolor II Programming and Reference Manual (copyright 1978 by Compucolor Corporation), it is presented here with a good deal of amplification, explanation and examples. In addition, there are tips, ideas and techniques in this book which are not available in that Manual.

I assume you have a Compucolor II on your desk fired up and ready to go. The short programs throughout the book are there not only to illustrate the points being made, but also to encourage you to sit down for constant hands-on experience. Try out some of the ideas presented here in your own programs. Fiddle around. Explore. Dazzle yourself with the Compucolor II's power and versatility.

The book has two main functions. First, and foremost, it is a tutorial. The BASIC programs herein are, for the most part, ~~program fragments~~ to demonstrate the use of the many tools at your disposal. There is little attempt to develop full-fledged applications: that is a matter in which your own needs and imagination are far better guides than I.

Second, the book is meant to be a reference to return to again and again in order to refresh your recollection of all the nitty-gritty of the material presented here. It is simply not possible, in one reading, to assimilate all the data you might make use of in your own work. If you come back to the book on occasion, you will no doubt re-discover interesting tidbits which you had forgotten and which could make your programming tasks easier.

The BASIC programs in this book were not written for efficiency in either speed or use of memory. Instead, they were written for instruction and clarification. And except for the quad-directional scrolling patch in Section 10 and Ben Barlow's no-echo patch in Programs 9.2, 12.8 and 12.9, there is no discussion or use of 8080 machine language.

All programs were written in Compucolor Disk BASIC 8001 V6.78, often with the aid of Compucolor Corporation's BASIC Text Editor.

TABLE OF CONTENTS

PREFACE.	(iii)
LIST OF PROGRAMS	(vii)
1. THE CURSOR AND WHERE IT GOES.	1
1.1 Cursor Control	1
1.2 Hiding the Cursor.	2
1.3 Erase Page	2
1.4 The PRINT Statement.	2
1.5 Character Heights.	3
1.6 Scroll Mode.	4
1.7 Page Mode.	4
1.8 Vertical Mode.	5
2. COLORS.	8
2.1 The FLAG Bit	8
2.2 The CCI Code	9
2.3 Color Keys	10
2.4 The Hatch Character.	12
2.5 Quiz	13
2.6 Mystery Program.	15
2.7 Blink.	15
2.8 PLOT 31.	16
2.9 Blink and the CCI Code	17
3. THE BLIND CURSOR.	18
3.1 Blind Cursor Addressing.	18
3.2 Reentry to Blind Cursor Mode	19
3.3 Double Height Characters in Blind Cursor Mode.	20
4. CHARACTERS AND THE PLOT STATEMENT	21
4.1 Exercise	21
4.2 Special Characters	21
4.3 Larger Characters.	24
5. TEST MODE	27
6. PLOT MODES.	30
6.1 PLOT 254 -- Character Plot	31
6.2 PLOT 253 -- Point Plot	31
6.3 An Artillery Game.	36
6.4 PLOT 251 -- Incremental Point Plot	41
6.5 Script	44
6.6 PLOT 250 -- X Bar Graph.	45
6.7 PLOT 247 -- Incremental X Bar Graph.	46
6.8 PLOT 246 -- Y Bar Graph and PLOT 243 -- Incremental Y Bar Graph.	51
6.9 PLOT 242 -- Vector Plot.	52
6.10 PLOT 240 -- Incremental Vector Plot	56
6.11 Flying Wedges	57

6.12 Reentering Plot Submodes.	58
6.13 PLOTting Character Strings.	64
7. PLOT 6,128 AND ABOVE.	65
8. SCREEN REFRESH.	69
9. CRT MODE.	75
9.1 CRT Plotting	75
9.2 Special Function Keys.	75
9.3 Saving Displays.	77
9.4 Duplicating Displays	78
9.5 Editing Displays	78
10. QUAD-DIRECTIONAL SCROLLING PATCH	84
10.1 What the Patch Does	84
10.2 The Patch	85
10.3 Scrolling Up.	87
10.4 Scrolling Down.	88
10.5 Scrolling Right and Left.	88
11. MISCELLANEOUS NOTES.	93
11.1 A Note on Menu Programs	93
11.2 A Note on Displaying Text	94
12. MISCELLANEOUS PROGRAMS	96
12.1 Variations on a Theme	96
12.2 Simulating a Radar Scope.	97
12.3 Radar Scope Using Incremental Vector Plot	99
12.4 Some Circles.	102
12.5 An Animated Joke.	103
12.6 Chess Pieces Using Character Plot Submode	106
12.7 A Real Time Lunar Lander.	111
12.8 Extra Large Characters.	121
12.9 Dice.	127
12.10 Dice -- The Fast Way	129
12.11 Quick Change Artistry.	131

APPENDICES

A. ASCII CHART.	135
B. CHARACTER SET.	136
C. EXTRA LARGE CHARACTERS	138
D. PLOT MODES	139
E. INCREMENTAL VALUES FOR INCREMENTAL POINT PLOT AND INCREMENTAL BAR GRAPH SUBMODES	140
F. INCREMENTAL VALUES FOR INCREMENTAL VECTOR PLOT SUBMODE	141
G. KEYBOARD ENCODING.	142
H. SOME HELPFUL EQUATIONS	146
I. SCREEN MEMORY LOCATIONS.	147
J. COLORS	148
INDEX.	149

LIST OF PROGRAMS

1.1 The PRINT Statement.	2
1.2 PRINTing Past the Right Side of the Screen	3
1.3 PRINTing More Than 64 Characters Without a CR/LF	3
1.4 Double Height Characters	4
1.5 The Vertical Mode.	5
1.6 CR/LF in Vertical Mode	5
1.7 Drawing a Border	6
2.1 Setting Foreground and Background Colors	8
2.2 The CCI Code	10
2.3 Explosion Effect Using Erase Page.	10
2.4 Color Changes From the Keyboard.	11
2.5 Color Changes in REMark Statements	12
2.6 The Hatch Character.	12
2.7 More of the Hatch Character.	14
2.8 Mystery Program.	15
2.9 Blinking Characters.	16
2.10 Out of Phase Blinking	16
3.1 Blind Cursor Addressing.	18
3.2 Interaction of Visible and Blind Cursors	19
3.3 Reentry to Blind Cursor Mode	19
3.4 Broken Characters.	20
4.1 Special Characters	21
4.2 Circle and Square.	22
4.3 Edges of Characters.	23
4.4 Borders Using Special Characters	24
4.5 Underlining.	24
4.6 Characters in 4 Sizes.	25
5.1 Demo Introduction Using the Test Mode.	27
5.2 Demo Introduction #2	27
5.3 Explosion Effect Using Test Mode	28
6.1 Character Plot	32
6.2 Demonstrating the Character Plot	32
6.3 Point Plot Submode	33
6.4 Resolution of Colors	35
6.5 Testing the Pseudo-random Number Generator	35
6.6 An Artillery Game.	38
6.7 Mystery Program Using the Incremental Point Plot	43
6.8 Script Using Incremental Point Plot.	44
6.9 A Triangle Using X Bar Graph	46
6.10 Triangle Using Incremental X Bar Graph.	46
6.11 Mystery Program Using Incremental X Bar Graph	46
6.12 X Bar Graph with FLAG On.	47
6.13 Erase Screen Using X Bar Graph.	48
6.14 Erase Screen With X Bar Graph and FLAG on	49
6.15 Sine Function in Y Bar Graph Mode	51

6.16 Incremental Y Bar Graph Demo.	51
6.17 Explosion Using Vector Plot Submode	52
6.18 Erase Screen Using Vector Plot -- I	54
6.19 Erase Screen Using Vector Plot -- II.	55
6.20 A Flying Wedge.	57
6.21 Random Flying Wedges.	58
6.22 Reentry to Plot Submodes.	58
6.23 Plot Mode Default Co-ordinates.	59
6.24 Graphing Widgeco Production	60
6.25 PLOTting Character Strings.	64
7.1 Verifying Plot Mode Output	65
7.2 Plot Mode.	66
7.3 Blind Cursor in Plot Mode.	66
7.4 Plot a Border Around the Screen.	67
7.5 Erase Page in Plot Mode.	68
8.1 PEEKing Screen Refresh Memory.	69
8.2 POKEing Into Screen RAM.	70
8.3 POKE Any Character Into Screen Memory.	71
8.4 Half Character Color Changes	72
8.5 POKEing CCI Codes Greater Than 127	73
8.6 Move a * Across the Screen	73
9.1 Dup a Screen Display	78
9.2 Display Create/Edit/Dup.	79
10.1 Quad-Directional Scrolling Patch.	85
10.2 Backwards Text Guessing Game.	92
11.1 Blank Menu.	93
11.2 Displaying Text	94
12.1 Converging Borders.	96
12.2 PLOTting a Circle	97
12.3 Simulating a Radar Scope.	99
12.4 Radar Scope Simulation Using Incremental Vector Plot.	101
12.5 Circles About the Corners	102
12.6 An Animated Joke.	103
12.7 Chess Pieces Using Character Plot	107
12.8 Real Time Lunar Lander.	112
12.9 Extra Large Characters.	121
12.10 Dice	127
12.11 Dice -- The Fast Way	129
12.12 Quick Change Artistry.	132

1. THE CURSOR AND WHERE IT GOES

1.1 CURSOR CONTROL

Characters are printed wherever the cursor is presently located. But your Intecolor® or Compucolor® is a cursor addressable machine, which means that you may put the cursor anywhere on the screen before printing a character. This is accomplished by a statement of the form `PLOT 3,X,Y`, where X is a number (or variable, or expression) which determines the horizontal location of the cursor, and Y is a number (or variable, or expression) which determines the vertical location of the cursor. For example,

`PLOT 3,5,10:PRINT "TEST"`

would position the cursor at the sixth column (counting 0 as the first) and the eleventh line down (counting 0 as the first) and print the word "TEST" beginning at that spot.

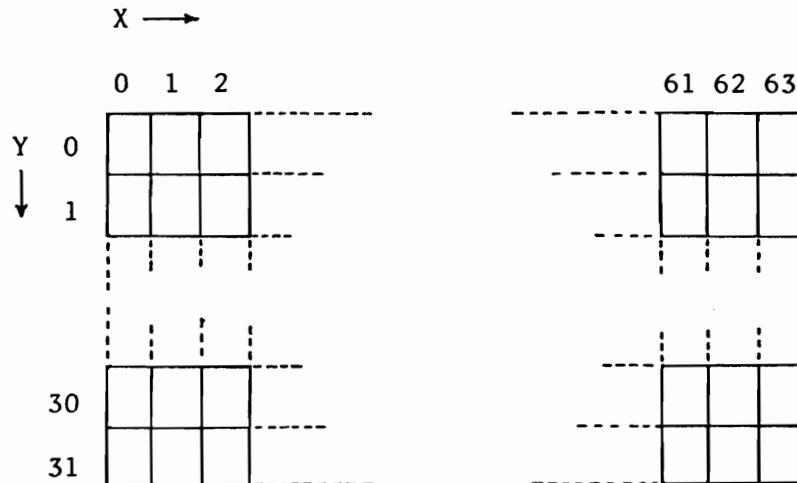


FIG. 1.1

It is not always necessary to use this cursor control in order to print a character at a desired spot on the screen. We will see two other methods later in Section 3 (Blind Cursor) and Section 8 (Screen Refresh Memory).

The cursor may be moved up, down, right or left by `PLOT 28`, `PLOT 10`, `PLOT 25`, or `PLOT 26`, respectively. `PLOT 13` returns the cursor to the beginning of the line it is presently on. `PLOT 8` places the cursor in the top, left hand corner of the screen. It is therefore equivalent to `PLOT 3,0,0`. (See Appendix A for a list of the `PLOT` commands.)

®Intecolor and Compucolor are trademarks of Intelligent Systems Corp.

1.2 HIDING THE CURSOR

The cursor may be taken off the screen, where it will remain until (1) explicitly moved elsewhere, (2) the end of the program is reached, or (3) a PRINT or INPUT statement is issued (except for an INPUT statement which prints nothing; that is, INPUT "X";A\$ will print "X" and, hence, put the cursor back on the screen; but INPUT "";A\$ will output nothing to the screen, and so the cursor will not be moved from wherever it is until a key is pressed).

To place the cursor off the screen, specify 64 for the X value. Thus, PLOT 3,64,0 will place the cursor just off the screen on line 0 (the top line), and PLOT 3,64,15 will place it off the screen on line 15, and so on.

1.3 ERASE PAGE

The entire screen is erased by the PLOT 12 instruction (or by pressing the ERASE PAGE key). The screen will be erased in the background color last used by filling all the character positions with spaces. In addition, the cursor will be homed--placed at co-ordinates X=0, Y=0. For additional ways of erasing the display, see Sections 5, 6.13, 6.14, 6.18 and 6.19.

1.4 THE PRINT STATEMENT

The PRINT statement in BASIC will print whatever follows, until the end of the program line (or a colon) is reached. Then a carriage return and line feed will automatically be performed. (A carriage return [CR] simply moves the cursor back to the left side of the screen, just like PLOT 13, and a line feed [LF] moves the cursor down to the next line, just like PLOT 10.) However, the CR/LF may be suppressed with either a semicolon or a comma following whatever is to be PRINTed. A semicolon will hold the cursor at its present location. A comma will move the cursor over to the next tab column. (The tab columns are columns 8, 16, 24, etc.) Note also that a semicolon or comma may be inserted before whatever is to be PRINTed.

```
5 REM PROGRAM 1.1
6 REM THE PRINT STATEMENT
10 PRINT "TEST #1":REM THIS DOES AN AUTOMATIC CR/LF
20 PRINT "TEST #2",:REM TAB OVER AFTER PRINTING
30 PRINT "TEST #3":REM CR/LF
40 PRINT,,"TEST #4":REM TAB TO COLUMN 16 BEFORE PRINTING
50 PRINT "TEST #5";:REM SUPPRESS CR/LF
60 PRINT "TEST #6"
70 FOR J=7 TO 10
80 PRINT "TEST #"J;
90 NEXT
100 PRINT
110 PRINT;"TEST #11"
```

What does line 100 do? If a PRINT statement has nothing to print, it

will just do a CR/LF. Notice that after the FOR-NEXT loop (lines 70-90) has executed, the cursor will be sitting at the end of the last thing printed, because it will have encountered a semicolon. If line 100 is omitted, the "Test #11" in line 110 will be printed on the same line as the four previous "TEST"s. (Try it and see.) Also notice that the semicolon in line 110 is superfluous. I don't even know why I put it in....

When the cursor has been moved past the right-most column, it wants to "wrap around" back to the left side of the screen and one line down. But in addition the computer does a CR/LF. Now, since the "wrap around" is, in effect, also a CR/LF, pushing the cursor past the right-most column is like suddenly issuing two carriage return/line feeds, and the result will be that a line will be skipped before the printing is continued.

```
5 REM PROGRAM 1.2
6 REM PRINTING PAST THE RIGHT SIDE OF THE SCREEN
10 FOR J=1 TO 32
20 PRINT "TEST";
30 NEXT
```

Notice that although "TEST" will be printed 32 times, the first 16 will fill up the first line, but the second 16 will be printed only after skipping a line.

If a series of PRINT statements outputs 64 or more characters, and if each of those PRINT statements is followed by a semicolon, then BASIC will insist on doing a CR/LF, even if the cursor has not reached the right side of the screen. (The PLOT 15 in the following program selects the small characters, and will be discussed in the next section.)

```
5 REM PROGRAM 1.3
6 REM PRINTING MORE THAN 64 CHARACTERS WITHOUT A CR/LF
10 PLOT 15,12:REM SMALL CHARACTERS; CLEAR SCREEN
20 FOR J=1 TO 15
30 PLOT 3,J,J:REM POSITION THE CURSOR
40 PRINT "LINE"J;:REM SUPPRESS THE CR/LF
50 NEXT
```

By the time the program is part way through PRINTing "LINE 11", 64 characters will have been PRINTed without a CR/LF, and so the CR/LF is automatically issued, resulting in the disruption of part of the text. (Notice that the value of J is printed, as all positive numbers are, with a leading space--negative numbers, of course, have a negative sign--and so that must be taken into account when counting out 64 characters.)

1.5 CHARACTER HEIGHTS

The computer can print 32 lines of 64 (regular height) characters per line, or it can print 64 double height characters per line, with only 16 lines. (When the power is first turned on, the double height characters are used.)

The double height characters may be selected in BASIC by PLOT 14. Any subsequent character (letter, number, special character, and even a space) will be double height. A PLOT 15 statement will set the computer to output

regular height characters.

The character height may also be set from the keyboard: the A7 ON key acts just like PLOT 14, and the BL/A7 OFF key is equivalent to PLOT 15. Incidentally, this key--or a PLOT 15--also turns off the BLINK. (See Section 2.7.) Be aware, though, that the BASIC interpreter will not understand these keys and will tell you so with "SN ERROR" (syntax error), unless the keystroke is in a REMark statement or enclosed in quotes. (See Section 2.3.)

Appendix B displays all the computer's characters in both regular and double height.

When the A7 is on (double character height), characters will be printed on 2 of the 32 regular height lines on the screen. Moreover, the bottom of a double height character will always appear on an odd-numbered line. Although you may direct the cursor to an even-numbered line, the character will nevertheless be printed on the odd-numbered line.

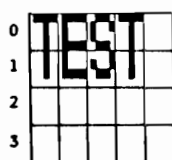


FIG. 1.2

```
5 REM PROGRAM 1.4
6 REM DOUBLE HEIGHT CHARACTERS
10 PLOT 14:REM DOUBLE HEIGHT
20 PLOT 3,0,3:PRINT "LINE 3"
30 PLOT 3,10,4:PRINT "LINE 4"
```

Giant sized letters can be fashioned by certain combinations of the computer's special characters. (See Section 4.3.)

1.6 SCROLL MODE

The monitor screen displays 64 characters per line (numbered 0-63) and 32 lines per "page" (numbered 0-31). When the cursor reaches the bottom of the screen and attempts to go down another line, it just stays at the bottom and the rest of the screen "scrolls" up. The top line consequently disappears. This is called the SCROLL (or ROLL UP) MODE and is the mode the computer is automatically in when power is first turned on. If the screen is subsequently taken out of the scroll mode, it may be brought back in BASIC by PLOT 27,11, or from the keyboard by the sequence ESC K.

1.7 PAGE MODE

Sometimes it is desirable not to have the screen scroll up, and in such cases the PAGE MODE is used. This is accomplished in BASIC by PLOT 27,24 or from the keyboard by the sequence ESC X. Now when the cursor passes below the bottom of the screen, it will reappear at the top, and the screen will

not scroll.

Programs which use the PAGE mode might conveniently end with PLOT 27,11 (return to scroll mode), for two reasons: (1) In debugging a program, you will want to list it, and if the listing is more than 32 lines, the page mode will cause the listing to overlay itself from above, which makes for difficult reading. (2) Also, if the computer is left in page mode, a subsequent program which requires the scroll mode will be foiled, unless it includes PLOT 27,11 near its beginning. (Not a bad idea--just in case. A program which takes too much for granted is a program likely to go awry.)

1.8 VERTICAL MODE

You can also set the display to print a string of characters in a vertical row. That is, anything to be printed will be displayed not horizontally, as usual, but vertically (top to bottom). In addition, the screen will not scroll.

The vertical mode is selected in BASIC by PLOT 27,10, or from the keyboard by ESC J. You may specify the cursor X,Y position where you wish to begin printing in the vertical mode, and this specification may take place whether you are already in the vertical mode or not.

```
5 REM PROGRAM 1.5
6 REM THE VERTICAL MODE
10 PLOT 12:REM CLEAR SCREEN
20 PLOT 27,10:REM VERTICAL MODE
30 PLOT 3,10,5:REM POSITION THE CURSOR
40 PRINT "TEST #1"
50 PLOT 3,20,5
60 PRINT "TEST #2"
68 REM NOW FOR PROOF THAT THE SCREEN DOES NOT
69 REM SCROLL WHEN IN VERTICAL MODE
70 PLOT 3,32,28:PRINT "TEST #3"
80 PLOT 27,11:REM BACK TO SCROLL MODE
```

Remember that if a PRINT statement is followed by a semicolon, the cursor is held at the end of whatever was printed, whereas if nothing occurs at the end of the PRINT statement, the cursor will return to the left side of the screen, one line down. This occurs in scroll, page and vertical modes.

```
5 REM PROGRAM 1.6
6 REM CR/LF IN VERTICAL MODE
10 PLOT 12,27,10:REM CLEAR SCREEN & SET TO VERTICAL MODE
20 PLOT 3,32,5:REM POSITION THE CURSOR
30 PRINT "TEST #1";:REM NOTE THE SEMICOLON
40 PRINT "TEST #2":REM NO SEMICOLON
50 PRINT "TEST #3"
60 PLOT 27,11:REM SCROLL MODE
```

In BASIC, an entire line of the display may be erased by PLOT 11. Or if you wish to spend more time doing it, PRINT SPC(64) will work. A still slower method is to generate a loop in which each character position is successively replaced by a space:


```
FOR J=0 TO 63:PRINT " ";:NEXT
```

But the PLOT 11 (or the ERASE LINE key) will erase a horizontal line, even if you are in the vertical mode. The PRINT SPC(32) method and the loop method will, however, erase a vertical line if you are in the vertical mode. (You need only 32 spaces in vertical mode, because a vertical column is 32 lines high.)

Suppose, now, we wish to draw a line around the entire screen. We could, of course, use some of the special characters for drawing the line, or we could use the PLOT mode. But the discussions of the special characters and of the PLOT mode will be reserved for later. Right now let us draw the line by drawing spaces in a different background color. Since the discussion of the various color commands will come later, it is enough at this point to attend to how the line is drawn, nevermind its color.

```
5 REM PROGRAM 1.7
6 REM DRAWING A BORDER
10 PLOT 15,6,6:REM SELECT REGULAR CHARACTER HEIGHT; SET COLOR
20 PLOT 12:REM CLEAR SCREEN
30 PLOT 27,24:REM PAGE MODE
40 PLOT 6,8:REM SET BACKGROUND COLOR TO RED
50 PLOT 11:REM ERASE THE TOP LINE (I.E., MAKE IT RED)
60 PLOT 3,0,31:REM MOVE THE CURSOR TO BOTTOM OF SCREEN
70 PLOT 11:REM ERASE BOTTOM LINE IN RED
80 PLOT 8:REM HOME CURSOR
90 PLOT 27,10:REM VERTICAL MODE
100 PRINT SPC(32)"":REM DRAW A LINE IN RED
110 PLOT 3,63,0:REM POSITION CURSOR AT TOP OF RIGHT HAND CORNER
120 PRINT SPC(32)"":REM DRAW A LINE
130 PLOT 8:REM HOME CURSOR AGAIN
140 PLOT 27,11:REM SCROLL MODE
150 PLOT 6,3:REM BACK TO A MANAGEABLE COLOR
```

Notice that when the PLOT 11 occurs in lines 50 and 70, the cursor remains at the beginning of the line it is presently on. Since, in line 50, that is the top line of the screen, and since the computer is set to page mode, we could move the cursor to the bottom line by moving it up one line: it will wrap around and appear at the bottom. (Actually, it will do this in vertical and scroll modes, too.) So line 60 could just as easily be:

```
60 PLOT 28:REM CURSOR UP
```

Line 80, which homes the cursor, i.e., places it at the top left of the screen, could also be accomplished by moving the cursor down one line from the bottom of the screen so that it wraps around to the top. (It won't do this in scroll mode!) On the other hand, why bother to put it at the top of the screen at all? Why can't we just print 32 spaces starting at the bottom? We can. In vertical mode, just as in page mode, the cursor wraps around from the bottom to the top. So let's delete line 80.

Line 110 positions the cursor in the upper right corner before drawing a vertical line. But line 100 has just drawn a vertical line on the left side of the screen. What happens if we move the cursor left from there? It will wrap around to the right side, which is just where we want it to be.

So we could change line 110 to:

```
110 PLOT 26:REM CURSOR LEFT
```


2. COLORS

The computer is equipped to display characters in any of eight colors against a choice of any of eight background colors. There are several ways of setting the colors in BASIC.

2.1 THE FLAG BIT

If a PLOT 29 is issued, then the FLAG bit is turned off, and subsequent color commands will determine the foreground color according to Table 2.1. The background color will not be changed. If a PLOT 30 is issued, then the FLAG bit is turned on, and subsequent color commands will determine the background color according to that same Table. The foreground color will not be changed.

PLOT	COLOR
====	=====
16	Black
17	Red
18	Green
19	Yellow
20	Blue
21	Magenta
22	Cyan
23	White

TABLE 2.1

```
5 REM PROGRAM 2.1
6 REM SETTING FOREGROUND AND BACKGROUND COLORS
10 PLOT 30:REM FLAG BIT ON
20 PLOT 17:REM SET BACKGROUND TO RED
30 PLOT 29:REM FLAG OFF
40 PLOT 20:REM SET FOREGROUND TO BLUE
50 PRINT "BLUE ON RED ";
60 PLOT 23:REM SET FOREGROUND TO WHITE
70 PRINT "WHITE ON RED ";
80 PLOT 30:REM FLAG ON
90 C1=20:C2=16
100 PLOT C1:REM SET BACKGROUND TO BLUE
110 PRINT "WHITE ON BLUE ";
120 PLOT C2:REM SET BACKGROUND TO BLACK
130 PRINT "WHITE ON BLACK"
```


2.2 THE CCI CODE

If a PLOT 6 is issued, then the next PLOT statement is taken as color information (commonly referred to as the CCI code) for both background and foreground colors according to Table 2.2. If the argument of the PLOT statement which follows the PLOT 6 is viewed as an eight bit binary number, then Table 2.3 indicates what color will be determined by which of the eight bits is on. (The functions of bits B6 and B7--blink and plot--will be discussed below, in Sections 2.9 and 7.)

Program 2.1 above could be rewritten as Program 2.2.

Bit:	B7	B6	B5	B4	B3	B2	B1	B0
	PLOT	BLINK	BACKGROUND COLOR			FOREGROUND COLOR		
			BLUE	GREEN	RED	BLUE	GREEN	RED
Decimal:	128	64	32	16	8	4	2	1

TABLE 2.2

128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255
 10 + 8 + 4 + 2 + 1 = 25
 ∴ PLOT = 6.01

PLOT 6,n		Color:	
Decimal	Binary	Background	Foreground
0	00000000	black	black
1	00000001	black	red
2	00000010	black	green
3	00000011	black	yellow
4	00000100	black	blue
5	00000101	black	magenta
6	00000110	black	cyan
7	00000111	black	white
8	00001000	red	black
9	00001001	red	red
⋮			
62	00111110	white	cyan
63	00111111	white	white

TABLE 2.3


```

5 REM PROGRAM 2.2
6 REM THE CCI CODE
10 PLOT 6,12:REM BACKGROUND RED, FOREGROUND BLUE
20 PRINT "BLUE ON RED ";
30 PLOT 6,15:REM BACKGROUND RED, FOREGROUND WHITE
40 PRINT "WHITE ON RED ";
50 PLOT 6,39:REM BACKGROUND BLUE, FOREGROUND WHITE
60 PRINT "WHITE ON BLUE ";
70 PLOT 6,7:REM BACKGROUND BLACK, FOREGROUND WHITE
80 PRINT "WHITE ON BLACK"

```

Note that the PLOT 6,n command will not affect the FLAG bit, so color commands like those in Program 2.2 will be unaffected. For example, add this to the above program:

```

90 PLOT 17
100 PRINT "RED ON BLACK"

```

Line 100 will indeed print red characters on a black background, unless the FLAG was on, in which case line 90 changed the background color to red, and line 100 consequently printed white characters on a red background. On the other hand, as Program 2.2 demonstrates, the condition of the FLAG bit will not affect the setting of foreground and background colors by means of the PLOT 6,n sequence.

If you erase the screen several times in succession using different background colors, you can create an explosion effect.

```

5 REM PROGRAM 2.3
6 REM EXPLOSION EFFECT USING ERASE PAGE
10 FOR J=1 TO 2
20 PLOT 6,56:REM BACKGROUND = WHITE
30 GOSUB 200:REM ERASE PAGE & PAUSE SLIGHTLY
40 PLOT 6,24:REM BACKGROUND = YELLOW
50 GOSUB 200
60 PLOT 6,0:REM BACKGROUND = BLACK
70 GOSUB 200
80 PLOT 6,24:REM BACKGROUND = YELLOW AGAIN
90 GOSUB 200
100 PLOT 6,0:REM BACKGROUND = BLACK AGAIN
110 GOSUB 200
120 PLOT 6,40:REM BACKGROUND = MAGENTA
130 GOSUB 200
140 NEXT
150 PLOT 6,2,12:REM BACK TO A MANAGEABLE COLOR
160 END
198
199 REM
200 PLOT 12:REM CLEAR SCREEN
210 FOR T=1 TO 25:NEXT:REM SLIGHT PAUSE
220 RETURN

```

Alternatively, if the FLAG bit is set, then the background colors may be selected by PLOT 16 for black, PLOT 17 for red, and so on, so that the above program could be re-written:


```

10 PLOT 30:REM FLAG ON
20 FOR J=1 TO 2
30 PLOT 23:REM BACKGROUND = WHITE
40 GOSUB 200
.
.
.
etc.

```

Since a previously run program may have affected the FLAG bit, and since, as well, the FLAG bit can be turned on or off from the keyboard, it is good programming practice to make sure your programs take nothing for granted: near the beginning of the program you could put in a statement which sets up everything the way you want it (especially colors, the FLAG bit, character heights, and page or scroll mode). If your program changes foreground colors frequently, you might put in a PLOT 29 early on, just to make sure the FLAG is off. Now you can change foreground colors the easy way with PLOT n, where n=17 for red, 18 for green, and so on. (If the FLAG is already off, a PLOT 29 statement will do no harm. Better safe than sorry, and all that.)

2.3 COLOR KEYS

You can specify colors from the keyboard, too, just by pressing the CONTROL key along with the appropriately colored key. If you have the color key pad at the left of the keyboard, you need only press the appropriate color key there. Once you have set the color in this manner, whatever you type in will be displayed in that color until the color is changed, either from the keyboard or by a program. The FLAG ON and FLAG OFF keys are there as well. If the FLAG is off, then any subsequent color key (or PLOT statement which affects the color--except for the PLOT 6,n statement) will select a foreground color. If you press the FLAG ON key, then any subsequent color key (or PLOT statement which affects the color) will change the background color.

All this is only to say that the computer does not care whether you set the FLAG or the color from the keyboard or from a BASIC statement. However, BASIC will not recognize such a change entered from the keyboard (and it will tell you so with "SN ERROR") unless the key stroke is in a REMark statement or enclosed in quotes. You may use quotes only in PRINT statements, INPUT statements, DATA statements, or string commands. In the following program, use the CONTROL-color key (or the color key in the color key pad) as specified in brackets: [].

```

5 REM PROGRAM 2.4
6 REM COLOR CHANGES FROM THE KEYBOARD
10 PLOT 29:REM TURN FLAG OFF, JUST IN CASE IT WAS ON
20 PRINT "[yel]YELLOW"
30 PRINT "[red]RED"
40 DATA "[wht]WHITE"
50 READ D$
60 PRINT D$

```


Be aware, though, that setting colors from the keyboard requires one byte for each key stroke. For example, D\$ in the above program is five characters in length plus one more for the [wht]. You can verify this by adding the following lines:

```

70 PRINT "LENGTH OF D$="LEN(D$)
80 DATA "WHITE":REM NO COLOR CHANGE FOR THIS ONE
90 READ W$
100 PRINT "LENGTH OF W$="LEN(W$)
110 IF D$=W$ THEN PRINT "D$=W$":GOTO 130
120 PRINT "D$<>W$"
130 IF D$="[wht]WHITE" THEN PRINT "OK!"

```

Color changes may also be included in REMark statements. Such changes will affect the color of the program listing (and so it is useful for highlighting different parts of a program listing), but since BASIC ignores everything after the REM, a color change there will not affect the colors which your program puts out when it runs.



```

5 REM PROGRAM 2.5
6 REM COLOR CHANGES IN REMARK STATEMENTS
10 PLOT 29:REM FLAG OFF
20 REM [red]
30 PRINT "[wht]WHITE"
40 PLOT 18:REM GREEN
50 REM [yel]
60 PRINT "GREEN"

```

There is one final way to change colors, and that is by POKEing suitable CCI codes into the screen refresh memory. We'll take up that interesting topic in Section 8.

2.4 THE "HATCH" CHARACTER

Although the computer has eight background and eight foreground colors, even more are available under certain conditions. Look at the character associated with shift-@ (FLAG off): . (PLOT 96 also does it.) That little character, like any other character, can be printed in any color on a background of any color. But because the background shows through only in between the hatches, the result is a new color which is a function of the foreground and background color combination. Use the PLOT 6 followed by a CCI code, followed by PLOT 96 (or PRINT "") and see what you can come up with. Program 2.6 will display all the possibilities.

```

5 REM PROGRAM 2.6
6 REM THE HATCH CHARACTER
10 PLOT 27,24:REM PAGE MODE
20 PLOT 30,16,29,16:REM SET COLORS
30 PLOT 15,12:REM SMALL CHARACTERS; CLEAR SCREEN
37 REM PRINT HEADINGS AT TOP. THE NUMBERS UNDER
38 REM 'BG' AND 'FG' ARE BACKGROUND AND FOREGROUND
39 REM COLORS FROM BLACK (16) TO WHITE (23)

```



```

40 FOR X=6 TO 54 STEP 16
50 PLOT 3,X,0,17:PRINT "BG ";:PLOT 22:PRINT "FG"
60 NEXT:PRINT
68 REM INITIALIZE BACKGROUND (BG) AND
69 REM FOREGROUND (FG) TO BLACK
70 BG=16:FG=16
80 FOR X=0 TO 48 STEP 16
90 FOR Y=1 TO 31 STEP 2
100 FG=FG+1:REM NEXT FG COLOR
110 IF FG=24 THEN FG=16:BG=BG+1:REM NEXT BG COLOR
120 IF BG=24 THEN X=48:Y=31:GOTO 170:REM ALL DONE
130 PLOT 30,BG,29,FG,3,X,Y:REM SET COLOR & CURSOR
140 PLOT 96,96,96,96,25:REM 4 HATCHES, THEN MOVE OVER
150 PLOT 6,1:PRINT BG;
160 PLOT 6,6:PRINT FG
170 NEXT Y,X
180 PLOT 6,2,3,48,31
190 INPUT "WAITING...";A$:REM WAIT FOR 'RETURN'
200 PLOT 27,11

```

2.5 QUIZ

OK. Time for a quiz.

- (1) Why the PLOT 27,24 (page mode) in line 10?
- (2) Why is there a PLOT 25 (cursor right) in line 140 instead of PLOT 32 (space)?
- (3) What good do lines 180 and 190 do? Why not just end the program?
- (4) Line 200 is supposed to return to scroll mode. Yet when the program ends, the READY prompt and the cursor appear at the top of the page. That is, it seems as if the screen is not scrolling. Why?

Half the fun of this computer business is discovering for yourself how things work. But in case you need assistance, here are some responses to the four questions. (No fair peeking until you've tried to answer them for yourself.)

- (1) Try running the program in scroll mode by changing line 10 to:

```
10 PLOT 27,11:REM SCROLL MODE
```

- (2) PLOT 32 would print a space, and a space is printed in whatever background color was last used. The results for this program would be unaesthetic. See for yourself. PLOT 25 is used because it just moves the cursor without printing anything, not even a space. (Nor will it erase anything.) Of course, PLOT 32 will work just fine if you first change the background color back to black. You could, for example, combine lines 140 and 150 into:

```
140 PLOT 96,96,96,96,6,1,32:PRINT BG;
```


While you're at it, you could simplify things even more. Remember that color changes from the keyboard may be entered into a BASIC program by placing them in REMark statements or within quotes. Let us, in effect, define a new character with this additional line (the "[]" indicate information which you should enter from the keyboard):

```
25 D$="██████[flg on][blk][flg off][red] "
```

And now delete line 150 and change line 140 to:

```
140 PRINT D$;BG;
```

(Note that the semicolon after D\$ is optional. You could also use PRINT D\$BG;.)

- (3) Omit lines 180 and 190 and see what happens. That's no good. (Omitting line 200 as well--return to the scroll mode--is also no good, since the READY prompt will appear near the top of the screen and erase part of your display.) The WAITING... is just a little touch to help clarify what's happening. You could use INPUT "";A\$ instead, but when the program has run, the cursor will just sit near the corner of the screen, blinking. You know that the program is just waiting, but would anybody else?
- (4) Line 200 really does return to the scroll mode, but it doesn't do so until it has received the dummy input from line 190, which causes a carriage return/line feed. Since the system is at that time still in page mode, the cursor "wraps around" to the top of the screen. If you wish the screen to scroll in the usual way at the end of the program, you can reverse the effect of the line feed by PLOT 28 (cursor up):

```
200 PLOT 28,27,11
```

The PLOT 28 moves the cursor up a space, but since it is already at the top of the screen, it "wraps around" to the bottom.

Notice that eight of the colors in the output of that program seem to be identical. For example, 17,19=19,17. Actually, the pairs are not quite the same, but nearly so. There is a slight difference in how the edges of the hatch characters meet up or mesh. Try this:

```
5 REM PROGRAM 2.7
6 REM MORE ON THE HATCH CHARACTER
10 PLOT 30,16,29,23,12:REM SET COLOR;FLAG OFF;CLEAR SCREEN
20 PLOT 15:REM REGULAR HEIGHT CHARACTERS
29 REM DRAW A RECTANGLE OF HATCH CHARACTERS (WHITE ON BLACK)
30 FOR Y=13 TO 21
40 PLOT 3,28,Y
50 FOR X=28 TO 35
60 PLOT 96:REM THE HATCH CHARACTER
70 NEXT: NEXT
80 PRINT
```



```

90 INPUT "PRESS RETURN TO CONTINUE ";A$
97 REM NOW INTERCHANGE BACKGROUND & FOREGROUND
98 REM COLORS & PRINT A SMALLER RECTANGLE OF
99 REM HATCH CHARACTERS INSIDE THE LARGER ONE
100 PLOT 6,56:REM BLACK ON WHITE
110 FOR Y=15 TO 19
120   PLOT 3,30,Y
130   FOR X=30 TO 33
140     PLOT 96
150   NEXT:NEXT
160 PLOT 6,2:REM BACK TO A MANAGEABLE COLOR

```

Notice how the edges of the hatch character do not quite mesh when the foreground and background colors are switched. This could be a useful way of highlighting various colored regions.

2.6 MYSTERY PROGRAM

Try to determine what this program will do. Then type it in and RUN it to see if you were correct.

```

5 REM PROGRAM 2.8
6 REM MYSTERY PROGRAM
10 PLOT 15,6,6,12,27,24
20 J=-1
30 FOR L=1 TO 63
40   PLOT 6,L
50   J=J+1:IF J>15 THEN J=0
60   PLOT 3,J,J
70   FOR K=1 TO 64-J-J:PLOT 96:NEXT
80   PLOT 26
90   PLOT 27,10
100  FOR K=1 TO 32-J-J:PLOT 96:NEXT
110  PLOT 27,24,28
120  FOR K=1 TO 63-J-J:PLOT 26,96,26:NEXT
130  PLOT 27,10
140  FOR K=1 TO 31-J-J:PLOT 28,96,28:NEXT
150  PLOT 27,24
160 NEXT
170 PLOT 6,2
180 END

```

2.7 BLINK

The foreground color of any single character or of any group of characters (including the entire screen) can be made to blink against the background color. Actually, this is not altogether accurate: the foreground color of the blinking character is not replaced with the background color as the character blinks; rather, the foreground color is turned on and off. The result is always a switch from the foreground color to black, back again,

and so on.

The blink mode is turned on in BASIC in one of two ways.

2.8 PLOT 31

Issuing a PLOT 31 will make any subsequent printable character (including special characters and graphics) blink. A PLOT 15 will turn off the blink, but only for subsequent characters: characters which are already blinking will not be affected. Thus, to stop a blinking character, you will have to turn off the blink and then reprint that character. Incidentally, PLOT 15 will also turn off the A7 bit, i.e. will cause further characters to be printed in regular height.

```
5 REM PROGRAM 2.9
6 REM BLINKING CHARACTERS
10 PLOT 15:REM REGULAR CHARACTER HEIGHT
20 PLOT 30,16,29,17:REM SET COLORS; FLAG OFF
30 PRINT "BLINK TEST (OFF)"
40 PLOT 31:REM BLINK ON
50 PRINT "BLINK TEST (ON)"
60 PLOT 30,19,29,17:REM RED ON YELLOW
70 PRINT "BLINK TEST (ON)"
80 PLOT 19:REM FOREGROUND TO YELLOW
90 PRINT "BLINK YELLOW ON YELLOW"
100 PLOT 15:REM BLINK OFF
110 PLOT 30,16,29,18:REM BACK TO A MANAGEABLE COLOR
```

Line 80 blinks yellow on yellow. But since a yellow character on a yellow background is perfectly camouflaged, it is not seen. Because these characters are blinking, they alternate between yellow and black. The result is an on-off message which has a slightly different effect from the message printed by line 60 (which changes colors, too, but is always there to be seen). This is no different from line 40, which also blinks its message. But since that message is red on black, it is not "camouflaged" until the red is turned off (in which case it becomes black on black), whereas the message in line 80 disappears only when its foreground color is turned on. These facts allow you to print messages which blink in two different ways at the same time.

```
5 REM PROGRAM 2.10
6 REM OUT OF PHASE BLINKING
10 PLOT 30,16,29,17:REM SET COLORS; FLAG OFF
20 PLOT 15,12:REM REGULAR CHARACTER HEIGHT; CLEAR SCREEN
30 PLOT 31:REM BLINK ON
40 FOR Y=14 TO 18 STEP 4
50   PLOT 3,25,Y
60   PRINT "*****"
70 NEXT
80 PLOT 27,10:REM VERTICAL MODE
90 FOR X=25 TO 37 STEP 12
100   PLOT 3,X,15
110   PRINT "****"
```



```

120 NEXT
130 PLOT 27,11:REM BACK TO SCROLL MODE
140 PLOT 30,17:REM BACKGROUND TO RED
150 PLOT 3,26,15:PRINT SPC(11)""
160 PLOT 3,26,16:PRINT " RED ALERT ":REM NOTE THE SPACES
170 PLOT 3,26,17:PRINT SPC(11)""
180 PLOT 15:REM BLINK OFF
190 PLOT 16:REM BACKGROUND TO BLACK (NOTE THAT THE
191 REM FLAG WAS LEFT ON FROM LINE 140)
200 PLOT 29:REM FLAG OFF

```

Incidentally, why not try some other character (or characters) for the blinking borders produced by lines 60 and 110? Try "#", "+", or some of the special characters.

2.9 BLINK AND THE CCI CODE

Blinking characters may also be selected by the PLOT 6,n instruction, where $63 < n < 128$. That is, just add 64 to the desired CCI code for the colors. Examples:

```

PLOT 6,64+1    blinks red on black;
PLOT 6,64+38   blinks cyan on blue.

```

A PLOT 6,n instruction will turn off the blink if n is less than 64. Or the blink may be turned off with PLOT 15 or with the BL/A7 OFF key.

3. THE BLIND CURSOR

There are two methods of putting characters on the screen without using the cursor. One way is to POKE appropriate numbers into the screen refresh memory; that will be discussed in Section 8. In this section we examine the blind cursor mode. The blind cursor mode can be entered in two ways.

3.1 BLIND CURSOR ADDRESSING

The PLOT 3,X,Y instruction usually moves the cursor to the given X,Y co-ordinate. But if X=64, then the cursor is placed off the screen at the indicated Y co-ordinate. (See Section 1.1.) And if X is greater than 80, the X value will be ignored entirely, the cursor will remain where it is, and the blind cursor mode will be entered. (Values of X from 65 through 80 will generate an AUTO command--LOAD "MENU":RUN.) Following a value of 81 or more for X, the next two numbers will be taken to be the X,Y co-ordinate of the blind cursor. (These numbers operate just as they would in the visible cursor mode, but the visible cursor is not moved from its present location.) The third number will be taken as the CCI code (just as it is in the PLOT 6,n form). For example,

```
PLOT 3,127,15,20,1
```

will position the blind cursor at X=15,Y=20 and set the CCI code to 1 (red on black). (Although the blind cursor will now print in red, the color for the visible cursor will remain unaffected.) The blind cursor mode is exited by PLOT 27,27 (or ESC,ESC).

```
PLOT 3,127,15,20,1:PRINT "BLIND CURSOR":PLOT 27,27
```

The visible cursor controls (PLOT 10, PLOT 28, PLOT 26, and so on), will not move the blind cursor, although they will move the visible cursor, even when you are in the blind cursor mode. The only blind cursor control is the explicit setting of the X and Y values. However, you need not exit the blind cursor mode and then reenter it in order to re-position the blind cursor. Instead, you may issue another statement of the form PLOT 3,127,X,Y,CCI code.

```
5 REM PROGRAM 3.1
6 REM BLIND CURSOR ADDRESSING
10 PLOT 6,6,12:REM SET COLOR & CLEAR SCREEN
20 FOR J=1 TO 7
30 PLOT 3,127:REM BLIND CURSOR
40 PLOT J,J,J:REM X,Y & CCI CODE
50 PRINT "LINE"J
60 NEXT
70 PLOT 27,27:REM EXIT BLIND CURSOR MODE
```

Notice, though, that as this program runs, the visible cursor moves down the screen as each line is printed in the blind cursor mode. You can slow

things down a bit for a better look by inserting:

```
55 FOR K=1 TO 200:NEXT:REM PAUSE
```

Even if you first put the visible cursor off the screen with, say,

```
15 PLOT 3,64,0
```

you will find that the visible cursor still appears. Why is that? The reason is not hard to find. Whenever a PRINT statement is executed, BASIC automatically follows it with a carriage return/line feed (CR/LF), i.e. PLOT 13,10, unless it is explicitly instructed otherwise with a semicolon or comma. (See Section 1.4.) But CR/LF are visible cursor controls. No wonder, then, that the visible cursor reappears. If you place a semicolon at the very end of line 50, then the visible cursor will not move. The semicolon, however, introduces difficulties of its own, for whenever 64 characters have been PRINTed without a CR/LF, BASIC thinks that it is time, no matter what, to go back down to the next line. And this it will do--with the visible cursor; the blind cursor will continue on its merry way. Compare the following program with Program 1.3 in Section 1.4.

```
5 REM PROGRAM 3.2
6 REM INTERACTION OF VISIBLE & BLIND CURSORS
10 PLOT 6,6,12:REM SET COLOR & CLEAR SCREEN
20 PLOT 3,64,0:REM HIDE THE VISIBLE CURSOR
21 REM (WHO NEEDS TO HIDE A BLIND CURSOR?)
30 FOR J=1 TO 15
40 PLOT 3,127:REM BLIND CURSOR MODE
50 PLOT J,J,1:REM BLIND CURSOR X,Y; CCI CODE=RED
60 PRINT "LINE"J;
70 FOR K=1 TO 200:NEXT:REM PAUSE
80 NEXT
90 PLOT 27,27:REM BACK TO VISIBLE CURSOR MODE
```

You can see that the visible cursor finally appears when J=11, but the blind cursor messages are not disrupted.

3.2 REENTRY TO BLIND CURSOR MODE

The second method of entering the blind cursor mode is PLOT 27,1 (or ESC A). This will position the blind cursor at its previous position and with its previous CCI code. (If the blind cursor has not yet been used, the values will default to the start of the screen refresh memory [see Section 8] with 1 as the CCI code.)

```
5 REM PROGRAM 3.3
6 REM REENTRY TO BLIND CURSOR MODE
10 PLOT 6,6,12:REM SET COLOR & CLEAR SCREEN
20 PLOT 3,127:REM BLIND CURSOR MODE
30 PLOT 15,21:REM X,Y
40 PLOT 1:REM CCI CODE = RED
50 PRINT "THIS IS A"
```



```

60 PLOT 27,27:REM EXIT BLIND CURSOR
70 PRINT "VISIBLE CURSOR"
80 PLOT 27,1:REM REENTRY TO BLIND CURSOR MODE
90 PRINT " BLIND CURSOR TEST"
100 PLOT 27,27:REM EXIT BLIND CURSOR

```

3.3 DOUBLE HEIGHT CHARACTERS IN BLIND CURSOR MODE

If the number following PLOT 3 is greater than 127 (i.e. 128-255), then the blind cursor will automatically print in double height characters. To make program listings easier to read, let us establish a convention of specifying regular height characters in the blind cursor mode by PLOT 3,127 and double height characters in the blind cursor mode by PLOT 3,128.

Try changing line 20 in the previous program to

```
20 PLOT 3,128
```

The double height characters require two lines on the screen. The bottom of the character is always printed on an odd-numbered line and the top of the character is printed one line above. (See Section 1.5.) This is so with the blind cursor as well, except when the A7 bit is off (regular height characters) and the blind cursor specifies that a double height character is to have the bottom half printed not on an odd-numbered line, but on an even-numbered line. In this situation, the bottom of the double height character will be printed over the top! This bug was corrected for V8.79 and later ROMs, and so the following program will produce "broken characters" only with the earlier, V6.78 ROMs.

```

5 REM PROGRAM 3.4
6 REM BROKEN CHARACTERS
10 PLOT 15,6,6,12:REM A7 OFF; SET COLOR; CLEAR SCREEN
20 PLOT 3,128:REM BLIND CURSOR, DOUBLE HEIGHT
30 PLOT 15,20:REM X,Y
40 PLOT 1:REM CCI CODE = RED
50 PRINT "THIS IS A TEST!"
60 PLOT 27,27:REM BACK TO VISIBLE CURSOR

```

This will allow you to put some strange new characters on the screen. It also demonstrates that the double height characters can be broken in half and printed separately, although the bottom half will always appear on an odd-numbered line, and the top half will always appear on an even-numbered line. More about this later, in Section 8.

4. CHARACTERS AND THE PLOT STATEMENT

The PLOT statement in BASIC can be used to access all the ASCII characters and controls available on the computer. (See Appendix A.) For example, the ASCII code for the letter "A" is 65, so PLOT 65 will be an instruction to print the letter "A" and will be equivalent to PRINT "A"; or PRINT CHR\$(65);. Note that the semicolon suppresses the carriage return/line feed. If we wish to include the CR/LF in the PLOT statement, we must do so explicitly: PLOT 65,13,10.

You can specify any number for the argument of the PLOT statement from 0 to 255, but if the number is larger than 127, then 128 will be subtracted from it. Thus, PLOT 128 is equivalent to PLOT 0, and PLOT 193 will print "A" just as PLOT 65 will.

4.1 EXERCISE

For each of instructions (a) through (c) below, refer to the ASCII chart (Appendix A) to determine what the instruction will do. Then type in the instruction and see if you were correct.

- (a) PLOT 68,73,68,32,89,79,85,32,71,69,84,32,73,84,63
- (b) A=28:B=11:PLOT A,B
- (c) PLOT 49,50,51,13,10,56,25,52,13,10,55,54,53

4.2 SPECIAL CHARACTERS

Some micro-computer systems display only the upper case letters. This makes text harder to read, but it does make things cheaper. Unless you have the lower case option, your computer will produce only the upper case letters, corresponding to the ASCII codes 65-90. The ASCII characters for the codes from 97 through 122 would produce the lower case letters, but the computer has a set of special characters instead. In fact, for each of the ASCII codes for the lower case letters, the computer has two special characters. The condition of the FLAG bit determines which of the groups of special characters will be used. The FLAG is turned off by PLOT 29 and will stay off until turned on by PLOT 30. You can duplicate part of Table 4.1 with this program:

```
5 REM PROGRAM 4.1
6 REM SPECIAL CHARACTERS
10 PLOT 29:REM FLAG OFF
20 FOR C=64 TO 95:REM FOR EACH OF THE ASCII
21     REM CHARACTERS "@" THROUGH "_"...
30     PLOT C,32:REM ...PRINT IT AND SKIP OVER A SPACE
40     PLOT 30:REM FLAG ON
50     PLOT C+32,32:REM SPECIAL CHARACTER THEN A SPACE
60     PLOT 29:REM FLAG OFF
70     PLOT C+32:REM OTHER SPECIAL CHARACTER
```



```

80 PLOT 13,10:REM CR/LF
90 NEXT

```

Be inventive with your use of these special characters. Although the

FLAG			FLAG		
PLOT	on	off	PLOT	on	off
96 (224)			112 (240)		
97 (225)			113 (241)		
98 (226)			114 (242)		
99 (227)			115 (243)		
100 (228)			116 (244)		
101 (229)			117 (245)		
102 (230)			118 (246)		
103 (231)			119 (247)		
104 (232)			120 (248)		
105 (233)			121 (249)		
106 (234)			122 (250)		
107 (235)			123 (251)		
108 (236)			124 (252)		
109 (237)			125 (253)		
110 (238)			126 (254)		
111 (239)			127 (255)		

TABLE 4.1

character whose ASCII code is 100 (FLAG off) is a diamond, it could also do as a circle, given the proper context. PLOT 115 (with FLAG off) might be a spade or a space ship. PLOT 112 (FLAG off) could be a chess pawn or a robot. And some of the special characters can serve a dual function, depending on the foreground and background colors used.

```

5 REM PROGRAM 4.2
6 REM CIRCLE AND SQUARE
10 PLOT 15,30,20,29,17,12:REM SET COLOR;FLAG OFF;CLEAR SCREEN
20 PLOT 3,30,10:REM POSITION CURSOR
29 REM DRAW A CIRCLE
30 GOSUB 200
39 REM INTERCHANGE FOREGROUND AND BACKGROUND COLORS

```



```

40 PLOT 6,12
50 PLOT 3,30,13:REM POSITION CURSOR
59 REM PRINT A SQUARE WITH A HOLE IN IT
60 GOSUB 200
70 END
199 REM
200 PLOT 116,117,10,26,26,118,119:RETURN

```

Special characters can also be used for filling gaps. The square with a hole in it in the program above is really not very good, since there are little slices in each side of the square. We can fill up the edges by putting some special characters around the outside. Add these lines to the above program:

```

70 PLOT 6,33:REM INTERCHANGE COLORS AGAIN
80 PLOT 30:REM FLAG ON
90 PLOT 3,30,12:REM POSITION CURSOR
100 PLOT 127,127,10,97,10,26,97
110 PLOT 26,26,26,26,28
120 PLOT 98,26,10,98
130 PLOT 10,101,101
140 PLOT 29:REM FLAG OFF
150 END

```

Letters and numbers are set to the right side of the character position. As a result, some foreground colors on some background colors can cause a character to be somewhat indistinct at its right edge. Consider the numeral "2" printed in black on a red background:

```

5 REM PROGRAM 4.3
6 REM EDGES OF CHARACTERS
10 PLOT 15,12:REM REGULAR HEIGHT; CLEAR SCREEN
20 PLOT 6,8:REM BLACK ON RED
30 PLOT 3,15,15:REM POSITION CURSOR
40 PRINT "2";
50 PLOT 6,2:REM BACK TO A MANAGEABLE COLOR

```

Now see how much more clear the numeral is if it is followed with a red space. Change line 40 to:

```

40 PRINT "2 ";

```

But that still looks a bit awkward. Instead, let's use the character produced by shift-A (PLOT 97) (FLAG on). Put line 40 back to its original version and then add these lines:

```

42 PLOT 6,1:REM RED ON BLACK
44 PLOT 30:REM FLAG ON
46 PLOT 97
48 PLOT 29:REM FLAG OFF, JUST FOR CONVENIENCE

```

Important sections of the screen may be set off by borders using a set of special characters (with the FLAG on) which produce straight or right-angled lines. While text printed in blue is usually difficult to read, borders in

blue are often effective.

```
5 REM PROGRAM 4.4
6 REM BORDERS USING SPECIAL CHARACTERS
10 PLOT 6,4,12:REM BLUE ON BLACK; CLEAR SCREEN
20 PLOT 30:REM FLAG ON
28 REM DRAW A BORDER AROUND A SECTION AT
29 REM THE TOP OF THE SCREEN
30 PLOT 110:REM TOP LEFT CORNER
40 FOR X=1 TO 30:PLOT 101:NEXT:REM TOP LINE
50 PLOT 111:REM TOP RIGHT CORNER
60 PLOT 26,10:REM BACK ONE SPACE AND DOWN
70 PLOT 27,10:REM VERTICAL MODE
80 FOR Y=1 TO 10:PLOT 98:NEXT:REM RIGHT SIDE
90 PLOT 109:REM BOTTOM RIGHT CORNER
100 PLOT 8,10:REM HOME & DOWN ONE LINE
110 FOR Y=1 TO 10:PLOT 97:NEXT:REM LEFT SIDE
120 PLOT 108:REM BOTTOM LEFT CORNER
130 PLOT 27,11:REM BACK TO SCROLL MODE
140 PLOT 28,25:REM ONE LINE UP & OVER A SPACE
150 FOR X=1 TO 30:PLOT 127:NEXT:REM BOTTOM LINE
160 PLOT 29:REM FOR CONVENIENCE, TURN FLAG OFF
170 PRINT
```

Words or phrases may be underlined by setting the FLAG on, and, in the line under the word to be emphasized, printing the shift-E character. Of course, nothing else can be printed in these same spaces. In many cases, however, that will not be cause for concern. Besides, leaving a space between lines of text is a good way of making text easier to read.

```
5 REM PROGRAM 4.5
6 REM UNDERLINING
10 PLOT 6,2:REM GREEN ON BLACK
20 PLOT 15:REM SMALL CHARACTERS
30 PLOT 12:REM CLEAR SCREEN
40 PRINT:PRINT
50 PRINT TAB(5)"WHEN BLANK LINES ARE LEFT BETWEEN EACH LINE,"
60 PRINT
70 PRINT TAB(5)"THE DISPLAY IS MUCH MORE PLEASING TO THE EYE,"
80 PLOT 30:REM FLAG ON
90 PRINT TAB(20)"eeee" :REM SHIFT-E
100 PLOT 29:REM FLAG OFF
110 PRINT TAB(5)"AND THE TEXT EASIER TO READ."
120 PRINT
```

4.3 LARGER CHARACTERS

When the A7 bit is off (PLOT 15), regular height characters are displayed. When the A7 is on, double height characters are printed. But you can use the special characters to produce still larger characters as illustrated in Appendix C. With the A7 off, these characters are double height and

double width. With the A7 on, they are quadruple height, double width.

The following program demonstrates four different character sizes by PLOTTing the numbers of the special characters of which they are composed. (See Appendix A.) Just to liven up the program a bit, lines 10 through 60 will cycle through all the foreground and background color combinations (except those where the foreground color is the same as the background color). Save this program; later on we'll add some things to it.

```
5 REM PROGRAM 4.6
6 REM CHARACTERS IN 4 SIZES
10 BG=16:FG=16:REM START WITH FOREGROUND & BACKGROUND=BLACK
20 FG=FG+1:IF FG=24 THEN FG=16:BG=BG+1
30 IF FG=BG THEN 20:REM NOTHING WILL BE SEEN IF FG=BG
40 IF BG=24 THEN 10:REM WHEN ALL DONE, DO IT AGAIN
50 PLOT 29,FG:REM FOREGROUND COLOR
60 PLOT 30,BG:REM BACKGROUND COLOR
70 PLOT 12:REM CLEAR SCREEN
80 PLOT 15:REM REGULAR CHARACTER HEIGHT
90 GOSUB 700:REM PAUSE BEFORE BEGINNING
100 GOSUB 300:REM PRINT MESSAGE IN REGULAR HEIGHT
110 PLOT 14:REM DOUBLE HEIGHT
120 GOSUB 300:REM PRINT MESSAGE
130 PLOT 15:REM REGULAR HEIGHT SPECIAL CHARACTERS
131 REM NOTE: THE SPECIAL CHARACTERS USED REQUIRE
132 REM THAT THE FLAG BE ON, WHICH IT IS
133 REM BECAUSE OF LINE 60.
140 PLOT 3,12,15:REM POSITION CURSOR
150 GOSUB 400:REM PRINT MESSAGE IN LARGE CHARACTERS
160 PLOT 14:REM A7 ON
170 PLOT 3,12,13:REM POSITION CURSOR
180 GOSUB 400
190 GOTO 20
298
299 REM SUBROUTINE FOR REGULAR & DOUBLE HEIGHT
300 PLOT 3,18,15:REM POSITION CURSOR
310 PRINT "C O L O R G R A P H I C S"
320 GOSUB 700:REM HIDE CURSOR & PAUSE
330 RETURN
398
399 REM SUBROUTINE FOR DOUBLE WIDTH CHARACTERS
400 RESTORE 800
410 FOR C=1 TO 14:REM FOR EACH OF THE 14 CHARACTERS...
420 GOSUB 600:REM PRINT TOP HALF OF CHARACTER
430 PLOT 26,26,10:REM RE-POSITION CURSOR FOR BOTTOM HALF
440 GOSUB 600:REM PRINT BOTTOM HALF
450 PLOT 32:REM SPACE
460 PLOT 26,28:REM BACK & UP
470 PLOT 32:REM SPACE
480 NEXT
490 GOSUB 700:REM HIDE CURSOR & PAUSE
500 RETURN
598
599 REM SUBROUTINE TO PRINT HALF OF A LARGE CHARACTER
600 READ A:PLOT A:READ A:PLOT A
```



```
610 RETURN
698
699 REM SUBROUTINE TO HIDE CURSOR AND PAUSE
700 PLOT 3,64,0
710 FOR T=1 TO 1500:NEXT
720 RETURN
797
798 REM DATA FOR THE 14 LARGE CHARACTERS
799 REM C
800 DATA 116,102,118,105
804 REM O
805 DATA 116,117,118,119
809 REM L
810 DATA 97,32,108,127
814 REM O
815 DATA 116,117,118,119
819 REM R
820 DATA 123,100,97,124
824 REM SPACE
825 DATA 32,32,32,32
829 REM G
830 DATA 116,102,118,125
834 REM R
835 DATA 123,100,97,124
839 REM A
840 DATA 126,124,110,111
844 REM P
845 DATA 123,100,97,32
849 REM H
850 DATA 97,98,110,111
854 REM I
855 DATA 111,101,109,127
859 REM C
860 DATA 116,102,118,105
864 REM S
865 DATA 99,102,103,100
```


5. TEST MODE

The test mode will fill the entire screen with any character you wish, using whatever colors and character height were last specified. The PLOT 27,25 instruction sets up the test mode. The next byte (whether in a PLOT or a PRINT statement) determines the character. The following two statements are identical in their results (except that the first will also perform a carriage return/line feed): they will fill the entire screen with the letter "A".

```
PLOT 27,25:PRINT "A"  
PLOT 27,25,65
```

The test mode may also be selected from the keyboard by ESC Y, but BASIC will give you a syntax error message when you finally hit RETURN.

Using the test mode can sometimes be a flashy way of introducing a program.

```
5 REM PROGRAM 5.1  
6 REM DEMO INTRODUCTION USING THE TEST MODE  
10 PLOT 14:REM DOUBLE HEIGHT CHARACTERS  
20 PLOT 6,33:REM RED ON BLUE  
30 PLOT 27,25:REM TEST MODE  
40 PLOT 12:REM SEE CHARACTER CHART, APPENDIX B  
50 PLOT 6,2:REM GREEN ON BLACK  
60 PLOT 3,26,4:REM POSITION CURSOR  
70 PRINT "A MAZE GAME"
```

Try variations using regular height characters for filling the screen and double height for the title, or vice-versa. Perhaps spaces surrounding the title would look better.

Many characters have unique and sometimes unexpected properties when the entire screen is filled with them.

```
5 REM PROGRAM 5.2  
6 REM DEMO INTRODUCTION #2  
10 GOSUB 100:REM SET UP  
20 GOSUB 200:REM FILL THE SCREEN  
30 GOSUB 300:REM PRINT A BORDER AROUND THE SCREEN  
40 GOSUB 400:REM SET UP SPACE FOR THE TITLE  
50 GOSUB 600:REM PRINT THE TITLE  
60 END  
98  
99 REM SUBROUTINE TO SET UP  
100 PLOT 15:REM REGULAR HEIGHT  
110 PLOT 6,2:REM GREEN ON BLACK  
120 PLOT 27,24:REM PAGE MODE  
130 RETURN  
198  
199 REM SUBROUTINE TO FILL THE SCREEN  
200 PLOT 27,25:REM TEST MODE  
210 PLOT 124:REM SEE CHARACTER CHART, APPENDIX B
```



```

220 RETURN
298
299 REM SUBROUTINE TO ADD A BORDER
300 PLOT 6,16:REM BACKGROUND TO GREEN
310 PLOT 8,11:REM HOME CURSOR & ERASE TOP LINE IN GREEN
320 PLOT 28,11:REM CURSOR UP (=BOTTOM LINE) & ERASE LINE
330 PLOT 27,10:REM VERTICAL MODE
340 GOSUB 390:REM ERASE A COLUMN (32 LINES)
350 PLOT 26:REM CURSOR LEFT (=RIGHT SIDE OF SCREEN)
360 GOSUB 390:REM ERASE A COLUMN
370 PLOT 27,24:REM BACK TO PAGE MODE
380 RETURN
388
389 REM SUBROUTINE TO PRINT 32 SPACES IN VERTICAL MODE
390 FOR J=1 TO 32:PLOT 32:NEXT:RETURN
398
399 REM SUBROUTINE TO SET UP SPACE FOR THE TITLE
400 PLOT 6,2:REM GREEN ON BLACK
410 PLOT 30:REM FLAG ON
420 PLOT 3,20,5:REM POSITION CURSOR
430 PLOT 110:REM TOP LEFT CORNER
440 FOR J=1 TO 24:PLOT 101:NEXT:REM TOP
450 PLOT 111:REM TOP RIGHT CORNER
460 PLOT 26,10,98:REM BACK ONE & DOWN, THEN PRINT RIGHT SIDE
470 PLOT 3,20,6:REM RE-POSITION CURSOR
480 PLOT 97,26,10:REM LEFT SIDE;CURSOR BACK & DOWN
490 PLOT 108:REM BOTTOM LEFT CORNER
500 FOR J=1 TO 24:PLOT 127:NEXT:REM BOTTOM
510 PLOT 109:REM BOTTOM RIGHT CORNER
520 PLOT 29:REM FLAG OFF FOR FUTURE CONVENIENCE
530 RETURN
598
599 REM SUBROUTINE TO PRINT THE TITLE
600 PLOT 6,3:REM YELLOW ON BLACK
610 PLOT 3,21,6:REM POSITION CURSOR
620 PRINT "ENGINEERING APPLICATIONS"
630 RETURN

```

The test mode can also be used to erase the screen:

```
PLOT 27,25,32
```

Notice, however, that when the screen is erased by a PLOT 12 (or by the ERASE PAGE key), the cursor is homed, whereas erasing the screen using the test mode does not disturb the cursor.

Erasing the screen several times in different background colors was used in Section 2.2 to heighten the effect of an explosion. A similar technique can be used with the test mode, but since the test mode fills the screen with any character, not just blanks, some further special effects can be added to an explosion.

```

5 REM PROGRAM 5.3
6 REM EXPLOSION EFFECT USING TEST MODE
10 PLOT 14:REM DOUBLE HEIGHT

```



```
20 PLOT 6,56:REM  BLACK ON WHITE
30 C$="+":REM  FILL CHARACTER
40 GOSUB 200:REM  EXPLOSION
50 PLOT 6,25
60 C$="-"
70 GOSUB 200
80 PLOT 6,15
90 C$="*"
100 GOSUB 200
110 PLOT 6,2
120 PLOT 12:REM  ERASE SCREEN
130 END
198
199 REM  TEST MODE SUBROUTINE
200 PLOT 27,25:PRINT C$
210 FOR J=1 TO 50:NEXT:REM  SLIGHT PAUSE
220 RETURN
```


FOR LARGE SIZE
CHARACTERS. NOT ON
SEE INDEX #1
MAN. FOR FORUM.

6. PLOT MODES

"A7 OFF"

Each regular size character rectangle on the display is composed of 48 small squares, six wide and eight high. (Figure 6.1) In the PLOT mode, the rectangle is divided into eight smaller rectangles--plot blocks--each of which is three squares wide and two squares high. (Figure 6.2.)

ASCII CHARACTERS
FOR MOST CHARACTERS
THE FIRST COLUMN AND
THE BOTTOM LINE ARE BLANK.
THEREBY FORMING LETTER
AND LINE SPACING WHEN
VIEWED ON THE SCREEN.

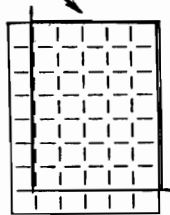


FIG. 6.1

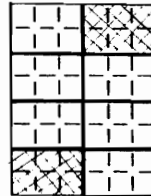


FIG. 6.2

CHARACTER PLOT MODE
PLOT 2, 254, -8, -16, 255.
i.e. Takes up full area 6x8 squares.

The PLOT 2 instruction sets the computer up for drawing points, lines and "characters" composed of these plot blocks. Any subsequent output to the display will be interpreted as a plot number. The plot number is any number from 0 through 254. (255 exits the plot mode.) The numbers from 240 through 254 have special significance. Let's start with PLOT 254.

General Plot Mode	2	General plot mode introduction. Unless instructed otherwise, the computer will automatically be in the point plot submenu.
Vectors	240	Incremental vector plot
	241	Y co-ordinate of vector end point
	242	X co-ordinate of vector end point
Y Bar Graph	243	Incremental Y bar graph
	244	Y co-ordinate of top of vert. bar
	245	X co-ordinate of vert. bar
	246	Y co-ordinate of bottom of vert. bar
X Bar Graph	247	Incremental X bar graph
	248	X co-ordinate of right of horiz. line
	249	Y co-ordinate of horiz. bar
	250	X co-ordinate of left of horiz. bar
Point Plot	251	Incremental point plot
	252	Y co-ordinate of point
	253	X co-ordinate of point
Character Plot	254	
Plot Mode Escape	255	

Table 6.1

6.1 PLOT 254 — CHARACTER PLOT

After entering the general plot mode with PLOT 2, the PLOT 254 instruction will take each subsequent number and print a "character" wherever the cursor is. (After each character is drawn, the cursor will move over a space--or down if you are in the vertical mode.) Since the number 3 will generate a character, it cannot be used to control the cursor, as it ordinarily is. If you wish to move the cursor to a given location, you will have to position the cursor first and then enter the general plot mode and the character plot submode. If you then wish to plot a character elsewhere, you will have to exit the plot mode (PLOT 255), re-position the cursor in the usual way, and then reenter the general plot mode and the character plot submode (PLOT 2,254).

What are these "characters" which the character plot produces? As previously mentioned, each character position on the screen is a rectangle divided into eight smaller rectangles. Each of these smaller rectangles has a number associated with it, and the plot number which follows the character plot submode introduction will determine which one or more of these rectangles will be printed. For example, according to Figure 6.3, the instruction PLOT 2,254,1,255 will enter the general plot mode, then the character plot submode, draw rectangle #1--that is, the rectangle in the upper left of the character position--and then exit the plot mode. PLOT

1	16
2	32
4	64
8	128

FIG. 6.3

1			
	32		
		4	
			128

$1+32=33$ $4+128=132$

FIG. 6.4

2,254,8+16,255 will produce the lower left rectangle and the upper right one together. PLOT 2,254,15,255 will produce the four left hand rectangles. ($15=1+2+4+8$.) And so on. We can produce a short, slanted line by plotting the two characters from Figure 6.4 with this instruction:

PLOT 2,254,33,132,255

Remember, the general plot mode is entered with a PLOT 2 instruction, and the character plot submode is specified by PLOT 254. All further PLOT instructions will be understood as character plot numbers (except for PLOT 255, which will exit the plot mode).

You can build all sorts of pictures using this character plot mode: lines, fences, brick buildings, space ships.... What you cannot do, however, is print an entire space. That is, if you wanted to plot all eight rectangles at once, you would have to PLOT 255 ($=1+2+4+8+16+32+64+128$). But 255 is the plot mode escape. If you wish to fill up an entire space, you will have to exit the plot mode, put the color you want into the background, print a space, and then, if you had more characters to plot, change back to the desired color and reenter the general plot mode and the character plot submode. (See the chess pieces using the character plot submode in Section 12.6.) The following program, for example, will print the characters in



FIG. 6.5



FIG. 6.6

Figure 6.5.

```

5 REM PROGRAM 6.1
6 REM CHARACTER PLOT
10 PLOT 6,1:REM RED ON BLACK
20 PLOT 2:REM GENERAL PLOT MODE
30 PLOT 254:REM CHARACTER PLOT SUBMODE
40 PLOT 96:REM FIRST PART
50 PLOT 255:REM EXIT PLOT MODE
60 PLOT 6,8:REM BACKGROUND TO RED
70 PLOT 32:REM SECOND PART (A SPACE)
80 PLOT 6,1:REM BACK TO RED ON BLACK
90 PLOT 2:REM GENERAL PLOT MODE
100 PLOT 254:REM CHARACTER PLOT SUBMODE AGAIN
110 PLOT 6:REM THIRD PART
120 PLOT 255:REM EXIT PLOT MODE

```

In this particular case, however, it would have been easier to have defined the characters as in Figure 6.6, and which could be accomplished by:

```
PLOT 6,1,2,254,246,111,255
```

In the following program, enter color changes from the keyboard as indicated in brackets: [].

```

5 REM PROGRAM 6.2
6 REM DEMONSTRATING THE CHARACTER PLOT
10 PLOT 15,30,16,29,22,12:REM SET UP
20 PRINT
30 PRINT TAB(15)"A DEMONSTRATION OF THE CHARACTER PLOT"
40 PRINT
50 PRINT TAB(25)"[yel]PLOT 2,254,N"
99
100 PLOT 3,0,5,11:REM ERASE ANY PREVIOUS INPUT
110 INPUT "[grn]SELECT A NUMBER FROM 0 TO 254:[red] ";N$
120 N=VAL(N$):IF N<0 OR N<>INT(N) OR N>254 THEN 100
130 PRINT
140 PLOT 11:REM ERASE ANY PREVIOUS DISPLAY
150 PRINT "[yel]PLOT 2,254,[red]"N" [yel]GIVES: [red]";
160 PLOT 2,254,N,255
170 PRINT:PRINT
180 PRINT "[wht]HERE ARE TWO LINES OF IT...[red]"
190 PRINT
200 PLOT 2,254:FOR J=1 TO 128:PLOT N:NEXT:PLOT 255
210 PLOT 3,10,16:PRINT "[wht]VERTICAL MODE:[red]"
220 PLOT 27,10:REM VERTICAL MODE
230 PLOT 3,30,15:GOSUB 300

```



```

240 PLOT 3,31,15:GOSUB 300
250 PLOT 27,11:REM RETURN TO SCROLL MODE
260 GOTO 100:REM BACK FOR MORE
298
299 REM SUBROUTINE TO PLOT A VERTICAL LINE
300 PLOT 2,254:FOR J=1 TO 10:PLOT N:NEXT:PLOT 255
310 RETURN

```

In some cases, characters can be plotted faster and more efficiently by using the PLOT 6,n instruction where $n > 128$. More on that in Section 7.

6.2 PLOT 253 — POINT PLOT

Since there are 32 regular height character lines on the screen, and since each character is four plot blocks high, the screen is $32 \times 4 = 128$ plot blocks high. Since there are 64 characters per line, and since each character is two plot blocks wide, the display is 128 plot blocks wide. As usual, numbering begins with zero. But whereas the cursor positions are numbered from left to right, top to bottom, the plot block positions are numbered from left to right, bottom to top.

After entering the general plot mode with PLOT 2, the PLOT 253 instruction tells the computer to take the next number as the X co-ordinate of a single plot block. After the X co-ordinate is given, the Y co-ordinate is automatically expected, which, if given, will cause the point at that location to be plotted. The computer will then automatically expect the X co-ordinate of another point, then the Y co-ordinate, and so on.

```

5 REM PROGRAM 6.3
6 REM POINT PLOT SUBMODE
10 PLOT 2:REM GENERAL PLOT MODE
20 PLOT 253:REM POINT PLOT SUBMODE
30 PLOT 0,0:REM X,Y OF BOTTOM LEFT CORNER
40 PLOT 127,0:REM X,Y OF BOTTOM RIGHT CORNER
50 PLOT 127,127:REM X,Y OF TOP RIGHT CORNER
60 PLOT 0,127:REM X,Y OF TOP LEFT CORNER
70 PLOT 255:REM EXIT PLOT MODE

```

Line 20 in the program above is not necessary. Upon entering the general plot mode, the computer will automatically be in the point plot submode unless instructed otherwise. (See for yourself by deleting line 20.) The various plot submodes are entered by a plot number from 240 to 254, so upon entering the general plot mode with PLOT 2, the next number is automatically taken to be the X co-ordinate of a point unless that number is equal to or greater than 240. But this means that the X co-ordinate can be in the range 0-239, yet the display is only 128 blocks wide. What will happen if the X co-ordinate exceeds the width of the display? Try it. It "wraps around" to the other side. That is, if a number greater than 128 is given, 128 is subtracted from it. Similarly for the Y co-ordinate.

The reason for having the plot number 253 available even though the computer is automatically expecting the X,Y co-ordinate of a point after entering the general plot mode, is to allow the point plot submode to be easily accessed from the other plot submodes without having to exit the

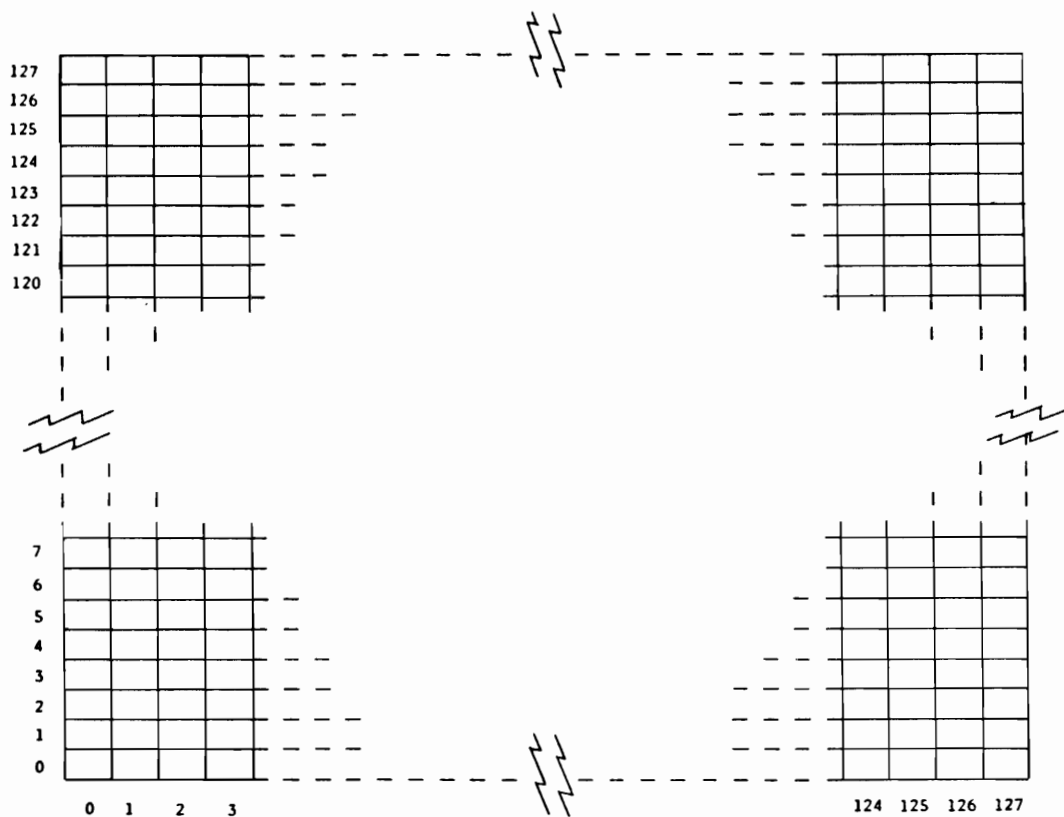


FIG. 6.7

plot mode and then reenter it. In fact, all the plot submodes (except the character plot submode) are directly accessible from any other simply by specifying its code number.

It is sometimes necessary to co-ordinate the printing of characters (either the ASCII characters or the character plot characters) with the plotting of point blocks. That is, it would be handy to be able to convert cursor co-ordinates to point plot co-ordinates, and vice-versa. You might find these equations useful:

$$\begin{aligned} X_p &= 2 \cdot X_c & X_c &= \text{INT}(X_p/2) \\ Y_p &= 127 - 4 \cdot Y_c & Y_c &= \text{INT}((127 - Y_p)/4) \end{aligned}$$

where X_p and Y_p are the X,Y co-ordinates of the top left plot block in a character position, and X_c and Y_c are the X,Y co-ordinates of a cursor position.

Although you can plot up to eight blocks per character rectangle, the plotting of a plot block in any character position will erase any ASCII character there. Similarly, the printing of an ASCII character will erase any and all plot blocks which happen to be at that location.

The foreground/background colors may be specified as usual for point plotting (but of course you will have to specify the colors before entering the general plot mode). However, colors are defined for an entire character

position. That is, if you have plotted a block in, say, red, and you subsequently plot, say, a green block in that same character position, both blocks will become green.

```
5 REM PROGRAM 6.4
6 REM RESOLUTION OF COLORS
10 PLOT 15,6,6,12:REM SET UP
20 PLOT 6,15:REM WHITE ON RED
30 PLOT 3,15,15:PRINT "TEST TEST TEST"
40 FOR J=1 TO 700:NEXT:REM PAUSE
50 PLOT 6,2:REM GREEN
60 PLOT 2:REM GENERAL PLOT MODE
69 REM PLOT GREEN BLOCKS AT THE TOP OF THE LINE
70 FOR X=0 TO 64
80 PLOT X,67
90 NEXT
100 PLOT 255:REM EXIT PLOT MODE
110 FOR J=1 TO 700:NEXT:REM PAUSE
120 PLOT 6,3:REM YELLOW
128 REM PLOT YELLOW BLOCKS TWO PLOT POSITIONS DOWN FROM
129 REM THE GREEN ONES
130 PLOT 2:REM GENERAL PLOT MODE
140 FOR X=0 TO 64
150 PLOT X,65
160 NEXT
170 PLOT 255:REM EXIT PLOT MODE
```

The RND(x) function in BASIC generates pseudo-random numbers between 0 and 1. The idea of the pseudo-random number generator is to produce numbers which are evenly distributed over the range 0 to 1. We can test the computer's pseudo-random number generator by PLOTting points at random co-ordinates. In that way, we can visually determine whether the numbers have been more or less evenly distributed. Let's assume that about 2000 PLOTs will be enough. Since the point plot co-ordinates go up to 127 along both axes, and since the pseudo-random number generator yields numbers less than 1, we can use this function

$X=128 \times \text{RND}(1)$

to produce numbers which will go as high as 127.999. Should we also take the integer of that number before PLOTting it? We could. But the plot mode will just ignore the fractional part when plotting. 127.9 will be plotted as 127. So let's not bother with the INT(x) function.

```
5 REM PROGRAM 6.5
6 REM TESTING THE PSEUDO-RANDOM NUMBER GENERATOR
10 PLOT 15,6,6,12:REM SET UP
20 PLOT 2:REM GENERAL PLOT MODE
30 FOR J=1 TO 2000
40 X=128*RND(1):Y=128*RND(1)
50 PLOT X,Y
60 NEXT
70 PLOT 255
```


6.3 AN ARTILLERY GAME

In order to illustrate further the point plot, and in order to introduce a new and convenient trick, let's design a very simple artillery game. Let's put a gun--a pill box--somewhere on the left of the screen and require the player to shoot projectiles at a target on the right. To represent the ground we can put the cursor at the bottom of the screen, set the background color to green, then PLOT 11 (erase line). We can orient the pill box at some random spot on the ground somewhere on the left and the target at some random spot on the ground somewhere on the right. Let's draw the pill box in red and make it a simple one by printing a space with the background set to red. We can use the rook character (shift-R) in light blue for the target. For the projectile we will be using the point plot mode.

Now we must deal with the mathematics of projectile motion. Just to simplify matters, let's not try to be too picky: let's assume there is no wind or air resistance. And let's assume that the muzzle velocity of the projectile is constant. Of course, you may refine the game to any degree you wish. You might even end up with something like Intelligent Systems Corporation's "SHOOT" game.

The only variable which the player may specify is the angle of inclination of the gun. Now, in order to plot the path of the projectile, we must know its varying X and Y co-ordinates and PLOT them as they change with time. We can't vary time continuously, so we'll choose a very small increment for T (time) so as to produce a smooth curve. If we start by knowing M, the muzzle velocity, and A, the angle of inclination of the gun, then the initial problem is to determine the X and Y component vectors in Figure 6.8.

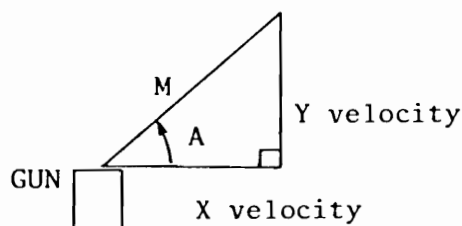


FIG. 6.8

From trigonometry we learn that $\text{SIN}(A)=Y/M$ and $\text{COS}(A)=X/M$. So we may write

$$Y=M*\text{SIN}(A), \quad X=M*\text{COS}(A).$$

Since wind or air resistance is to be neglected, the X vector will remain constant, and this means that

$$X1=X0+VX*T,$$

where $X0$ is a given X co-ordinate, VX is the horizontal velocity $=M*\text{COS}(A)$, and T is the time increment for PLOTting. We know where the gun is to start with, and so we know what $X0$ is to start with. The program will compute $X1$, PLOT the projectile, set $X0=X1$, and go back to figure out the new $X1$,

and so on.

The vertical velocity is more complicated, because, due to the pull of gravity, it does not remain constant. If we let the acceleration due to gravity be $G=-32$ ft/sec, then we might use this equation for determining the vertical component at the end of a time increment:

$$VY = VY + .5 * G * T * T.$$

And (fudging things a bit) the new Y co-ordinate of the projectile at the end of a time increment will be

$$Y1 = Y0 + VY * T,$$

where $Y0$ is the old co-ordinate. As with the X co-ordinate, we start off knowing the initial $Y0$, calculate $Y1$, go back to calculate a new $Y1$, and so on.

If we let the time increment, T , be small enough (say, $T=.1$), then the plot of the projectile will be fairly even.

Figure 6.9 shows the logic of PLOTting the projectile. The logic of the whole program is given in Figure 6.10.

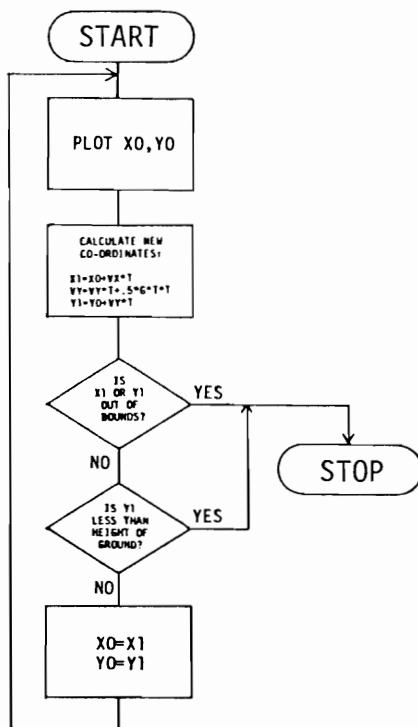


FIG. 6.9

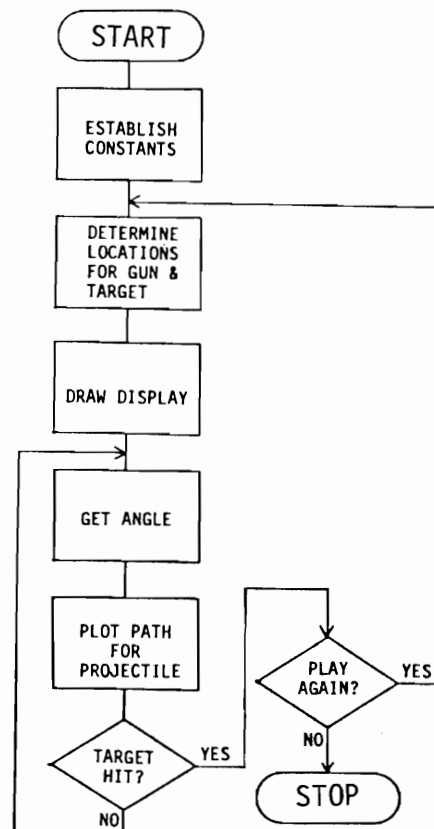


FIG. 6.10

If the projectile passes through the target, then the target will of course disappear. In Section 8 we will see how to PEEK directly into the screen memory locations. (If you wish, you may skip ahead to read that Section, but it will not really be necessary.) For now, you may accept that lines 70 and 80 of the program really do check to see whether the rook (the

target) is still on the screen. If it is not on the screen where it was first printed, then it must have been erased by a projectile, and we may then say that it was hit.

A few more details must be cleared up. We know that the ground level is a green line at the bottom of the screen, and so we know that the projectile will have hit the ground whenever its newly calculated Y co-ordinate is less than or equal to 3. (See Figure 6.11.) However, what

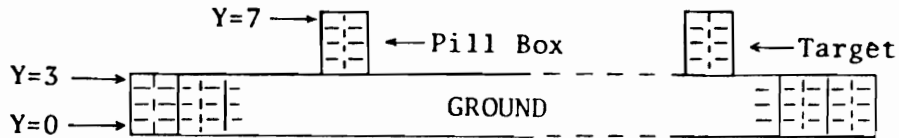


FIG. 6.11

if the Y co-ordinate is between 3 and 4? If it is, say, 3.64, then it will be PLOTTed at Y=3. (Remember, the plotting function neglects the fractional element.) So we had better check to see if the projectile's Y co-ordinate is less than 4.

Note that although BASIC can handle the necessary trigonometric functions, it does so in radians instead of degrees. Since we can expect the player to enter his gun elevation in degrees, we must convert to radians. One degree is equal to $\text{Pi}/180$ radians, so the player's chosen angle, A, will convert to $A \cdot \text{Pi}/180$.

Here is the program. Some improvements will be offered later, but for now, type it in and RUN it. As usual, the brackets indicate information which you should enter from the keyboard.

```

5 REM PROGRAM 6.6
6 REM AN ARTILLERY GAME
10 PLOT 15,30,16,29,22,12,27,24:REM SET UP
20 PRINT TAB(27)"ARTILLERY"
28
29 REM SET UP THE CONSTANTS
30 GOSUB 1000
38
39 REM SET UP THE DISPLAY
40 GOSUB 2000
48
49 REM GET ANGLE FOR EACH SHOT
50 GOSUB 3000
58
59 REM PLOT PROJECTILE PATH
60 GOSUB 4000
68
69 REM CHECK TO SEE IF TARGET WAS HIT
70 SC=28672+128*TY+TX+TX:REM SCREEN RAM LOCATION FOR TARGET
80 IF PEEK(SC)=114 THEN 50:REM IT'S STILL THERE
88
89 REM PLAYER HAS HIT TARGET
90 PRINT:PRINT "[mag]GOOD SHOT!":PRINT
100 INPUT "[grn]CARE TO PLAY AGAIN? ";A$
110 A$=LEFT$(A$,1)

```



```

120 IF A$="Y" OR A$="O" THEN RUN:REM ALLOW ANSWER OF "OK"
130 IF A$<>"N" THEN 100
140 END
997
998 REM ***** SUBROUTINE TO SET UP CONSTANTS *****
999
1000 G=-32:REM GRAVITY
1010 T=.1:REM TIME INCREMENT
1020 M=15:REM MUZZLE VELOCITY (YOU MIGHT WANT TO CHANGE THIS)
1027
1028 REM GET RANDOM X CO-ORDINATE BETWEEN
1029 REM 3 AND 25 FOR THE PILL BOX
1030 PX=INT(22*RND(1))+3
1038
1039 REM Y CO-ORDINATE OF PILL BOX IS ALWAYS 30
1040 PY=30
1047
1048 REM GET RANDOM X CO-ORDINATE BETWEEN
1049 REM 40 AND 55 FOR THE TARGET
1050 TX=INT(15*RND(1))+40
1058
1059 REM TARGET Y CO-ORDINATE IS ALWAYS 30
1060 TY=30
1070 RETURN
1996
1997 REM ***** SUBROUTINE TO PRINT THE DISPLAY *****
1998
1999 REM GROUND
2000 PLOT 6,16:REM BACKGROUND TO GREEN
2010 PLOT 3,0,31,11
2018
2019 REM PILL BOX
2020 PLOT 6,8:REM BACKGROUND TO RED
2030 PLOT 3,PX,PY,32:REM PRINT A SPACE IN RED FOR THE PILL BOX
2039 REM TARGET
2040 PLOT 6,6:REM CYAN ON BLACK
2050 PLOT 3,TX,TY,114:REM USE ROOK FOR THE TARGET
2060 RETURN
2997
2998 REM ***** SUBROUTINE TO GET ANGLE *****
2999
3000 PLOT 6,2
3010 PLOT 3,0,1,11:REM ERASE ANY PREVIOUS INPUT
3020 INPUT "ANGLE?[red] ";A$
3029 REM IF A<0 THEN ABORT AND START AGAIN
3030 A=VAL(A$):IF A<0 THEN RUN
3039 REM TRANSLATE INTO RADIANS
3040 A=A*3.1415926/180
3049 REM CALCULATE INITIAL X AND Y VELOCITIES OF PROJECTILE
3050 VX=M*COS(A)
3060 VY=M*SIN(A)
3070 RETURN
3994
3995 REM ***** SUBROUTINE TO PLOT PATH OF PROJECTILE *****

```



```

3996
3997 REM FIRST SET UP INITIAL X CO-ORDINATE. LET'S
3998 REM HAVE IT BEGIN JUST ABOVE THE UPPER RIGHT
3999 REM OF THE PILL BOX
4000 X0=2*PX+1
4008
4009 REM INITIAL Y CO-ORDINATE IS ALWAYS 8
4010 Y0=8
4019
4020 PLOT 6,7,2:REM COLOR=WHITE; ENTER GENERAL PLOT MODE
4030 PLOT X0,Y0:REM PLOT PROJECTILE
4038
4039 REM CALCULATE NEW X CO-ORDINATE
4040 X1=X0+VX*T
4048
4049 REM CHECK FOR OUT OF BOUNDS
4050 IF X1<0 OR X1>127 THEN 4200
4058
4059 REM CALCULATE NEW Y VELOCITY
4060 VY=VY+.5*G*T*T
4068
4069 REM CALCULATE NEW Y CO-ORDINATE
4070 Y1=Y0+VY*T
4078
4079 REM CHECK FOR TOO HIGH
4080 IF Y1>127 THEN 4200
4088
4089 REM CHECK FOR HITTING THE GROUND
4090 IF Y1<4 THEN 4200
4098
4099 REM LET THE NEW CO-ORDINATES NOW BE THE OLD ONES
4100 X0=X1:Y0=Y1
4108
4109 REM BACK FOR NEXT PLOT & NEXT COMPUTATIONS
4110 GOTO 4030
4198
4199 REM END OF PLOT
4200 PLOT 255:REM EXIT PLOT MODE
4210 RETURN

```

Variations on this game will naturally occur to you. Try varying the time increment (T) and the muzzle velocity (M). You might even let the player choose the muzzle velocity for each shot.

Note that the entire projectile path is plotted and remains on the screen. Each time a new projectile is fired, a new path is plotted. Pretty soon the display will be a mess of paths. But we will be able to do something about that. If, before entering the general plot mode, the FLAG is turned on (PLOT 30), then if a plot block is PLOTted twice, it will disappear. That is, PLOTting a point will turn it on, and PLOTting it again will turn it off. This is a most convenient tool for the artillery game. Just before entering the general plot mode (line 4020), add PLOT 30. And after exiting the plot mode in line 4200, add PLOT 29 to turn the FLAG off. Now just before putting the new co-ordinates into X0 and Y0 (line 4100), PLOT the projectile a second time at the old location. You might add

4095 PLOT X0,Y0:REM ERASE PROJECTILE AT LAST LOCATION

That should make the game much better.

Also, now note that if the newest co-ordinates are out of bounds, the PLOTting routine will be exited. But this can leave the projectile printed up in the air in its last location before going out of bounds, and that's no good. To get rid of it, you could change line 4200:

4200 PLOT X0,Y0,255

I hope you will fiddle with this program--adding little goodies, changing colors, trying out new ideas--until you have it just the way it ought to be. You must have an aesthetic appreciation for what goes on the display. It does not have to be Rembrandt, but it ought to be artful. Later on I will suggest a few additions to make the game more lively.

6.4 PLOT 251 — INCREMENTAL POINT PLOT

After PLOTting a point, the instruction PLOT 251 will set the computer up to move that point. Just how it is moved will be determined by what PLOT number is subsequently issued. It may be anywhere from 0 through 239. (240-255, remember, are instructions to enter one of the other plot submodes.) Actually, the number will define two movements of the point. If we interpret the number as an eight bit binary number, then the four most significant (i.e. left most) bits determine the increment for PLOTting the first of the two additional blocks, and the four least significant (i.e. right most) bits determine the increment for the second block from the first.

Since two bits are used for determining the increment for each co-ordinate, there are exactly four possibilities for each. They are defined in Table 6.2

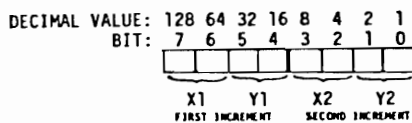


FIG. 6.12

If the binary value of the two bits is...	...then the increment of that co-ordinate will be...
00	None
01	-
10	+
11	None

TABLE 6.2

Let's take an example. Suppose we have PLOTted a point in the point plot submode and wish to PLOT a second point just to the right of it and a third point down and to the right of the second. That is, we wish the

result shown in Figure 6.13. Since the incremental point plot submode will move the original block twice for us, we needn't PLOT all three points by separate instructions in the point plot submode. Instead, after having drawn the first point, we enter the incremental point plot submode with PLOT 251. For the first increment we wish the X co-ordinate to increase but the Y co-ordinate to stay the same. According to the Table above, we want the two bits for X1 to be 10 and the two bits for Y1 to be 00. We have now determined the four most significant bits for the number we want to use. (See Figure 6.14.) The X co-ordinate for the second block is to have

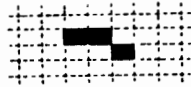


FIG. 6.13

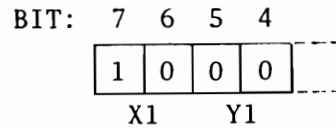


FIG. 6.14

a positive increment (10), and the Y co-ordinate is to decrease (01). The four least significant bits of the eight bit number can now be added. (Figure 6.15.) To get the decimal equivalent of this binary number, just add together the decimal values for all bits containing a 1. (Table 6.3.)

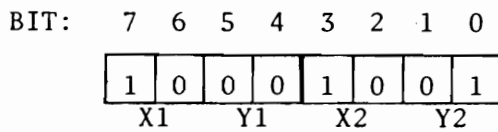


FIG. 6.15

BIT	DECIMAL VALUE
7	128
3	8
0	1
<hr/>	
Total	= 137

TABLE 6.3

So issuing a PLOT 137 will do the job. Suppose the original block is to be PLOTTed at X=63,Y=63. Then the following instruction will do the whole thing:

PLOT 2,63,63,251,137,255

Note also that you can string any number of incremental instructions together. After PLOT 251 is given, the computer will automatically take each subsequent PLOT number to be a plot increment, unless, of course, the number is greater than 239.

As an exercise, enter this instruction and see if you can determine why it does what it does.

PLOT 2,0,15,251:FOR J=1 TO 63:PLOT 169:NEXT:PLOT 255

Another exercise: see if you can determine what this program will do. Then type it in and RUN it to see if you were correct.


```

5 REM PROGRAM 6.7
6 REM MYSTERY PROGRAM USING INCREMENTAL POINT PLOT
10 PLOT 6,6,12
20 PLOT 2,0,127,251
30 INC=154:GOSUB 100
40 INC=89:GOSUB 100
50 INC=101:GOSUB 100
60 INC=166:GOSUB 100
70 PLOT 255
80 END
99
100 FOR J=1 TO 63:PLOT INC:NEXT:RETURN

```

You can spruce up that program by adding these lines

```

15 FOR K=1 TO 7:PLOT 6,K
.
.
.
75 NEXT K

```

Figure 6.16 gives the incremental point plot numbers for incrementing a block twice in a given direction.

If the four most significant bits of the increment number are zero, the X and Y co-ordinates of the first block will not change. This allows you to increment a point only once instead of twice, according to Figure

Incremental values
for incrementing
twice in the same
direction.

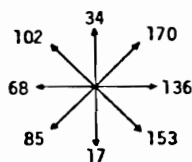


FIG. 6.16

Incremental values
for incrementing
only X2,Y2
(X1=0, Y1=0).

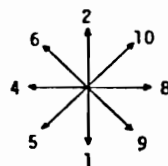


FIG. 6.17

Incremental values
for incrementing
without plotting
(X2=0, Y2=0).

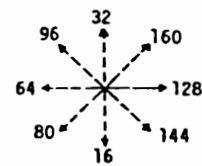


FIG. 6.18

6.17.

If the four least significant bits are zero, then the number for the first block will determine the increment, but no block will actually be PLOTted on the screen. This allows you to skip a block in any of eight directions according to Figure 6.18.

The following instructions will plot a dotted line going up and to the right from the center of the screen.

```

PLOT 2,63,63,251:FOR J=1 TO 10:PLOT 160,10:NEXT:PLOT 255

```


6.5 SCRIPT

In addition to regular height characters, double height characters and extra large characters, you can produce script. This will take more memory and time to create, but it can have an interesting effect. Let's write "Color Graphics" in script and add it to Program 4.6.

To produce script we could simply PLOT each point in the point plot submode, but that would require specifying the co-ordinates of each point. Let's try the incremental point plot submode so that only one number (the increment value) need be used for each of two new points. The result will be a substantial savings in memory over the point plot method. Since Program 4.6 printed the bottom of its message on line 15, we should, for the sake of good looks, also print the script message there. Figure 6.19 shows how the first part of the script message looks. The PLOtting begins at the upper right tip of the capital "C", and the arrows show the way the rest is PLOtted using incremental plots. These lines should be added to Program 4.6. (Save the result. We'll add even more goodies later on.)

```
184 REM PROGRAM 6.8
185 GOSUB 1000:REM SCRIPT
997
998 REM SCRIPT SUBROUTINE USING INCREMENTAL POINT PLOT
999
1000 PLOT 29:REM FLAG MUST BE OFF
1010 PLOT 12:REM CLEAR SCREEN
1020 PLOT 3,64,0:REM HIDE CURSOR
1030 RESTORE 1100
1040 PLOT2:REM GENERAL PLOT MODE
1050 PLOT 10,78:REM FIRST POINT ON THE CAPITAL "C"
1060 PLOT 251:REM INCREMENTAL POINT PLOT SUBMODE
1070 READ A:PLOT A:IF A<255 THEN 1070
1080 GOSUB 700:REM PAUSE SUBROUTINE FROM PROGRAM 4.6
1090 RETURN
1096
1097 REM DATA FOR INCREMENTAL POINT PLOT
1098
1099 REM C
1100 DATA 38,100,68,85,81,17,17,17,17,25,25,24,136,138,42
1104 REM O
1105 DATA 130,34,34,162,130,20,21,17,17,25,25,136,162,162,34,34,98
,100,81,153,136
1109 REM L
1110 DATA 170,40,40,34,34,101,17,17,17,17,17,17,17,154,170
1114 REM O
1115 DATA 162,34,42,40,65,81,17,17,145,152,138,42,34,34,38,38,69,2
5,152,138
1119 REM R
1120 DATA 138,170,33,24,152,17,81,17,17,152,138,42
1124 REM SPACE
1125 DATA 160,160,160,160,160,160,160,160,160,160,160
1129 REM G
1130 DATA 130,102,68,69,85,17,17,17,17,17,145,145,136,136,162,162,
34,34,36,69,25,136,138
```



```

1134 REM R
1135 DATA 170,170,33,24,152,17,81,17,17,152
1139 REM A
1140 DATA 138,42,130,34,34,170,168,152,70,69,85,17,17,25,25,136,13
8,162
1145 DATA 34,34,34,34,17,17,17,17,17,152
1149 REM P
1150 DATA 170,42,34,34,34,33,17,17,17,17,17,17,17,17,17,34,34,34,3
4,34,34,34,34
1155 DATA 170,168,153,25,17,21,21,84,70,152,136,136,138
1159 REM H
1160 DATA 162,162,34,34,34,34,34,34,17,17,17,17,17,17,17,17,34,34,
34
1165 DATA 42,170,136,145,17,17,17,17,152,162
1169 REM I
1170 DATA 162,42,34,34,17,17,17,17,25,162
1174 REM C
1175 DATA 162,162,34,162,168,137,150,100,69,21,17,17,17,153,136,13
6,170
1179 REM S
1180 DATA 42,42,42,42,42,165,25,25,17,17,81,85,102,153,136,138,170
1184 REM SKIP BACK TO DOT THE I
1185 DATA 253,102,77
1189 REM PLOT MODE ESCAPE
1190 DATA 255

```

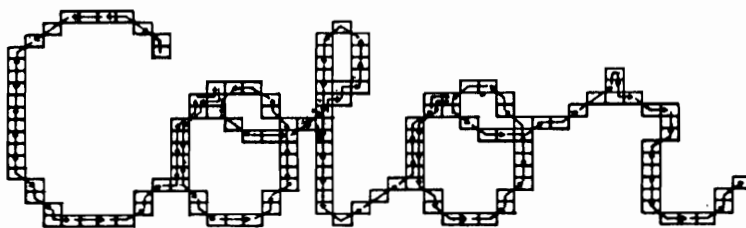


Fig. 6.19

6.6 PLOT 250 — X BAR GRAPH

A horizontal line may be drawn from a given point, X_0, Y_0 , to another point, X_1, Y_0 , by using the PLOT 250 instruction (after entering the general plot mode). After this instruction, your computer is expecting three numbers (less than 240) in this order: X_0, Y_0, X_1 . For example,

```
PLOT 2,250,0,63,127,255
```

will draw a horizontal line across the screen, halfway up. Only after X_1 is given will the line be drawn. Moreover, the computer will then automatically take the next two numbers as the Y_0, X_1 of another line (unless, of course, either of those numbers is greater than 239). That is, the same value for X_0 will be assumed. If you wish to give it a different X_0 , you will have to respecify the X bar graph submode with PLOT 250.

This program will draw a large, solid triangle near the center of the screen:


```

5 REM PROGRAM 6.9
6 REM A TRIANGLE USING X BAR GRAPH
10 PLOT 6,6,12:REM SET UP
20 PLOT 2:REM GENERAL PLOT MODE
30 PLOT 250:REM X BAR GRAPH SUBMODE
40 PLOT 30:REM X0
50 FOR Y=30 TO 97
60 PLOT Y,127-Y:REM Y0,X1 OF BAR GRAPH
70 NEXT
80 PLOT 255

```

6.7 PLOT 247 -- INCREMENTAL X BAR GRAPH

After plotting an X bar graph, the instruction PLOT 247 sets the computer up to take all subsequent numbers (less than 240) as incremental values. The incremental values will do for the bar graph what they did for the point plot, and they refer to the X1,Y0 co-ordinate of the bar graph. Thus,

```
PLOT 2,250,0,63,127
```

will draw a horizontal line across the screen, and if it is followed by

```
PLOT 247,85,255
```

two more lines will be drawn, the first just under the original but with its end point shifted to the left one block, and the second just under the first with its end point shifted one block more. As with the incremental point plot submode, an increment number whose four least significant bits are zero will increment without actually drawing the line.

This program will draw the same triangle as the previous program, but this time the incremental X bar graph submode is employed.

```

5 REM PROGRAM 6.10
6 REM TRIANGLE USING INCREMENTAL X BAR GRAPH
10 PLOT 6,6,12:REM SET UP
20 PLOT 2:REM GENERAL PLOT MODE
30 PLOT 250:REM X BAR GRAPH SUBMODE
40 PLOT 30,30,97:REM BOTTOM LINE OF TRIANGLE
50 PLOT 247:REM INCREMENTAL X BAR GRAPH SUBMODE
60 FOR J=1 TO 33
70 PLOT 102:REM INCREMENT
80 NEXT
90 PLOT 255

```

See if you can determine what this program will do. Then type it in and RUN it to see if you were correct.

```

5 REM PROGRAM 6.11
6 REM MYSTERY PROGRAM
10 PLOT 6,6,12,3,64,0
20 PLOT 2

```

↑

GEN VECTOR GRAPH ROUTINE

START FROM CCI

LOAD SCREEN

CURSOR

MINUS CURSOR

IF GREEN

ALL TOP

-46-


```

184 REM PROGRAM 6.8
185 GOSUB 1000: REM SCRIPT

998 REM
999 REM SCRIPT SUBROUTINE USING INCREMENTAL POINT PLOT
1000 PLOT 29: REM FLAG MUST BE OFF
1010 PLOT 12: REM CLEAR SCREEN
1020 PLOT 3,64,0: REM HIDE THE CURSOR
1030 RESTORE 1100
1040 PLOT 2: REM GENERAL PLOT MODE
1050 PLOT 24,72: REM FIRST POINT OF CAPITAL "C"
1060 PLOT 251: REM INCREMENTAL POINT PLOT SUBMODE
1070 READ A: PLOT A: IF A< 255 THEN 1060
1080 GOSUB 700: REM PAUSE
1090 RETURN
1097 REM
1098 REM DATA FOR INCREMENTAL POINT PLOT
1099 REM C
1100 DATA 100,68,69,65,81,17,25,152,136,138,170
1104 REM 0
1105 DATA 170,136,68,81,17,152,136,162,34,100,17,152,138
1109 REM M
1110 DATA 170,129,17,17,34,34,168,145,17,18,34,42,137,17,17,136
1114 REM P
1115 DATA 42,42,33,17,17,17,17,34,34,34,42,168,137,17,85,68,136,
    136,138
1119 REM U
1120 DATA 162,34,17,17,152,138,34,34,17,17,24,138
1124 REM C
1125 DATA 170,42,136,137,100,68,81,17,152,136,138
1129 REM 0
1130 DATA 162,170,136,68,85,17,152,136,162,34,101,25,136
1134 REM L
1135 DATA 138,162,34,42,129,21,21,17,25,138
1139 REM 0
1140 DATA 170,42,136,68,81,17,152,138,162,38,81,152,136
1144 REM R
1145 DATA 162,162,25,129,81,152,10
1149 REM SPACE (INCREMENT WITHOUT PLOTTING)
1150 DATA 160,160,160,32,32,32
1154 REM I
1155 DATA 168,136,136,68,81,17,17,21,72,136,136
1159 REM SPACE (INCREMENT WITHOUT PLOTTING)
1160 DATA 160,160,32,32,32,32,32
1164 REM I
1165 DATA 168,136,136,68,81,17,17,21,72,136,136
1169 REM PLOT MODE ESCAPE
1170 DATA 255

```

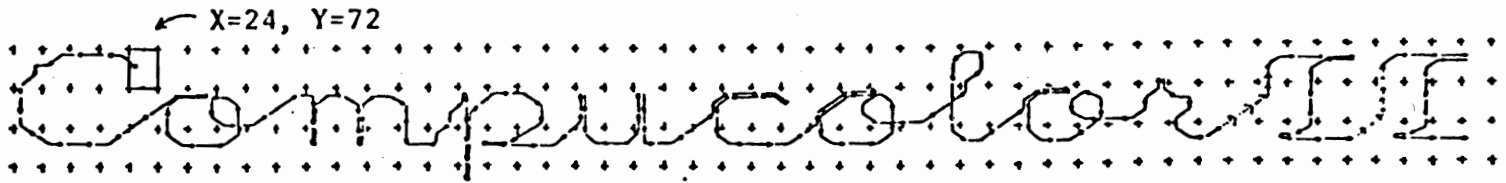



FIG. 6.19

6.6 PLOT 250--X BAR GRAPH

A horizontal line may be drawn from a given point, X_0, Y_0 , to another point, X_1, Y_0 , by using the PLOT 250 instruction (after entering the general plot mode). After this instruction, the Compucolor II is expecting three numbers (less than 240) in this order: X_0, Y_0, X_1 . For example, PLOT 2,250,0,63,127,255 will draw a horizontal line across the screen, halfway up.

X_0, Y_0 ————— X_1, Y_0

FIG. 6.20

Only after the X_1 number is given will the line be drawn. Moreover, the computer will then automatically take the next two numbers as the Y_0, X_1 of another line (unless, of course, either of those numbers is greater than 239). That is, the same value for X_0 will be assumed. If you wish to give it a different X_0 , you will have to respecify the X bargraph submenu with PLOT 250.

~~This program will draw a large, solid triangle near the center of the screen:~~

```
5 REM PROGRAM 6.9
6 REM A TRIANGLE USING X BAR GRAPH
10 PLOT 6,6,12: REM SET COLOR & CLEAR SCREEN
20 PLOT 2: REM GENERAL PLOT MODE
30 PLOT 250: REM X BAR GRAPH SUBMODE
40 PLOT 30: REM X0
50 FOR Y=30 TO 97
60 PLOT Y,127-Y: REM Y0,X1 OF BAR GRAPH
70 NEXT
80 PLOT 255
```

6.7 PLOT 247--INCREMENTAL X BAR GRAPH

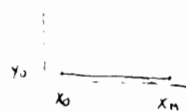
After plotting an X bar graph, the instruction PLOT 247 sets the computer up to take all subsequent numbers (less than 240) as incremental values. The incremental values will do for the bar graph what they did for the point plot, and they refer to the X_1, Y_0 co-ordinate of the bar graph.


```

30 PLOT 250
40 PLOT 30,0,125
50 PLOT 247
60 FOR Y=0 TO 63
70   PLOT 34
80 NEXT
90 PLOT 2
100 PLOT 255
110 PLOT 6,4
120 PLOT 2
130 S=0
139
140 FOR Y=0 TO 112 STEP 16
149
150   FOR X=30 TO 114 STEP 24
160     PLOT 250
170     PLOT X+12*S,Y,X+12*S+11
180     PLOT 247
190     FOR J=1 TO 7:PLOT 34:NEXT
200     PLOT 2
210   NEXT
219
220   S=1+(S=1):REM TOGGLE
230 NEXT
239
240 PLOT 255

```

BAR GRAPH MODE



CAN 100 PLOT BLOCK LINES

0010 0010
X Y X Y
C A N C A N

GEN VECTOR DATA PLOT MODE

EXIT

CCL STATUS BLUE FUNCTIONS

GEN VECTOR PLOT MODE

STEP?

XY POINT PLOT WITH GMP



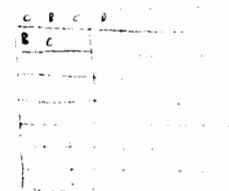
CAN BACKGROUND

RWE SQUARE

X BAR GRAPH

INCREMENTAL BAR GRAPH

0010 0100
X Y X Y
1001 0100
0 + - 0



ALT CUL SP - CAN RWE

Recall that if the FLAG bit is set prior to entering the general plot mode, PLOTTing a point twice will turn that point on, then off. Will that happen with a bar graph? Try this experiment:

```

5 REM PROGRAM 6.12
6 REM X BAR GRAPH WITH FLAG ON
10 PLOT 6,6,12:REM SET UP
20 PLOT 30:REM FLAG ON
30 PLOT 2:REM GENERAL PLOT MODE
40 PLOT 250:REM X BAR GRAPH SUBMODE
50 PLOT 0,63,127:REM DRAW THE BAR
60 PLOT 63,127:REM AND AGAIN
61 REM NOTE THAT THE INITIAL VALUE GIVEN TO X0 IS ASSUMED
70 PLOT 255:REM EXIT PLOT MODE
80 PLOT 29:REM FOR CONVENIENCE, TURN FLAG OFF

```

One way to erase the screen is to use PLOT 12. You could also erase a line at a time:

```
PLOT 8:FOR J=1 TO 32:PLOT 11,10:NEXT
```

Or you could plot a horizontal (or vertical, see next section) bar graph in black (or whatever color the screen is to be erased in) and move it down (or up, or across) the screen. This can sometimes give a bit of a "cinemagraphic" touch to your programs.


```

5 REM PROGRAM 6.13
6 REM ERASE SCREEN USING X BAR GRAPH
10 PLOT 6,2:REM SET COLOR
17
18 REM NOW, FOR THE SAKE OF THE DEMONSTRATION,
19 REM USE THE TEST MODE TO FILL THE SCREEN WITH SOMETHING
20 PLOT 27,25:REM TEST MODE
30 PLOT 65:REM FILL THE SCREEN WITH "A"
39
40 PLOT 3,64,0:REM HIDE THE CURSOR
50 PLOT 6,0:REM COLOR=BLACK
59
60 PLOT 2:REM GENERAL PLOT MODE
70 PLOT 250:REM X BAR GRAPH SUBMODE
80 PLOT 0:REM X0=0
88
89 REM ERASE THE SCREEN
90 FOR Y=0 TO 125 STEP 4
100 PLOT Y,127:REM Y0 AND X1
110 NEXT
119
120 PLOT 255
130 PLOT 6,2:REM RETURN TO SOME COLOR

```

Note that the FOR-NEXT loop in line 90 has a step value of 4. This will guarantee that one bar graph line will go across each character line on the screen; if a PLOTTed block or line goes into a character position, any ASCII character there will disappear. So we need not PLOT the bar graph for all four blocks within the height of a character position. On the other hand, you could do so if you wished the erasing to go slightly slower. Change line 90 to

```
90 FOR Y=0 TO 127
```

You could also use the incremental bar graph mode. Establish the first bar by changing line 80 to

```
80 PLOT 0,0,127
```

Enter the incremental bar graph submode with

```
85 PLOT 247
```

Line 100 will then contain the increment number:

```
100 PLOT 34
```

Remember that the bar graph increments twice each time through the loop, so if you leave line 90 as a loop from Y=0 to 127, the lines will pass the top edge of the screen, "wrap around" to the bottom, and erase the screen a second time. A loop from Y=0 to 61 will just be sufficient.

Still another variation is to use the regular bar graph, but have the FLAG bit set so that you can PLOT a line and then go back and rePLOT (=erase) the previous one. This way, you can use a bar graph in some color other than black, and the result will be a colored line moving up the screen, erasing as it goes.

```

5 REM PROGRAM 6.14
6 REM ERASE SCREEN WITH X BAR GRAPH & FLAG ON
10 PLOT 30:REM FLAG ON
20 PLOT 6,2:REM CHOOSE COLOR
30 PLOT 27,25:REM TEST MODE
40 PLOT 65:REM FILL THE SCREEN WITH SOMETHING
50 PLOT 3,64,0:REM HIDE THE CURSOR
60 PLOT 6,6:REM CHOOSE A COLOR FOR THE ERASING BAR
70 PLOT 2:REM GENERAL PLOT MODE
80 PLOT 250:REM X BAR GRAPH
90 PLOT 0,0,127:REM FIRST BAR
99
100 FOR Y=1 TO 127
110 PLOT Y-1,127:REM ERASE LAST BAR
120 PLOT Y,127:REM PLOT NEW BAR
130 NEXT
139
140 PLOT Y-1,127:REM ERASE VERY LAST BAR
150 PLOT 255:REM PLOT MODE ESCAPE
160 PLOT 29:REM FLAG OFF, FOR CONVENIENCE

```

Or, if you wish to go faster, use STEP 4 in the loop. But in that case you will have to make some slight changes elsewhere:

```

100 FOR Y=4 TO 125 STEP 4
110 PLOT Y-4,127:REM ERASE PREVIOUS BAR
.
.
.
140 PLOT Y-4,127:REM ERASE VERY LAST BAR

```

You could also use the incremental X bar graph by choosing increment values so that you first PLOT a bar, increment it to PLOT a new one, decrement it to rePLOT (=erase) the previous one, increment without PLOTting, and so on.

If there are plot blocks already on the screen, then using the moving bar graph with the FLAG on to erase the screen will erase those blocks and then immediately rePLOT them in the color of the moving bar. The following lines provide a variation on the incremental method. Add them to Program 6.8 (which has been added to Program 4.6):


```

1085 GOSUB 2000:REM MOVING X BAR WITH FLAG ON
.
.
.
1996
1997 REM ***** SUBROUTINE TO PLOT MOVING X BAR GRAPH *****
1998
1999 REM FIRST DRAW A BLUE BORDER AROUND THE SCRIPT
2000 PLOT 6,4:REM BLUE
2010 PLOT 2:REM GENERAL PLOT MODE
2020 PLOT 0,89:REM X,Y OF TOP LEFT CORNER OF BORDER
2030 PLOT 242:REM VECTOR PLOT SUBMODE (SEE SECTION 6.9)
2040 PLOT 0,50,127,50,127,89,0,89:REM PLOT REST OF BORDER
2050 PLOT 255:REM PLOT MODE ESCAPE
2059
2060 PLOT 30:REM FLAG ON
2070 GOSUB 700:REM PAUSE ROUTINE IN PROGRAM 4.6
2080 C=FG-16+(BG-16)*8:REM PUT COLOR INTO CCI FORMAT
2090 PLOT 6,C
2099
2100 PLOT 2:REM BACK TO GENERAL PLOT MODE
2110 PLOT 250:REM X BAR GRAPH SUBMODE
2120 PLOT 2,52,125:REM X0,Y0,X1 OF FIRST BAR BAR
2130 PLOT 247:REM INCREMENTAL X BAR GRAPH SUBMODE
2138
2139 REM NOW MOVE BAR UP THROUGH SCRIPT
2140 FOR Y=52 TO 86
2150   PLOT 2:REM INCREMENT ONCE & PLOT
2160   PLOT 1:REM DECREMENT AND ERASE PREVIOUS BAR
2170   PLOT 32:REM INCREMENT WITHOUT DRAWING THE BAR
2180 NEXT
2188
2189 REM MOVE BAR BACK DOWN THROUGH MESSAGE
2190 FOR Y=86 TO 52 STEP -1
2200   PLOT 1:REM MOVE ONCE & DRAW NEW BAR
2210   PLOT 2:REM ERASE PREVIOUS BAR
2220   PLOT 16:REM MOVE WITHOUT DRAWING
2230 NEXT
2239
2240 PLOT 255:REM EXIT PLOT MODE
2250 C=C+1:IF C=64 THEN RETURN:REM ALL DONE
2259
2260 PLOT 6,C:REM NEW COLOR
2270 PLOT 2:REM GENERAL PLOT MODE AGAIN
2280 PLOT 247:REM INCREMENTAL BAR GRAPH WILL PICK UP
2281   REM FROM WHERE IT LEFT OFF. (SEE SECTION 6.12)
2290 GOTO 2140:REM REPEAT IN THE NEW COLOR

```


**6.8 PLOT 246 -- Y BAR GRAPH and
PLOT 243 -- INCREMENTAL Y BAR GRAPH**

Everything which applied to the X bar graph now applies to the Y bar graph plot submode. Just make the necessary switches from horizontal lines to vertical lines.

The following program will plot the area under a sine function.

```
5 REM PROGRAM 6.15
6 REM SINE FUNCTION IN Y BAR GRAPH MODE
10 PLOT 6,6,12:REM SET UP
20 PLOT 2:REM GENERAL PLOT MODE
30 PLOT 246:REM Y BAR GRAPH SUBMODE
40 PLOT 0:REM Y0 OF Y BAR GRAPH
49
50 FOR X=0 TO 127
60 PLOT X:REM X OF Y BAR GRAPH
70 PLOT 63+40*SIN(X/10):REM SCALE THE Y1
80 NEXT
89
90 PLOT 255:REM EXIT PLOT MODE
```

The following program uses the incremental Y bar graph mode to produce what, with some imagination, might be... um... "modern art" impressions of industrial plants.

```
5 REM PROGRAM 6.16
6 REM INCREMENTAL Y BAR GRAPH DEMO
10 DEF FN R(R)=INT(R*RND(1))+1
20 Y0=25:REM Y0 IS KEPT CONSTANT, JUST FOR LOOKS
30 PLOT 6,FN R(63),12:REM ERASE SCREEN IN A RANDOM COLOR
40 PLOT 3,64,0:REM HIDE CURSOR
49
50 FOR J=1 TO FN R(20)+5:REM RANDOM NUMBER OF FIGURES
60 PLOT 6,FN R(63):REM CHOOSE RANDOM COLOR FOR EACH
68
69 REM RANDOMLY SET THE FLAG ON OR OFF
70 PLOT 29:IF RND(1)>.5 THEN PLOT 30
79
80 PLOT 2:REM GENERAL PLOT MODE
90 PLOT 246:REM Y BAR GRAPH SUBMODE
98
99 REM CHOOSE RANDOM X AND Y1, BUT NOT TOO CLOSE TO BORDERS
100 X=FN R(80)+20
110 Y1=FN R(80)+20
119
120 PLOT Y0,X,Y1:REM DRAW A VERTICAL BAR
130 PLOT 243:REM INCREMENTAL Y BAR GRAPH SUBMODE
140 INC=FN R(239):REM RANDOM INCREMENTAL VALUE
150 T=FN R(20):REM RANDOM NUMBER OF THOSE INCREMENTS
159
160 FOR K=1 TO T:PLOT INC:NEXT
169
170 PLOT 255:REM PLOT MODE ESCAPE
```



```

180 NEXT
189
190 FOR J=1 TO 2500:NEXT:REM PAUSE
199
200 GOTO 30

```

6.9 PLOT 242 — VECTOR PLOT

The vector plot submode is more versatile than either X or Y bar graph submodes because a line (vector) may be drawn from any point to any other point on the screen. The computer will plot the closest approximation to a straight line between the points. The first point of the line is specified by PLOTTing a point in the point plot mode. Then, after receiving a PLOT 242 instruction, the computer will take the next two numbers to be the X,Y co-ordinates of the other end of the vector and draw the line. It is then automatically set up to receive two more numbers as the end point of another line to be drawn from the end of the first, and so on. Thus,

```
PLOT 2,0,0,242,127,0,127,127,0,127,0,0,255
```

will draw a line around the screen.

A combination of point plots and vector plots in several colors will allow us to create an animated explosion effect for the artillery game developed earlier. The explosion will shoot out colored vectors from the site of the explosion, but to avoid having any of these vectors try to go off the screen, the explosion site for this particular explosion routine must be no closer than six plot blocks (three character positions) from either edge of the screen. If we use the explosion only when the target has been hit, then we need be sure only that the target is sufficiently in from the side of the screen. (Notice that the artillery game program already does that.)

As always, to change colors you must exit the plot mode, change the color and then reenter the plot mode. The explosion will not look very good if the flag is on, so we must make sure it is off. X and Y are the PLOT co-ordinates of the explosion site. (If you wish to incorporate this routine into the artillery game, use only the lines 5000-5201 in the following program, and in the artillery program add 85 X=2*TX:Y=4:GOSUB 5000.)

```

5 REM PROGRAM 6.17
6 REM EXPLOSION USING VECTOR PLOT SUBMODE
10 PLOT 15,6,6,12:REM SET UP
20 PLOT 27,24:REM PAGE MODE
29
30 PLOT 6,16:REM BACKGROUND = GREEN
40 PLOT 3,0,31,11:REM ERASE BOTTOM LINE IN GREEN
50 Y=4:REM EXPLOSION IS ALWAYS AT GROUND LEVEL
99
100 PLOT 6,6
110 PLOT 8,11:REM ERASE ANY PREVIOUS INPUT
120 INPUT "X CO-ORDINATE (6-120)? [red] ";X$
130 X=VAL(X$)

```



```

140 IF X<6 OR X>120 THEN 100
149
150 GOSUB 5000
160 GOTO 100
4997
4998 REM ***** SUBROUTINE TO PLOT EXPLOSION *****
4999
5000 PLOT 29:REM FLAG OFF
5010 PLOT 19:REM YELLOW
5020 PLOT 2:REM GENERAL PLOT MODE
5029
5030 PLOT X,Y,242:REM FIRST POINT, THEN ENTER VECTOR PLOT
5038
5039 REM SOME VECTORS
5040 PLOT X-2,Y+21,253,X,Y,242,X+3,Y+7,253,X,Y,242,X-3,Y+15
5049
5050 FOR J=1 TO 150:NEXT:REM SLIGHT PAUSE
5058
5059 NOW FOR A FEW POINTS
5060 PLOT 253,X,Y+13,X-2,Y+7,X+2,Y+20
5068
5069 REM ANOTHER VECTOR
5070 PLOT X,Y,242,X+5,Y+20
5079
5080 PLOT 255,16:REM CHANGE COLOR TO BLACK
5088
5089 REM ERASE ONE OF THE VECTORS
5090 PLOT 2,X,Y,242,X-3,Y+15
5099
5100 PLOT 255,23:REM NEW COLOR = WHITE
5108
5109 REM SOME MORE POINTS
5110 PLOT 2,X-1,Y+20,X+3,Y+25
5118
5119 REM SOME MORE VECTORS
5120 PLOT X,Y,242,X+2,Y+11,253,X,Y,242,X-1,Y+30
5130 PLOT 253,X,Y,242,X+1,Y+8
5139
5140 PLOT 255,16:REM BLACK AGAIN TO ERASE EVERYTHING
5149
5150 PLOT 2,X,Y,242,X+2,Y+11,253,X,Y,242,X+5,Y+20
5160 PLOT 253,X,Y,242,X+1,Y+8,253,X,Y,242,X-1,Y+30
5169
5170 FOR J=1 TO 50:NEXT:REM SLIGHT PAUSE
5179
5180 PLOT 253,X,Y+13,X-2,Y+7,X+2,Y+20,X,Y,242,X-2,Y+21
5190 PLOT 253,X,Y,242,X+3,Y+7,253,X-1,Y+20,X+3,Y+25,255
5199
5200 RETURN:REM NOTE THAT THIS SUBROUTINE RETURNS WITH
5201 REM COLOR=BLACK AND THE FLAG OFF

```

Try specifying several explosions right next to each other. What happens? The explosions become more "chunky". The reason is not hard to discover. The routine PLOTs points and vectors in yellow and white, then erases them

by rePLOTting them in black. The result is that all those points and vectors are still there, but now they are perfectly camouflaged, since the background is also black. Now, when another explosion near the same site PLOTs its vectors and points, there will be some character positions with plot blocks in them (in black) as a result of the previous explosion. Recall that colors are defined for an entire character position. Thus, when the new explosion PLOTs a block in white or yellow, any other plot blocks in that same character position will also turn white or yellow.

The problem is avoided by clearing the entire screen before each explosion. Clearing the screen makes each character position a space (ASCII 32), thus wiping out any plot blocks. If you change line 160 to

```
160 GOTO 10
```

then the explosions will always be cleanly PLOTted.

To the list of interesting ways to erase the screen we can add the following. Draw a black vector which travels around the screen in smaller and smaller rectangles, erasing as it goes.

```
5 REM PROGRAM 6.18
6 REM ERASE SCREEN USING VECTOR PLOT -- I
10 PLOT 6,2:REM SET COLOR
20 PLOT 27,25,65:REM FILL THE SCREEN USING TEST MODE
30 PLOT 3,64,0,6,0:REM HIDE CURSOR & SET COLOR FOR VECTOR
39
40 PLOT 2:REM GENERAL PLOT MODE
50 PLOT 0,0:REM FIRST POINT
60 PLOT 242:REM VECTOR PLOT SUBMODE
70 Y=-4
79
80 FOR X=0 TO 30 STEP 2
90 Y=Y+4
91 REM THUS, 1ST TIME THRU THE LOOP, X=0, Y=0.
92 REM X GOES BY 2'S AND Y BY 4'S BECAUSE A VECTOR
93 REM WHICH GOES THRU A CHARACTER POSITION WILL
94 REM ERASE ANY ASCII CHARACTER THERE
100 PLOT X,Y:REM BOTTOM LEFT (NEW)
110 PLOT 127-X,Y:REM BOTTOM RIGHT
120 PLOT 127-X,127-Y:REM TOP RIGHT
130 PLOT X,127-Y:REM TOP LEFT
140 PLOT X,Y:REM BOTTOM LEFT (OLD)
150 NEXT
159
160 PLOT 255:REM PLOT MODE EXIT
170 PLOT 6,2:REM BACK TO SOME COLOR
```

The program can erase in slightly different proportions by changing the amount by which Y is incremented in the loop. But you'll have to make appropriate changes in lines 70 and 80. Try this:

```
70 Y=-2
80 FOR X=0 TO 60 STEP 2
90 Y=Y+2
```


Still another possibility presents itself. PLOT a series of black vectors from the center of the screen out to each point along the edge.

```
5 REM PROGRAM 6.19
6 REM ERASE SCREEN USING VECTOR PLOT -- II
10 PLOT 29:REM FLAG OFF
20 PLOT 6,2:REM SET COLOR TO GREEN
30 PLOT 27,25,65:REM FILL SCREEN USING TEST MODE
40 PLOT 3,64,0,6,0:REM HIDE CURSOR AND SET COLOR=BLACK
49
50 PLOT 2:REM GENERAL PLOT MODE
58
59 REM UPPER TRIANGLE OF SCREEN
60 Y=127
69
70 FOR X=0 TO 127:GOSUB 200:NEXT
78
79 REM RIGHT TRIANGLE OF SCREEN
80 X=127
89
90 FOR Y=127 TO 0 STEP -1:GOSUB 200:NEXT
98
99 REM LOWER TRIANGLE OF SCREEN
100 Y=0
109
110 FOR X=127 TO 0 STEP -1:GOSUB 200:NEXT
118
119 REM LEFT TRIANGLE OF SCREEN
120 X=0
129
130 FOR Y=0 TO 127:GOSUB 200:NEXT
139
140 PLOT 255:REM EXIT PLOT MODE
150 PLOT 6,2:REM BACK TO SOME COLOR
160 END
197
198 REM ***** SUBROUTINE TO DRAW A VECTOR *****
199
200 PLOT 253,63,63:REM FIRST POINT OF VECTOR IS CENTER OF SCREEN
210 PLOT 242:REM VECTOR PLOT SUBMODE
220 PLOT X,Y:REM END POINT OF VECTOR AT EDGE OF SCREEN
230 RETURN
```

Experimenting with the computer's graphics can often lead to some unexpected and sometimes beautiful results. Suppose we alter the previous program in a few simple ways. First, set the FLAG on instead of off. Erase the display using a black vector as in the above program. Then erase the screen again using a blue vector. Finally, erase the screen a third time using a red vector. I think you'll find the results to be very interesting. These are the changes you might make:


```

.
.
.
10 PLOT 30:REM FLAG ON
.
.
.
40 PLOT 3,64,0,6,0:GOSUB 50:REM HIDE CURSOR; BLACK VECTOR
42 PLOT 6,4:GOSUB 50:REM BLUE VECTOR
44 PLOT 6,1:GOSUB 50:REM RED VECTOR
46 PLOT 29:END
.
.
.
150 RETURN

```

The final pattern which is produced could be achieved immediately by erasing the screen the first time using some color other than black for the vector. (With the FLAG on, the two patterns toggle.) But the final red pattern emerging over the blue presents an especially interesting effect. We have a tendency to see blue objects as though they were further in the distance when compared to red objects (in this case, the emerging pattern). You might also try different X1,Y1 values for the vectors in line 200. (If you set X1=0 and Y1=0 then lines 100-130 will not be necessary.) Finally, try using different background colors for one or more of the vector sweeps. (Blue might be a good choice, because red on blue has that special effect.)

6.10 PLOT 240 -- INCREMENTAL VECTOR PLOT

Like points and bar graphs, vectors can be PLOTted and then incremented. The incremental codes are nearly identical to the codes for incrementing points and bar graphs. The difference is that if the four most significant bits of the increment number are zero, then the X1,Y1 co-ordinate of the vector will be incremented, but the vector will not be drawn, whereas for the incremental point plot and the incremental bar graph modes, the point or bar will be incremented but not drawn only if the four least significant bits are zero. (If the four most significant bits are zero, the first of the two increments of the point or bar will remain unchanged, giving the point or bar only one increment, whereas for vectors there is always only one increment at most.) Furthermore, if the four least significant bits of the vector increment number are zero, the vector will be neither incremented nor drawn. (This is contrary to the account given in ISC's "Compucolor II Programming and Reference Manual", Rev. 2, p. 80.)

A vector, unlike a bar graph, may have different increments for its two end points. This allows a vector to be rotated about an axis and to be propelled this way or that even while being rotated. Suppose we wish to rotate a vector counterclockwise about its midpoint. (Figure 6.20.) X1 should be increased while Y1 is decreased, but vice-versa for X2 and Y2. Recall the numbers necessary to make these changes. (See Figure 6.21 and Table 6.4.) The number we wish is therefore $128+16+4+2=150$. This instruction will draw a vector from X=40,Y=63 to X=80,Y=63 and increment it 20 times:

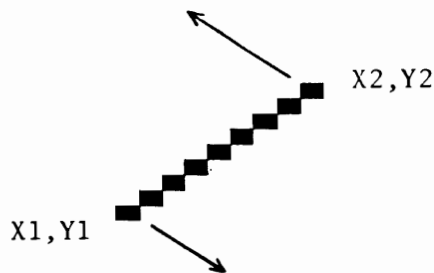


FIG. 6.20

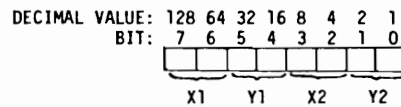


FIG. 6.21

If the binary value of the two bits is...	...then the increment of that co-ordinate will be...
00	None
01	-
10	+
11	None

TABLE 6.4

```
PLOT 2,40,63,242,80,63,240:FOR J=1 TO 20:PLOT 150:
NEXT:PLOT 255
```

The missing blocks in the resulting display are caused by the "best fit" line drawing routine from one end point to the other; an absolutely straight line cannot always be drawn, so the rotation of the vector, incrementing a plot block at a time, causes some plot blocks to be passed over.

6.11 FLYING WEDGES

If you PLOT a certain vector and then increment its X1 co-ordinate while decrementing the other three co-ordinates, after a number of such changes the result will be what I am calling a "flying wedge". Here's a simple one.

```
5 REM PROGRAM 6.20
6 REM A FLYING WEDGE
10 PLOT 6,6,12:REM SET UP
20 PLOT 2:REM GENERAL PLOT MODE
30 PLOT 63,63:REM X1,Y1 OF VECTOR
40 PLOT 242:REM VECTOR PLOT SUBMODE
50 PLOT 93,93:REM X2,Y2 OF VECTOR
60 PLOT 240:REM INCREMENTAL VECTOR PLOT
69
70 FOR J=1 TO 30
80 PLOT 149:REM INCREMENT X1, DECREMENT THE OTHERS
90 NEXT
99
100 PLOT 255
```

Now try adding:

```
15 PLOT 30:REM FLAG ON.
```

Flying wedges are to me interesting shapes and illustrate what can be accomplished with very simple instructions. For variations, change the number of times the program increments the vector (line 70 in the program above). The following program draws random flying wedges in random colors.


```

5 REM PROGRAM 6.21
6 REM RANDOM FLYING WEDGES
10 C=63*RND(1)+1:REM CHOOSE RANDOM COLOR
20 PLOT 6,C
30 PLOT 12:REM ERASE SCREEN
40 PLOT 3,64,0:REM HIDE CURSOR
49
50 N=5*RND(1)+2:REM 2 TO 6 FLYING WEDGES
59
60 FOR J=1 TO N
70 GOSUB 200:REM DRAW IT
80 NEXT
89
90 FOR J=1 TO 2500:NEXT:REM PAUSE
99
100 GOTO 10:REM BACK FOR SOME MORE
194
195
196 REM ***** SUBROUTINE TO DRAW A FLYING WEDGE *****
197
198 REM DON'T GET TOO CLOSE TO EDGES OF SCREEN, AND,
199 REM JUST FOR LOOKS, USE 50 AS MINIMUM Y
200 X1=75*RND(1):Y1=25*RND(1)+50
210 X2=X1+10*RND(1)+40:Y2=Y1+10*RND(1)+40
219
220 PLOT 29,8*RND(1)+16:REM CHOOSE RANDOM FOREGROUND COLOR
230 IF RND(1)>.5 THEN PLOT 30:REM FLAG RANDOMLY ON
239
240 PLOT 2,X1,Y1,242,X2,Y2:REM DRAW A VECTOR
250 PLOT 240:REM INCREMENTAL VECTOR PLOT SUBMODE
259
260 FOR K=1 TO 45
270 PLOT 149:REM INCREMENT
280 NEXT
289
290 PLOT 255:REM PLOT MODE ESCAPE
300 RETURN

```

6.12 REENTERING PLOT SUBMODES

Excluding the character plot submode, you may enter any plot submode directly from any other plot submode simply by issuing the proper PLOT number (253 for point plot, 242 for vector plot, and so on). For example, you may draw a vector and then immediately enter the incremental point plot submode. The point which will be incremented is the X2,Y2 of the last vector.

```

5 REM PROGRAM 6.22
6 REM REENTRY TO PLOT SUBMODES
10 PLOT 6,6,12:REM SET UP
20 PLOT 2:REM GENERAL PLOT MODE
28
29 REM DRAW A VECTOR FROM 0,0 TO 63,63

```



```

30 PLOT 0,0,242,63,63
39
40 PLOT 251:REM INCREMENTAL POINT PLOT SUBMODE
49
50 FOR J=1 TO 10
60 PLOT 96:REM INCREMENT WITHOUT PLOTTING
70 PLOT 6:REM INCREMENT ONLY ONCE
80 NEXT
89
90 PLOT 255

```

You may even exit the plot mode entirely in order to change colors or to print text, then reenter any plot submode and continue where you left off. Add to the above program these lines:

```

100 PLOT 6,1:REM RED
110 PLOT 2:REM GENERAL PLOT MODE AGAIN
120 PLOT 251:REM BACK TO INCREMENTAL POINT PLOT SUBMODE
129
130 FOR J=1 TO 10
140 PLOT 96,6:REM CONTINUE THE DOTTED LINE
150 NEXT
159
160 PLOT 255
169
170 PLOT 6,2:REM GREEN
180 PLOT 2:REM REENTER GENERAL PLOT MODE
190 PLOT 242:REM VECTOR PLOT MODE
200 PLOT 0,0:REM DRAW A VECTOR BACK TO STARTING POINT
209
210 PLOT 255

```

If a program does not specify the starting co-ordinates for a point or line, the starting point defaults to the last co-ordinates, which will be 0,0 if no other value has previously been used since the computer was turned on (or since you hit CPU RESET). Type in this:

```
PLOT 2,0,0,255.
```

Now type in this program:

```

5 REM PROGRAM 6.23
6 REM PLOT MODE DEFAULT CO-ORDINATES
10 PLOT 6,6,12:REM SET UP
20 PLOT 2:REM GENERAL PLOT MODE
30 PLOT 251:REM INCREMENTAL POINT PLOT SUBMODE
39
40 FOR J=1 TO 20
50 PLOT 10:REM INCREMENT UP AND TO THE RIGHT
60 NEXT
69
70 PLOT 255

```

Now RUN it. RUN it again. And again. And again. Now hit CPU RESET, ESC E (BASIC reset) and RUN it again. Interesting. Or try adding these lines:


```

15 PLOT 2,0,0,255
17 FOR K=1 TO 6
.
.
.
80 PLOT 12
90 NEXT

```

The following program uses these features for reentering the Y bar graph submode in order to draw bar graphs for the production history of a fictitious company, Widgeco.

```

5 REM PROGRAM 6.24
6 REM GRAPHING WIDGECO PRODUCTION
10 DIM EC(30),DE(30):REM 30 YEARS PRODUCTION OF ECONOMY
11 REM AND DELUXE MODEL WIDGETS
19
20 PLOT 14:REM DOUBLE HEIGHT
30 PLOT 30,16,29,19:REM SET COLORS; FLAG OFF
40 PLOT 12,27,24:REM ERASE SCREEN & SET TO PAGE MODE
50 PRINT TAB(33)"WIDGECO"
60 PLOT 15,18:REM REGULAR HEIGHT; COLOR=GREEN
70 PRINT TAB(27)"PRODUCTION HISTORY"
78
79 REM SET UP GRAPH BORDERS
80 GOSUB 3000
88
89 REM LABEL VERTICAL AXIS
90 GOSUB 2200
98
99 REM PRINT LEGEND
100 GOSUB 2000
108
109 REM GIVE INSTRUCTIONS
110 GOSUB 1700
118
119 REM GET DATA
120 GOSUB 4000
298
299 REM GET STARTING YEAR DESIRED
300 GOSUB 1600
308
309 REM CHECK FOR INPUT ERRORS
310 IF SY>1949 AND SY<1980 THEN 500
319
320 PLOT 28,11:REM ERASE BAD INPUT
330 PLOT 6,65:REM BLINK RED
340 PRINT TAB(17)"DATA NOT AVAILABLE"
350 PLOT 6,1,3,64,0:REM BLINK OFF; HIDE CURSOR
359
360 FOR T=1 TO 1000:NEXT:REM PAUSE
369
370 GOTO 300

```



```

498
499 REM ERASE GRAPH AREA & YEARS
500 GOSUB 1500
508
509 REM DISPLAY NEW DATA
510 GOSUB 1000
518
519 REM BACK FOR MORE
520 GOTO 300
997
998 REM ***** SUBROUTINE TO DISPLAY NEW DATA *****
999
1000 PLOT 2:REM GENERAL PLOT MODE
1010 PLOT 246,28:REM Y0 OF Y BAR GRAPH
1020 PLOT 34:REM X OF Y BAR GRAPH
1030 PLOT 255:REM EXIT PLOT MODE
1039
1040 PLOT 3,16,26:REM POSITION CURSOR
1049
1050 FOR X=0 TO 8:REM FOR EACH OF THE 9 YEARS DISPLAYED...
1060 IF SY+X>1979 THEN X=8:GOTO 1160
1069
1070 PLOT 23:REM WHITE
1080 PRINT SY+X,:REM PRINT THE YEAR
1090 PLOT 21:REM MAGENTA
1100 Y1=EC(SY+X-1949)*4+27:REM HEIGHT OF Y BAR FOR
1101 REM ECONOMY MODEL
1102
1110 GOSUB 1200:REM PLOT THE BAR
1120 PLOT 19:REM YELLOW FOR DELUXE MODEL
1130 Y1=DE(SY+X-1949)*4+27
1140 GOSUB 1200
1150 PLOT 2,243,128,128,255:REM SKIP OVER & EXIT PLOT MODE
1160 NEXT
1169
1170 PRINT
1180 RETURN
1197
1198 REM ***** SUBROUTINE TO PLOT A BAR *****
1199
1200 PLOT 2:REM GENERAL PLOT MODE
1210 PLOT 244,Y1:REM HEIGHT OF THE BAR
1220 PLOT 243:REM INCREMENTAL Y BAR GRAPH SUBMODE
1230 PLOT 136:REM INCREMENT TWICE TO THE RIGHT
1240 PLOT 128,128:REM SKIP WITHOUT PLOTTING
1250 PLOT 255:REM EXIT PLOT MODE
1260 RETURN
1497
1498 REM ***** SUBROUTINE TO ERASE GRAPH AREA & YEARS *****
1499
1500 PLOT 3,0,26,11:REM ERASE YEARS
1510 PLOT 3,64,0,16:REM HIDE CURSOR; COLOR=BLACK
1519
1520 PLOT 2:REM GENERAL PLOT MODE

```



```

1530 PLOT 246,28:REM Y0 OF Y BAR GRAPH
1540 PLOT 32:REM X OF Y BAR GRAPH
1550 PLOT 111:REM HEIGHT OF Y BAR GRAPH
1560 PLOT 243:REM INCREMENTAL Y BAR GRAPH SUBMODE
1569
1570 FOR X=1 TO 47:PLOT 136:NEXT
1579
1580 PLOT 255:REM EXIT PLOT MODE
1590 RETURN
1596
1597 REM ***** SUBROUTINE TO GET STARTING YEAR *****
1598
1599 REM FIRST ERASE ANY PREVIOUS INPUT
1600 PLOT 3,0,30,11
1609
1610 INPUT "[cyn]STARTING YEAR?[wht] ";SY$
1620 SY=VAL(SY$)
1630 RETURN
1697
1698 REM ***** SUBROUTINE TO GIVE INSTRUCTIONS *****
1699
1700 PLOT 18:REM GREEN
1710 Y=8:REM INITIAL CURSOR Y POSITION LESS 2
1720 GOSUB 1850:REM POSITION CURSOR FOR NEXT LINE OF TEXT
1730 PRINT "FIGURES FOR WIDGECO'S PRODUCTION ARE AVAILABLE"
1740 GOSUB 1850
1750 PRINT "FOR THE YEARS 1950-1979. WHEN REQUESTED, PLEASE"
1760 GOSUB 1850
1770 PRINT "INDICATE THE STARTING YEAR FOR WHICH GRAPHS OF"
1780 GOSUB 1850
1790 PRINT "THOSE FIGURES ARE DESIRED. THAT YEAR AND UP TO"
1800 GOSUB 1850
1810 PRINT "EIGHT ADDITIONAL YEARS' DATA WILL BE DISPLAYED."
1820 RETURN
1847
1848 REM **** SUBROUTINE TO SKIP A LINE & REPOSITION CURSOR ****
1849
1850 Y=Y+2:PLOT 3,16,Y:RETURN
1997
1998 REM ***** SUBROUTINE TO PRINT THE LEGEND *****
1999
2000 PLOT 3,21,28:REM POSITION CURSOR
2010 PLOT 18:REM GREEN
2020 PLOT 30,21:REM FLAG ON; BACKGROUND=MAGENTA
2030 PLOT 32,32,32:REM THREE SPACES IN MAGENTA
2040 PLOT 16:REM BACKGROUND=BLACK (FOREGROUND IS STILL GREEN)
2050 PRINT " ECONOMY MODEL ";
2059
2060 PLOT 19:REM FLAG IS STILL ON, SO BACKGROUND BECOMES YELLOW
2070 PLOT 32,32,32:REM THREE SPACES IN+ YELLOW
2080 PLOT 16,29:REM BACKGROUND TO BLACK; FLAG OFF
2090 PRINT " DELUXE MODEL "
2099
2100 RETURN

```



```

2197
2198 REM ***** SUBROUTINE TO LABEL VERTICAL AXIS *****
2199
2200 PLOT 23:REM WHITE
2209
2210 FOR Y=5 TO 20 STEP 5
2220   PLOT 3,11,25-Y:REM POSITION CURSOR
2230   PRINT Y
2240 NEXT
2249
2250 PLOT 22:REM CYAN
2260 PLOT 3,0,9:REM POSITION CURSOR
2269
2270 PRINT " NUMBER"
2280 PRINT
2290 PRINT " OF"
2300 PRINT
2310 PRINT " WIDGETS"
2320 PRINT
2330 PRINT "( X1000 )"
2339
2340 RETURN
2997
2998 REM ***** SUBROUTINE TO SET UP GRAPH BORDERS *****
2999
3000 PLOT 22:REM CYAN
3010 PLOT 30:REM FLAG ON FOR SPECIAL CHARACTERS
3020 PLOT 27,10:REM VERTICAL MODE
3030 PLOT 3,14,5:REM POSITION CURSOR
3038
3039 REM DRAW VERTICAL AXIS
3040 FOR Y=5 TO 24:PLOT 98:NEXT
3049
3050 PLOT 27,24:REM BACK TO PAGE MODE
3060 PLOT 3,15,25:REM POSITION CURSOR
3068
3069 REM DRAW HORIZONTAL AXIS
3070 FOR X=15 TO 63:PLOT 101:NEXT
3079
3080 PLOT 29:REM FLAG OFF
3090 RETURN
3997
3998 REM ***** SUBROUTINE TO GET DATA *****
3999
4000 RESTORE 4100
4009
4010 FOR J=1 TO 30
4020   READ EC(J),DE(J)
4030 NEXT
4039
4040 RETURN
4096
4097 REM ***** THE DATA *****
4098

```



```

4099 REM 1950-1959
4100 DATA 5,3,7,4,7,5,6,5,8,5,9,6,8,5,10,5,11,5,12,6
4104 REM 1960-1969
4105 DATA 12,7,13,9,14,10,15,11,15,12,14,11,9,7,8,5,9,4,10,6
4109 REM 1970-1979
4110 DATA 11,7,13,9,17,15,16,14,16,10,15,8,16,8,17,8,16,8,16,10

```

6.13 PLOTTING CHARACTER STRINGS

After entering the general plot mode with PLOT 2, all subsequent output to the screen will be PLOTTed. The PRINT instruction is one way of sending information to the screen, and so any instruction to PRINT will be PLOTTed, one byte (character) at a time, according to the ASCII values of the characters. That is, PRINT "AA" is accomplished also by PLOT 65,65,13,10. (65 is the ASCII value for "A"; and don't forget the carriage return/line feed sequence, which, however, can be suppressed with a semicolon after whatever is to be PRINTed.) Thus,

```
PLOT 2:PRINT "AA";:PLOT 255
```

will put a plot block at X=65,Y=65.

This will allow you to predefine certain graphics as character strings. Some of the lower numbers will not be available, however, since you won't find characters equivalent to ASCII 0 through 13 for use in character strings. (But ISC's FREDI, the BASIC Text Editor, may help you out.) Nor will you be able to use characters with values greater than 191 unless you have the deluxe keyboard with the special function keys. (See Appendix G.) (By the way, the use of those keys in character strings will produce a nearly undecipherable program listing.) Thus, the instruction

```
PLOT 2,32,32,242,90,32,61,90,32,32,255
```

will produce a triangle. So will

```
TR$="Z =Z ":PLOT 2,32,32,242:PRINT TR$;:PLOT 255
```

```

5 REM PROGRAM 6.25
6 REM PLOTTING CHARACTER STRINGS
10 PLOT 6,6,12:REM SET UP
20 PLOT 30:REM FLAG ON
29
30 FOR J=1 TO 500
40 PLOT 6,64*RND(1):REM SELECT ARBITRARY COLOR
50 PLOT 2:REM GENERAL PLOT MODE
60 PLOT 251:REM INCREMENTAL POINT PLOT SUBMODE
70 PRINT 1000*-COS(RND(1));
80 PLOT 255
90 NEXT
99
100 GOTO 30

```


7. PLOT 6,128 AND ABOVE

The CCI codes (PLOT 6,n) from 0 through 63 represent the 64 possible foreground/background color combinations. If 64 is added to such a number, the foreground will blink. (See Section 2.7.) That accounts for the numbers from 0 through 127. But an eight bit binary number, which is what the CCI code is, can go up to 255. What results with PLOT 6,128 and PLOT 6,129 and ... PLOT 6,255?

Such numbers will do two things. First they will specify the foreground/background colors. Subtract 128 to get the usual number. That is, PLOT 6,129 will select red on black, because $129-128=1$, and PLOT 6,1 selects red on black. If the resulting number is greater than 64, the foreground color will blink. Second, output to the screen will be in the form of character plot. Now, although things will appear to be in the character plot mode, you ought to issue all subsequent instructions as though you were not in the character plot mode. This may be a bit confusing. Just imagine that PLOT 6,129 (for example) is a way not only of setting the color to red, but also of translating English into Plot-English, a new language spoken by the computer. Consequently, if you type in

PLOT 6,129

the result will be as shown in Figure 7.1. That's Plot-English for "READY".

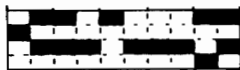


FIG. 7.1

(If the A7 bit was on, you got a double row of Plot-English.) Now type in PLOT 6,1, and you will get "READY". (That's English-English.) You can verify that "READY" was correctly translated into Plot-English by using the character plot mode and PLOtting the ASCII values of those five letters. (See Table 7.1.)

<u>Letter</u>	<u>ASCII Value</u>
R	82
E	69
A	65
D	68
Y	89

TABLE 7.1

```
5 REM PROGRAM 7.1
6 REM VERIFYING PLOT MODE OUTPUT
10 PLOT 2,254:REM CHARACTER PLOT SUBMODE
20 PLOT 82,69,65,68,89:REM R,E,A,D,Y
30 PLOT 255
```


Notice that if you are in the (ordinary) plot mode, the plot mode will automatically be exited if the program ends or if it is interrupted. Type in this:

PLOT 2,254

You still get the "READY" prompt back. BASIC automatically exited the plot mode. It will do the same when you're in Plot-English as well.

PLOT 6,129,2,254

will also come back with "READY". That is, if you are in Plot-English, you will stay in Plot-English unless you issue an appropriate CCI code (like PLOT 6,1) or you enter the general plot mode. Hence, line 60 in the following program is not necessary.

```
5 REM PROGRAM 7.2
6 REM PLOT MODE
10 PLOT 6,129:REM ENTER PLOT
20 PRINT "THIS IS PLOT-ENGLISH"
29
30 PLOT 2:REM GENERAL PLOT MODE
40 PLOT 63,63:REM PLOT A POINT
50 PLOT 255:REM EXIT PLOT MODE
59
60 PLOT 6,1:REM BACK TO ENGLISH-ENGLISH
70 PRINT " BACK TO NORMAL"
```

In Plot-English, all the regular plot codes apply as usual, but you will always end up in Plot-English (unless, to repeat, you escape it with a PLOT 6,n instruction, where n is less than 128, or unless you enter the general plot mode), and all letters and characters will appear as character plots--except for those in the blind cursor.

```
5 REM PROGRAM 7.3
6 REM BLIND CURSOR IN PLOT MODE
10 PLOT 6,131:REM YELLOW AND PLOT
20 PLOT 3,15,11:REM POSITION CURSOR
30 PRINT "TEST #1";:REM SUPPRESS THE CR/LF
40 PLOT 10:REM CURSOR DOWN
50 PLOT 25,25:REM CURSOR TO THE RIGHT TWICE
60 PRINT "TEST #2"
69
70 PLOT 3,127:REM BLIND CURSOR
80 PLOT 15,19:REM X,Y
90 PLOT 1:REM CCI CODE
100 PRINT "BLIND CURSOR"
110 PLOT 27,27:REM EXIT BLIND CURSOR
119
120 PRINT:PRINT
130 PRINT "TEST #3"
140 PLOT 6,2:REM GREEN; BACK TO NORMAL
150 PRINT "TEST #4"
```


The blind cursor has its own CCI code. You may use Plot-English in the blind cursor mode, too, if you wish, by specifying the appropriate CCI code there. Change line 90 in the program above to:

```
90 PLOT 129:REM BLIND CURSOR CCI CODE
```

Experiment with this:

```
5 REM PROGRAM 7.4
6 REM PLOT A BORDER AROUND THE SCREEN
10 PLOT 12:REM ERASE SCREEN
20 PLOT 6,134:REM CYAN AND PLOT
30 PLOT 27,24:REM PAGE MODE
39
40 N=8:REM FOR THE LOOP IN LINE 200
50 GOSUB 200:REM PLOT TOP BORDER
59
60 PLOT 3,0,31:REM POSITION CURSOR
70 GOSUB 200:REM PLOT BOTTOM BORDER
79
80 PLOT 27,10:REM VERTICAL MODE
90 PLOT 8:REM HOME CURSOR
100 N=4
110 GOSUB 200:REM PLOT LEFT BORDER
119
120 PLOT 3,63,0
130 GOSUB 200:REM PLOT RIGHT BORDER
139
140 PLOT 27,11:REM SCROLL MODE
150 PLOT 6,2:REM REGULAR GREEN
159
160 END
198
199
200 FOR J=1 TO N
210 PRINT "A BORDER";
220 NEXT
229
230 PRINT
240 RETURN
```

Notice that in the program above the screen must be cleared before entering plot, because to clear the screen is to put ASCII 32 into all the character positions; but when the computer is speaking Plot-English, 32 shows up as in Figure 7.2. On the other hand, if the test mode is used (see



FIG. 7.2



FIG. 7.3

Section 5) and zero is specified, each character position on the screen will be filled with nothing (provided the A7 is off--i.e., regular character

height), thereby clearing the screen.

```
5 REM PROGRAM 7.5
6 REM ERASE PAGE IN PLOT MODE
10 PLOT 15:REM A7 MUST BE OFF
20 PLOT 6,129:REM PLOT MODE; RED ON BLACK
30 PLOT 27,25:REM TEST MODE
40 PLOT 0:REM FILL SCREEN WITH NOTHING
50 PLOT 6,2:REM BACK TO NORMAL
```

If you have some characters which are built up of plot blocks, and if you can manage to divide each of those characters into two-by-four rectangles (= one character position; or a double row of them if the A7 is on) such that the values of each will be equivalent to some printable ASCII character, then you may be able to PRINT these special characters using the Plot-English mode. For example, you could, in Plot-English, PRINT the two blocks shown in Figure 7.3. The value is 65 and is therefore equivalent to "A", which is a printable character if ever there was one. But you could not PRINT the upper left block alone, for that would be equivalent to PLOT 1, which is the instruction to LOAD a "Menu" program from disk. Nor could you PRINT all eight of the blocks, for that would be the plot mode escape (255). In fact, you cannot plot the lower right rectangle at all, either by itself or with others, for once the numbers reach 128, 128 is subtracted from them. (See Appendix A.) Thus, 128 is equivalent to PLOT 0, and 129 will LOAD the "Menu".

In general, what you can print with Plot-English is far more limited than what you can do in the character plot mode. On the other hand, everything else being equal, the character plot mode is usually slower and more costly in terms of memory. In addition, characters PLOTted in the character plot mode require the use of the cursor; if you wish to eliminate it and still print plot characters, you will have to use the blind cursor and a blind cursor CCI code greater than 127.

8. SCREEN REFRESH

Anything on the screen is held in a certain part of memory--the screen refresh memory. If you change what is in that memory you change what is on the screen. The screen refresh begins at decimal 28672 and goes through 32767. This area of the Compucolor II computer's memory was originally intended to be a "slow" refresh, and the locations from 24576 through 28671 to be a "fast" refresh. The reason is not clear; but anyway it was not implemented. As a result, the Compucolor II computer's "fast" refresh memory actually maps to the same **physical** memory locations as its "slow" refresh, even though the addresses seem to specify otherwise.

Screen refresh locations do not always coincide with character positions on the screen. If the screen is cleared, then 28672 will be the location in memory of the character at the top left of the screen. But if the screen scrolls up one line, 28672 will be at the bottom left. Each time the screen is scrolled up one line, 28672 will move up, too, until it reaches the top and "wraps around" once more to the bottom.

Just to make sure you know which position on the screen coincides with which location in refresh memory, you ought first to clear the screen. Now 28672 will be in the top left character position. In order to keep it there, set the display to page mode (PLOT 27,24).

Each character position on the screen is associated with two locations in refresh memory. The first contains the character which is on the screen, and the second contains the CCI code for that character.

```
5 REM PROGRAM 8.1
6 REM PEEKING SCREEN REFRESH MEMORY
10 PLOT 15:REM SMALL CHARACTERS
20 PLOT 6,6,12:REM COLOR=CYAN; CLEAR SCREEN
30 PRINT "A":REM THIS WILL APPEAR AT X=0, Y=0
39
40 PRINT "THE CHARACTER # IS"PEEK(28672)
50 PRINT "THE CCI CODE IS"PEEK(28673)
```

The result of this program will tell you that "A" is equivalent to ASCII 65, and the CCI code for cyan is 6. Well, you already knew that.

If we can PEEK at memory to see what's on the screen, we can also POKE numbers into memory to put characters on the screen. Since the CCI code for green is 2,

```
POKE 28673,2
```

should turn that cyan "A" into a green "A".

Since there are 64 characters per line, and since there are two memory locations per character, there are 128 memory locations per line of the screen display. So the memory location for the character at X=0,Y=1 will be 28672+128=28800. Let's put a yellow "B" there:

```
POKE 28800,ASC("B"):POKE 28801,3
```

In general, the memory location corresponding to any cursor X,Y position is given by this formula: **28672+128*Y+X+X**

Thus,

```
PLOT 3,15,5:PRINT "A"
```

could also be accomplished by

```
POKE 28672+128*5+15+15,65
```

Ordinarily when you wish to change the color of something which is already on the screen, you will specify the new color and redraw (or rePRINT) whatever it is. An alternative is now available: POKE the new CCI code into the memory locations associated with the display. (This will be slower than PRINTing, but it can in many cases be more elegant. Besides, you won't have to know just what is on the screen in order to change its color by POKEing into the screen memory.)

```
5 REM PROGRAM 8.2
6 REM POKEING INTO SCREEN RAM
10 PLOT 15:REM REGULAR HEIGHT CHARACTERS
20 PLOT 6,0,12:REM COLOR=BLACK; ERASE SCREEN
29
30 X=15:Y=10:REM X,Y OF TOP LEFT OF DISPLAY TO BE DRAWN
40 GOSUB 200:REM DRAW IT
49
50 SC=28673+128*Y+X+X:REM SCREEN RAM LOCATION FOR CCI CODE OF
51 REM TOP LEFT CHARACTER OF WHATEVER
52 REM WAS DRAWN
59
60 FOR C=1 TO 127:REM FOR EACH CCI CODE THRU BLINK
69
70 FOR J=0 TO 9:REM FOR EACH LINE OF THE DISPLAY...
80 P=SC+128*J:REM SCREEN RAM FOR START OF NEW LINE
89
90 FOR K=0 TO 18 STEP 2:REM FOR EVERY OTHER RAM LOCATION...
100 POKE P+K,C:REM PUT IN THE NEW CCI CODE
110 NEXT
119
120 NEXT
129
130 NEXT
139
140 PLOT 6,2
149
150 END
159
198 REM **** SUBROUTINE TO PRINT SOMETHING (10X10) IN BLACK ****
199
200 FOR J=Y TO Y+10
210 PLOT 3,X,J:REM POSITION CURSOR AT START OF EACH LINE
219
220 FOR K=1 TO 10:REM 10 CHARACTERS PER LINE
230 PLOT 96:REM HATCH CHARACTER (IF FLAG IS OFF)
240 NEXT
```



```

249
250 NEXT
259
260 PLOT 3,X+2,Y+2:PRINT "THIS IS"
270 PLOT 3,X+2,Y+3:PRINT "A TEST "
280 PLOT 3,64,0:REM HIDE THE CURSOR
290 RETURN

```

Here are some interesting changes to that program:

```

60 delete this line
.
.
.
95 C=128*RND(1):REM PICK A RANDOM COLOR FROM 0 THRU 127
.
.
.
130 GOTO 70

```

The ASCII characters begin at 32 (space) and go through 95 ("_"). If we POKE one of these numbers into the screen memory, we get the corresponding character on the screen. But what about the numbers from 0 to 31? And what about the numbers from 96 to 255? (We can't go higher than 255 using an 8 bit binary number.)

```

5 REM PROGRAM 8.3
6 REM POKE ANY CHARACTER INTO SCREEN MEMORY
10 PLOT 6,6,12:REM SET UP
19
20 PLOT 3,0,5:REM POSITION CURSOR
30 PLOT 11:REM ERASE ANY PREVIOUS INPUT
40 INPUT "POKE WHAT NUMBER? ";N$:N=VAL(N$)
50 IF N<0 OR N<>INT(N) OR N>255 THEN 20
59
60 PLOT 11:REM ERASE NEXT LINE
70 PRINT "CHARACTER ="N
80 POKE 28672,N
89
90 GOTO 20

```

You will discover that POKEing values from 0 through 31 gives some of the computer's special characters--those, namely, associated with shift-@, shift-A, shift-B, etc., with the FLAG on. And values from 96-127 yield the special characters you would get with the FLAG off. (See Appendix B.)

But what about the other 128 numbers, from 128-255? You will get the same characters as before, but now each character will be a double height character. However, since each double height character requires two lines of the display, if you POKE such a character's number into line 0 (28672, for example), you will get only the top half of the character. If you POKE that same value into 28800 (one line down) you will get that character's bottom half. Consequently, you now have available a number of "new" special characters which you can place on the screen. The drawback, of course, is that half of them can appear only on even numbered lines and the other half

on odd numbered lines. (See Section 1.5 on double height characters.) Even with this restriction you might be able to make use of some of these characters. For example, some Greek symbols are available: the top half of the double height "&" (=166) gives a pretty good "ø". "π" is available with the bottom half of the double height "#" (=163), and "ψ" comes from the top half of the double height "*" (=170).

```

5 REM PROGRAM 8.4
6 REM HALF CHARACTER COLOR CHANGES
10 PLOT 14:REM DOUBLE HEIGHT
20 PLOT 6,2,12:REM SET UP
29
30 PLOT 3,18,15:REM POSITION CURSOR
40 PRINT "C O L O R   G R A P H I C S"
49
50 PLOT 3,64,0:REM HIDE CURSOR
59
60 C=0:REM CCI CODE
69
70 FOR Y=14 TO 15:REM FOR EACH OF THE TWO
71     REM (REGULAR HEIGHT) LINES...
79
80     FOR X=18 TO 44:REM FOR EACH OF THE 26 CHARACTERS...
90         C=C+1:IF C=8 THEN C=1:REM NEXT CCI CODE
100        POKE 28673+128*Y+X,X,C:REM POKE IT INTO SCREEN
110    NEXT
119
120 NEXT
129
130 GOTO 70

```

The CCI code associated with each character in screen memory is the same CCI code we have been dealing with all along. Thus, if PEEK(28673)=65, then the character at 28672 is blinking red on black. If the CCI code is greater than 127, then the character is a plot character and not an ASCII character. Thus,

POKE 28672,66

will place a "B" on the screen unless 28673 holds a CCI code greater than 127, in which case the 66 represents the plot character shown in Figure

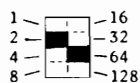


FIG. 8.1

8.1. (See Section 7.) A program such as 8.2 might go awry if it tries to change colors of portions of the screen which contain plot characters. Statements will have to be included to determine whether the CCI code at a given address is greater than 127. (If the display is entirely ASCII or entirely plot, there is, of course, no need for such additional checking.) The following program plots a series of vectors in green and then POKES 129 into the screen memory as a CCI code in order to change the vectors to

red. Notice what happens.

```
5 REM PROGRAM 8.5
6 REM POKEING CCI CODES GREATER THAN 127
10 PLOT 6,2:REM SELECT COLOR=GREEN
20 PLOT 15,12:REM ERASE SCREEN
30 PLOT 3,64,0:REM HIDE CURSOR
37
38 REM DRAW SEVERAL VECTORS FROM TOP LEFT CORNER
39 REM OUT FOR ABOUT 15 PLOT BLOCKS
40 PLOT 2,0,127:REM FIRST POINT
50 PLOT 242:REM VECTOR PLOT
60 PLOT 15,125:REM DRAW A VECTOR
70 PLOT 253,0,127,242,15,120:REM ANOTHER VECTOR
80 PLOT 253,0,127,242,15,115:REM A THIRD
90 PLOT 255:REM EXIT PLOT MODE
99
100 FOR Y=0 TO 4:REM FOR 1ST 4 LINES OF DISPLAY...
109
110 FOR X=0 TO 7:REM FOR 8 CHARACTERS/LINE...
120 POKE 28673+128*Y+X+X,129:REM RED AND PLOT
130 NEXT
139
140 NEXT
```

How did those new plot blocks manage to appear? The answer is given in Section 7. The action of the PLOT 12 in line 20 is to fill each character position with 32, the ASCII code for space. But if the CCI code is greater than 127, 32 will be interpreted as a plot number--i.e., one of the eight plot blocks per character position. The problem is solved using the test mode. (See Section 5.) Leave in line 20, because it sets the beginning of screen memory to the top left character position. But now clear the screen again using the test mode with a CCI code greater than 127 (and the A7 off). You might add these lines:

```
22 PLOT 6,128:REM BLACK AND PLOT
24 PLOT 27,25:REM TEST MODE
26 PLOT 0:REM FILL SCREEN WITH NO PLOT BLOCKS
28 PLOT 6,2:REM EXIT PLOT
```

The ability to know at any time what is on the screen can be used in a variety of ways. The "Display Create/Edit/Dup" program in Section 9.5 makes use of this for printing a message and then replacing it with the user's display. The following program prints a random display and then moves a "*" (ASCII 42) across the screen, replacing those parts of the display it erased as it moves.

```
5 REM PROGRAM 8.6
6 REM MOVE A "*" ACROSS THE SCREEN
9
10 PLOT 15:REM SMALL CHARACTERS
19
20 PLOT 6,63*RND(1)+1:REM RANDOM COLOR
30 PLOT 12,27,24:REM ERASE SCREEN & SET TO PAGE MODE
```



```

39
40 FOR J=1 TO 300:REM 300 RANDOM CHARACTERS
50 X=64*RND(1)
60 Y=32*RND(1)
70 COLR=63*RND(1)+1
80 CHAR=95*RND(1)+32:REM RANDOM ASCII CHARACTER
90 PLOT 6,COLR,3,X,Y,CHAR:REM PRINT THE CHARACTER
100 NEXT
109
110 PLOT 3,64,0:REM HIDE CURSOR
119
120 Y=INT(32*RND(1)):REM PICK A RANDOM LINE
130 SC=28672+128*Y:REM SCREEN RAM AT START OF THAT LINE
139
140 C1=PEEK(SC):REM GET CHARACTER THERE
150 R1=PEEK(SC+1):REM GET ITS CCI CODE
159
160 POKE SC,42:REM DISPLAY THE "*"
170 POKE SC+1,1:REM MAKE IT RED ON BLACK
179
180 FOR X=1 TO 63:REM FOR EACH CHARACTER POSITION ON THE LINE...
190 C2=PEEK(SC+X+X):REM GET NEXT CHARACTER
200 R2=PEEK(SC+X+X+1):REM GET ITS CCI CODE
209
210 POKE SC+X+X,42:REM DISPLAY THE "*" AT THE NEW SPOT
220 POKE SC+X+X+1,1:REM MAKE IT RED
229
230 POKE SC+X+X-2,C1:REM PUT BACK PREVIOUS CHARACTER
240 POKE SC+X+X-1,R1:REM PUT BACK ITS CCI CODE
249
250 C1=C2:R1=R2:REM NEW BECOMES OLD
260 NEXT
269
270 POKE SC+126,C1:REM PUT BACK VERY LAST CHARACTER
280 POKE SC+127,R1:REM PUT BACK ITS CCI CODE
289
290 GOTO 20:REM REPEAT

```

If the "*" moves too fast for you, add this line:

```

255 FOR T=1 TO 50:NEXT:REM SLIGHT PAUSE

```

Note that sometimes the "*" will seem to be preceded by a faint flash. That is because the "*" will appear in whatever foreground/background color is at its new location, and then it will be turned red. Although the flash is usually barely noticable, it can be cured by interchanging lines 210 and 220.

9. CRT MODE

Pressing ESC CRT takes you out of BASIC and lets you draw anything you wish on the screen: almost anything you can program to draw, you can do from the keyboard in the CRT mode. You can control the cursor, color, character height, and so on. These sequences may be important aids:

ESC K	Scroll Mode
ESC J	Vertical Mode
ESC X	Page Mode
ESC B	Character Plot Mode
ESC ESC	Plot Mode Exit
CONTROL B	Plot Mode (special function keys only)
F15	Plot Mode exit (special function keys only)

9.1 CRT PLOTTING

If you have the color key pad, then, in plot mode (ESC B), pressing one of those keys will turn on the corresponding plot block in the character position wherever the cursor is presently located: the eight color keys are taken to represent the eight plot blocks in a character position. If you don't have the color key pad, you must use the CONTROL key along with Q for red, W for white, and so on, according to Figure 9.1. Pressing the

(P) Black		Blue (T)
(Q) Red		Magenta (U)
(R) Green		Cyan (V)
(S) Yellow		White (W)

FIG. 9.1

same key a second time will turn the plot block off--just as though the FLAG were on. While you are in the plot mode, you may still use the cursor controls, print text, etc., but you may not change colors unless you exit the plot mode (ESC ESC).

9.2 SPECIAL FUNCTION KEYS

If you have the deluxe model keyboard with the special function keys along the top, you may use them to enter the various plot submodes. First you will have to enter the general plot mode with CONTROL B. You may consider that to be equivalent to the PLOT 2 instruction in BASIC. This automatically puts you in the point plot mode unless you specify otherwise by means of one of the special function keys. The ASCII values of the next two keys will determine the X,Y co-ordinates of a point plot. So, for example,

pressing AA in the point plot mode will plot a plot block at X=65,Y=65. Pressing the F2 key is equivalent to issuing PLOT 242 in BASIC (vector plot), so the next two keys will determine the end points of a vector drawn from 65,65. The F15 key (plot escape) is equivalent to PLOT 255 in BASIC. For each number from 0 through 255 there is some key, either by itself or with the shift key, or with the CONTROL key, or with both shift and CONTROL keys together (i.e., the COMMAND key) which will output that number to the CRT. (See Appendix G.) As an example of the use of these keys, perform the following key strokes:

<u>KEY</u>	<u>COMMENT</u>
ESC CRT	...enter CRT mode. You could also hit CPU RESET, which has the advantage of clearing the screen and resetting the screen refresh memory.
BG ON BLUE FG ON RED	...set colors.
ERASE PAGE	...clear the screen in blue.
CONTROL B	...enter general plot mode.
F13	...point plot submode. Just as in BASIC, this is optional.
CONTROL @	...=0.
DELETE CHAR	...=127. A plot block now appears at X=0,Y=127.
F2	...enter vector plot submode.
DELETE CHAR	...=127.
DELETE CHAR	...=127. A vector is now drawn from 0,127 to 127,127.
F0	...enter incremental vector plot submode.
SHIFT/MAGENTA	...or SHIFT/CONTROL/U. =149, which increments the vector down and in at both ends.
(Continue SHIFT/MAGENTA until the vector reaches the bottom of the screen.)	

9.3 SAVING DISPLAYS

The drawing of displays used by a program will often be done in the program itself. But you can also draw the display in the CRT mode and save it on disk. This way, your program need only load the display from disk rather than take up memory space (and time) required to draw it.

The business of saving a display is done in FCS by saving the contents of the screen refresh memory. This simple command will suffice:

```
FCS> SAVE SCREEN.DSP 7000 1000
```

The file type ".DSP" indicates to you when you read the disk directory that the file is a screen display. You needn't use ".DSP", however; you could use ".PIC" for "picture", or anything else you felt appropriate. The "7000" specifies (in hexadecimal) the starting location in memory where the file should be loaded, and the "1000" (hex again) indicates the byte count. The screen refresh memory begins at decimal 28672 (=7000 hex) and contains decimal 4096 (=1000 hex) bytes. Since 28672 is the starting location, you should make sure to clear the screen (and perhaps set it to page mode) before drawing and saving the display, otherwise the display might be loaded into the wrong part of the screen.

Saving the display will save everything on the screen. (Unless you specify otherwise. You could save part of the display by specifying a different starting address or a different byte count. For example, SAVE SCREEN.DSP 7000 7FF would save only the top half of the display.) If you go into FCS by ESC D after drawing your picture, and if you type in the command to save your display, then the **FCS>** and your command will appear on the screen, and so they will be part of the display as saved. That is not, I presume, what you wish to happen. One answer to the problem is to first write a program in BASIC:

```
10 PLOT 27,4:REM ENTER FCS
20 PRINT "SAVE SCREEN.DSP 7000 1000"
30 PLOT 27,27:REM EXIT FCS
```

Then go into the CRT mode, draw the picture, go back into BASIC with ESC E (BASIC reset), and RUN the program. Unfortunately, when you reenter BASIC, a carriage return/line feed will automatically happen, the "READY" prompt will appear, and then two more CR/LF's will occur. In addition, you will have to type in "RUN". Your display will contain the words "READY" and "RUN". However, if, before reentering BASIC, you set the foreground and background colors so that "READY" and "RUN" will be camouflaged, and then move the cursor so that when reentering BASIC those two (now invisible) words will not write over any part of your display, then you can save your display intact.

Of course, if you don't have any space at the left of your display for the "READY" and "RUN", then you've got troubles. (The solution is to use the "DISPLAY CREATE/EDIT/DUP" program in Section 9.5 below.)

Loading displays can be done in FCS or in BASIC. In FCS just type in:


```
FCS> [erase page]
FCS> LOAD SCREEN.DSP
```

And in BASIC it will be:

```
10 PLOT 12:REM ERASE PAGE & RESET SCREEN MEMORY
20 PLOT 27,4:REM ENTER FCS
30 PRINT "LOAD SCREEN.DSP"
40 PLOT 27,27:REM EXIT FCS
```

9.4 DUPLICATING DISPLAYS

You can take a display stored on one disk and duplicate it to another disk. If a DUP instruction is available in your FCS, and if you have two disk drives, then you can duplicate an entire disk. Of course, you will also copy over all other programs on that disk. If you have but one disk drive, you may use ISC's "COPY" program (on older machines, a COPY instruction is available in FCS), or else you may write your own:

```
5 REM PROGRAM 9.1
6 REM DUP A SCREEN DISPLAY
10 PLOT 12:REM ERASE SCREEN & RESET SCREEN MEMORY
19
20 PLOT 27,4:REM FCS
30 PRINT "LOAD SCREEN.DSP"
39
40 PLOT 27,27:REM EXIT FCS
49
50 INPUT "";A$:REM WAIT
51
52 REM AT THIS POINT, SWITCH DISKS AND THEN PRESS
53 REM "RETURN". DOING SO WILL PRINT NOTHING ON
54 REM THE SCREEN.
55
60 PLOT 27,4:REM BACK TO FCS
70 PRINT "SAVE SCREEN.DSP 7000 1000"
79
80 PLOT 27,27:REM EXIT FCS
```

Of course, you could alter one or both of lines 30 and 70 to LOAD or SAVE the display on a different drive.

9.5 EDITING DISPLAYS

How will you alter a display you have previously saved? First write the BASIC program to save the display, then LOAD the display into the screen. Then type ESC CRT to enter the CRT mode, make the necessary changes, and finally save the display as previously described. This procedure involves

problems in addition to the usual ones of having to set the foreground and background colors so that the "READY" prompt is invisible, and so on. For when you enter the CRT mode, a whole line of your display is erased.

The following program will aid you in saving, editing and duplicating screen displays. The main idea is to mimic the CRT mode by using a no-echo routine written by Ben Barlow and published in DATACHIP, Vol.1, No.3, March, 1979. (DATACHIP is the newsletter of the Compucolor Users' Group of Rochester, NY and is available by writing to the editor, Ben Barlow, 161 Brookside Dr., Rochester, NY 14618.) While the echo is turned off, the program takes all inputs from the keyboard and PLOT's them instead of echoing them. It's about that simple. You can change colors, move the cursor, change character size, blink, and so on. ESC B (plot) lets you use the color key pad (or CONTROL with the appropriate color key) to plot character plots. (Remember to use ESC ESC to exit that plot mode.) If you have the deluxe keyboard, you can still use the special function keys. In short, you're in a pseudo-CRT mode, and you probably won't be able to tell it from the genuine CRT mode.

When you are done creating or editing the display, you type in a predefined character (in my version it is "!"), and a message appears asking whether you wish to continue editing or whether you wish to now save the display. To save the display, mount the appropriate disk, type "S", and press "RETURN". The message will be replaced with that section of your display which it erased when it was printed, and the display will be saved. As usual, the brackets, [], in the following program indicate information which you should enter from the keyboard.

```
5 REM PROGRAM 9.2
6 REM DISPLAY CREATE/EDIT/DUP
7
10 REM NOTE: INPUT ERROR CHECKING IS
11 REM      NOT FULLY IDIOT PROOF
12
13 REM MAJOR VARIABLES
14
15 REM KB.....LOCATION OF KEYBOARD CHARACTER.
16
17 REM KF.....LOCATION OF KEYBOARD CHARACTER FLAG.
18
19 REM SC.....2 LESS THAN START OF SCREEN REFRESH.
20
21 REM STP.....ASCII VALUE OF WHATEVER CHARACTER YOU WISH
22 REM      TO USE TO STOP THE DRAW (EDIT) ROUTINE AND
23 REM      BEGIN THE SAVE DISPLAY ROUTINE. I'VE
24 REM      CHOSEN THE "!".
25
26 REM OD$.....NAME OF OLD DISPLAY.
27
28 REM OV$.....VERSION OF OLD DISPLAY.
29
30 REM ND$.....NAME OF NEW DISPLAY.
31
32 REM NV$.....VERSION OF NEW DISPLAY.
33
34 REM CH( )...CHARACTERS PEEKED FROM SCREEN (SEE LINE 940).
35
```



```

36 REM CO( )...CCI CODES PEEKED FROM SCREEN (SEE LINE 940).
37
38
39 REM POKE IN THE NO-ECHO PATCH
40 GOTO 63000
49
50 DIM CH(49),CO(49)
60 KB=33278:KF=33247:SC=28670
70 STP=ASC("!")
79
80 PLOT 14:REM LARGE CHARACTERS
90 PLOT 30,16,29,22,12:REM SET COLORS; FLAG OFF; CLEAR SCREEN
100 PLOT 27,11:REM SCROLL MODE
109
110 PRINT:PRINT TAB(20)"DISPLAY CREATE/EDIT/DUP"
120 PRINT TAB(20)"[grn]DAVID B. SUITS, 12 A.L.":PRINT
129
130 PLOT 15: REM SMALL CHARACTERS
139
140 PRINT:PRINT TAB(15)"[yel]CREATE A NEW DISPLAY . . . . . [red]
1"
150 PRINT:PRINT TAB(15)"[yel]EDIT AN OLD DISPLAY . . . . . [red]
2"
160 PRINT:PRINT TAB(15)"[yel]DUP A DISPLAY TO ANOTHER DISK. . [red]
3"
170 PRINT
180 PRINT TAB(35);:INPUT "[mag]YOUR CHOICE? [yel]";CH$
190 CH=VAL(CH$)
200 IF CH<1 OR CH<>INT(CH) OR CH>3 THEN PLOT 28,11:GOTO 180
210 ON CH GOTO 300,400,400
298
299 REM

GET FILE NAME FOR THE DISPLAY

300 T$="NEW":GOSUB 700:ND$=D$:NV$=V$
308
309 REM CREATE AND SAVE THE DISPLAY
310 GOTO 800
397
398 REM

EDIT/DUP A DISPLAY

399 REM GET FILE NAME OF OLD DISPLAY
400 PRINT
410 INPUT "PLEASE MOUNT DISK WITH DISPLAY AND PRESS RETURN ";A$
419
420 PLOT 18:REM GREEN
430 PLOT 27,4:REM FCS
440 PRINT "DIR"
450 PLOT 27,27:REM EXIT FCS
459
460 T$="OLD":GOSUB 700:OD$=D$:OV$=V$

```



```

468
469 REM GET FILE NAME FOR NEW DISPLAY
470 T$="NEW":GOSUB 700:ND$=D$:NV$=V$
478
479 REM DOES USER WISH TO EDIT THE DISPLAY?
480 IF CH=2 THEN 800
488
489 REM NO, SO JUST DUP THE DISPLAY
490 GOTO 870
698
699 REM

GET NAME OF DISPLAY

700 PRINT:PRINT "[mag]WHAT IS THE SIMPLE NAME OF THE [wht]"T$"[mag]
DISPLAY?"
710 PRINT "(MAXIMUM 6 CHARACTERS. DO NOT INCLUDE '.DSP' OR THE"
720 INPUT "VERSION NUMBER):[yel] ";D$
729
730 IF LEN(D$)>6 OR LEFT$(D$,1)<"A" OR LEFT$(D$,1)>"Z" THEN 700
740 PRINT
750 INPUT "[mag]VERSION NUMBER?[yel] ";V$
760 RETURN
798
799 REM

CREATE/EDIT THE DISPLAY

800 PRINT:PRINT:PRINT
810 PRINT TAB(15)"***** PSEUDO-CRT MODE *****"
820 PRINT:PRINT
830 PRINT "[mag]USE THE KEYBOARD JUST AS YOU WOULD IN CRT MODE."
840 PRINT "TYPE [yel]";:PLOT STP:PRINT "[mag] WHEN YOU'RE FINISHED.
"
850 PRINT
860 INPUT "PRESS RETURN TO BEGIN ";A$
868
869 REM ERASE SCREEN AND SET TO PAGE MODE
870 PLOT 12,27,24:IF CH=1 THEN 900
879
880 PLOT 27,4:PRINT "LOA "OD$".DSP;"OV$:PLOT 27,27
889
890 IF CH=3 THEN 900
898
899 REM

MIMIC THE CRT MODE

900 POKE KF,31:REM NO-ECHO
910 POKE KB,0
919
920 A=PEEK(KB):IF A=0 THEN 920
929
930 IF A<>STP THEN PLOT A:GOTO 910

```



```

935
936 REM

END OF PSEUDO-CRT MODE

937 REM GET 1ST 49 CHARACTERS & CCI CODES OF SCREEN FOR
938 REM REDRAWING THE DISPLAY AFTER THE MESSAGE BELOW
939 REM IS PRINTED
940 FOR J=1 TO 49:CH(J)=PEEK(SC+J+J):CO(J)=PEEK(SC+J+J+1):NEXT
947
948 REM SMALL CHARACTERS (& TURN OFF POSSIBLE BLINK)
949 REM SET COLORS; HOME CURSOR
950 PLOT 15,30,16,29,21,8
959
960 PRINT " TYPE [wht] E[mag] TO EDIT, OR MOUNT DISK & TYPE [wht] S[m
ag] TO SAVE [wht] ";

970 PLOT 26:INPUT " ";A$
978
979 REM DOES USER WISH TO SAVE THE DISPLAY?
980 IF A$="S" THEN 1100
988
989 REM NO, SO JUST GIVE REMINDER INSTRUCTION
990 PLOT 8:PRINT " [grn]TYPE [wht] ";:PLOT STP
1000 PRINT " [grn]WHEN YOU'RE DONE EDITING"SPC(16)""
1008
1009 REM NO-ECHO WHILE DELAYING FOR MESSAGE
1010 POKE KF,31:FOR J=1 TO 2000:NEXT
1018
1019 REM REPLACE MESSAGE WITH DISPLAY & GOTO PSEUDO-CRT MODE
1020 GOSUB 1200:GOTO 910
1098
1099 REM REPLACE MESSAGE WITH DISPLAY
1100 GOSUB 1200
1108
1109 REM SAVE THE DISPLAY
1110 PLOT 27,4:PRINT "SAVE "ND$".DSP;"NV$" 7000 1000":PLOT 27,27
1119
1120 PLOT 27,11:REM BACK TO SCROLL MODE
1128
1129 REM PUT TOP OF MEMORY BACK WHERE IT WAS
1130 TM=256*PEEK(32941)+PEEK(32940)+7
1140 HI=INT(TM/256):POKE 32941,HI
1150 POKE 32940,TM-256*HI
1159
1160 END:REM (OR ELSE LOAD "MENU" & RUN)
1197
1198
1199 REM REPLACE THE 49 CHARACTERS PEEKED FROM SCREEN
1200 FOR J=1 TO 49
1210 POKE SC+J+J,CH(J):POKE SC+J+J+1,CO(J)
1220 NEXT
1229
1230 RETURN

```


62998
62999 REM

BEN BARLOW'S NO-ECHO PATCH

```
63000 DATA 245,175,50,255,129,241,201
63010 TM=256*PEEK(32941)+PEEK(32940)-7
63020 RESTORE 63000
63030 FOR X=1 TO 7
63040   READ D:POKE TM+X,D
63050 NEXT
63060 BR=INT(TM/256)
63070 POKE 33221,195:POKE 33222, TM-BR*256+1
63080 POKE 33223, BR:POKE 32941, BR:POKE 32940, TM-BR*256
63090 CLEAR 50:GOTO 50
63091
63092 REM  FOR NO-ECHO, POKE 33247,31
63093 REM  TO RETURN TO ECHO, POKE 33247,12.
63094 REM  (AN INPUT STATEMENT OR THE END
63095 REM  OF THE PROGRAM WILL ALSO DO IT.)
```

*press N2 keys (such as in games)
without N2 showing on screen! R4 CUVAC
JAN/FEB 83.*

*See also GAMER3 attached to Port 2
on 'GAMES' disk —
patch sits on top of BASIC programs
& disallows any inputs to the screen
unless prompted from an INPUT
statement or is terminated by an
END command.*

10. QUAD-DIRECTIONAL SCROLLING PATCH

When the screen is in the scroll mode, new information may be continually printed out under whatever was printed before; the old part of the display rolls up like a sheet of paper in a typewriter. For many applications, however, you will wish to retain a great deal of old information without scrolling it up and off the screen. This is one reason for setting the display to page mode. But there is a drawback to page mode, namely, old information which you do not want to retain must be erased. That can be done as long as your program remembers exactly where that old information was printed.

The best of these two modes can be combined in a convenient manner by using a special machine language routine which will scroll only a certain portion of the screen while leaving the remainder undisturbed. The routine which I am presenting here is a modified version of the scrolling patch which was published in *COLORCUE*, Vol 1, No. 1.

10.1 WHAT THE PATCH DOES

The action of the routine is quite straightforward: you specify any rectangular area as the special scrolling area. Suppose you have chosen a seven character wide by three line high area whose top left character position is at cursor address X,Y. (This is a very small area indeed. I'm using it only as an illustration.) Now suppose you have printed three lines of characters in that rectangle (Figure 10.1) and you wish to scroll the

X,Y
→ ABCDEFG
HIJKLMN
OPQRSTU

FIG. 10.1

X,Y
→ HIJKEFG
HIJKLMN
OPQRSTU

FIG. 10.2

X,Y
→ HIJKLMN
OPQRSTU
OPQRSTU

FIG. 10.3

whole thing upwards one line and print a new line at the bottom of the rectangle, just as though the rectangle by itself were an ordinary display screen set to scroll mode. The machine language routine would, in effect, PEEK at the screen memory location one line down from the top left character position, take whatever was there and copy it into the character position above. Then it would move over to the right one character, PEEK at that location and copy that character into the line above, and so on for the entire line. Then it would go back down to the start of the next line and begin over again. And so on for all the lines. Figure 10.2 shows the state of affairs when the routine is about halfway along the first line. Figure 10.3 shows the final result. Notice that the bottom line is unaffected. You will have to explicitly erase this by a statement in your BASIC program. (More on this later.)

It ought to be easy to see how scrolling down instead of up can also be accomplished: just copy each character in a line into the line below.

Similarly, scrolling to the right or to the left is accomplished by moving all the characters in a given column to the right (or left).

The Quad-Directional Scrolling Patch sets up a machine language routine which is CALLED from your BASIC program to scroll a pre-selected area of the screen in any of four directions. In addition, you may specify whether or not you wish the color--that is, the CCI codes which follow each character in the screen refresh memory--to be scrolled as well.

By the way, you can also write a routine to do this in BASIC. (An interesting programming exercise.) But beware of the results. Compared to machine language, BASIC is incredibly slow. The machine language routine will scroll an area of the screen in the wink of an eye, whereas you will have time to whistle "Dixie" waiting for BASIC to do the same thing.

10.2 THE PATCH

There are three parts to the implementation of this scrolling routine: (1) the BASIC program to POKE the required numbers into the top 34 bytes of memory; (2) the BASIC routine to decide on the variables--the size and placement of the screen area to be scrolled, and so on; and (3) the BASIC routine for PRINTing whatever you want in the scrolling area and for CALLing the machine language routine at the appropriate times to scroll that area.

Part (1) should be done at the very beginning of your program, and it need be done only once. Part (2) can be done at any time prior to CALLing the machine language routine, and need be done only once, unless you wish to change the position or direction of the scroll. Part (3) will be used each time you wish to PRINT something in the scrolling area.

First, then, to part (1). Here is the patch itself. (After you have verified that it is working properly with the examples I will give below, you might wish to save it on disk.)

PRG# 10.1

```
0 GOTO 65000:REM POKE IN THE PATCH
1 REM THERE MUST BE A LINE #1 BECAUSE THERE IS A
2 REM "GOTO 1" IN LINE 65140.
3
4 REM NOW SPECIFY X,Y,W,H,C,D AND GOSUB 64000.
5
6 REM X,Y = CURSOR POSITION FOR TOP LEFT OF SCROLLING AREA.
7 REM W = WIDTH OF SCROLLING AREA.
8 REM NOTE: W MUST BE >1 FOR SCROLLING LEFT OR RIGHT.
9 REM H = HEIGHT OF SCROLLING AREA.
10 REM NOTE: H MUST BE >1 FOR SCROLLING UP OR DOWN.
11 REM C = 1 IF THE COLOR IS ALSO TO BE SCROLLED, 0 IF NOT.
12
13 REM D = 1 SCROLL UP.
14 REM D = 2 SCROLL DOWN.
15 REM D = 3 SCROLL RIGHT.
16 REM D = 4 SCROLL LEFT.
```

.
.
.

the rest of your program goes here

.


```

.
.
61997
61998 REM BEFORE ENDING THE PROGRAM, PUT THE TOP
61999 REM OF MEMORY BACK WHERE IT WAS.
62000 GOSUB 65410
62010 AD=ER:Z=TM+34:GOSUB 65400
62019
62020 END
62021
63999 REM QUAD-DIRECTIONAL SCROLLING PATCH
64000 GOSUB 65410
64010 Z=28672+X+X:IF D<>2 THEN Z=Z+128*Y:IF D=3 THEN Z=Z+W+W-(2-C)
64015 IF D=2 THEN Z=Z+128*(Y+H-1)
64020 AD=TM+2:GOSUB 65400
64030 POKE TM+5,H+(D<3):POKE TM+7,W*(C+1)+(D>2)*(1+C)
64040 POKE TM+9,128+126*(D=4)-126*(D=3):POKE TM+10,-255*(D=2 OR D=3
)
64050 POKE TM+14,128+126*(D=3)-126*(D=4):POKE TM+15,-255*(D=1 OR D=
4)
64060 POKE TM+19,-(2-C)*(D<>3)-(254+C)*(D=3):POKE TM+20,-255*(D=3)
64070 IF D=1 OR D=4 THEN Z=130+(2*(D=1))-W-W:GOTO 64080
64075 Z=128+2*(D=3)+W+W:IF D=2 THEN Z=256-Z
64080 POKE TM+27,Z:POKE TM+28,-255*(D=2)
64085 RETURN
64999
65000 GOSUB 65410:RESTORE 65010
65010 DATA 33,-1,-1,6,-1,14,-1,17,-1,-1,25
65020 DATA 126,17,-1,-1,25,119,17,-1,-1,25,13
65030 DATA 194,-1,-1,17,-1,-1,25,5,194,-1,-1,201
65040 IF TM>65501 THEN TM=TM-34:GOTO 65080
65050 FOR J=1 TO 34:READ A
65060 IF A=>0 AND A<>PEEK(TM+J) THEN J=34:TM=TM-34
65070 NEXT
65080 RESTORE 65010
65090 FOR J=1 TO 34:READ A:POKE TM+J,A-(A<0):NEXT
65100 Z=TM+1:AD=33283:GOSUB 65400
65110 Z=TM:AD=ER:GOSUB 65400
65120 Z=TM+6:AD=TM+32:GOSUB 65400
65130 Z=TM+8:AD=TM+24:GOSUB 65400
65140 CLEAR 100:GOTO 1
65399
65400 ZZ=INT(Z/256):POKE AD,Z-256*ZZ:POKE AD+1,ZZ:RETURN
65410 ER=32940:TM=256*PEEK(ER+1)+PEEK(ER):RETURN

```

Now for part (2) of implementing the routine, in which you will determine the variables to be used. Let X and Y be the cursor co-ordinates of the top left character position of the scrolling area. (See Figure 10.4.) Let H be the height, or number of lines, in the scrolling area. (NOTE: H must be greater than 1 for scrolling up or down.) Let W be the width, or the number of characters per line, of the scrolling area. (NOTE: W must be greater than 1 for scrolling right or left.) Let C=1 if you wish the CCI codes to be scrolled with the characters, and C=0 if you do not. Finally, let D=1 for scrolling up, D=2 for scrolling down, D=3 for scrolling right,

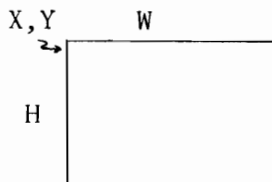


FIG. 10.4

and D=4 for scrolling left.

After setting all these variables, GOSUB 64000 to give them to the machine language routine. For example:

```
0 GOTO 65000
1 X=16:Y=8:W=32:H=16:C=1
2 D=1:GOSUB 64000
.
.
.
```

Now the scrolling routine is set up and ready to go. Each time your BASIC program encounters a CALL statement, the designated area will be scrolled.

10.3 SCROLLING UP

How do we make use of this nifty device? First of all, we note that many different parts of a program may wish to PRINT messages in the scrolling area, so we'll set up a subroutine to position the cursor and then return. But in addition we want to keep track of which line of the scrolling area we're on in order to PRINT each succeeding message one line down in the scrolling area and to scroll the whole area once the bottom line has been reached.

The top line of the area is at cursor position X,Y, and the bottom line is at cursor position X,Y+H-1. If we initialize a variable, L, to keep track of what line we are on, our subroutine might look something like this (for convenience, since the program will probably have to GOSUB to it often, we'll give it a small number):

```
3 L=Y-1:REM L STARTS ONE LINE ABOVE TOP OF SCROLLING AREA
4 GOTO 100:REM MAIN PROGRAM STARTS AT 100
5 REM SCROLLING ROUTINE
6 L=L+1:IF L<Y+H THEN PLOT 3,X,L:RETURN
7 ZZ=CALL(0):PLOT 3,X,Y+H-1:PRINT SPC(W)""
8 PLOT 3,X,Y+H-1:RETURN
.
.
.
100 REM THE PROGRAM
.
.
```


Line 7 CALLs the machine language routine which scrolls everything up. But remember, the last line is left on the screen and must be explicitly erased. (The CALL function returns a value for ZZ. That doesn't concern us at all, except that it might change the value of ZZ, so we should not use that variable elsewhere in the program.) Each time you wish to PRINT a message in the scrolling area, simply GOSUB 6 first. For example:

```

100 PLOT 12
110 FOR J=1 TO 50
120   GOSUB 6:PRINT "TEST LINE #"J
130 NEXT
140 GOSUB 6
150 GOSUB 6:PRINT "THAT'S ALL"
160 GOTO 62000:REM  END

```

By the way, line 140 is there merely in order to skip a line in the scrolling area. An easier way to do this is to have another line, say:

```

5 GOSUB 6

```

Now if you GOSUB 6 you will move down to the next line before printing, but if you GOSUB 5 you will move down two lines.

10.4 SCROLLING DOWN

Scrolling down merely involves a bit of fiddling.

```

0 GOTO 65000
1 X=16:Y=8:W=32:H=16:C=1
2 D=2:GOSUB 64000
3 L=Y+H:REM  L BEGINS ONE LINE BELOW BOTTOM OF SCROLLING AREA
4 GOTO 100:REM  TO MAIN PROGRAM
5 GOSUB 6
6 L=L-1:IF L=>Y THEN PLOT 3,X,L:RETURN
7 ZZ=CALL(0):PLOT 3,X,Y:PRINT SPC(W)""
8 PLOT 3,X,Y:RETURN
.
.
.

```

10.5 SCROLLING RIGHT AND LEFT

If you wish, you may scroll right or left to make an electronic billboard effect using only one line of text. (If the text is to be PRINTed in double height characters, then you must specify the height of the scrolling area as H=2. And remember that the bottom half of a double height character is printed on odd numbered lines.) In this case we want to PRINT not one line

at a time, but only one character at a time. This can be handled in BASIC by the MID\$ function.

```
0 GOTO 65000
1 X=16:Y=8:W=32:H=1:C=1
2 D=4:GOSUB 64000:REM SCROLL LEFT
3 GOTO 100
4
6 ZZ=CALL(0):PLOT 3,X+W-1,Y,32,26:RETURN
99
100 M$(1)="THIS IS A TEST OF THE SCROLLING PATCH.  "
110 M$(2)="SCROLL LEFT FOR ELECTRONIC BILLBOARD.  "
119
120 PLOT 15,6,6,12:REM SET UP
129
130 FOR J=1 TO 2
139
140   FOR K=1 TO LEN(M$(J))
150     GOSUB 6:PRINT MID$(M$(J),K,1)
160   NEXT
169
170 NEXT
179
180 GOTO 62000:REM END
```

In this example it was not necessary to keep track of where in the scrolling area a character was, since the message begins at the far right side and scrolls immediately and continuously.

Let's make a slight alteration for double height characters. Line 120 becomes:

```
120 PLOT 14,6,6,12:REM SET UP; DOUBLE HEIGHT
```

And H must equal 2, since there are two lines to be scrolled (= one double height line). So,

```
1 X=16:Y=8:W=32:H=2
```

The machine language routine is instructed to scroll two lines, at Y=8 and Y=9. That's just what we want, since the double height characters will be PRINTed on those two lines. Notice that even though Y=8 and the subroutine at 6 positions the cursor at Y=8, the bottom of the double height characters will be printed at Y=9. We could not specify Y=9 in line 1, for then the machine language scrolling routine would take its instructions to mean scrolling the two lines at Y=9 and Y=10. Try it and see for yourself.

The electronic billboard would look better if only that cursor could be got out of the way. Why not use the blind cursor?

```
0 GOTO 65000
1 X=16:Y=8:W=32:H=2:C=1
2 D=4:GOSUB 64000
3 GOTO 100
4
6 ZZ=CALL(0):PLOT 3,128:REM BLIND CURSOR, DOUBLE HEIGHT
```



```

7 PLOT X+W-1,Y,1:REM X,Y; CCI CODE = RED
8 PLOT 32:REM ERASE LAST CHARACTER
9 PLOT 3,128,X+W-1,Y,1:REM SET UP BLIND CURSOR FOR NEXT LETTER
10 RETURN
99
100 M$(1)="ELECTRONIC BILLBOARD IN BLIND CURSOR"
109
110 FOR J=1 TO 32
120   M$(2)=M$(2)+". "
130 NEXT
139
140 M$(3)="EAT AT JOE'S"
150 M$(4)=M$(2)
159
160 PLOT 6,6,12,14:REM SET UP
169
170 FOR J=1 TO 4
179
180   FOR K=1 TO LEN(M$(J))
190     GOSUB 6:PLOT ASC(MID$(M$(J),K,1))
200   NEXT
209
210 NEXT
219
220 PLOT 27,27:REM EXIT BLIND CURSOR
229
230 GOTO 62000:REM END

```

Of course, the same thing can be done with regular height characters by specifying H=1 and introducing the blind cursor mode with PLOT 3,127.

The cursor can be effectively hidden by setting the background color to white. The following program uses the vertical mode to print columns of the hatch character in random colors.

```

0 GOTO 65000
1 X=8:Y=16:W=32:H=8:C=1
2 D=4:GOSUB 64000
3 L=X-1:REM L BEGINS ONE SPACE LEFT OF THE SCROLLING AREA
4 GOTO 100
5
6 L=L+1:IF L<X+W THEN PLOT 3,L,Y:RETURN
7 ZZ=CALL(0):PLOT 3,X+W-1,Y:PRINT SPC(H)""
8 PLOT 3,X+W-1,Y:RETURN
99
100 PLOT 15,29:REM SMALL CHARACTERS; FLAG OFF
110 PLOT 6,56,12:REM CLEAR SCREEN IN WHITE
120 PLOT 27,10:REM VERTICAL MODE
129
130 FOR J=1 TO 100
140   GOSUB 6
149
150   FOR K=1 TO H
160     PLOT 6,64*RND(1):REM RANDOM COLORS
170     PLOT 96:REM HATCH CHARACTER

```



```

180 NEXT
189
190 NEXT
199
200 PLOT 27,11:REM SCROLL MODE
210 PLOT 6,2:REM BACK TO A MANAGEABLE COLOR
219
220 GOTO 62000:REM END

```

You will see that in this case the second statement in line 7 serves no purpose, since each successive line fills up an entire column and replaces whatever was there before. There is a need to erase a column (or line, if you are scrolling up or down) only if it is not known whether the next column (or line) will print its message into all the character positions.

It is just as easy to scroll right as it is to scroll left. The following program PRINTs random characters (ASCII 32-127) in random colors. This time, however, let's set C=0 so that the CCI codes do not scroll.

```

0 GOTO 65000
1 X=8:Y=16:W=32:H=8:C=0
2 D=3:GOSUB 64000:REM SCROLL LEFT
3 L=X+W:REM L STARTS ONE SPACE TO RIGHT OF SCROLLING AREA
4 GOTO 100
5
6 L=L-1:IF L=>X THEN PLOT 3,L,Y:RETURN
7 ZZ=CALL(0):PLOT 3,X,Y:RETURN
99
100 PLOT 15
110 PLOT 6,56,12:REM CLEAR SCREEN IN WHITE
120 PLOT 27,10:REM VERTICAL MODE
129
130 FOR J=1 TO 100
140 GOSUB 6
149
150 FOR K=1 TO H
160 PLOT 6,64*RND(1):REM RANDOM COLORS
170 PLOT 96*RND(1)+32:REM RANDOM ASCII CHARACTER
180 NEXT
189
190 NEXT
199
200 PLOT 27,11:REM SCROLL MODE
210 PLOT 6,2:REM BACK TO A MANAGEABLE COLOR
219
220 GOTO 62000:REM END

```

If we use the electronic billboard idea but scroll right instead of left, we can have messages roll by going backwards, printed backwards. This might be a cute game. First, load the patch. Next ask the player for the message to be printed. Then ask the speed at which the message should be printed. The other player then watches as the message goes by backwards and must identify what it says.


```

0 GOTO 65000
1 CLEAR 300
2 X=16:Y=8:W=32:H=1:C=1
3 D=3:GOSUB 64000
4
5 REM PROGRAM 10.2
6 REM BACKWARDS TEXT GUESSING GAME
10 PLOT 15,6,6,12:REM SET UP
18
19 REM DRAW A BORDER AROUND THE MESSAGE DISPLAY AREA
20 PLOT 2,31,97,242,31,90,97,90,97,97,31,97,255
28
29 REM GET THE MESSAGE
30 PLOT 8,6,2:INPUT"MESSAGE: ";M$:IF M$="END" THEN 62000:REM END
37
38 REM ADD 32 SPACES TO THE MESSAGE SO THAT IT
39 REM WILL SCROLL OFF ENTIRELY
40 FOR J=1 TO 32
50 M$=M$+" "
60 NEXT
67
68 REM HOME CURSOR AND ERASE 2 LINES, JUST IN CASE THE
69 REM PREVIOUS MESSAGE WAS MORE THAN 1 LINE LONG
70 PLOT 8,11,10,11
78
79 REM NOW GET THE SPEED
80 PLOT 8:INPUT "SPEED (1=FAST, 100=SLOW)? ";SP$
90 SP=VAL(SP$):IF LEFT$(SP$,1)="E" THEN 62000:REM END
98
99 REM CHECK FOR ERRORS
100 IF SP<1 OR SP>100 THEN PLOT 28,11:GOTO 80
108
109 REM NOW PRINT THE MESSAGE IN RED FOR GOOD CONTRAST
110 PLOT 6,1
119
120 FOR J=1 TO LEN(M$)
130 ZZ=CALL(0)
140 PLOT 3,X,Y
150 PRINT MID$(M$,J,1)
158
159 REM DELAY
160 FOR K=1 TO SP:NEXT
170 NEXT
179
180 PLOT 6,2,3,0,1
190 INPUT "SAME MESSAGE OR A NEW ONE (S/N)? ";A$
200 IF LEFT$(A$,1)="S" THEN 70
210 IF LEFT$(A$,1)="E" THEN 62000:REM END
220 IF LEFT$(A$,1)<>"N" THEN PLOT 28,11:GOTO 190
230 PLOT 8,11,10,11:GOTO 30

```


11. MISCELLANEOUS NOTES

11.1 A NOTE ON MENU PROGRAMS

Your MENU programs introduce both you and others to the existence of a number of other programs. Many programs, once they have run, return to the MENU. Your MENU programs, then, ought to look good and at the same time perform some housekeeping chores. Some programs leave the display in page mode, or end up with the FLAG on. Or a program which has set the display to page mode or has turned the FLAG on might be interrupted by the user who wishes to return to the MENU. In such cases the MENU program itself can be the main housecleaner, although it is of course good programming practice to have each of your programs take nothing for granted.

The following program is a "blank" menu. You might save it (or an altered version) on disk for easy preparation of a MENU when you need one. As usual, color changes made from the keyboard are indicated with brackets: [].

```
5 REM PROGRAM 11.1
6 REM BLANK MENU
9
10 CLEAR 2000
20 DIM F$(27):REM MAXIMUM OF 27 FILES
29
30 FOR J=1 TO 52:D$=D$+" ":NEXT
33
34 REM ENTER THE FILE NAMES STARTING AT LINE 40. FOR EXAMPLES:
35
36 REM F$(1)="SUPER-HYPER REAL TIME SPACE OPERA"
37 REM F$(2)="SANSKRIT — LESSON II"
38 REM ETC.
39
499
500 PLOT 15:REM SMALL CHARACTERS; POSSIBLE BLINK OFF
510 PLOT 6,3,29:REM SET COLORS; FLAG OFF
520 PLOT 12,27,11:REM ERASE SCREEN; SET TO SCROLL MODE
528
529 REM FIND OUT HOW MANY FILES THERE ARE
530 K=0
539
540 FOR J=27 TO 1 STEP -1
550 IF F$(J)<>" " THEN K=J:J=1
560 NEXT
569
570 IF K<6 THEN PLOT 14:REM DOUBLE HEIGHT IF THERE'S ROOM
579
580 PRINT TAB(27)"M E N U[grn]":PRINT
589
590 IF K=0 THEN PRINT:PRINT "NO FILES AVAILABLE!":END
597
598 REM PRINT THE FILE NAMES
599
```



```

600 FOR J=1 TO K
610 PRINT TAB(2)LEFT$(F$(J)+D$,52)J
618
619 REM SPACE BETWEEN SELECTIONS IF THERE'S ROOM
620 IF K<15 THEN PRINT
630 NEXT
638
639 REM SKIP A LINE AFTER ALL FILE NAMES HAVE BEEN PRINTED
640 IF K>14 THEN PRINT
649
650 PRINT TAB(2)"[cyn]ENTER YOUR SELECTION NUMBER AND ";
660 INPUT "PRESS [wht]RETURN [cyn]..... [red]";A$
670 F=VAL(A$)
678
679 REM CHECK FOR INPUT ERROR
680 IF F<1 OR F<>INT(F) OR F>K THEN PLOT 28,11:GOTO 650
686
687 REM REPRINT CHOSEN FILE NAME IN RED
688
689 REM FIRST GET Y CO-ORDINATE OF FILE NAME
690 IF K<6 THEN Y=4*F+1
700 IF K>5 AND K<15 THEN Y=2*F
710 IF K>14 THEN Y=F+1
719
720 PLOT 3,2,Y:PRINT LEFT$(F$(F)+D$,52)F
729
730 PLOT 3,64,0:REM HIDE THE CURSOR
739
740 ON F GOTO 801,802,803,804,805,806,807,808,809,810
750 ON F-10 GOTO 811,812,813,814,815,816,817,818,819,820
760 ON F-20 GOTO 821,822,823,824,825,826,827
799
800 REM ENTER LOAD AND RUN INSTRUCTIONS BEGINNING AT 801

```

11.2 A NOTE ON DISPLAYING TEXT

Type in the program below and RUN it. (The arrow in line 40 is pointing to an extra space which must not be omitted.)

```

5 REM PROGRAM 11.2
6 REM DISPLAYING TEXT
10 PLOT 15:REM REGULAR HEIGHT
20 PLOT 6,33:REM RED ON BLUE
30 PLOT 12:REM ERASE PAGE
39
40 PRINT "WHEN YOU HAVE A LOT OF TEXT TO DISPLAY--INSTRUCTIONS TO
THE"
50 PRINT "USER, FOR EXAMPLE--YOU WILL WANT TO PRINT THE TEXT IN SUCH
A"
60 PRINT "WAY THAT THE DISPLAY IS BOTH PLEASING AND LEGIBLE. USING

```



```

THE"
70 PRINT "DOUBLE HEIGHT CHARACTERS (PLOT 14) CERTAINLY MAKES THE WO
RDS"
80 PRINT "EASIER TO SEE--AN IMPORTANT CONSIDERATION IF THE USER'S E
YE-"
90 PRINT "SIGHT IS VERY BAD. ON THE OTHER HAND, THE COMPACTNESS OF
THE"
100 PRINT "SMALLER CHARACTERS MAKES THEM LESS 'RAGGED' THAN THE DOU
BLE-"
110 PRINT "HEIGHT CHARACTERS."

```

The text has been right and left justified. A nice touch if you can manage it. And it's not very hard: add an extra space here or there; change the wording slightly; leave only one space between sentences instead of the usual two; and so on.

Yet the display could be improved. Legibility can be enhanced by leaving a blank line between each line of text. Add lines 45, 55, 65 and so on, all of which are only PRINT statements.

With backgrounds other than black the display looks much better when the text is surrounded by margins. Add line 35 PRINT to give a margin at the top. Then add a space at the beginning of each line of text. Or else add lines 39, 49, 59 and so on, all of which are

```
PRINT " ";
```

The choice of colors is also important. Experiment with line 20. In particular, try using blue on black and black on blue. The blue-black combination can be quite "soft", but for that very reason it should be avoided; legibility requires a higher contrast.

12. MISCELLANEOUS PROGRAMS

The following programs further illustrate various techniques described in this book. I hope they will provide you with ideas you can use in your own work.

12.1 VARIATIONS ON A THEME

```
5 REM PROGRAM 12.1
6 REM CONVERGING BORDERS
10 DEF FN R(X)=INT(48*RND(1))+1
20 PLOT 15,6,6,12,27,24
29
30 FOR J=1 TO 15
40   GOSUB 100
50 NEXT
59
60 GOTO 30
98
99
100 PLOT 6, FN R(X)
109
110 PLOT 3,J,J
120 PRINT SPC(64-J-J) ""
129
130 PLOT 3,J,31-J
140 PRINT SPC(64-J-J) ""
149
150 PLOT 27,10
160 PLOT 3,J,J
170 PRINT SPC(32-J-J) ""
179
180 PLOT 3,63-J,J
190 PRINT SPC(32-J-J) ""
199
200 PLOT 27,24
210 RETURN
```

VARIATION #1: Change lines 120 and 140 to PLOT 11.

VARIATION #2: Same as #1, but also change lines 170 and 190 to PRINT SPC(32) "".

VARIATION #3: Same as #2, but change line 30 to FOR J=0 TO 31.

VARIATION #4: Same as #3, but add line 155 PLOT 6, FN R(X).

VARIATION #5: Same as #4, but change lines 170 and 190 to PRINT SPC(64) "".

12.2 SIMULATING A RADAR SCOPE

Let us write a program to simulate a radar scope. First we must be able to plot a circle. Since the computer deals with radians and not degrees, we might as well leave everything in radians. (If you wish to make the necessary changes to degrees, you will have to do so explicitly. One degree is equal to $3.1415926/180$ radians.)

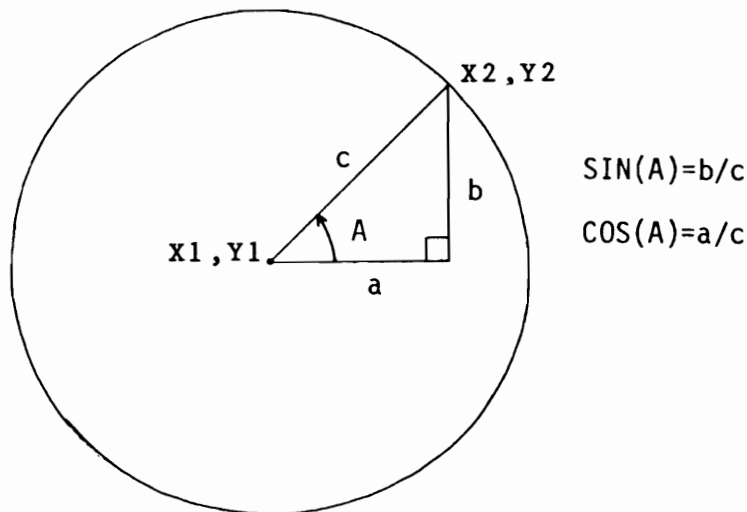


FIG. 12.1

According to Figure 12.1, if the center of a circle is at co-ordinates $X1, Y1$, then any point on the circumference will be at $X2, Y2$, where $X2=X1+C*\text{COS}(A)$ and $Y2=Y1+C*\text{SIN}(A)$. Let's plot a circle in the center of the screen: $X1=63, Y1=63$. Since there are $2*3.1415926$ radians in a circle, all we need is a loop of the form:

```
FOR J=0 TO 2*3.1415926
  plot a point
NEXT
```

However, since J will range from 0 to just a little over 6, we would be PLOTting only 6 or 7 points on the circumference, so we'll have to STEP that loop down. A STEP size of .05 will yield almost all the points of the circumference. A smaller STEP size would be more accurate, but the smaller the STEP size, the slower the radar beam will make its sweep.

We can now write a program just for a circle.

```
5 REM PROGRAM 12.2
6 REM PLOTTING A CIRCLE
9
10 PLOT 6,6,12:REM SET UP
19
20 R=2*3.1415926:REM NUMBER OF RADIAN IN A CIRCLE
30 S=.05:REM STEP SIZE
```



```

40 C=30:REM CIRCLE RADIUS
49
50 PLOT 2:REM GENERAL PLOT MODE
59
60 FOR J=0 TO R STEP S
70 PLOT 63+C*COS(J),63+C*SIN(J)
80 NEXT
89
90 PLOT 255

```

This program will plot some sort of oval, and not a circle as planned. What went wrong? There's nothing wrong with the mathematics. It is the computer's screen which is the culprit. Each plot block is not a square, but a rectangle. So, while there are the same number of plot blocks horizontally as there are vertically, the horizontal dimension is greater. When we PLOT each successive plot block of the circle's circumference, we move a greater distance in the horizontal direction than we do in the vertical direction. We'll have to squeeze the oval into a circle by increasing the factor by which we multiply SIN(J) (or by decreasing the factor of COS(J)). I have found 40 to be about right, but you might want to try other numbers. If we define F=40, then line 70 will become:

```

70 PLOT 63+C*COS(J),63+F*SIN(J)

```

To have a radar scope you have to have a beam which sweeps the circle. We can use the vector plot for that. Each time through the loop, we'll draw a vector from the center of the circle to the computed point on its circumference. We need only add to the above program this line:

```

65 PLOT 253,63,63,242

```

But this does not give rise to a single beam. To get a single beam sweeping around, we could try something like Figure 12.2. Well? Try it. Or, we could set the FLAG on so that PLOTting a vector a second time will erase it. (Figure 12.3.) The second method seems to me a bit easier, so let's try that.

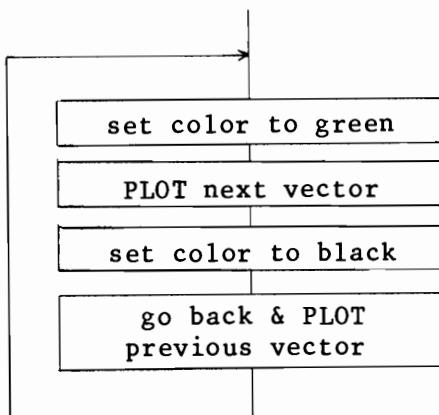


FIG. 12.2

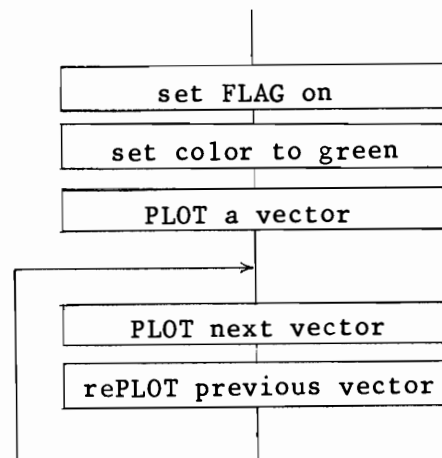


FIG. 12.3


```

5 REM PROGRAM 12.3
6 REM SIMULATING A RADAR SCOPE
9
10 PLOT 6,2,12,30:REM COLOR=GREEN; CLEAR SCREEN; FLAG ON
19
20 R=2*3.1415926:REM RADIANS IN A CIRCLE
30 S=.05:REM STEP SIZE
40 C=30:REM CIRCLE RADIUS
50 F=40:REM SCALING FACTOR FOR Y CO-ORDINATE
59
60 X1=63:Y1=63:REM INITIALIZE A "PREVIOUS VECTOR" OF LENGTH=0
69
70 PLOT 2:REM GENERAL PLOT MODE
79
80 FOR J=0 TO R STEP S
90 PLOT 253:REM POINT PLOT
100 PLOT 63,63:REM CENTER OF CIRCLE
110 PLOT 242:REM VECTOR PLOT
118
119 REM GET NEXT VECTOR'S POINTS & PLOT IT
120 X2=63+C*COS(J):Y2=63+F*SIN(J)
130 PLOT X2,Y2
138
139 REM ERASE PREVIOUS VECTOR
140 PLOT 253,63,63,242,X1,Y1
148
149 REM NEW VECTOR BECOMES "PREVIOUS VECTOR"
150 X1=X2:Y1=Y2
160 NEXT
169
170 GOTO 80:REM BACK FOR ANOTHER SWEEP

```

12.3 RADAR SCOPE SIMULATION USING INCREMENTAL VECTOR PLOT

Just for the sake of exercise, let's take a different approach to the simulation of a radar beam. This time we'll use the incremental vector plot submode. We are handicapped at the outset, however, because we cannot easily increment the vector in such a way as to have its end points mark out the circumference of a circle. So we'll invent a rectangular screen for the radar scope and have the vector follow the perimeter of that rectangle.

A spot near the center of the screen will be the axis for the sweep. The beam will follow the four sides of the rectangle back around to the starting point, where it will begin the process again. A sweep of about 60 units per side ought to look good. And just for the sake of nicety, we'll PLOT a rectangular border to outline the sweep area. We'll have to PLOT the sweep vector (radar beam) in four separate quadrants, because the X and Y increments will change as the beam moves.

We will PLOT the first vector from the center point to the upper left corner of the sweep area. As the beam sweeps around, one point of the vector remains constant at the center of the screen. Since that is the X1,Y1

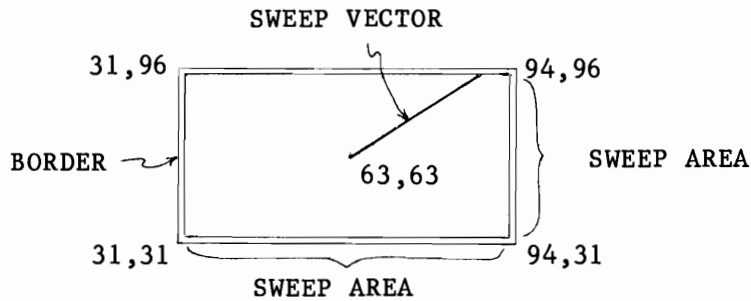


FIG. 12.4

co-ordinate of the beam, the increment numbers we use in the incremental vector plot will have to indicate that there is no change to these co-ordinates. But we cannot use zero for the increment number because, remember, if either the four most significant bits or the four least significant bits are zero, the vector will not be drawn. (See Section 6.10.) How can we indicate "no change" without using zero? Use binary 11. That is a non-zero number, and it, too, indicates no change, so it is just what we need. Since it is the X_1, Y_1 co-ordinate which is to be unincremented, we must use binary 11 for X_1 or for Y_1 or for both. But notice that if we make both X_1 and Y_1 equal to binary 11, the resulting increment number will be at least 240 (more, depending on the values for X_2 and Y_2). But the numbers from 240-255 are instructions for entering other plot submodes (or for exiting the plot mode altogether). If we make the two Y_1 bits equal to binary 11, and the two X_1 bits equal to binary 00, then the increment number will not be too large, and we still satisfy the requirements of having the four most significant bits not all equal to zero, yet indicating no change.

How should the vector be incremented? Figure 12.5 divides the sweep

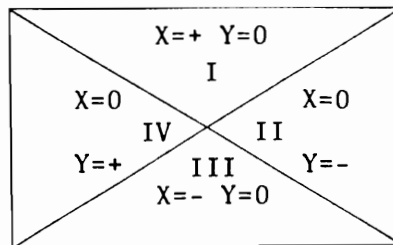


FIG. 12.5

area into four quadrants. If the beam moves clockwise, then in quadrant I its X_2 co-ordinate must be incremented while its Y_2 co-ordinate is unchanged. The increment number must accordingly be:

DECIMAL:	128	64	32	16	8	4	2	1	
BIT:	7	6	5	4	3	2	1	0	
	0	0	1	1	1	0	0	0	$32+16+8 = 56$
	X_1		Y_1		X_2		Y_2		

A little thought yields the increment numbers for the other three quadrants:

II	49
III	52
IV	50

In order to create a single beam which sweeps around, let us PLOT a vector, then increment it backwards and PLOT the previous vector once again. If the FLAG is on, then the previous vector will disappear. Then we will increment the vector without drawing it, and continue the cycle anew. The decrement numbers work out to be:

QUADRANT	I	52
	II	50
	III	56
	IV	49

And the numbers for incrementing without drawing will be:

QUADRANT	I	8
	II	1
	III	4
	IV	2

Now we can write the program.

```
5 REM PROGRAM 12.4
6 REM RADAR SCOPE USING INCREMENTAL VECTOR PLOT
9
10 PLOT 6,6,12:REM COLOR=CYAN; CLEAR SCREEN
20 PLOT 3,64,0:REM HIDE CURSOR
30 PLOT 30:REM FLAG ON
39
40 PLOT 2:REM GENERAL PLOT MODE
48
49 REM DRAW BORDER AROUND SWEEP AREA
50 PLOT 31,96,242,94,96,94,31,31,31,31,95
51 REM THAT LAST NUMBER MUST BE 95. IF THE VECTOR
52 REM WENT BACK TO 96, THE FIRST PLOT BLOCK OF THE
53 REM BORDER WOULD BE ERASED BECAUSE THE FLAG IS ON.
59
60 PLOT 255,6,2:REM SET COLOR TO GREEN
68
69 REM DRAW FIRST VECTOR
70 PLOT 2,63,63,242,32,95
79
80 PLOT 240:REM INCREMENTAL VECTOR PLOT SUBMODE
86
87 REM NOW THE 4 QUADRANT SWEEP
88
89 REM QUADRANT I
90 FOR J=0 TO 60
100 PLOT 56:REM INCREMENT & PLOT
```



```

110 PLOT 52:REM DECREMENT & PLOT
120 PLOT 8:REM INCREMENT WITHOUT DRAWING
130 NEXT
138
139 REM QUADRANT II
140 FOR J=0 TO 62
150 PLOT 49,50,1
160 NEXT
168
169 REM QUADRANT III
170 FOR J=0 TO 60
180 PLOT 52,56,4
190 NEXT
198
199 REM QUADRANT IV
200 FOR J=0 TO 62
210 PLOT 50,49,2
220 NEXT
229
230 GOTO 90:REM CONTINUE

```

12.4 SOME CIRCLES

```

5 REM PROGRAM 12.5
6 REM CIRCLES ABOUT THE CORNERS
9
10 R=2*3.1415926:REM RADIANS IN A CIRCLE
19
20 PLOT 6,6,12,3,64,0:REM SET UP
29
30 C=5*RND(1)+1:REM C IS RADIUS OF CIRCLE
39
40 PLOT 6,7*RND(1)+1:REM RANDOM COLOR
49
50 FOR X=0 TO 127 STEP 127
59
60 FOR Y=0 TO 127 STEP 127
70 GOSUB 200:REM DRAW A CIRCLE AROUND A CORNER
80 NEXT
89
90 NEXT
99
100 C=C+10*RND(1)+1:REM INCREASE RADIUS OF NEXT CIRCLE
110 IF C<60 THEN 40:REM MORE CIRCLES UNTIL THEY INTERSECT
119
120 FOR T=1 TO 2000:NEXT:REM PAUSE
129
130 GOTO 20:REM START OVER
197
198
199 REM SUBROUTINE TO DRAW A CIRCLE

```



```

200 PLOT 2
209
210 FOR J=0 TO R STEP .1
220   X1=X+C*COS(J):IF X1<0 OR X1>127 THEN 250
230   Y1=Y+1.3*C*SIN(J):IF Y1<0 OR Y1>127 THEN 250
240   PLOT X1,Y1
250 NEXT
259
260 PLOT 255:RETURN

```

12.5 AN ANIMATED JOKE

The following program uses some of the computer's special characters obtained with shift-A, shift-B, and so on. They are given in the program listing as lower case letters.

```

5 REM PROGRAM 12.6
6 REM AN ANIMATED JOKE
8
9 REM SET UP
10 GOSUB 100
18
19 REM PAUSE
20 GOSUB 200
28
29 REM BIG BIRD
30 GOSUB 300
38
39 REM PAUSE
40 GOSUB 200
48
49 REM LITTLE BIRDS
50 GOSUB 500
58
59 REM FIRST MESSAGE
60 GOSUB 700
68
69 REM SECOND MESSAGE
70 GOSUB 800
78
79 REM PAUSE
80 GOSUB 200
88
89
90 PLOT 6,2:END
95
96
97 REM ***** SUBROUTINES *****
98
99 REM —— SET UP
100 PLOT 15,30,22,12:REM SMALL CHARACTERS; CLEAR SCREEN IN CYAN

```



```

110 PLOT 18,3,0,31,11:REM ERASE BOTTOM LINE IN GREEN
120 PLOT 22,29,17:REM BG=CYAN, FG=RED; FLAG OFF
130 RETURN
198
199 REM ----- PAUSE
200 PLOT 8:FOR T=1 TO 2000:NEXT:RETURN
298
299 REM ----- BIG BIRD
300 X=40
309
310 FOR Y=0 TO 29
320 PLOT 3,X,Y:PRINT "      iidii      "
330 PLOT 3,X,Y:PRINT "    iiggfdfggii    "
339
340 GOSUB 470:REM SLIGHT PAUSE
349
350 PLOT 3,X,Y:PRINT "iiiiggggggdggggggiii"
360 PLOT 3,X,Y:PRINT "iggggggggdgggggggi"
370 PLOT 3,X,Y:PRINT "ffggiiiiidiiiiiggff"
380 PLOT 3,X,Y:PRINT "eeffggiiidiiiggffee"
398
390 GOSUB 470:REM SLIGHT PAUSE
399
400 PLOT 3,X,Y:PRINT "ggggggggdgggggggg"
410 PLOT 3,X,Y:PRINT "iiiiggfffdfffgggiii"
419
420 PLOT 28,11
430 X=X-1
440 NEXT
449
450 PLOT 3,X+9,30:PRINT "d"
459
460 RETURN
468
469 REM ----- SLIGHT PAUSE
470 FOR T=1 TO 30:NEXT:RETURN
498
499 REM ----- LITTLE BIRDS
500 FOR J=29 TO 3 STEP -2
510 X=J:R=30*RND(1)
519
520 FOR Y=2 TO 31
530 IF Y<30 THEN PLOT 3,X,Y:PRINT "ggggsgggg"
540 PLOT 3,X-3,Y-2:PRINT " igfsfgi"
550 IF Y<30 THEN PLOT 3,X,Y:PRINT " igfsfgi"
559
560 FOR T=1 TO R:NEXT
569
570 PLOT 3,X-3,Y-2:PRINT "ggggsgggg"
580 IF Y<30 THEN PLOT 3,X,Y:PRINT "ggggsgggg"
590 PLOT 3,X-3,Y-2:PRINT "efgisigfe"
600 IF Y<30 THEN PLOT 3,X,Y:PRINT "efgisigfe"
610 PLOT 3,X-3,Y-2:PRINT "ggggsgggg"
620 IF Y<30 THEN PLOT 3,X,Y:PRINT SPC(9)""

```



```

630     PLOT 3,X-3,Y-2:PRINT SPC(9)""
640     IF Y=29 THEN PLOT 3,X+5,30:PRINT "a"
649
650     X=X+1
660     NEXT
669
670     PLOT 3,X+1,30:PRINT "a"
679
680     FOR T=1 TO 2500*RND(1):NEXT
689
690 NEXT:RETURN
698
699 REM —— FIRST MESSAGE
700 PLOT 16:REM FOREGROUND=BLACK
710 PLOT 2,37,15,242,34,25,255
719
720 PLOT 3,10,24:PRINT "YOU MAY WONDER WHY I'VE"
730 PLOT 3,8,25:PRINT "BROUGHT YOU ALL HERE TODAY"
739
740 GOSUB 200:REM PAUSE
749
750 GOSUB 930:REM ERASE MESSAGE
759
760 RETURN
798
799 REM —— SECOND MESSAGE
800 PLOT 2,85,17,242,83,24,253,100,20,242,90,30
810 PLOT 253,110,20,242,110,29,255
820 PLOT 14:REM DOUBLE HEIGHT CHARACTERS
829
830 PLOT 3,40,24:PRINT "YES"
840 PLOT 3,43,22:PRINT "YES!"
849
850 PLOT 15:REM REGULAR HEIGHT
860 PLOT 30:REM FLAG ON FOR QUADRUPLE HEIGHT CHARACTERS TO FOLLOW
870 PLOT 3,53,23
880 PLOT 124,126,32,123,101,32,99,102
890 PLOT 3,53,24
900 PLOT 98,32,32,108,127,32,103,100
910 PLOT 29:REM FLAG OFF
919
920 GOSUB 200:REM PAUSE
928
929 REM ERASE MESSAGE
930 FOR Y=22 TO 29:PLOT 3,0,Y,11:NEXT
939
940 RETURN

```


12.6 CHESS PIECES USING THE CHARACTER PLOT SUBMODE

Program 6.11, you will have discovered, draws a chess board. Now let's add some chess pieces. (You are left to your own devices to write the chess playing program!)

Each of the board squares drawn by Program 6.11 is 12 plot blocks wide and 16 plot blocks high; this corresponds to 6 character positions wide and 4 character positions high. The board appears on the display at the cursor positions shown in Figure 12.6.

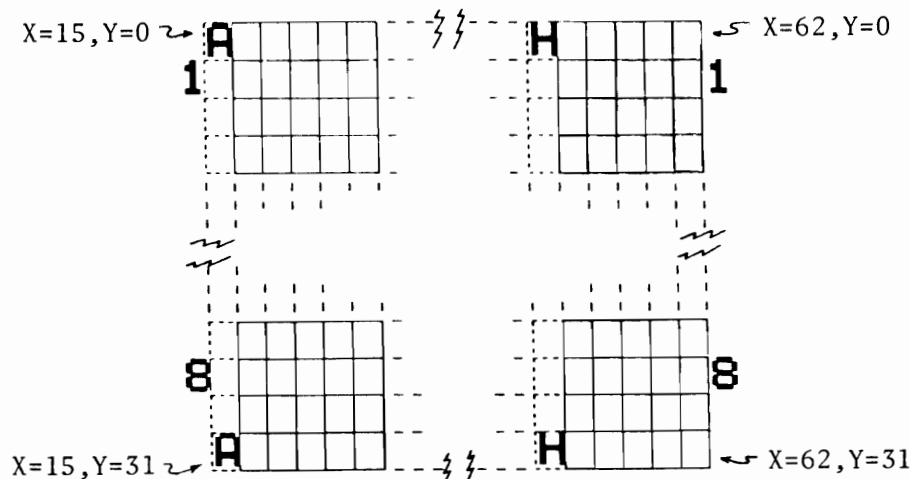


FIG. 12.6

The player must be able to refer to board squares, and, instead of the English, or descriptive, system, an algebraic notation can easily be used here. The columns are labeled "A" through "H", and the rows are numbered 1 through 8 (beginning at White's side of the board). We can use the columns on both vertical edges of the board for the numbers. Where can the letters go? They can be printed in the squares themselves: "A"-"H" will be printed in the top left character position of the top row of squares and in the bottom left character position of the bottom row of squares. In cyan squares, the letters will be printed in blue on cyan, and in blue squares the letters will be printed in cyan on blue. The numbers on each side of the board might be in white.

The chess pieces will be drawn (using the character plot submode) on the right 5x4 block of characters within each square. This will allow them to be drawn and erased as necessary without interfering with the board square reference letters. It also provides ample spacing between the pieces for the sake of good looks. The pieces themselves are shown in Figure 12.7, and the 20 character plot numbers for each of the six kinds of pieces are given in Table 12.1

Notice that except for the pawn, each chess piece has one or more character positions which are completely filled in. This is impossible to do using the character plot submode. The number 255 in Table 12.1 is used to represent such a completely filled in character. When the chess piece drawing routine in your program encounters the 255, it will exit the character plot submode, print a space in the appropriate color, and then return to the character plot submode to continue drawing the piece.

Instead of black and white pieces, it is suggested that black and red

PAWN					KNIGHT				
0	0	0	0	0	0	128	204	206	8
0	192	206	0	0	236	239	255	255	15
0	50	63	2	0	16	200	254	63	1
64	100	103	68	0	102	119	119	103	6

BISHOP					ROOK				
0	200	134	12	0	136	128	136	128	8
0	169	191	9	0	247	254	255	254	7
0	240	255	0	0	0	255	255	15	0
102	119	119	103	6	118	119	119	119	6

QUEEN					KING				
132	140	140	140	4	0	68	78	4	0
0	217	223	9	0	128	238	239	142	0
0	240	255	0	0	0	240	255	0	0
102	119	119	103	6	102	119	119	103	6

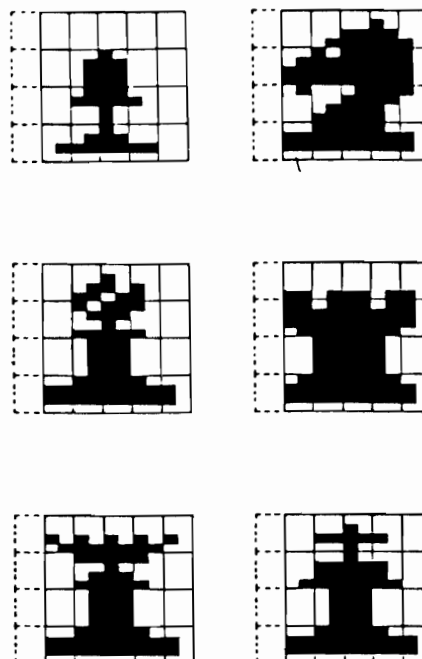


TABLE 12.1

FIG. 12.7

be used. These colors provide sufficient contrast and definition against either cyan or blue backgrounds (squares). You might wish to experiment with other colors, but the choices are rather limited.

```

5 REM PROGRAM 12.7
6 REM CHESS PIECES USING CHARACTER PLOT
9
10 DIM P(6,20):REM 20 DATA FOR DRAWING EACH OF 6 PIECES
98
99 REM DRAW THE BOARD
100 GOSUB 1000
108
109 REM GET DATA FOR THE 6 PIECES
110 GOSUB 1500
118
119 REM GET PLAYER'S CHOICE OF COLORS
120 GOSUB 1300
128
129 REM DRAW THE PIECES
130 GOSUB 1600
138
139
140 PLOT 27,11:REM FOR CONVENIENCE, RETURN TO SCROLL MODE
150 END
151
996
997 REM ***** SUBROUTINES *****
998
999 REM ——— DRAW THE BOARD

```



```

1000 PLOT 6,6,12,15,3,64,0:REM SET UP
1008
1009 REM DRAW THE WHOLE BOARD IN CYAN
1010 PLOT 2,250,30,0,125,247
1019
1020 FOR Y=0 TO 63:PLOT 34:NEXT
1029
1030 PLOT 2,255
1038
1039 REM NOW ADD THE BLUE SQUARES
1040 PLOT 6,4
1050 S=0
1060 PLOT 2
1069
1070 FOR Y=0 TO 112 STEP 16
1079
1080     FOR X=30 TO 114 STEP 24
1090         PLOT 250,X+12*S,Y,X+12*S+11,247
1100         FOR J=1 TO 7:PLOT 34:NEXT
1110         PLOT 2
1120     NEXT
1129
1130     S=1+(S=1)
1140 NEXT
1149
1150 PLOT 255
1158
1159 REM ADD REFERENCE LETTERS
1160 C=52:REM START WITH BLUE ON CYAN, BUT THIS WILL
1161     REM TOGGLE IMMEDIATELY TO CYAN ON BLUE
1169
1170 FOR Y=0 TO 31 STEP 31
1180     L=64:REM ASCII VALUES, BEGINNING WITH 1 LESS THAN "A"
1190     C=52+14*(C=52):REM TOGGLE COLORS
1199
1200     FOR X=15 TO 57 STEP 6
1210         C=52+14*(C=52):REM TOGGLE COLORS
1220         PLOT 6,C
1230         L=L+1:REM NEXT LETTER
1240         PLOT 3,X,Y,L
1250     NEXT
1259
1260 NEXT
1269
1270 RETURN
1297
1298 REM —— GET PLAYER'S CHOICE OF COLORS AND
1299 REM     ADD REFERENCE NUMBERS.
1300 PLOT 6,3,8:REM COLOR = YELLOW; HOME CURSOR
1310 PRINT "DO YOU WISH TO"
1320 INPUT "PLAY WHITE? ";A$
1330 A$=LEFT$(A$,1)
1340 IF A$="Y" THEN N1=56:N2=49:S=-1:GOTO 1370
1350 IF A$="N" THEN N1=49:N2=56:S=1:GOTO 1370

```



```

1360 PLOT 28:PRINT SPC(15)"":PLOT 28:GOTO 1320
1369
1370 PLOT 6,7:REM WHITE ON BLACK
1380 PLOT 27,10:REM VERTICAL MODE
1389
1390 FOR X=14 TO 63 STEP 49
1400   PLOT 3,X,1
1409
1410   FOR N=N1 TO N2 STEP S:REM ASCII VALUES OF NUMBERS 1-8
1420     PLOT N,32,32,32
1430   NEXT
1439
1440 NEXT
1449
1450 PLOT 27,24,8:REM PAGE MODE; HOME CURSOR
1458
1459 REM ERASE INPUT
1460 PRINT SPC(14)"":PRINT SPC(14)""
1469
1470 W=(A$="Y"):REM W WILL NOW BE A FLAG = 0 IF PLAYER
1471           REM IS PLAYING BLACK AND = -1 IF PLAYER
1472           REM IS PLAYING WHITE
1473
1480 RETURN
1498
1499 REM ----- GET DATA FOR THE 6 KINDS OF PIECES
1500 FOR P=1 TO 6
1509
1510   FOR N=1 TO 20:READ P(P,N):NEXT
1519
1520 NEXT
1529
1530 RETURN
1536
1537 REM ----- THE DATA
1538
1539 REM PAWN
1540 DATA 0, 0, 0, 0, 0
1542 DATA 0,192,206, 0, 0
1544 DATA 0, 50, 63, 2, 0
1546 DATA 64,100,103, 68, 0
1548
1549 REM KNIGHT
1550 DATA 0,128,204,206, 8
1552 DATA 236,239,255,255, 15
1554 DATA 16,200,254, 63, 1
1556 DATA 102,119,119,103, 6
1558
1559 REM BISHOP
1560 DATA 0,200,134, 12, 0
1562 DATA 0,169,191, 9, 0
1564 DATA 0,240,255, 0, 0
1566 DATA 102,119,119,103, 6
1568

```



```

1569 REM  ROOK
1570 DATA 136,128,136,128, 8
1572 DATA 247,254,255,254, 7
1574 DATA 0,255,255, 15, 0
1576 DATA 118,119,119,119, 6
1578
1579 REM  QUEEN
1580 DATA 132,140,140,140, 4
1582 DATA 0,217,223, 9, 0
1584 DATA 0,240,255, 0, 0
1586 DATA 102,119,119,103, 6
1588
1589 REM  KING
1590 DATA 0, 68, 78, 4, 0
1592 DATA 128,238,239,142, 0
1594 DATA 0,240,255, 0, 0
1596 DATA 102,119,119,103, 6
1598
1599 REM  ----- DRAW BEGINNING POSITION
1600 BG=22:FG=17+W:REM  SET COLORS FOR ROOK AT TOP LEFT.
1610 PLOT 3,16,0:REM  POSITION CURSOR AT TOP LEFT OF
1611          REM  BOARD AND ONE SPACE IN FROM EDGE
1612
1620 P=4:GOSUB 2000:REM  DRAW A ROOK
1630 P=2:GOSUB 2000:REM  KNIGHT
1640 P=3:GOSUB 2000:REM  BISHOP
1650 P=6+W:GOSUB 2000:REM  QUEEN OR KING
1660 P=5-W:GOSUB 2000:REM  KING OR QUEEN
1668
1669 REM  NOW THE RIGHT BISHOP, KNIGHT & ROOK
1670 P=3:GOSUB 2000:P=2:GOSUB 2000:P=4:GOSUB 2000
1678
1679 REM  NOW FOR A ROW OF PAWNS AT THE TOP
1680 BG=20:PLOT 3,16,4
1690 P=1:FOR K=1 TO 8:GOSUB 2000:NEXT
1698
1699 REM  NOW FOR THE PIECES AT THE BOTTOM
1700 BG=22:FG=16-W:REM  SET COLORS FOR BOTTOM ROW OF PAWNS
1710 PLOT 3,16,24:REM  POSITION CURSOR
1718
1719 REM  DRAW BOTTOM ROW OF PAWNS
1720 P=1:FOR K=1 TO 8:GOSUB 2000:NEXT
1729
1730 BG=20:PLOT 3,16,28
1740 P=4:GOSUB 2000:REM  ROOK
1750 P=2:GOSUB 2000:REM  KNIGHT
1760 P=3:GOSUB 2000:REM  BISHOP
1770 P=6+W:GOSUB 2000:REM  QUEEN OR KING
1780 P=5-W:GOSUB 2000:REM  KING OR QUEEN
1788
1789 REM  NOW FOR THE RIGHT 3 PIECES
1790 P=3:GOSUB 2000:P=2:GOSUB 2000:P=4:GOSUB 2000
1799
1800 PLOT 6,2,8

```



```

1810 RETURN
1995
1996 REM —— DRAW A PIECE
1997
1998 REM BG AND FG ASSUMED ASSIGNED.
1999 REM P=PIECE #(1-6). CURSOR IS ASSUMED POSITIONED.
2000 PLOT 30,BG,29,FG:REM SET COLORS
2009
2010 FOR Y=0 TO 3
2020 PLOT 2,254:REM CHARACTER PLOT
2029
2030 FOR X=1 TO 5
2040 PLOT P(P,5*Y+X):IF P(P,5*Y+X)<255 THEN 2080
2049
2050 PLOT 30,FG,32:REM PRINT A SPACE IN THE FG COLOR
2060 PLOT BG,29:REM RETURN ORIGINAL BG COLOR; FLAG OFF
2070 PLOT 2,254:REM RETURN TO CHARACTER PLOT
2079
2080 NEXT:PLOT 255
2089
2090 PLOT 26,26,26,26,26,10:REM CURSOR BACK & DOWN
2100 NEXT
2108
2109 REM POSITION CURSOR FOR NEXT PIECE
2110 FOR J=1 TO 4:PLOT 25,28:NEXT:PLOT 25,25
2120 BG=22+2*(BG=22):REM TOGGLE BG COLOR... CYAN—BLUE
2129
2130 RETURN

```

12.7 A REAL TIME LUNAR LANDER GAME

I think it must be a Law of Nature that a programmer must, at some point, write a lunar lander game. Excellent speed, accuracy and graphics can be implemented on the computer in assembly language. But something can be done in BASIC, too. The program presented here is not very sophisticated: the physics of lunar landers, for example, is given up in favor of speed.

The lunar surface is drawn in vertical mode using the hatch character. The space ship is drawn in incremental point plot submode with the FLAG on, so that PLOTting it twice will draw it and then erase it. Keyboard input is prevented from being echoed on the screen by making use of a no-echo patch. When information is entered from the keyboard, it is stored in location 33278. The program periodically checks that location to see if anything is there. If there is something there, it decides whether it is valid or not and calculates the new position and velocity accordingly. If the player manages to land the ship at the lunar base with a very small velocity (less than 8.5 ft/sec), a special treat is provided by having a little moon walker (a short line drawn in incremental point plot mode) "walk" from the space ship to the highest point on the surface and raise a little green flag. After each attempt at landing, statistics are provided, and the player is rated according to various factors such as initial speed, final speed, whether landing was made at the moon base, and the degree of

difficulty chosen. The degree of difficulty determines the initial horizontal and vertical velocities; the height of the lunar surface from the bottom of the screen (the higher the surface, the effectively less time there is to slow the ship and land); the amount of fuel initially available; and finally the placement of the lunar base on the surface: in easy simulations, the lunar base is level with the surface of the moon, so that the ship can land partly on the base and partly on the surface; in the medium level of difficulty, one side of the base is level with the lunar surface, but the other is not. This therefore requires a bit more skill to land at the base without tipping over. In the advanced version, the lunar base is not level with the lunar surface on either of its sides: the ship must land precisely on the base.

The program is quite large, requiring a 16K machine. But if you remove all unnecessary spaces and REMark statements, and if you put as many statements on a line as you can (without destroying the destination of GOSUB's and GOTO's), you might squeeze it into an 8K machine. As usual, information which you should enter from the keyboard is indicated in brackets: [].

```

5 REM PROGRAM 12.8
6 REM REAL TIME LUNAR LANDER
7
8 REM BY DAVID B. SUITS, 12 ANNO LUNAE
9
10 GOTO 63000:REM POKE IN THE NO-ECHO PATCH
99
100 DIM Y(127):REM HEIGHT OF LUNAR SURFACE FOR EACH OF
101 REM THE 127 HORIZONTAL PLOT POSITIONS.
110 KB=33278:REM LOCATION OF KEYBOARD CHARACTER.
120 KF=33247:REM LOCATION OF KEYBOARD FLAG.
130 MAX=-1.5:REM MAX SAFE LANDING VELOCITY
138
139 REM GIVE INSTRUCTIONS
140 GOSUB 8000
148
149 REM GET DEGREE OF DIFFICULTY AND SET UP LUNAR SURFACE
150 GOSUB 7000
157
158 REM DETERMINE INITIAL FUEL; PLOT THE SHIP; PRINT
159 REM AMOUNT OF FUEL REMAINING
160 GOSUB 6000
168
169 REM CALCULATE EFFECTS OF ANY RETRO-FIRE
170 GOSUB 2000
178
179 REM CALCULATE NEW CO-ORDINATES
180 GOSUB 1000
188
189 REM IF SHIP IS STILL ABOVE GROUND, CONTINUE
190 IF Y>Y(X-1)+V2 AND Y>Y(X+1)+V2 THEN 170
198
199 REM SHIP HAS LANDED (OR CRASHED)
200 V1=V:Y=Y1
210 GOSUB 1050:REM ERASE IT

```



```

218
219 REM SET Y=HIGHEST POINT UNDER SHIP + 4
220 Y=Y(X-1):IF Y(X)>Y(X-1) THEN Y=Y(X)
230 IF Y(X+1)>Y(X) THEN Y=Y(X+1)
240 Y=INT(Y+4):X=INT(X)
250 GOSUB 1050:PLOT 255:REM DRAW SHIP AT FINAL SPOT
259
260 PLOT 29,22:PRINT "FUEL=";:PLOT 14,19:PRINT F:PLOT 3,64,0
268
269 REM VELOCITY TOO GREAT FOR SAFE LANDING?
270 IF V1<MAX-.0001 THEN TD=0:GOSUB 3000:GOTO 410
278
279 REM NO. BUT IS SHIP STABLE ON SURFACE?
280 IF Y(X-1)<>Y(X+1) THEN TD=0:GOSUB 4000:GOTO 410
288
289 REM SAFE LANDING!
290 PLOT 3,18,1:PRINT "*** TOUCHDOWN ***":TD=1
298
299 REM SEE IF ANY PART OF SHIP IS ON MOON BASE
300 FLG=-1
309
310 FOR J=-1 TO 1
319
320 FOR K=MB*2 TO MB*2+3
330 IF X+J=K THEN FLG=0:REM YES
340 NEXT
349
350 NEXT
359
360 IF FLG THEN PLOT 15,3,15,3:PRINT "(BUT YOU MISSED THE MOON BASE
):GOTO 410

369
370 PLOT 6,3
380 PLOT 3,15,3:PRINT "SUCCESSFUL LANDING AT MOON BASE"
388
389 REM TURN OFF BLINK AT MOON BASE
390 PLOT 15,30,6,33,3,MB,31-INT(Y(MB*2)/4),101,101,6,3,29,3,64,0
396
397 REM IF SHIP LANDS AT MOON BASE WITH VELOCITY <.85 DOWN, A
398 REM LITTLE MOON WALKER GOES TO THE HIGHEST POINT ON THE
399 REM SURFACE AND RAISES A FLAG
400 IF V1=>-.85 THEN GOSUB 5000
408
409 REM GIVE STATISTICS
410 GOSUB 600
418
419 REM PLAY AGAIN?
420 INPUT"[mag]AGAIN? ";A$:A$=LEFT$(A$,1)
430 IF A$="Y" OR A$="O" THEN 150
440 IF A$<>"N" THEN PLOT 28,11:GOTO 420
448
449 REM PUT TOP OF MEMORY BACK WHERE IT WAS
450 TM=256*PEEK(32941)+PEEK(32940)+7

```



```

460 POKE 32941,INT(TM/256):POKE 32940,TM-256*INT(TM/256)
469
470 END
471
596
597 REM ***** SUBROUTINES *****
598
599 REM ——— GIVE STATISTICS
600 PLOT 3,64,5:REM HIDE CURSOR
610 FOR J=1 TO 1000:NEXT:REM PAUSE
620 PLOT 15
629
630 PRINT "[cyn]INITIAL VELOCITY:[mag]"TAB(22)-VI*10" FT/SEC"
640 PRINT "[cyn]VELOCITY AT ";:IF TD THEN PRINT "TOUCHDOWN";:GOTO 6
60
650 PRINT "IMPACT";
660 PRINT ":[mag]"TAB(22)-VI*10" FT/SEC"
670 PRINT "[cyn]FUEL EXPENDED:[mag]"TAB(22)FI-F" UNITS"
678
679 REM DETERMINE RATING
680 DM=X-MB*2:IF DM>-2 AND DM<5 THEN DM=0:REM DM=DIST. FROM MB
690 RA=127/(ABS(DM)+1)+VI*10*(F/FI)+ABS(HI)-VI
700 RA=RA+RA*FLG*(V1=>-.85):RA=-INT(RA*D+.5)*TD
710 PRINT
720 PRINT "[mag]YOUR RATING IS:[wht]"TAB(22)RA
730 TR=TR+RA:LNDG=LNDG+1:AVG=INT(TR/LNDG)
740 PRINT
750 PRINT "[yel]AVERAGE RATING"
760 PRINT "AFTER"LNDG" LANDING";
770 IF LNDG>1 THEN PRINT "S";
780 PRINT ":[red]"TAB(22)AVG
790 PRINT
800 RETURN
998
999 REM ——— CALCULATE NEW X,Y AND DRAW SHIP
1000 GOSUB 1050:REM ERASE IT AT OLD LOCATION FIRST
1010 X=X+H:IF X>125 THEN X=125
1020 IF X<2 THEN X=2
1030 Y=Y+V:IF Y>127 THEN Y=127
1040 Y1=Y:V=V-.31:V2=ABS(V)+4
1049
1050 PLOT 253,X,Y,251,17,5,128,8
1059
1060 RETURN
1998
1999 REM ——— RETRO-FIRE?
2000 IF F=0 THEN RETURN:REM NO FUEL LEFT
2010 A=PEEK(KB)-48:IF A=39 THEN A=.2:GOTO 2100
2020 IF A=21 THEN A=-.2:GOTO 2100
2030 IF A<1 OR A>9 THEN RETURN:REM NONSENSE KEY WAS STRUCK,
2031 REM OR ELSE NO KEY AT ALL.
2039 REM VERTICAL RETRO
2040 IF F-A<0 THEN A=F
2048

```



```

2049 REM DON'T PLOT RETRO IF SHIP IS TOO CLOSE TO GROUND
2050 F=F-A:V=V+.2*A-.3:IF Y<Y(X)+8 THEN 2070
2058
2059 REM DRAW YELLOW LINE UNDER SHIP FOR RETRO
2060 PLOT 255,6,3,2,X,Y-5,251,17,16,34,2
2069
2070 GOTO 2120
2098
2099 REM LATERAL RETRO
2100 F=F-1:H=H+A
2108
2109 REM DRAW YELLOW LINE FOR FIRE
2110 PLOT 255,6,3,2,X-5*A+(A>0),Y-1,251,8,4,8
2118
2119 REM PRINT FUEL REMAINING; BLINK IF FUEL <50
2120 PLOT 255,6,6:PRINT "FUEL=";:PLOT 14,6,3-64*(F<50)
2130 PRINT F" ":PLOT 15,6,6,3,64,0,2
2140 POKE KB,0:REM CLEAR KEYBOARD LOCATION
2150 RETURN
2998
2999 REM ----- PLOT AN EXPLOSION AT SITE OF CRASH
3000 PLOT 29,19
3010 PLOT 2
3018
3019 REM IS CRASH SITE TOO CLOSE TO EDGE OF SCREEN?
3020 IF X<6 THEN G=1:GOTO 3200
3030 IF X>121 THEN G=-1:GOTO 3200
3039
3040 PLOT X,Y+13,X-2,Y+7,X+2,Y+20,X,Y,242,X+5,Y+20,255
3050 GOSUB 3400:PLOT 16
3060 PLOT 2,X,Y,242,X-3,Y+15,255
3070 PLOT 23
3080 PLOT 2,X-1,Y+20,X+3,Y+25,X,Y,242,X+2,Y+11
3090 PLOT 253,X,Y,X-1,Y+30,X,Y,242,X+1,Y+8,255
3100 GOSUB 3400:GOSUB 3400:PLOT 16
3110 PLOT 2,X,Y,242,X+2,Y+11,253,X,Y,242,X+5,Y+20
3120 PLOT 253,X,Y,242,X+1,Y+8,253,X,Y,242,X-1,Y+30
3130 GOSUB 3400
3140 PLOT 253,X,Y+13,X-2,Y+7,X+2,Y+20,X,Y,242,X-2,Y+21
3150 PLOT 253,X,Y,242,X+3,Y+7,253,X-1,Y+20,X+3,Y+25,X+1,Y
3160 PLOT X-1,Y
3170 PLOT 255
3180 RETURN
3198
3199 REM EXPLOSION AT SIDE OF SCREEN
3200 PLOT X,Y,242,X,Y+4,253,X,Y,242,X+2*G,Y+2
3210 PLOT 253,X,Y,X+5*G,Y+6
3220 PLOT 253,X,Y,X-G,Y+2,255
3229
3230 FOR K=1 TO 3:GOSUB 3400:NEXT
3239
3240 PLOT 23
3250 PLOT 2,X+G,Y+6,X-G,Y+6,X,Y,242,X+5*G,Y+13
3260 PLOT 253,X,Y,242,X,Y+20

```



```

3270 PLOT 253,X+4*G,Y+8,X+6*G,Y+10,X+10*G,Y+13,255
3279
3280 GOSUB 3400:PLOT 16
3289
3290 PLOT 2,X,Y,242,X+5*G,Y+6,253,X,Y,242,X+2*G,Y+2
3300 PLOT 253,X,Y,242,X,Y+4,253,X,Y,242,X,Y+20
3310 PLOT 253,X+10*G,Y+13,X+6*G,Y+10,X-G,Y+6
3319
3320 GOSUB 3400:GOSUB 3400
3329
3330 PLOT X+G,Y+6,X,Y,242,X-G,Y+2
3340 PLOT 253,X+4*G,Y+8,X,Y,242,X+5*G,Y+13
3350 PLOT 253,X-G,Y
3360 PLOT 255
3370 RETURN
3398
3399 REM ----- PAUSE
3400 FOR J=1 TO INT(50*RND(1))+20:NEXT:RETURN
3997
3998 REM ----- VELOCITY AT TOUCHDOWN OK, BUT SHIP IS
3999 REM          UNSTABLE AND FALLS
4000 PLOT 3,64,0,6,6,30,2,251
4010 Q=1+2*(Y(X-1)<Y(X+1))
4020 IF Q=1 THEN 4200
4028
4029 REM SHIP FALLS TO LEFT
4030 PLOT 4,32,32,2,16,16,81:GOSUB 4300
4110 PLOT 160,32,36:GOSUB 4300
4120 PLOT 72,24,9:GOSUB 4300
4130 PLOT 64,102,89,144,1:GOSUB 4300
4140 PLOT 32,164,70,25:GOSUB 4300
4150 PLOT 255
4159
4160 PLOT 29,18
4170 X=INT(X+Q+Q):Y=Y-4
4180 GOTO 3000:REM FOR AN EXPLOSION
4198
4199 REM SHIP FALLS TO RIGHT
4200 PLOT 18,4,32,32,2:GOSUB 4300
4210 PLOT 24:GOSUB 4300
4220 PLOT 132,20,5:GOSUB 4300
4230 PLOT 128,170,149,80,1:GOSUB 4300
4240 PLOT 32,104:GOSUB 4300
4249
4250 GOTO 4150
4298
4299 REM ----- SLIGHT PAUSE
4300 FOR J=1 TO 100:NEXT:RETURN
4998
4999 REM ----- PLOT THE MOON WALKER
5000 PLOT 3,64,0
5010 PLOT 6,6
5020 PLOT 30:REM FLAG ON
5030 G=1:IF X>HP THEN G=-1:REM HP IS X OF HIGHEST POINT OF SURFACE

```



```

5040 PLOT 2
5049
5050 FOR J=X+2*G TO HP STEP G
5059
5060   FOR K=1 TO 2
5070     PLOT J,Y(J)+1,251,34,253
5080     FOR Z=1 TO 20:NEXT
5090   NEXT
5097
5098   REM IF AT BOTTOM OR TOP OF CLIFF,
5099   REM GO UP OR DOWN THE SIDE OF IT
5100   IF Y(J)<>Y(J+G) AND (J<>HP) THEN GOSUB 5300
5110 NEXT
5119
5120 FOR Z=1 TO 150:NEXT
5128
5129 REM RAISE THE FLAG
5130 PLOT 255,29
5138
5139 REM FLAG POLE
5140 PLOT 2,HP,Y(HP)+1,242,HP,Y(HP)+9,255
5148
5149 REM THE FLAG
5150 PLOT 3,127:REM BLIND CURSOR
5160 PLOT (HP+1)/2,30-INT((Y(HP)+1)/4):REM X,Y
5170 PLOT 2:REM CCI CODE
5180 PLOT 96,96:REM 2 HATCH CHARACTERS
5190 PLOT 27,27:REM EXIT BLIND CURSOR
5200 RETURN
5298
5299 REM ----- MOON WALKER GOES UP OR DOWN THE SIDE OF A CLIFF
5300 Q=-(Y(J)>Y(J+G))
5309
5310 FOR S=1 TO 4
5319
5320   FOR T=1 TO 2
5330     PLOT J+Q*G,Y(J)+1+S*(1-2*Q)
5340     FOR Z=1 TO 50:NEXT
5350   NEXT
5359
5360   FOR Z=1 TO 10:NEXT
5370 NEXT
5379
5380 RETURN
5997
5998 REM DETERMINE INITIAL FUEL, INITIAL VELOCITY,
5999 REM AND INITIAL STARTING CO-ORDINATES
6000 F=200+(10-(D-1))*20:REM F IS ALWAYS FUEL REMAINING
6010 FI=F:REM REMEMBER INITIAL FUEL
6019
6020 X=INT(95*RND(1))+25:Y=127:REM STARTING X,Y FOR SHIP
6029
6030 H=.5*D*RND(1)-(.3*D):REM HORIZONTAL VELOCITY
6040 HI=H:REM REMEMBER INITIAL HORIZONTAL VELOCITY

```



```

6049
6050 V=-.5*D*RND(1)-(.5*D):REM VERTICAL VELOCITY
6060 VI=V:REM REMEMBER INITIAL VERTICAL VELOCITY
6069
6070 POKE KB,0:POKE KF,31:REM TURN OFF ECHO
6078
6079 REM DRAW THE SHIP
6080 PLOT 8,10
6090 PLOT 2:GOSUB 1050
6099
6100 GOSUB 2120:REM PRINT FUEL REMAINING
6109
6110 RETURN
6998
6999 REM ----- GET DEGREE OF DIFFICULTY & SET UP LUNAR SURFACE
7000 PRINT
7010 INPUT "[grn]WHICH WOULD YOU LIKE: EASY, MEDIUM OR ADVANCED? ";
D$
7020 D$=LEFT$(D$,1)
7030 D=2:IF D$="M" THEN D=5:GOTO 7060
7040 IF D$="A" THEN D=10:GOTO 7060
7050 IF D$<>"E" THEN PLOT 28,11:GOTO 7010
7059
7060 PLOT 12
7070 PRINT TAB(26)"[red]STAND BY"
7080 PLOT 3,64,0
7089
7090 S=INT(7*RND(1)+.5*D):REM SURFACE HEIGHT IS INFLUENCED
7091 REM BY DEGREE OF DIFFICULTY
7092
7100 HP=1:REM HIGHEST POINT ON SURFACE SO FAR
7109
7110 FOR X=0 TO 63 STEP 2:REM 2 COLUMNS AT A TIME
7120 N=S+INT(6*RND(1))-3:REM NEXT COLUMN UP OR DOWN
7128
7129 REM DON'T GO TOO LOW
7130 IF N<INT(.5*D)+1 THEN N=INT(.5*D)+1
7138
7139 REM DON'T GO TOO HIGH
7140 IF N>INT(.5*D)+7 THEN N=INT(.5*D)+7
7147
7148 REM TRANSLATE HEIGHT OF TWO COLUMNS
7149 REM INTO PLOT CO-ORDINATES
7150 FOR J=0 TO 3:Y(2*X+J)=(N-1)*4+3:NEXT
7158
7159 REM KEEP TRACK OF HIGHEST POINT
7160 IF Y(2*X+3)>Y(HP) THEN HP=2*X+3
7169
7170 S=N:REM NEXT POINT OF SURFACE STARTS FROM PRESENT HEIGHT
7180 NEXT
7187
7188 REM NOW FIND A SUITABLE LOCATION FOR THE MOON BASE.
7189 REM MB IS CURSOR X OF MOON BASE. N IS A COUNTER.
7190 MB=INT((30+2*D)*RND(1)+(20-2*D)):N=0

```



```

7197
7198 REM LUNAR SURFACE IS CREATED TWO COLUMNS AT A TIME,
7199 REM SO MAKE SURE MB BEGINS AT AN EVEN X
7200 IF MB/2<>INT(MB/2) THEN MB=MB+1
7208
7209 REM DON'T GO TOO FAR RIGHT WITH MB
7210 MB=MB+2:IF MB>60 THEN MB=2
7220 N=N+1
7228
7229 REM DON'T PUT MB TOO CLOSE TO HIGHEST POINT
7230 IF ABS(MB*2-HP)<4 THEN 7210
7238
7239 REM IF NO SUITABLE SPOT FOR MB IS FOUND, START OVER
7240 IF N>31 THEN 7090
7249
7250 M=MB*2:REM M IS X PLOT CO-ORDINATE OF MOON BASE
7257
7258 REM CHECK APPROPRIATE DEGREE OF DIFFICULTY FOR
7259 REM PROPER ENVIRONMENT FOR MOON BASE
7260 FLG=0:ON INT(D/5)+1 GOSUB 7500,7520,7540
7268
7269 REM IF BASE NOT IN GOOD SPOT, MOVE RIGHT & TRY AGAIN
7270 IF FLG THEN 7210
7278
7279 REM BASE POSITIONED PROPERLY; NOW DRAW EVERYTHING
7280 PLOT 6,7,12
7290 PLOT 27,10:REM VERTICAL MODE
7299
7300 FOR X=0 TO 63
7310 N=INT(Y(X*2)/4):REM HEIGHT OF SURFACE TRANSLATED FROM
7311 REM PLOT BLOCKS TO CHARACTER POSITIONS
7312
7320 PLOT 3,X,31-N:REM POSITION CURSOR N CHARACTER POSITIONS
7321 REM ABOVE BOTTOM OF SCREEN
7322
7330 FOR K=0 TO N:PLOT 96:NEXT:REM DRAW A COLUMN
7339
7340 NEXT
7348
7349 REM DRAW BLINKING MOON BASE, FLAG ON
7350 PLOT 27,11,6,97,30
7360 PLOT 3,MB,31-INT(Y(MB*2)/4)
7370 PLOT 101,101
7379
7380 PLOT 6,6
7390 RETURN
7495
7496 REM —— CHECK POSITION OF MOON BASE
7497
7498 REM EASY—BOTH EDGES OF TOP SURFACE OF BASE MUST BE
7499 REM LEVEL WITH LUNAR SURFACE
7500 FLG=(Y(M)<>Y(M-1)) OR (Y(M)<>Y(M+4))
7510 RETURN
7517

```



```

7518 REM MEDIUM--TOP OF BASE MUST BE LEVEL WITH LUNAR
7519 REM SURFACE ON ONE SIDE BUT NOT THE OTHER
7520 FLG=((Y(M)=Y(M-1)) AND (Y(M)=Y(M+4))) OR ((Y(M)<>Y(M-1)) AND (
Y(M)<>Y(M+4)))
7530 RETURN
7537
7538 REM ADVANCED--BOTH EDGES OF TOP OF MOON BASE MUST BE
7539 REM UNEQUAL WITH LUNAR SURFACE
7540 FLG=(Y(M)=Y(M-1)) OR (Y(M)=Y(M+4))
7550 RETURN
7998
7999 REM ----- INSTRUCTIONS
8000 PLOT 29,14,6,6,12
8010 PRINT TAB(20)"LUNAR LANDING SIMULATION"
8020 PLOT 15
8030 PLOT 6,2
8040 PRINT
8050 PRINT "YOU ARE THE PILOT OF A SPACE SHIP LANDING ON THE MOON."
8060 PRINT
8070 PRINT "UNFORTUNATELY, YOUR LANDING COMPUTER WENT BERZERK AND Y
OU HAD TO"
8080 PLOT 28
8090 PRINT "SHOOT IT. NOW YOU MUST LAND THE SHIP MANUALLY. THE I
DEA IS TO"
8100 PLOT 28
8110 PRINT "LAND (SAFELY!) AS CLOSE AS YOU CAN TO THE MOON BASE."
8120 PRINT
8130 PRINT "YOU MAY FIRE YOUR RETRO-ROCKETS BY PRESSING A NUMBER FR
OM [wht]1[grn] TO [wht]9[grn]"
8140 PLOT 28
8150 PRINT "(NO NEED TO HIT RETURN). [wht]1[grn] GIVES YOU THE MI
NIMUM THRUST, WHILE"
8160 PLOT 28
8170 PRINT "[wht]9[grn] GIVES YOU THE MAXIMUM. YOU CAN MOVE HORIZ
ONTALLY BY PRESSING"
8180 PLOT 28
8190 PRINT "[wht]W[grn] OR [wht]E[grn], GIVING YOU A SMALL BURST OF
FIRE ON THE [wht]WEST[grn] OR THE [wht]EAST[grn]"
8200 PLOT 28
8210 PRINT "SIDE OF YOUR SHIP."
8220 PRINT
8230 PRINT "IF YOUR VELOCITY AS YOU HIT THE SURFACE IS GREATER THAN
[red]"MAX*-10" FT/S[grn],"
8240 PLOT 28
8250 PRINT "YOU CRASH. SINCE YOUR COMPUTER IS OUT OF ORDER, YOU
WON'T KNOW"
8260 PLOT 28
8270 PRINT "YOUR EXACT VELOCITY DURING DESCENT. YOU'LL HAVE TO EYE
-BALL IT."
8280 PRINT "YOUR FUEL GAUGE IS, HOWEVER, STILL FUNCTIONING. THE NU
MBERS YOU"
8290 PLOT 28
8300 PRINT "TYPE IN FOR RETRO-FIRES ARE THE NUMBERS OF FUEL UNITS W
HICH WILL"

```



```

8310 PLOT 28
8320 PRINT "BE FIRED -- EXCEPT FOR [wht]W[grn] AND [wht]E[grn], WHI
CH USE 1 UNIT EACH TO FIRE."
8330 PRINT
8340 PRINT "THE AMOUNT OF FUEL YOU BEGIN WITH, AS WELL AS YOUR INIT
IAL ALTI-"
8350 PLOT 28
8360 PRINT "TUDE AND VELOCITY, WILL BE DETERMINED BY HOW DIFFICULT
A SIMULA-"
8270 PLOT 28
8280 PRINT "TION YOU WISH TO DO."
8390 RETURN
62998
62999 REM ----- BEN BARLOW'S NO-ECHO PATCH
63000 RESTORE 63000:DATA 245,175,50,255,129,241,201
63010 TM=256*PEEK(32941)+PEEK(32940)-7
63020 FOR X=1 TO 7:READ D:POKE TM+X,D:NEXT
63030 BR=INT(TM/256):POKE 33221,195:POKE 33222,TM-BR*256+1
63040 POKE 33223,BR:POKE 32941,BR:POKE 32940,TM-BR*256
63050 CLEAR 50:GOTO 100
63051
63052 REM FOR NO-ECHO, POKE 33247,31.
63053 REM TO RETURN TO ECHO, POKE 33247,12.
63054 REM (AN INPUT STATEMENT OR THE END
63055 REM OF THE PROGRAM WILL ALSO DO IT.)

```

12.8 EXTRA LARGE CHARACTERS

The following program prints extra large characters according to the chart in Appendix C. The no-echo patch is used so that full cursor control is allowed, which means that 88 possible keyboard inputs are recognized, from HOME through "_", excluding TAB and XMIT. ESCAPE ends the program.

In response to a keyboard input, the program will print the appropriate character in double width and double height (quadruple height if the A7 is on), or else issue the proper cursor or color command. The program prints out instructions for its use.

For the sake of good looks, each extra large character is printed with an additional space to its right (lines 120 and 140); a blank line is left between each row of printed characters(line 150).

Some of the large characters require the use of a special character with the FLAG off. These are listed in Appendix C (and in the DATA statements in lines 600-774) as negative numbers. Line 130 turns the FLAG on or off for each of the four special characters used in printing one extra large character.

```

0 GOTO 63000:REM POKE IN THE NO-ECHO PATCH
1
5 REM PROGRAM 12.9
6 REM EXTRA LARGE CHARACTERS
8

```



```

9 REM GIVE INSTRUCTIONS
10 GOSUB 300
18
19 REM GET THE DATA
20 GOSUB 500
28
29 REM NO ECHO
30 POKE KF,31
38
39 REM DRAW ----- MAIN ROUTINE
40 POKE KB,0
50 A=PEEK(KB):IF A=0 THEN 50
60 IF A<8 OR A>95 THEN 40:REM ILLEGAL INPUT
68
69 REM IF [ESC] IS PRESSED, END THE PROGRAM
70 IF A=27 THEN 9000
78
79 REM CHECK FOR CONTROL CHARACTER
80 A=A-7:REM "HOME" IS SMALLEST ASCII VALUE PERMITTED, NOW = 1
90 IF A<25 THEN GOSUB 200:GOTO 150
93
94 REM IGNORE UNUSED KEYS SUCH AS TAB
95 IF C(A,1)=0 THEN 40
98
99 REM NON-CONTROL KEYS
100 PLOT 28:REM EACH CHARACTER BEGINS AT THE TOP LEFT OF 2X2 ARRAY
109
110 FOR J=1 TO 4:REM FOR EACH OF THE 4 SPECIAL CHARACTERS...
118
119 REM READY FOR 2ND ROW?
120 IF J=3 THEN PLOT 32,26,26,26,10
129
130 PLOT 29*-(C(A,J)<0)+30*-(C(A,J)>0),ABS(C(A,J))
140 NEXT:PLOT 32
146
147 REM IF RIGHT OF SCREEN IS REACHED, THEN DO CR/LF
148 REM (THE COMPUCOLOR KEEPS TRACK OF THE X CO-ORDINATE
149 REM OF THE CURSOR IN LOCATION 33227)
150 IF PEEK(33227)>60 THEN PLOT 13,10,10,10
158
159 REM BACK FOR MORE
160 GOTO 40
196
197 REM ***** SUBROUTINES *****
198
199 REM ----- CONTROL KEYS
200 FOR J=1 TO 4
210 PLOT C(A,J)
220 NEXT
228
229 REM IF [ERASE LINE], THEN 1 MORE "PLOT 10" IS REQUIRED
230 IF A=4 THEN PLOT 10:REM REMEMBER, A=A-7 BY LINE 80
239
240 RETURN

```



```

298
299 REM ----- INSTRUCTIONS
300 PLOT 14
310 PLOT 6,6,29,12:REM SET UP
320 PRINT TAB(22)"EXTRA LARGE CHARACTERS"
330 PRINT:PLOT 15,18
340 PRINT TAB(2)"THIS PROGRAM ALLOWS YOU TO TYPE DOUBLE WIDTH, DOUB
LE HEIGHT"
350 PRINT
360 PRINT TAB(2)"CHARACTERS (WITH A7 OFF), OR DOUBLE WIDTH, QUADRUP
LE HEIGHT"
370 PRINT
380 PRINT TAB(2)"CHARACTERS (WITH A7 ON). "
390 PRINT
400 PRINT
410 PRINT TAB(2)"CURSOR AND COLOR KEYS MAY ALSO BE USED, EXCEPT THA
T 'FG ON'"
420 PRINT
430 PRINT TAB(2)"OR 'BG ON' MUST ALWAYS BE PRESSED BEFORE A COLOR C
HANGE."
440 PRINT
450 PRINT
460 PRINT TAB(2)"PRESS [ESC] TO END THE PROGRAM."
470 PRINT:PRINT:PRINT
480 PLOT 17:PRINT TAB(25)"PLEASE STAND BY"
490 RETURN
498
499 REM ----- GET THE DATA
500 DIM C(88,4):REM 4 DATA FOR EACH OF 88 CHARACTERS OR CONTROLS
510 KB=33278:REM LOCATION OF KEYBOARD CHARACTER
520 KF=33247:REM LOCATION OF KEYBOARD CHARACTER FLAG
529
530 RESTORE 600
539
540 FOR A=1 TO 88
550   FOR B=1 TO 4:READ C(A,B):NEXT
560 NEXT
569
570 PLOT 19,28,11
580 INPUT "PRESS RETURN TO BEGIN...";A$
590 PLOT 12,10:RETURN
596
597 REM ----- DATA FOR EACH OF THE 88 KEYS
598
599 REM HOME
600 DATA 8,0,0,0
601 REM TAB (NOT USED)
602 DATA 0,0,0,0
603 REM CURSOR DOWN
604 DATA 10,10,10,0
605 REM ERASE LINE
606 DATA 13,11,28,11
607 REM ERASE PAGE
608 DATA 12,10,0,0

```



```

609 REM CR/LF
610 DATA 13,10,10,10
611 REM A7 ON
612 DATA 14,0,0,0
613 REM BLINK/A7 OFF
614 DATA 15,0,0,0
615 REM BLACK
616 DATA 16,0,0,0
617 REM RED
618 DATA 17,0,0,0
619 REM GREEN
620 DATA 18,0,0,0
621 REM YELLOW
622 DATA 19,0,0,0
623 REM BLUE
624 DATA 20,0,0,0
625 REM MAGENTA
626 DATA 21,0,0,0
627 REM CYAN
628 DATA 22,0,0,0
629 REM WHITE
630 DATA 23,0,0,0
631 REM XMIT (NOT USED)
632 DATA 0,0,0,0
633 REM CURSOR RIGHT
634 DATA 25,25,25,0
635 REM CURSOR LEFT
636 DATA 26,26,26,0
637 REM ESCAPE (NOT NEEDED)
638 DATA 0,0,0,0
639 REM CURSOR UP
640 DATA 28,28,28,0
641 REM FG ON/FLAG OFF
642 DATA 29,0,0,0
643 REM BG ON/FLAG ON
644 DATA 30,0,0,0
645 REM BLINK ON
646 DATA 31,0,0,0
647 REM SPACE
648 DATA 32,32,32,32
649 REM !
650 DATA 32,-110,32,33
651 REM "
652 DATA 39,39,32,32
653 REM #
654 DATA 43,43,43,43
655 REM $
656 DATA 99,110,109,100
657 REM %
658 DATA 79,126,126,79
659 REM &
660 DATA 99,100,99,120
661 REM '
662 DATA 32,39,32,32

```



```

663 REM (
664 DATA 116,32,118,32
665 REM )
666 DATA 32,117,32,119
667 REM *
668 DATA 32,42,32,32
669 REM +
670 DATA 109,108,111,110
671 REM ,
672 DATA 32,32,32,44
673 REM -
674 DATA 32,127,32,32
675 REM .
676 DATA 32,32,32,46
677 REM /
678 DATA 32,126,126,32
679 REM 0
680 DATA 96,117,118,120
681 REM 1
682 DATA 39,97,32,97
683 REM 2
684 DATA 104,100,121,127
685 REM 3
686 DATA 101,122,103,100
687 REM 4
688 DATA 126,97,101,110
689 REM 5
690 DATA 123,101,103,100
691 REM 6
692 DATA 116,102,-123,100
693 REM 7
694 DATA 101,122,-122,32
695 REM 8
696 DATA 99,100,99,100
697 REM 9
698 DATA 99,-125,103,119
699 REM :
700 DATA 32,46,32,46
701 REM ;
702 DATA 32,46,32,44
703 REM <
704 DATA 32,126,32,124
705 REM =
706 DATA -95,-95,-101,-101
707 REM >
708 DATA 124,32,126,32
709 REM ?
710 DATA 39,100,32,97
711 REM @ (NOT PROVIDED)
712 DATA 0,0,0,0
713 REM A
714 DATA 116,117,110,111
715 REM B
716 DATA 123,100,123,100

```


717 REM C
718 DATA 116,102,118,105
719 REM D
720 DATA 110,117,108,119
721 REM E
722 DATA 123,101,108,127
723 REM F
724 DATA 123,101,97,32
725 REM G
726 DATA 116,102,118,125
727 REM H
728 DATA 97,98,110,111
729 REM I
730 DATA 101,110,127,108
731 REM J
732 DATA 32,98,103,119
733 REM K
734 DATA -98,104,97,124
735 REM L
736 DATA 97,32,108,127
737 REM M
738 DATA 112,113,97,98
739 REM N
740 DATA 112,98,97,115
741 REM O
742 DATA 116,117,118,119
743 REM P
744 DATA 123,100,97,32
745 REM Q
746 DATA 116,117,118,120
747 REM R
748 DATA 123,100,97,124
749 REM S
750 DATA 99,102,103,100
751 REM T
752 DATA 101,110,32,97
753 REM U
754 DATA 97,98,118,119
755 REM V
756 DATA 97,98,124,126
757 REM W
758 DATA 97,98,114,115
759 REM X
760 DATA 124,126,126,124
761 REM Y
762 DATA 124,126,98,32
763 REM Z
764 DATA 101,122,121,127
765 REM [
766 DATA 110,32,108,32
767 REM \
768 DATA 124,32,32,124
769 REM]
770 DATA 32,111,32,109


```

771 REM ^
772 DATA 126,124,32,32
773 REM _
774 DATA 32,32,95,95
8998
8999 REM ——— PUT TOP OF MEMORY BACK WHERE IT WAS
9000 TM=256*PEEK(32941)+PEEK(32940)+7
9010 POKE 32941,INT(TM/256):POKE 32940,TM-256*INT(TM/256)
9018
9019 REM RESET COLORS, ETC.
9020 PLOT 15,6,2,29
9029
9030 END
9031
62998
62999 REM ——— BEN BARLOW'S NO-ECHO PATCH
63000 RESTORE 63000:DATA 245,175,50,255,129,241,201
63010 TM=256*PEEK(32941)+PEEK(32940)-7
63020 FOR X=1 TO 7:READ D:POKE TM+X,D:NEXT
63030 BR=INT(TM/256):POKE 33221,195:POKE 33222,TM-BR*256+1
63040 POKE 33223,BR:POKE 32941,BR:POKE 32940,TM-BR*256
63050 CLEAR 50:GOTO 10
63051
63052 REM FOR NO-ECHO, POKE 33247,31.
63053 REM TO RETURN TO ECHO, POKE 33247,12.
63054 REM (AN INPUT STATEMENT OR THE END OF
63055 REM THE PROGRAM WILL ALSO DO IT)

```

12.9 DICE

The following program prints a pair of dice showing a randomly chosen number. Each die is drawn separately by the subroutine starting at line 1000. Figure 12.8 shows a 15 cell die with the location of the seven possible spots.

For a cleaner looking display, you could rewrite the program to hide the cursor and draw the dice in blind cursor mode.

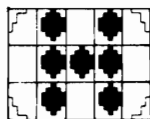


FIG. 12.8

```

5 REM PROGRAM 12.10
6 REM DICE
9
10 PLOT 15,6,2,29,12:REM SET UP; FLAG OFF
19

```



```

20 Y=6:REM  CURSOR Y OF EACH DIE
29
30 PLOT 3,18,15
40 INPUT "[yel]PRESS [wht]RETURN[yel] TO ROLL THE DICE ";A$
50 PLOT 28,11:REM  ERASE THE PROMPT
58
59 REM  CHOOSE RANDOM # OF TIMES DICE WILL ROLL
60 T=10*RND(1)
68
69 REM  ROLL 'EM
70 FOR J=1 TO T
79
80   FOR X=25 TO 32 STEP 7:REM  CURSOR X OF EACH DIE
90     N=INT(6*RND(1))+1:REM  GET A NUMBER FOR THE DIE
100    GOSUB 1000:REM  DRAW THE DIE
110    NEXT
119
120 NEXT
129
130 GOTO 30:REM  BACK FOR MORE
994
995 REM  ----- SUBROUTINE TO PRINT A DIE
996
997 REM  X,Y=CURSOR CO-ORDINATES OF TOP LEFT OF DIE.
998 REM  N IS THE NUMBER THE DIE IS TO SHOW.
999 REM  FLAG MUST BE OFF.
1000 PLOT 3,X,Y
1009
1010 PLOT 6,7:REM  SET COLOR FOR CELL #1
1020 PLOT 116:REM  PRINT CELL #1
1029
1030 PLOT 6,56:REM  SET COLOR FOR CELLS 2-4
1040 PLOT 32-68*(N=2 OR N>3):REM  CELL #2
1050 PLOT 32:REM  CELL #3
1060 PLOT 32-68*(N>2):REM  CELL #4
1069
1070 PLOT 6,7:REM  SET COLOR FOR CELL #5
1080 PLOT 117:REM  PRINT CELL #5
1088
1089 REM  MOVE CURSOR BACK AND DOWN FOR THE 2ND ROW
1090 GOSUB 1500
1099
1100 PLOT 6,56:REM  SET COLOR FOR CELLS 6-10
1110 PLOT 32:REM  CELL #6
1120 PLOT 32-68*(N=6):REM  CELL #7
1130 PLOT 32-68*(N=1 OR N=3 OR N=5):REM  CELL #8
1140 PLOT 32-68*(N=6):REM  CELL #9
1150 PLOT 32:REM  CELL #10
1158
1159 REM  MOVE CURSOR BACK AND DOWN FOR FINAL ROW
1160 GOSUB 1500
1169
1170 PLOT 6,7:REM  SET COLOR FOR CELL #11
1180 PLOT 118:REM  PRINT CELL #11

```



```

1189
1190 PLOT 6,56:REM SET COLOR FOR CELLS 12-14
1200 PLOT 32-68*(N>2):REM CELL #12
1210 PLOT 32:REM CELL #13
1220 PLOT 32-68*(N=2 OR N>3):REM CELL #14
1229
1230 PLOT 6,7:REM SET COLOR FOR CELL #15
1240 PLOT 119:REM CELL #15
1249
1250 RETURN
1498
1499 REM ----- MOVE CURSOR BACK & DOWN FOR NEXT ROW
1500 PLOT 26,26,26,26,26,10:RETURN

```

12.10 DICE — THE FAST WAY

The Dice Program in the previous section is fine, but slow. One of the fastest ways to put something on the screen in BASIC is to PRINT it. The following program defines six die faces (based upon Figure 12.8) as six character strings, D\$(1)...D\$(6). Each character string has embedded within it all the necessary PLOT commands. For example, an instruction such as

```
PLOT 30,16,29,23
```

can alternatively be coded as:

```
D$=CHR$(30)+CHR$(16)+CHR$(29)+CHR$(23):PRINT D$
```

Each of the six die face character strings incorporates color controls, cursor controls and special characters according to the ASCII chart in Appendix A. You may adapt the routine for your own uses by changing the colors for the dice (lines 30-50) and the co-ordinates (lines 60-70).

```

5 REM PROGRAM 12.11
6 REM DICE -- THE FAST WAY
9
10 CLEAR 500:REM (YOU'LL NEED IT)
19
20 DIM D$(6):REM ONE DIE FACE FOR EACH POSSIBLE NUMBER
28
29 REM SET UP THE CONSTANTS
30 FG=23:REM FOREGROUND COLOR OF DIE
40 BG=16:REM COLOR OF BACKGROUND AGAINST WHICH DIE IS PRINTED
50 SP=16:REM COLOR OF THE SPOTS ON THE DIE
59
60 X(1)=26:Y(1)=3:REM CURSOR X AND Y OF FIRST DIE
70 X(2)=33:Y(2)=3:REM CURSOR X AND Y OF SECOND DIE
78
79 REM ESTABLISH THE SIX DIE FACES
80 GOSUB 9000
88

```



```

89 REM SET UP THE SCREEN
90 PLOT 15:REM REGULAR HEIGHT. DOUBLE HEIGHT WORKS, TOO, BUT
91 REM THE DICE WON'T BE AS WELL PROPORTIONED
92
100 PLOT 6,6,12,29
109
110 PLOT 3,18,15
120 INPUT "[cyn]PRESS [wht]RETURN [cyn]TO ROLL THE DICE ";A$
130 PLOT 28,11:REM ERASE THE PROMPT
138
139 REM ROLL THE DICE A RANDOM NUMBER OF TIMES
140 FOR ROLL=1 TO 20*RND(1)+5:GOSUB 8000:NEXT
148
149 REM BACK FOR MORE
150 GOTO 110
7998
7999 REM ----- PRINT THE TWO DICE
8000 FOR DIE=1 TO 2
8010 N=INT(6*RND(1))+1:REM RANDOM NUMBER (1-6) FOR THE DIE
8020 PLOT 3,X(DIE),Y(DIE):REM POSITION THE CURSOR
8030 PRINT D$(N):REM PRINT IT
8040 NEXT
8049
8050 RETURN
8998
8999 REM ----- ESTABLISH THE SIX DIE FACES AS CHARACTER STRINGS
9000 RESTORE 9500
9009
9010 FOR FACE=1 TO 6
9019 REM TOP ROW OF DIE
9020 D$(FACE)=CHR$(30)+CHR$(BG)+CHR$(29)+CHR$(FG)+CHR$(116)
9030 D$(FACE)=D$(FACE)+CHR$(30)+CHR$(FG)+CHR$(29)+CHR$(SP)
9040 GOSUB 9300
9050 D$(FACE)=D$(FACE)+" "
9060 GOSUB 9300
9070 D$(FACE)=D$(FACE)+CHR$(30)+CHR$(BG)+CHR$(29)+CHR$(FG)+CHR$(1
17)
9078
9079 REM BACK AND DOWN FOR SECOND ROW
9080 GOSUB 9400
9088
9089 REM NOW FOR THE SECOND ROW
9090 D$(FACE)=D$(FACE)+CHR$(30)+CHR$(FG)+CHR$(29)+CHR$(SP)+" "
9100 FOR K=1 TO 3:GOSUB 9300:NEXT
9110 D$(FACE)=D$(FACE)+" "
9118
9119 REM BACK AND DOWN FOR THIRD (AND FINAL) ROW
9120 GOSUB 9400
9128
9129 REM THE THIRD ROW
9130 D$(FACE)=D$(FACE)+CHR$(30)+CHR$(BG)+CHR$(29)+CHR$(FG)+CHR$(1
18)
9140 D$(FACE)=D$(FACE)+CHR$(30)+CHR$(FG)+CHR$(29)+CHR$(SP)
9150 GOSUB 9300
9160 D$(FACE)=D$(FACE)+" "

```



```

9170 GOSUB 9300
9180 D$(FACE)=D$(FACE)+CHR$(30)+CHR$(BG)+CHR$(29)+CHR$(FG)+CHR$(1
19)
9190 NEXT
9199
9200 RETURN
9298
9299 REM -----
9300 READ A:D$(FACE)=D$(FACE)+CHR$(A):RETURN
9398
9399 REM -----
9400 FOR K=1 TO 5:D$(FACE)=D$(FACE)+CHR$(26):NEXT
9410 D$(FACE)=D$(FACE)+CHR$(10)
9420 RETURN
9498
9499 REM 1
9500 DATA 32,32,32,100,32,32,32
9504 REM 2
9505 DATA 32,100,32,32,32,100,32
9509 REM 3
9510 DATA 100,32,32,100,32,32,100
9514 REM 4
9515 DATA 100,100,32,32,32,100,100
9519 REM 5
9520 DATA 100,100,32,100,32,100,100
9524 REM 6
9525 DATA 100,100,100,32,100,100,100

```

12.11 QUICK CHANGE ARTISTRY

If a program of yours makes use of more than one screen display loaded from disk, or if your program makes use of both disk-loaded displays and displays generated by the program itself, or (how long can I go on?) if your program makes use of several internally generated displays, then you might consider a really flashy way of presenting them.

The method is straightforward: put one display on the screen, as usual, but put the other one into high memory. Now you can have quick access to either display simply by interchanging them. To make it quick, you'll have to use a machine language patch, but that will be fairly simple. Here is how I have done it:

```

      PUSH    H           ;Save H&L on stack.
      LXI     B,1000H     ;Use B&C pair as counter for the
                          ;4096 (decimal) bytes of a display.
      LXI     H,AUX       ;H&L point to start of AUX, which
                          ;is 4096 bytes at top of memory.
      LXI     D,7000H     ;D&E point to start of screen ram.
LOOP: LDAX    D           ;Get a byte from the screen and
      PUSH    PSW         ;save it temporarily on the stack.

```



```

MOV     A,M      ;Get byte from AUX and
STAX    D        ;put it onto the screen.
POP     PSW      ;Reclaim screen byte from stack and
MOV     M,A      ;put it into AUX.
INX     H        ;Point to next byte in AUX.
INX     D        ;Point to next byte on screen.
DCX     B        ;Decrement counter and continue until all
MOV     A,B      ;4096 bytes have been interchanged.
ORA     C
JNZ     LOOP

POP     H
RET

```

The BASIC program which follows incorporates a patch to implement that routine. The program is only a demo to show you what goes on by using two of your screen displays stored on disk. No doubt you will discover your own applications. For example, suppose you have one display (either loaded from disk or else generated by your program) which is occasionally updated (charts or graphs, for examples). And suppose you also want to be able to print a great deal of text, get input, and so on. You might erase the whole display and draw it later in order to update it. (Or you might confine all text to a small portion of the screen by using the scrolling patch as a little text window.)

Quick change artistry can offer an alternative here by allowing you to store the updated display and recall it whenever you wish. In this way, you can flip back and forth from one display to another, and the displays will not be lost.

By changing the patch in suitable ways (another one of those "exercises for the reader") you could interchange only part of a screen display while leaving the remainder undisturbed. Or you could call up one of a selection of auxiliary displays (or parts thereof) depending on the needs of your application.

```

0 GOTO 65000:REM  LOAD THE PATCH
1
5 REM  PROGRAM 12.12
6 REM  QUICK CHANGE ARTISTRY
8
9 REM SET UP
10 PLOT 14,30,16,29,18,12
19
20 PRINT TAB(10) "PROGRAM TO INTERCHANGE SCREEN DISPLAYS"
27
28 REM  GET NAME OF AUX DISPLAY
29 REM  NOTE: THERE'S NOT MUCH INPUT ERROR CHECKING
30 T$="AUXILIARY STORAGE":GOSUB 1000
38
39 REM  GET HEX VALUE FOR LOCATION OF AUX
40 GOSUB 65410:TM=TM+27
50 R$=""
60 A=INT(TM/16)
70 R=INT(TM-A*16):IF R>9 THEN R$=CHR$(55+R)+R$:GOTO 90
80 R$=RIGHT$(STR$(R),1)+R$

```



```

90 IF A>0 THEN TM=A:GOTO 60
98
99 REM LOAD THE AUX DISPLAY
100 GOSUB 2000:PLOT 27,4:PRINT "LOA "D$ "R$:PLOT 27,27
108
109 REM NOW GET THE DISPLAY FOR THE SCREEN
110 T$="THE SCREEN":GOSUB 1000
119
120 PRINT "AFTER THE SCREEN DISPLAY IS LOADED, SIMPLY PRESS"
130 PRINT "RETURN TO EXCHANGE THE TWO DISPLAYS."
140 PRINT "(ALSO TRY TYPING SOMETHING ONTO THE DISPLAY.)"
150 PRINT:PRINT "TYPE 'END' TO END THE PROGAM."
160 PRINT
170 INPUT "NOW PRESS RETURN TO LOAD THE SCREEN DISPLAY...";A$
179
180 PLOT 12,27,4:PRINT "LOA "D$:PLOT 27,27,3,64,0
189
190 INPUT "";A$
200 IF A$<>"END" THEN PLOT 3,64,0:Z=CALL(0):GOTO 190
218
219 REM PUT TOP OF MEMORY BACK WHERE IT WAS
220 GOSUB 65410:Z=TM+4122:AD=ER:GOSUB 65400
230 END
231
998
999 REM ----- GET DISPLAY NAME
1000 PRINT
1010 PRINT "I NEED THE NAME AND TYPE (AND VERSION, IF YOU WISH)--AS
THEY"
1020 PRINT "APPEAR IN THE DIRECTORY--OF THE SCREEN DISPLAY TO BE LO
ADED"
1030 PRINT "INTO "T$"."
1040 PRINT
1050 INPUT "WOULD YOU LIKE TO SEE THE DIRECTORY? ";A$
1060 IF LEFT$(A$,1)="N" THEN 1090
1070 GOSUB 2000
1080 PLOT 27,4:PRINT "DIR":PLOT 27,27
1089
1090 PRINT:PRINT
1100 INPUT "PLEASE ENTER NAME.TYPE OF THE DISPLAY: ";D$
1109
1110 IF LEN(D$)<5 OR LEN(D$)>13 THEN 1100
1119
1120 FOR J=1 TO LEN(D$)
1130 IF MID$(D$,J,1)="." THEN J=999
1140 NEXT:IF J<999 THEN 1100
1149
1150 PRINT:RETURN
1999
2000 PRINT
2010 INPUT "PLEASE MOUNT THE PROPER DISK AND PRESS RETURN...";A$
2020 PRINT:RETURN
64998
64999 REM ----- HERE'S THE PATCH
65000 GOSUB 65410:RESTORE 65010

```



```

65010 DATA 229,1,0,16,17,0,112,33,-1,-1,26
65020 DATA 245,126,18,241,119,35,19,11,120
65030 DATA 179,194,-1,-1,225,201
65038
65039 REM CHECK TO SEE IF PATCH IS ALREADY LOADED
65040 IF TM>61413 THEN TM=TM-4122:GOTO 65080
65049
65050 FOR J=1 TO 26:READ A
65060 IF A=>0 AND A<>PEEK(TM+J) THEN J=26:TM=TM-4122
65070 NEXT
65078
65079 REM POKE IN THE PATCH
65080 RESTORE 65010
65090 FOR J=1 TO 26:READ A:POKE TM+J,A-(A<0):NEXT
65098
65099 REM SET UP JUMP TO THE PATCH
65100 Z=TM+1:AD=33283:GOSUB 65400
65108
65109 REM SET UP NEW TOP OF MEMORY
65110 Z=TM:AD=ER:GOSUB 65400
65118
65119 REM SET UP ADDRESS OF START OF AUX DISPLAY STORAGE
65120 Z=TM+27:AD=TM+9:GOSUB 65400
65128
65129 REM SET UP ADDRESS OF LOOP IN THE PATCH
65130 Z=TM+11:AD=TM+23:GOSUB 65400
65139
65140 CLEAR 50:GOTO 10
65399
65400 ZZ=INT(Z/256):POKE AD,Z-256*ZZ:POKE AD+1,ZZ:RETURN
65409
65410 ER=32940:TM=256*PEEK(ER+1)+PEEK(ER):RETURN

```


APPENDIX A: ASCII CHART

PLT OF EITHER
PRODUCES FUNCTION.

0 (128) NULL
1 (129) AUTO
2 (130) PLOT
3 (131) CURSOR X,Y
4 (132) NOT USED
5 (133) NOT USED
6 (134) CCI
7 (135) NOT USED
8 (136) HOME CURSOR
9 (137) TAB
10 (138) CURSOR DOWN
11 (139) ERASE LINE
12 (140) ERASE PAGE
13 (141) RETURN
14 (142) A7 ON
15 (143) BLINK OFF/A7 OFF
16 (144) BLACK
17 (145) RED
18 (146) GREEN
19 (147) YELLOW
20 (148) BLUE
21 (149) MAGENTA
22 (150) CYAN
23 (151) WHITE
24 (152) XMIT
25 (153) CURSOR RIGHT
26 (154) CURSOR LEFT
27 (155) ESCAPE
28 (156) CURSOR UP
29 (157) FLAG OFF (FOREGROUND)
30 (158) FLAG ON (BACKGROUND)
31 (159) BLINK ON

PLT OF EITHER
PRODUCES SYMBOL

32 (160)	!
33 (161)	"
34 (162)	#
35 (163)	\$
36 (164)	%
37 (165)	&
38 (166)	'
39 (167)	(
40 (168))
41 (169)	*
42 (170)	+
43 (171)	,
44 (172)	-
45 (173)	.
46 (174)	/
47 (175)	0
48 (176)	1
49 (177)	2
50 (178)	3
51 (179)	4
52 (180)	5
53 (181)	6
54 (182)	7
55 (183)	8
56 (184)	9
57 (185)	:
58 (186)	;
59 (187)	<
60 (188)	=
61 (189)	>
62 (190)	?

PLT OF EITHER
PRODUCES SYMBOL

64 (192)	@
65 (193)	A
66 (194)	B
67 (195)	C
68 (196)	D
69 (197)	E
70 (198)	F
71 (199)	G
72 (200)	H
73 (201)	I
74 (202)	J
75 (203)	K
76 (204)	L
77 (205)	M
78 (206)	N
79 (207)	O
80 (208)	P
81 (209)	Q
82 (210)	R
83 (211)	S
84 (212)	T
85 (213)	U
86 (214)	V
87 (215)	W
88 (216)	X
89 (217)	Y
90 (218)	Z
91 (219)	[
92 (220)	\
93 (221)]
94 (222)	^
95 (223)	_

PLT OF EITHER
PRODUCES SYMBOL

PLT 30
PLT 29

FLAG
On Off

96 (224)	~
97 (225)	~
98 (226)	~
99 (227)	~
100 (228)	~
101 (229)	~
102 (230)	~
103 (231)	~
104 (232)	~
105 (233)	~
106 (234)	~
107 (235)	~
108 (236)	~
109 (237)	~
110 (238)	~
111 (239)	~
112 (240)	~
113 (241)	~
114 (242)	~
115 (243)	~
116 (244)	~
117 (245)	~
118 (246)	~
119 (247)	~
120 (248)	~
121 (249)	~
122 (250)	~
123 (251)	~
124 (252)	~
125 (253)	~
126 (254)	~
127 (255)	~

SHIFT
+---

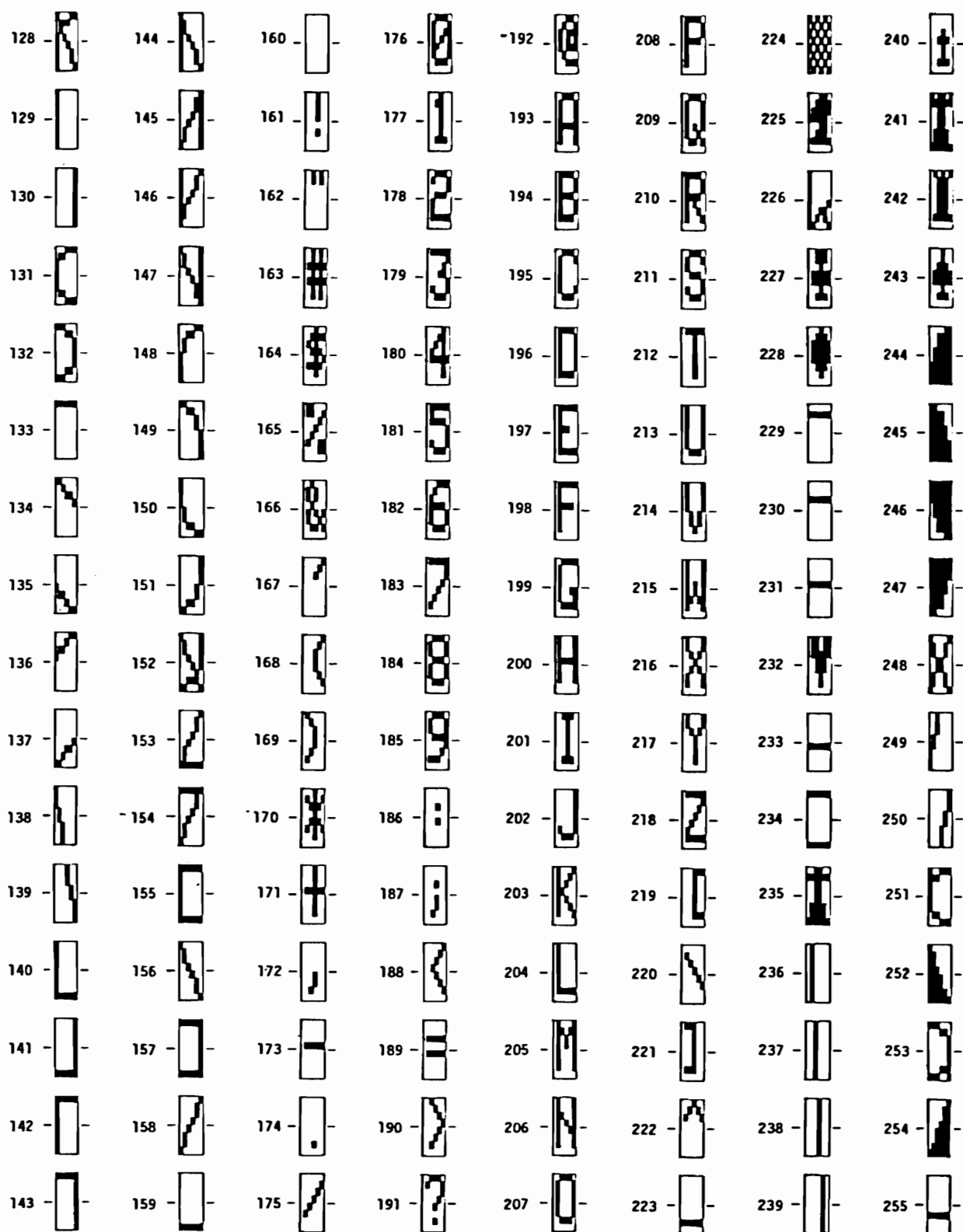
@
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
[
\
]
^

— used before

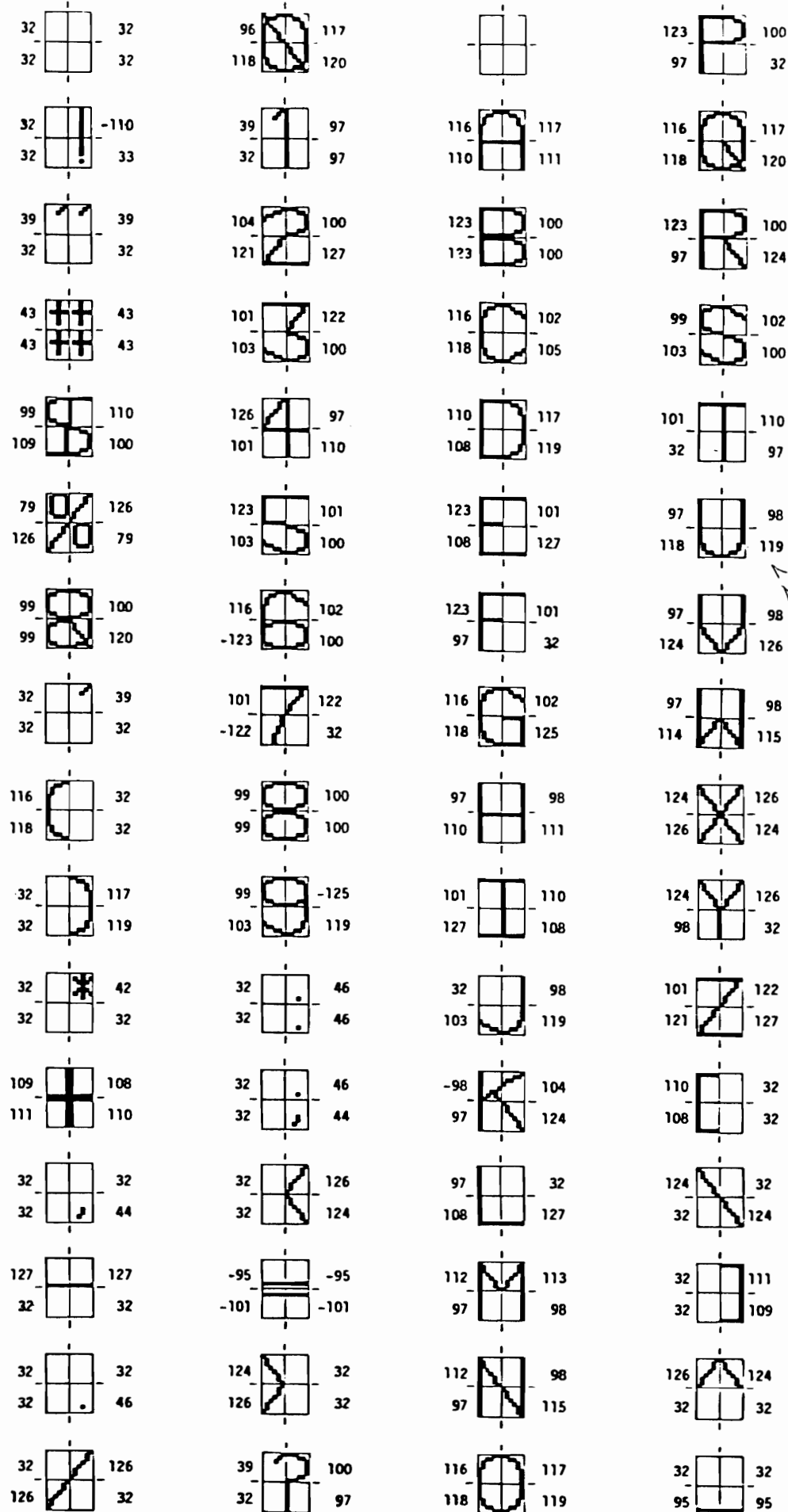
APPENDIX B: CHARACTER SET

NOT
PLOT #5

0		16		32		48		64		80		96		112	
1		17		33		49		65		81		97		113	
2		18		34		50		66		82		98		114	
3		19		35		51		67		83		99		115	
4		20		36		52		68		84		100		116	
5		21		37		53		69		85		101		117	
6		22		38		54		70		86		102		118	
7		23		39		55		71		87		103		119	
8		24		40		56		72		88		104		120	
9		25		41		57		73		89		105		121	
10		26		42		58		74		90		106		122	
11		27		43		59		75		91		107		123	
12		28		44		60		76		92		108		124	
13		29		45		61		77		93		109		125	
14		30		46		62		78		94		110		126	
15		31		47		63		79		95		111		127	



Positive numbers: PLOT with FLAG on. Negative numbers: PLOT with FLAG off.



PLOT 9, 30, 97, 98, 10
26, 26, 118, 119

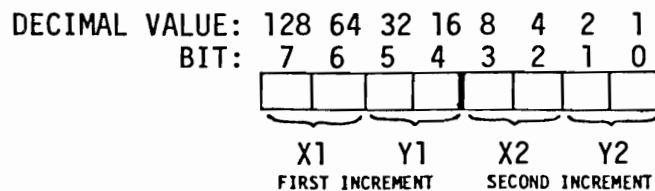
TND, FLAG ON SHIFT+A
SHIFT+B, SHIFT+C, SHIFT+D, SHIFT+E
SHIFT+F, SHIFT+G, SHIFT+H, SHIFT+I
SHIFT+J, SHIFT+K, SHIFT+L, SHIFT+M
SHIFT+N, SHIFT+O, SHIFT+P, SHIFT+Q
SHIFT+R, SHIFT+S, SHIFT+T, SHIFT+U
SHIFT+V, SHIFT+W, SHIFT+X, SHIFT+Y
SHIFT+Z, SHIFT+[, SHIFT+\, SHIFT+]
SHIFT+~, SHIFT+_

SHIFT NOT REQ IF
CNS LOCK UP.

APPENDIX D: PLOT MODES

General Plot Mode	2	General plot mode introduction. Unless instructed otherwise, the computer will automatically be in the point plot submode.
Vectors	240	Incremental vector plot
	241	Y co-ordinate of vector end point
	242	X co-ordinate of vector end point
Y Bar Graph	243	Incremental Y bar graph
	244	Y co-ordinate of top of vert. bar
	245	X co-ordinate of vert. bar
	246	Y co-ordinate of bottom of vert. bar
X Bar Graph	247	Incremental X bar graph
	248	X co-ordinate of right of horiz. line
	249	Y co-ordinate of horiz. bar
	250	X co-ordinate of left of horiz. bar
Point Plot	251	Incremental point plot
	252	Y co-ordinate of point
	253	X co-ordinate of point
Character Plot	254	
Plot Mode Escape	255	

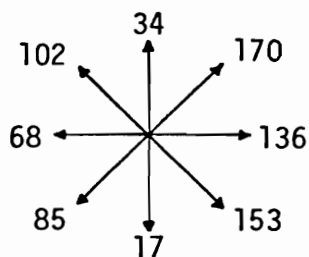
APPENDIX E:
INCREMENTAL VALUES FOR
INCREMENTAL POINT PLOT AND INCREMENTAL BAR GRAPH SUBMODES



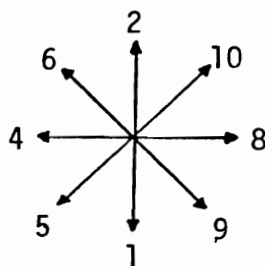
If the binary value of the two bits is...	...then the increment of that co-ordinate will be...
00	None*
01	-
10	+
11	None

- * If X1=0 and Y1=0 then the point or bar graph will be incremented once according to the values of X2 and Y2.
- * If X2=0 and Y2=0 then the point or bar graph will be incremented once according to the values of X1 and Y1, but the new point or bar graph will not be drawn.

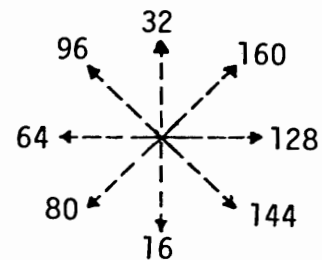
Incremental values
for incrementing
twice in the same
direction.



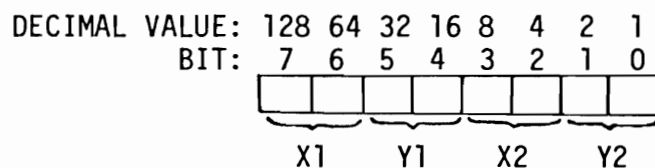
Incremental values
for incrementing
only X2,Y2
(X1=0, Y1=0).



Incremental values
for incrementing
without plotting
(X2=0, Y2=0).



**APPENDIX F:
INCREMENTAL VALUES FOR
INCREMENTAL VECTOR PLOT SUBMODE**



If the binary value of the two bits is...	...then the increment of that co-ordinate will be...
00	None*
01	-
10	+
11	None

- * If X1=0 and Y1=0 then the X1, Y1 co-ordinates of the vector will be incremented, but the vector will not be drawn.
- * If X2=0 and Y2=0 then the vector will be neither incremented nor drawn.

NO. 11
VALVE

APPENDIX G: KEYBOARD ENCODING

ALL CONTROL
FUNCTIONS

COMMAND

NUMBER
KEYBOARD
VOLUME

	KEY	KEY WITH SHIFT	KEY WITH CONTROL	KEY WITH SHIFT AND CONTROL
0			MULL E	
1	AUTO	AUTO	AUTO A	AUTO
2			PUT B	
3	INSERT LINE	INSERT LINE	INSERT LINE C	INSERT LINE
4	DELETE LINE	DELETE LINE	DELETE LINE D	DELETE LINE
5	INSERT CHAR	INSERT CHAR	INSERT CHAR E	INSERT CHAR
6			CCI F	
7			BELL G	
8	HOME	HOME	HOME H	
9	TAB		TAB I	
10	+		+	
11	ERASE LINE	ERASE LINE	ERASE LINE K	
12	ERASE PAGE	ERASE PAGE	ERASE PAGE L	
13	RETURN	RETURN	RETURN M	
14	A7 ON	A7 ON	A7 ON N	
15	KL/A7 OFF	KL/A7 OFF	KL/A7 OFF O	
16	BLACK		BLACK P	
17	RED		RED Q	
18	GREEN	GREEN	GREEN R	
19	YELLOW	YELLOW	YELLOW S	
20	BLUE		BLUE T	
21	MAGENTA		MAGENTA U	
22	CYAN		CYAN V	
23	WHITE	WHITE	WHITE W	
24			XMIT X	
25	+		+	
26	+		+	
27	ESC	ESC	ESC Y	
28	+	+	+	
29	PG ON/PLG OFF	PG ON/PLG OFF	PG ON/PLG OFF Z	
30	PG ON/PLG ON	PG ON/PLG ON	PG ON/PLG ON [
31	BLINK ON	BLINK ON	BLINK ON]	

	KEY	KEY WITH SHIFT	KEY WITH CONTROL	KEY WITH SHIFT AND CONTROL
32	SPACE	SPACE	SPACE	SPACE
33		1		
34		2		
35		3		
36		4		
37		5		
38		6		
39		7		
40		8		
41		9		
42	* (keypad)	* (keypad)	* (keypad)	* (keypad)
43	+ (keypad)	+ (keypad)	+ (keypad)	+ (keypad)
44	,			
45	= (keypad)			
46	.			
47	/ (keypad)			
48	0			
49	1			
50	2			
51	3			
52	4			
53	5			
54	6			
55	7			
56	8			
57	9			
58	.			
59	,			
60				
61	= (keypad)	= (keypad)	= (keypad)	= (keypad)
62	.			
63	/ (keypad)			

ASCII
VALUE

ASCII
VALUE

SYMBOL
PRODUCED ON
VDU

	KEY	KEY WITH SHIFT	KEY WITH CONTROL	KEY WITH SHIFT AND CONTROL	FLAG OFF	KEY	KEY WITH SHIFT	KEY WITH CONTROL	KEY WITH SHIFT AND CONTROL	FLAG ON	
64	e				96	e				X	
65	A			KNIGHT	97	A					a
66	B				98	B					b
67	C			CLUB	99	C					c
68	D			BISHOP	100	D					d
69	E				101	E					e
70	F				102	F					f
71	G				103	G					g
72	H			KNIGHT	104	H			used in ASCII SRR file!		h
73	I				105	I					i
74	J				106	J					j
75	K			KING	107	K					k
76	L				108	L					l
77	M				109	M					m
78	N				110	N					n
79	O				111	O					o
80	P			PAWN	112	P					p
81	Q			QUEEN	113	Q					q
82	R			ROOK	114	R					r
83	S			SPACE/BISHOP	115	S					s
84	T				116	T					t
85	U				117	U					u
86	V				118	V					v
87	W				119	W					w
88	X				120	X					x
89	Y				121	Y					y
90	Z				122	Z					z
91	[123	[[
92	\				124	\					\
93]				125]]
94	^				126	^					^
95	underline				127	DELETE CHAR	DELETE CHAR	DELETE CHAR	DELETE CHAR		delete

Called BASIC T-GENS
acceptable to BASIC EDITING disk

WORD
PRODUCED
IN VDU

WORD
PRODUCED
IN VDU

NOT NAIL
FROM PLOT
MODE

NOT NAIL
FROM PLOT
MODE

	KEY	KEY WITH SHIFT	KEY WITH CONTROL	KEY WITH SHIFT AND CONTROL	OR COMMAND	KEY	KEY WITH SHIFT	KEY WITH CONTROL	KEY WITH SHIFT AND CONTROL	OR COMMAND
END 128				E	END 160				0	
FOR 129				A	THEN 161				1	
NEXT 130				B	NOT 162				2	
DATA 131				C	STEP 163				3	
INPUT 132				D	+ 164				4	
DIM 133				E	- 165				5	
READ 134				F	* 166				6	
FILE 135				G	/ 167				7	
GOTO 136	HOME			R HOME	^ 168				8	
RUN 137	TAB			I TAB	AND 169				9	
IF 138	+			J +	OR 170				:	
RESUME 139	ERASE LINE			K ERASE LINE	> 171				;	
GOSUB 140	ERASE PAGE			L ERASE PAGE	= 172				,	
RETURN 141	RETURN			M RETURN	< 173				- (keypad)	
REM 142	A7 ON			N A7 ON	SEN 174				.	
GET 143	KL/A7 OFF			O KL/A7 OFF	INT 175				/ (keypad)	
OUT 144	BLACK			P BLACK	ANS 176				0	
PUT 145	RED			Q RED	CALL 177				1	
PLOT 146	GREEN			R GREEN	PRE 178				2	
SAVE 147	YELLOW			S YELLOW	INP 179				3	
LOAD 148	BLUE			T BLUE	POS 180				4	
POKE 149	MAGENTA			U MAGENTA	SPR 181				5	
PRINT 150	CYAN			V CYAN	RND 182				6	
LIST 151	WHITE			W WHITE	LOG 183				7	
CONT 152				X	EXP 184				8	
CLEAR 153	→			Y →	COS 185				9	
DEF 154	←			Z ←	SIN 186				:	
WAIT 155	ESC			[ESC	TAN 187				;	
ON 156	↑			↑ \	ATN 188				,	
TAB(157	PG ON/PLG OFF			J PG ON/PLG OFF	PEEK 189				- (keypad)	
TO 158	PG ON/PLG ON			K PG ON/PLG ON	LEN 190				.	
FN 159	BLINK ON			L BLINK ON	STR # 191				/ (keypad)	

WORD
PROCESSOR
IN VDU

NOT AVAILABLE
FROM PLOT
MODE

NOT AVAILABLE

FROM PLOT

USE PLOT 2 240

MODE

DIRECT

KEYS

KEY
KEY WITH
SHIFT
KEY WITH
CONTROL
KEY WITH
SHIFT AND
CONTROL

KEY
KEY WITH
SHIFT
KEY WITH
CONTROL
KEY WITH
SHIFT AND
CONTROL

VAL 192			F0	
ASC 193			F1	
CHR 194			F2	
195			F3	
196			F4	
197			F5	
198			F6	
199			F7	
200			F8	
201			F9	
202			F10	
203			F11	
204			F12	
205			F13	
206			F14	
207			F15	
208	F0			VECTOR INCREM
209	F1			VECTOR X
210	F2			VECTOR Y
211	F3			Y BAR INCREM
212	F4			Y BAR TOP Y
213	F5			Y BAR X
214	F6			Y BAR STAY
215	F7			X BAR INCREM
216	F8			X BAR R.H
217	F9			X BAR Y
218	F10			X BAR L.H
219	F11			POINT PLOT INC
220	F12			POINT PLOT Y
221	F13			POINT PLOT X
222	F14			CHARACTER PLOT
223	F15			EXIT PLOT MODE

224				F0
225				F1
226				F2
227				F3
228				F4
229				F5
230				F6
231				F7
232				F8
233				F9
234				F10
235				F11
236				F12
237				F13
238				F14
239				F15
240	F0			
241	F1			
242	F2			
243	F3			
244	F4			
245	F5			
246	F6			
247	F7			
248	F8			
249	F9			
250	F10			
251	F11			
252	F12			
253	F13			
254	F14			
255	F15			

↑

NE IS TAKEN
14-1-82

COINCIDENCE?

F15 = FF(H) = 255(D)

F13 = FF(H) = 243(D)

F0 = 00(H) = 0(D)

APPENDIX H: SOME HELPFUL EQUATIONS

Plot vs. Cursor

$$\begin{array}{l} X_{\text{plot}} = 2 * X_{\text{cursor}} \\ Y_{\text{plot}} = 127 - 4 * Y_{\text{cursor}} \end{array} \quad \left. \vphantom{\begin{array}{l} X_{\text{plot}} = 2 * X_{\text{cursor}} \\ Y_{\text{plot}} = 127 - 4 * Y_{\text{cursor}} \end{array}} \right\} \text{ see note}$$

$$\begin{array}{l} X_{\text{cursor}} = \text{INT}(X_{\text{plot}}/2) \\ Y_{\text{cursor}} = \text{INT}((127 - Y_{\text{plot}})/4) \end{array}$$

Screen Memory Location (M) vs. Cursor

$$M = 28672 + 128 * Y_{\text{cursor}} + 2 * X_{\text{cursor}}$$

$$\begin{array}{l} X_{\text{cursor}} = \text{INT}((M - 28672 - 128 * Y_{\text{cursor}})/2) \\ Y_{\text{cursor}} = \text{INT}((M - 28672)/128) \end{array}$$

Screen Memory Location (M) vs. Plot

$$M = 28672 + 128 * \text{INT}((127 - Y_{\text{plot}})/4) + 2 * \text{INT}(X_{\text{plot}}/2)$$

$$\begin{array}{l} X_{\text{plot}} = M - 28672 - 128 * \text{INT}((M - 28672)/128) \\ Y_{\text{plot}} = 127 - 4 * \text{INT}((M - 28672)/128) \end{array} \quad \left. \vphantom{\begin{array}{l} X_{\text{plot}} = M - 28672 - 128 * \text{INT}((M - 28672)/128) \\ Y_{\text{plot}} = 127 - 4 * \text{INT}((M - 28672)/128) \end{array}} \right\} \text{ see note}$$

NOTE: where $X_{\text{plot}}, Y_{\text{plot}}$ will reference the top left plot block in a character position.

Some Trigonometric Values

$$\pi = 3.1415926$$

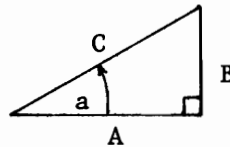
$$\text{Degrees} = \pi / 180 * \text{Radians}$$

$$\pi / 180 = .0174533$$

$$\text{SIN}(a) = B/C$$

$$\text{COS}(a) = A/C$$

$$\text{TAN}(a) = B/A$$



APPENDIX I: SCREEN MEMORY LOCATIONS

		X										
cursor		0	1	2				61	62	63		
Y		0 28672	28674	28676	.	.	.	28794	28796	28798	0	
	1	28800								28926	1	
	2	28928								29054	2	
	3	29056								29182	3	
	4	29184								29310	4	
	5	29312								29438	5	
	6	29440								29566	6	
	7	29568								29694	7	
	8	29696								29822	8	
	9	29824								29950	9	
	10	29952								30078	10	
	11	30080								30206	11	
	12	30208								30334	12	
	13	30336								30462	13	
	14	30464								30590	14	
	15	30592								30718	15	
	16	30720								30846	16	
	17	30848								30974	17	
	18	30976								31102	18	
	19	31104								31230	19	
	20	31232								31358	20	
	21	31360								31486	21	
	22	31488								31614	22	
	23	31616								31742	23	
	24	31744								31870	24	
	25	31872								31998	25	
	26	32000								32126	26	
	27	32128								32254	27	
	28	32256								32382	28	
	29	32384								32510	29	
	30	32512								32638	30	
	31	32640	32642	32644	.	.	.	32762	32764	32766	31	
		0	1	2				61	62	63		

(i) FLAG

If the FLAG bit is off (PLOT 29) then foreground colors are used.

If the FLAG is on (PLOT 30) the background colors are selected.

Colors are determined by subsequent PLOT values:

16	Black
17	Red
18	Green
19	Yellow
20	Blue
21	Magenta
22	Cyan
23	White

(ii) CCI code

Bit:	B7	B6	B5	B4	B3	B2	B1	B0
	PLOT	BLINK	BACKGROUND COLOR			FOREGROUND COLOR		
			BLUE	GREEN	RED	BLUE	GREEN	RED
Decimal:	128	64	32	16	8	4	2	1

PLOT 6,n		Color:	
Decimal	Binary	Background	Foreground
0	00000000	black	black
1	00000001	black	red
2	00000010	black	green
3	00000011	black	yellow
4	00000100	black	blue
5	00000101	black	magenta
6	00000110	black	cyan
7	00000111	black	white
8	00001000	red	black
9	00001001	red	red
	.		
	.		
62	00111110	white	cyan
63	00111111	white	white

(iii) FLAG/CCI code conversions

```
C = (FG-16)+8*(BG-16)
BG = INT(C/8)
FG = C-8*BG
```

where FG is the foreground color code,
BG is the background color code, and
C is the CCI code.

INDEX

Numbers in regular type are page numbers.
Numbers in boldface are program numbers.

Animated joke 103, **12.6**
Artillery game 36f, **6.6**
 explosion routine 52, **6.17**
A7 bit 3, **1.4**, 16, 20, 24, 65, 67, 71
AUTO 18, 68
Backwards text 91f, **10.2**
Bar graphs **see** Plot submodes
Blank Menu 93, **11.1**
Blind cursor 18ff
 addressing 18, **3.1**
 and double height characters 20, **3.4**
 and plot 66f, **7.3**
 reentry to 19, **3.3**
 in scrolling patch 89
 vs. visible cursor 18, **3.2**
Blink 15f
 and CCI code 17
 out of phase 16, **2.10**
 PLOT 31 16, **2.9**, **2.10**
CALL (x) 87
Carriage return 2, **1.1** - **1.3**, 21
 in vertical mode 5, **1.6**
CCI code 9
 and blind cursor 18
 and blink 17
 >127 65, **7.2** - **7.5**
 and screen refresh 69, **8.1**, **8.4**, **8.5**
Character plot submode **see** Plot submodes
Character strings, plotting 64, **6.25**
Characters 135f
 double height 3
 in blind cursor 20, **3.4**
 in screen memory 69f
 extra large 24, **4.6**, 121, **12.9**, 138
 heights 3, **1.4**, **4.6**
 in blind cursor 20, **3.4**
 and plot 21
 in screen memory 71, **8.4**
 special 21, **4.1**, **4.2**, **4.4** - **4.6**, **5.1**, **5.2**, 103, **12.6**,
 121, **12.9**, 135f
 and test mode 27, **5.1**, **5.2**
Chess board **6.11**, 106, **12.7**
Chess pieces 106, **12.7**
Circles 97, **12.2**, **12.5**
Colors 8
 and CCI code 9, **2.2**
 and CONTROL key 75

- and FLAG bit 8, 2.1
- and hatch character 12, 2.6, 2.7
- keys 11, 2.4, 2.5, 14
- and plot mode 34
- resolution of 6.4
- Comma 2
- COMMAND key 76
- CONTROL B 75
- CPU reset 59, 76
- CRT mode 75
- Cursor
 - blind **see** Blind cursor
 - visible 1
 - addressing 1
 - and screen refresh 69, 146, 147
 - vs. plot co-ordinates 34, 146
 - controls 1
 - and blind cursor 18, 3.2
 - and character plot 31
 - hiding 2, 90
 - and test mode 28
- Degrees 36, 97, 146
- Dice 127, 12.10, 129, 12.11
- Displays (screen)
 - duplicating 78, 79, 9.1, 9.2
 - editing 78, 9.2
 - interchanging 131, 12.12
 - saving 77, 79, 9.2
- Double height characters 3
 - in blind cursor 20, 3.4
 - in screen memory 71
 - in scrolling patch 89
- Editing screen displays 78, 9.2
- Erase line 5
 - and erase screen 47
- Erase page 2
 - and bar graph submodes 47, 6.13, 6.14
 - and erase line 47
 - explosion effect 10, 2.3
 - in plot 67, 7.5, 73
 - and test mode 28, 67
 - and vector plot submode 54, 6.18, 6.19
- Explosion effect
 - with erase page 10, 2.3
 - with point and vector plot 52, 6.17, 12.8
 - with test mode 28, 5.3
- Extra large characters 24, 4.6, 121, 12.9
- Escape
 - ESC A 19
 - ESC B 75
 - ESC CRT 75, 78
 - ESC D 77
 - ESC E 59, 77
 - ESC ESC 18, 75

- ESC J 5, 75
- ESC K 4, 75
- ESC X 4, 75
- ESC Y 27
- FCS 77
- FLAG bit 8, 40, 47, 6.12, 6.14, 71, 12.3, 12.4, 121, 12.9
- FLAG keys 11, 14
- Flying wedge 57, 6.20
- Greek letters 72
- Hatch character 12, 2.6, 2.7
- Incremental plot submodes **see** Plot submodes
- INPUT statement 2
- INTeger function 35
- Keyboard
 - CONTROL with color 75
 - encoding 142f
 - keys 11, 2.4, 2.5, 14, 75
 - special function 64, 75
- Line feed 2, 21
 - in vertical mode 1.6
- Lunar lander game 111, 12.8
- Menu programs 93, 11.1
- Mystery program 2.8, 6.11
- No-echo patch 79, 9.2, 12.8, 12.9
- Null character **see** Hatch character
- Page mode 4
- Plot characters in screen refresh 72, 8.5
- Plot co-ordinates 34
 - vs. cursor co-ordinates 34, 147
- Plot-English 65, 7.2, 7.4, 7.5
 - and blind cursor 66, 7.3
- Plot submodes 139f
 - character 31, 6.1, 6.2, 65, 106, 12.7, 139
 - in CRT mode 75
 - default values 58, 6.23
 - incremental point 41, 6.7, 6.8, 12.8, 139, 140
 - incremental vector 56, 6.20, 12.4, 139, 141
 - incremental x bar graph 46, 6.10, 6.11, 12.7, 139, 140
 - incremental y bar graph 51, 6.16, 6.24, 139, 140
 - point 33, 6.3, 6.5, 139
 - reentry to 58, 6.22, 6.24
 - vector 52, 6.17, 6.18, 6.19, 12.3, 139
 - x bar graph 45, 6.9, 6.12, 6.13, 6.14, 138
 - y bar graph 51, 6.15, 138
- PLOTting character strings 64, 6.25
- Point plot submode **see** Plot submodes
- PRINT statement 2, 1.1 - 1.3
 - and blind cursor 18
 - in Plot-English 65
 - in plot mode 64
- Pseudo-random numbers 35, 6.5
- Quad-directional scrolling patch 84, 10.1, 10.2
- Quick change artistry 131, 12.12
- Quiz 13

- Radar scope
 - using vector plot 97, 12.2
 - using incremental vector plot 99, 12.4
- Radians 38, 97, 146
- Random numbers 35, 6.5
- Refresh memory *see* Screen refresh
- Regular height characters *see* Characters
- REMark statements 11
- Roll up mode 4
- Screen displays *see* Displays
- Screen refresh 69, 8.1 - 8.6
 - vs. cursor co-ordinates 69, 146, 147
- Script 44, 6.8
- Scroll mode 4
- Scrolling patch 84, 10.1, 10.2
- Semicolon 2, 14
 - and blind cursor 19
- Special characters *see* Characters
- Special function keys 64, 75
- TAB 2
- Test mode 27, 5.1 - 5.3
 - and plot 67, 73
- Text 24, 94 11.2
 - backwards 91, 10.2
- Trigonometric values 146
- Underlining 24, 4.5
- Vector plot *see* Plot submodes
- Vertical mode 5, 1.5 - 1.7
- Wrap around
 - in plot mode 33, 48
 - in PRINT statements 3, 5, 1.2
 - screen refresh 69
- X bar graph *see* Plot submodes
- Y bar graph *see* Plot submodes

COMPUCOLOR KEYS

Chris Teo has found this useful summary of the ways that the keys can be used to produce different effects.

base	shift control command
------	-----------------------------

COMPUCOLOR KEYBOARD VALUES (CAPS LOCK ON)

64	96 ✓ null 0 ✓ END 128 ✓	P	112 ✓ black 16 ✓ OUT 144 ✓	44	60 ✓ math= ✓172 ATH ✓188	F0	200 VAL ✓192 224
A	97 ✓ auto 1 ✓ FOR 129 ✓	Q	113 ✓ red 17 ✓ PUT 145 ✓	45	61 ✓ math < ✓173 PEAK ✓189	F1	209 ASC ✓193 225
B	98 ✓ plot 2 ✓ NEXT 130 ✓	R	114 ✓ green 18 ✓ PLOT 146 ✓	46	62 ✓ SCH ✓174 LEN ✓190	F2	210 CHRS ✓194 226
C	99 ✓ Cur XY 3 ✓ DATA 131 ✓	S	115 ✓ yellow 19 ✓ SAVE 147 ✓	47	63 ✓ INT ✓175 STRS ✓191	F3	211 LEFT ✓195 227
D	100 ✓ Data Ch 4 ✓ INPUT 132 ✓	T	116 ✓ blue 20 ✓ LOAD 148 ✓	48	64 ✓ space 32 ✓ ABS ✓176 SPC(✓180	F4	212 RIGHT ✓196 228
E	101 ✓ Insert ch 5 ✓ DIM 133 ✓	U	117 ✓ magenta 21 ✓ POKE 149 ✓	49	65 ✓ CALL ✓177 THEN ✓161	F5	213 MID ✓197 229
F	102 ✓ CCI 6 ✓ READ 134 ✓	V	118 ✓ cyan 22 ✓ PRINT 150 ✓	50	66 ✓ " 34 ✓ FRE ✓178 NOT ✓162	F6	214 RIGHT ✓198 230
G	103 ✓ Ball 7 ✓ FILE 135 ✓	W	119 ✓ white 23 ✓ LIST 151 ✓	51	67 ✓ # 35 ✓ INP ✓179 STEP ✓163	F7	215 RIGHT ✓199 231
H	104 ✓ home 8 ✓ GOTO 136 ✓	X	120 ✓ xmit 24 ✓ CONT 152 ✓	52	68 ✓ 8 36 ✓ POS ✓180 math+ ✓164	F8	216 RIGHT ✓200 232
I	105 ✓ tab 9 ✓ RUII 137 ✓	Y	121 ✓ + 25 ✓ CLEAR 153 ✓	53	69 ✓ % 37 ✓ SOR ✓181 math- ✓165	F9	217 RIGHT ✓201 233
J	106 ✓ tab (t) 10 ✓ IF 138 ✓	Z	122 ✓ + 26 ✓ DEF 154 ✓	54	70 ✓ & 38 ✓ RID ✓182 math* ✓166	F10	218 RIGHT ✓202 234
K	107 ✓ er,ln,11 ✓ REST. 139 ✓	[123 ✓ escape ✓27 ✓ WAIT 155 ✓	55	71 ✓ ' 39 ✓ LOC ✓183 math/ ✓167	F11	219 RIGHT ✓203 235
L	108 ✓ er,pg,12 ✓ GOSUB 140 ✓	\	124 ✓ I ✓28 ✓ ON 156 ✓	56	72 ✓ (40 ✓ EXP ✓184 math ^ ✓168	F12	220 RIGHT ✓204 236
M	109 ✓ cr 13 ✓ RETRN 141 ✓]	125 ✓ 3 ✓29 ✓ TAB(157 ✓	57	73 ✓) 41 ✓ COS ✓185 AND ✓169	F13	221 RIGHT ✓205 237
N	110 ✓ A7 on 14 ✓ REM 142 ✓	^	126 ✓ bc on 30 ✓ TO 158 ✓	58	74 ✓ ° 42 ✓ SIN ✓186 OR ✓170	F14	222 RIGHT ✓206 238
O	111 ✓ A7 off 15 ✓ GET 143 ✓	_	127 ✓ # ✓31 ✓ FN 159 ✓	59	75 ✓ + 43 ✓ TAN ✓187 math > ✓171	F15	223 RIGHT ✓207 239

F0-F15
ONLY.

INCREMENTAL
VECTOR

Y₀
VECTOR

X₀
VECTOR

INCREMENTAL
Y BAR GRAPH

Y_{max} of
Y BAR GRAPH

X of Y BAR
GRAPH

Y₀ of
Y BAR GRAPH

INCREMENTAL
X BAR GRAPH

X_{max} of
X BAR GRAPH

Y of
X BAR GRAPH

X₀ of
X BAR GRAPH

X-Y
INCREMENTAL

Y POINT

X POINT

CHARACTER
PLOT

PLOT
ESCAPE

graphics unless lower case option

! = SEPARATE CONTROL KEY
(CONTROL + SHIFT = COMMAND)

COLOR GRAPHICS DISKETTE.....\$20.00

**CONTAINS ALMOST ALL THE PROGRAMS
FROM DAVID SUITS' BOOK. SAVES MANY
HOURS OF TYPING.**

BE SURE TO TELL WHAT COMPUTER MODEL

**ORDER FROM : JOSEPH J CHARLES PUBLISHING
P.O. BOX 750, HILTON NY 14468**

	03	4P4	.BAS;01	0026	0003	4A
	03	4P5	.BAS;01	0029	000F	4C
	03	5P1	.BAS;01	0038	0003	0B
	03	5P2	.BAS;01	003B	000C	45
	03	5P3	.BAS;01	0047	0003	58
	03	6P1	.BAS;01	004A	0003	80
	03	6P2	.BAS;01	004D	0006	42
	03	6P3	.BAS;01	0053	0003	1C
	03	6P4	.BAS;01	0056	0004	74
L	03	6P5	.BAS;01	005A	0002	40
A	03	6P6	.BAS;01	005C	0015	39
B	03	6P7	.BAS;01	0071	0002	55
E	03	6P8	.BAS;01	0073	000D	3A
L	03	6P9	.BAS;01	0080	0002	57
	03	6P10	.BAS;01	0082	0003	1A
S	03	6P11	.BAS;01	0085	0003	6E
I	03	6P12	.BAS;01	0088	0003	4B
D	03	6P13	.BAS;01	008B	0004	80
E	03	6P14	.BAS;01	008F	0005	06
	03	6P15	.BAS;01	0094	0003	24
	03	6P16	.BAS;01	0097	0007	42
	03	6P17	.BAS;01	009E	000C	42
	03	6P18	.BAS;01	00AA	0006	4F
	03	6P19	.BAS;01	00B0	0007	26
	03	6P20	.BAS;01	00B7	0003	31
	03	6P21	.BAS;01	00BA	0007	1F
	03	6P22	.BAS;01	00C1	0006	6B
	03	6P23	.BAS;01	00C7	0002	69
	03	6P24	.BAS;01	00C9	0024	0E
	03	6P25	.BAS;01	00ED	0003	1C
	03	7P1	.BAS;01	00F0	0002	06
	03	7P2	.BAS;01	00F2	0002	78
	03	7P3	.BAS;01	00F4	0004	28
	03	7P4	.BAS;01	00F8	0005	1C
	03	7P5	.BAS;01	00FD	0002	47
	03	8P1	.BAS;01	00FF	0002	6E
	03	8P2	.BAS;01	0101	0008	78
	03	8P3	.BAS;01	0109	0003	23
	03	8P4	.BAS;01	010C	0004	71
	03	8P5	.BAS;01	0110	0005	3D
	03	8P6	.BAS;01	0115	0009	6E
	03	9P1	.BAS;01	011E	0004	0C
	03	9P2	.BAS;01	0122	0022	66
	03	1P1	.BAS;01	0144	0003	40
	03	1P2	.BAS;01	0147	0001	69
	03	1P3	.BAS;01	0148	0002	55
	03	1P4	.BAS;01	014A	0001	7A
	03	GRPHCS.DSP	.DSP;01	014B	0020	80
	03	MENU	.BAS;01	016B	0002	26
	01	<FREE SPACE>		016D	0023	

	03	2P6	.BAS;01	0087	0007	0E
	03	2P7	.BAS;01	008E	0005	54
B	03	1P5	.BAS;01	0093	0003	37
A	03	1P6	.BAS;01	0096	0002	76
C	03	2P8	.BAS;01	0098	0003	42
K	03	2P9	.BAS;01	009B	0003	76
	03	10P2	.BAS;01	009E	0011	70
S	03	11P1	.BAS;01	00AF	000C	3F
I	03	11P2	.BAS;01	00BB	0006	1A
D	03	12P1	.BAS;01	00C1	0003	55
E	03	12P2	.BAS;01	00C4	0003	18
	03	12P3	.BAS;01	00C7	0006	44
	03	12P4	.BAS;01	00CD	0008	39
	03	12P5	.BAS;01	00D5	0006	26
	03	12P6	.BAS;01	00DB	0013	5B
	03	12P7	.BAS;01	00EE	0023	0B
	03	12P9	.BAS;01	0111	002A	07
	03	12P10	.BAS;01	013B	000D	06
	03	12P12	.BAS;01	0148	0013	2B
	03	12P11	.BAS;01	015B	0012	3C
	03	GRPHCS.DSP	.DSP;01	016D	0020	80
	03	MENU	.BAS;01	018D	0002	26
	01	<FREE SPACE>		018F	0001	

Label sideBack side

1P1-1P4
2P10
3P1-3P4
4P1-4P6
5P1-5P3
6P1-6P25
7P1-7P5
8P1-8P6
9P1-9P2

1P5-1P7
2P1-2P9

10P1-10P2
11P1-11P2
12P1-12P12

DIRECTORY CD0: GRAPHICS 0A

ATR NAME TYPE VR SBLK SIZE LBC

03 2P10 .BAS;01 000A 0005 37
 03 3P1 .BAS;01 000F 0002 5B
 03 3P2 .BAS;01 0011 0003 6A
 03 3P3 .BAS;01 0014 0003 52
 03 3P4 .BAS;01 0017 0002 6E
 03 4P1 .BAS;01 0019 0003 66
 03 4P2 .BAS;01 001C 0003 5E
 03 4P3 .BAS;01 001F 0002 43
 03 4P4 .BAS;01 0021 0005 7E
 03 4P5 .BAS;01 0026 0003 4A
 03 4P6 .BAS;01 0029 000F 4C
 03 5P1 .BAS;01 0038 0003 0B
 03 5P2 .BAS;01 003B 000C 45
 03 5P3 .BAS;01 0047 0003 58
 03 6P1 .BAS;01 004A 0003 80
 03 6P2 .BAS;01 004D 0006 42
 03 6P3 .BAS;01 0053 0003 1C
 03 6P4 .BAS;01 0056 0004 74
 03 6P5 .BAS;01 005A 0002 40
 03 6P6 .BAS;01 005C 0015 39
 03 6P7 .BAS;01 0071 0002 55
 03 6P8 .BAS;01 0073 000D 3A
 03 6P9 .BAS;01 0080 0002 57
 03 6P10 .BAS;01 0082 0003 1A
 03 6P11 .BAS;01 0085 0003 6E
 03 6P12 .BAS;01 0088 0003 4B
 03 6P13 .BAS;01 008B 0004 80
 03 6P14 .BAS;01 008F 0005 06
 03 6P15 .BAS;01 0094 0003 24
 03 6P16 .BAS;01 0097 0007 42
 03 6P17 .BAS;01 009E 000C 42
 03 6P18 .BAS;01 00AA 0006 4F
 03 6P19 .BAS;01 00B0 0007 26
 03 6P20 .BAS;01 00B7 0003 31
 03 6P21 .BAS;01 00BA 0007 1F
 03 6P22 .BAS;01 00C1 0006 6B
 03 6P23 .BAS;01 00C7 0002 69
 03 6P24 .BAS;01 00C9 0024 0E
 03 6P25 .BAS;01 00ED 0003 1C
 03 7P1 .BAS;01 00F0 0002 06
 03 7P2 .BAS;01 00F2 0002 78
 03 7P3 .BAS;01 00F4 0004 28
 03 7P4 .BAS;01 00F8 0005 1C
 03 7P5 .BAS;01 00FD 0002 47
 03 8P1 .BAS;01 00FF 0002 6E
 03 8P2 .BAS;01 0101 0008 78
 03 8P3 .BAS;01 0109 0003 23
 03 8P4 .BAS;01 010C 0004 71
 03 8P5 .BAS;01 0110 0005 3D
 03 8P6 .BAS;01 0115 0009 6E
 03 9P1 .BAS;01 011E 0004 0C
 03 9P2 .BAS;01 0122 0022 66
 03 1P1 .BAS;01 0144 0003 40
 03 1P2 .BAS;01 0147 0001 69
 03 1P3 .BAS;01 0148 0002 55
 03 1P4 .BAS;01 014A 0001 7A
 03 GRPHCS.DSP;01 014B 0020 80
 03 MENU .BAS;01 016B 0002 26
 01 <FREE SPACE> 016D 0023

DIRECTORY CD0: GRAPHICS 0A

ATR NAME TYPE VR SBLK SIZE LBC

03 10P1 .BAS;01 000A 000D 42
 03 12P8 .BAS;01 0017 005A 24
 03 1P7 .BAS;01 0071 0005 38
 03 2P1 .BAS;01 0076 0004 12
 03 2P2 .BAS;01 007A 0003 39
 03 2P3 .BAS;01 007D 0004 77
 03 2P4 .BAS;01 0081 0004 33
 03 2P5 .BAS;01 0085 0002 0F
 03 2P6 .BAS;01 0087 0007 0E
 03 2P7 .BAS;01 008E 0005 54
 B 03 1P5 .BAS;01 0093 0003 37
 A 03 1P6 .BAS;01 0096 0002 76
 C 03 2P8 .BAS;01 0098 0003 42
 K 03 2P9 .BAS;01 009B 0003 76
 03 10P2 .BAS;01 009E 0011 70
 S 03 11P1 .BAS;01 00AF 000C 3F
 I 03 11P2 .BAS;01 00BB 0006 1A
 D 03 12P1 .BAS;01 00C1 0003 55
 E 03 12P2 .BAS;01 00C4 0003 18
 03 12P3 .BAS;01 00C7 0006 44
 03 12P4 .BAS;01 00CD 0008 39
 03 12P5 .BAS;01 00D5 0006 26
 03 12P6 .BAS;01 00DB 0013 5B
 03 12P7 .BAS;01 00EE 0023 0B
 03 12P9 .BAS;01 0111 002A 07
 03 12P10 .BAS;01 013B 000D 06
 03 12P12 .BAS;01 0148 0013 2B
 03 12P11 .BAS;01 015B 0012 3C
 03 GRPHCS.DSP;01 016D 0020 80
 03 MENU .BAS;01 018D 0002 26
 01 <FREE SPACE> 018F 0001

Label side

Back side

1P1-1P4
 2P10
 3P1-3P4
 4P1-4P6
 5P1-5P3
 6P1-6P25
 7P1-7P5
 8P1-8P6
 9P1-9P2

1P5-1P7
 2P1-2P9

10P1-10P2
 11P1-11P2
 12P1-12P12

**COLOR GRAPHICS
FOR
INTECOLOR 3651 AND
COMPUCOLOR II COMPUTERS**

This book is intended for those who want to learn the ins and outs of color graphics for the Intecolor 3651 or Compucolor II computer. An introductory knowledge of BASIC programming is assumed.

David B. Suits has unravelled the mysteries of graphics on the Intecolor 3651 and Compucolor II computers better, perhaps, than anyone else. Suits guides the reader through the graphics capabilities of these computers with cleverness, style, and touches of genius. His appreciation of these computers infects the reader--and a wonderful infection it is! His writing is clear, understandable, witty, and engaging. The teaching ability of a good teacher shines through.

The depth and subtleties of the graphics capabilities shared with the reader must be experienced to be appreciated. There is so much more to be had if one knows how.

COLOR GRAPHICS begins with a discussion of cursor control and then proceeds through color control, test mode, plot modes, incremental plotting, screen refresh memory, and CRT mode. **COLOR GRAPHICS** contains over eighty tutorial programs and ten useful appendices. It takes up where **BASIC Training for Compucolor Computers** leaves off and explains many little-understood aspects of the machine.

David B. Suits received a Ph.D in Philosophy in 1977 from the University of Waterloo, Ontario, Canada. He presently teaches Logic and Philosophy at Rochester Institute of Technology, Rochester, New York. He has published articles in both philosophy and computer journals and is the author of a number of Intelligent Systems Corporation's programs, including BOUNCE, MAZEMASTER, LINKO, and others. His special interests include artificial intelligence, science fiction, games, and music.